ReportPlus Embedded

Web SDK Guide

Disclaimer

ReportPlus™ Embedded – Web SDK Guide 1.4

Table of Contents

Introduction

Welcome to the ReportPlus Web Embedding Guide.

The goal of this document is to describe how to embed ReportPlus Web inside an external (host) web application. The document will give a brief introduction about the ReportPlus Dashboard Viewer and Dashboards Web Repository, describing what their purpose and core functionality. The document will then describe how to embed ReportPlus Web inside a hosting web application.

Chapter 1
ReportPlus Web

Introduction

For the purposes of this document, the term *ReportPlus Server* refers to three distinct applications: the **Dashboards Web Repository**, **ReportPlus Dashboard Viewer and the ReportPlus Embed Server.**

Dashboards Web Repository

The Dashboards Web Repository is an application that allows users to store and administer folders and dashboards. The Dashboards Web Repository web interface enables users to create, edit, and delete folders and dashboards and manage the repository's permissions configuration. The repository server can be integrated with an organization's existing User/Group management to administer the repository's permissions. It can also be accessed in ReportPlus iOS, ReportPlus Android, and ReportPlus Desktop, as the main dashboards repository can be shared among many users and devices. The following is a screenshot of ReportPlus demo environment's Dashboards Web Repository:

ReportPlus Dashboard Viewer

The ReportPlus Dashboard Viewer is the web application used to render a dashboard. Dashboards are stored in the web repository. It consists of two parts: an ASP.NET webserver running on the backend and a jQuery widget on the browser frontend. The backend supplies visualization information over the network to the frontend which renders the dashboards in the users' browsers.

In more detail, the ASP.NET server is tasked with processing the dashboards, connecting to and managing the dashboards' data sources, retrieving data from those data sources, and sending it to the jQuery widget after transforming the data accordingly. This architecture makes for a very lightweight frontend, as all the data processing is performed server side.

ReportPlus Embed Server

The ReportPlus Embed Server is a component of ReportPlus installed separately for cases where you want to embed the ReportPlus Dashboard Viewer in a public facing internet or extranet application. To use ReportPlus in these types of solutions the administrator of the ReportPlus application must installer the ReportPlus Embed Server and configure one or more application users using the ReportPlus Repository Permission dialog.

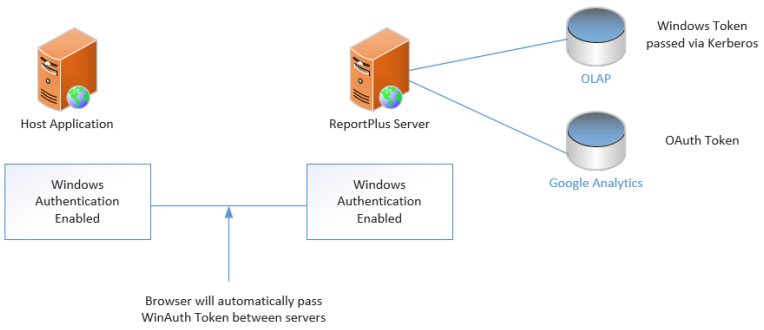
Security

ReportPlus supports two types of authentication when embedding in a host application:

- **Windows Authentication**, which is recommended when embedding ReportPlus into an internal application.
- Token-based **Single Sign On (SSO)**, which is the recommended approach for public-facing and extranet solutions.

Windows Authentication

This is the default authentication mode for embedding and should be used by internal applications**. Both the host application and ReportPlus server must be configured to use Windows Authentication with *anonymous authentication disabled***. When a host application renders a dashboard, the browser will pass the end user's windows credentials token to the server via CORS (Cross-Origin Resource Sharing).



Note: For browsers other than Internet Explorer, the users of the host application will be prompted to enter in their Windows Authentication credentials; these may be cached by their browser for later use until the user's Windows password expires. Currently, Apple's Safari browser does not support Windows Authentication over CORS. For applications that require Safari, the SSO token-based authentication is the recommend approach.

Dashboard Access Level

Once the end user has been authenticated, the ReportPlus server will verify if the end user has the necessary permissions to view the requested dashboard. To configure access, use the ReportPlus Server Repository screen to define what folders and dashboards the user can access. Once the permissions have been granted, the user will be able to view the dashboards using the standard ReportPlus Repository screen or from the host (embedding) application.

Data Access Level

Users must have the appropriate credentials to access the dashboard's data sources. There are several ways for those credentials to be provided; in all cases, the authentication and authorization is delegated to the data sources themselves.

ReportPlus supports three security models when accessing data:

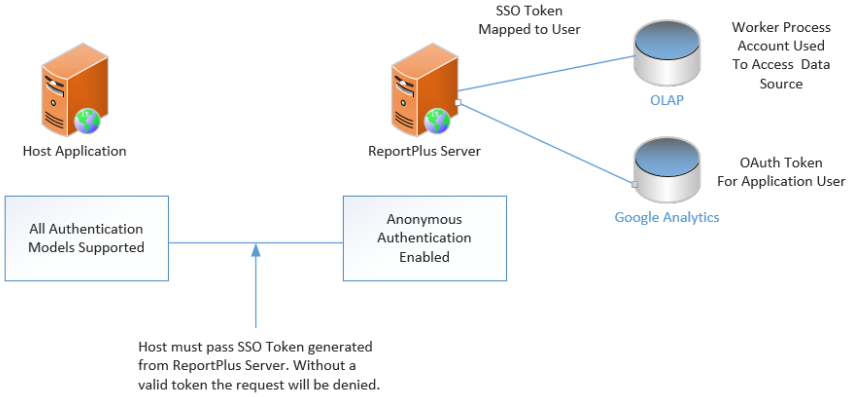
- **User Name/Password-based credentials.** If by the time the dashboard is shown users have not defined the appropriate credentials for the dashboard's data sources, they will be redirected to an HTTPS modal dialog where they can specify them. These credentials are then stored using a per-user basis encryption in a secure store.
- **OAuth based credentials.** If by the time the dashboard is shown users have not defined the appropriate credentials for an OAuth-based data provider, they will be redirected to the data source (e.g. Google, Dropbox, etc.) authentication screen. Once ReportPlus has been granted access by the OAuth data provider, the access security token returned will be encrypted and stored in the secure store for later use.
- **Kerberos Delegation**. Use Windows Single Sign-On authentication to access enterprise data sources that support it.

Note: For additional information about the required security configuration for ReportPlus, see the ReportPlus Web – Kerberos Sign-On Configuration and ReportPlus Web – OAuth Configuration guides.

Single Sign-On (SSO) Token Authentication

This is the authentication mode recommend for public-facing and extranet-based solutions that are not using Windows Authentication or host applications that want to manage users outside of the ReportPlus enterprise repository. **When using SSO authentication, the ReportPlus Embed Server web application must be installed and configured in Internet Information Services (IIS) web server with *Windows authentication disabled and anonymous authentication enabled***. See the ReportPlus Server installation document for more details about installing ReportPlus for embedding.

The host application can be configured to use any authentication protocol they want and will be responsible for managing their users independently of ReportPlus Server. When displaying a dashboard, the host application must supply a SSO Token using the generated security key provided by ReportPlus server. The dashboard viewer will pass the SSO token to the ReportPlus server which will use the token to authenticate and authorize the application and which dashboards can be displayed.

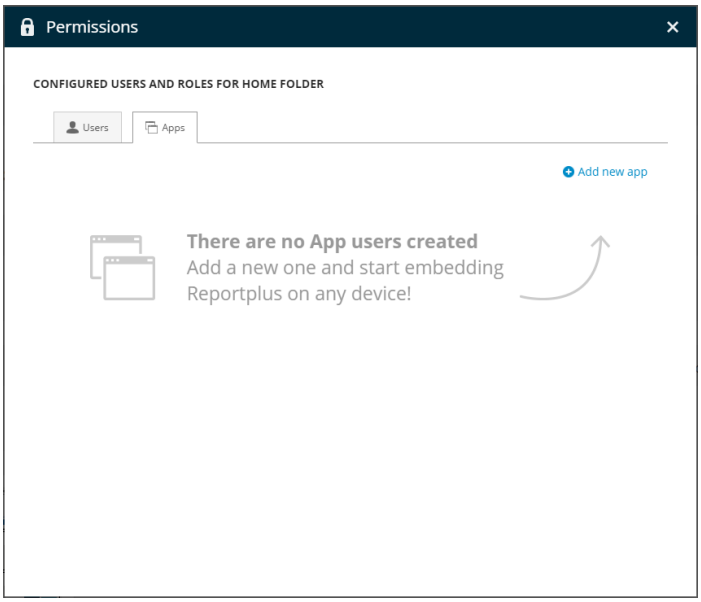


Note: Currently ReportPlus uses JSON web tokens (https://jwt.io/introduction/) for SSO authentication. All browsers, including Apple's Safari support this authentication approach and the end user will not be prompted when the dashboard viewer is displayed. It is up to the host application to manage end user authentication and map the users to one or more SSO tokens.

Creating an SSO Token

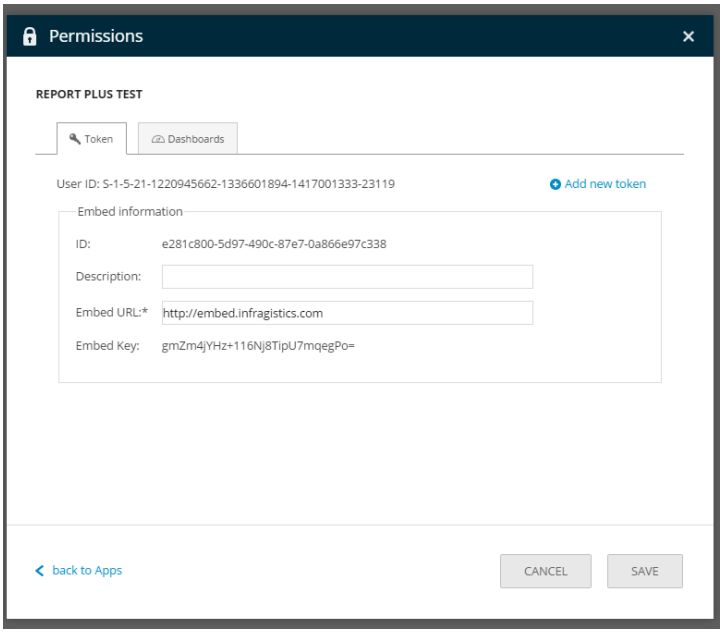
When using a SSO Token, ReportPlus will verify if the supplied token maps to a valid application user in the ReportPlus enterprise repository. Once the application has been authenticated, the ReportPlus server will verify if the application user has the necessary authorization to view the requested dashboard.

The administrator of ReportPlus or the developer of the host application will be responsible for generating a security key for an application user. To configure an application and grant access to one or more dashboards, the ReportPlus Permission Dialog must be loaded from the root folder in the web repository screen.



To authenticate using SSO, an application user must be granted access to ReportPlus, and one or more security token keys must be generated. The application user must exist as a standard Windows user in Active Directory before being added to ReportPlus through the Permission Dialog.

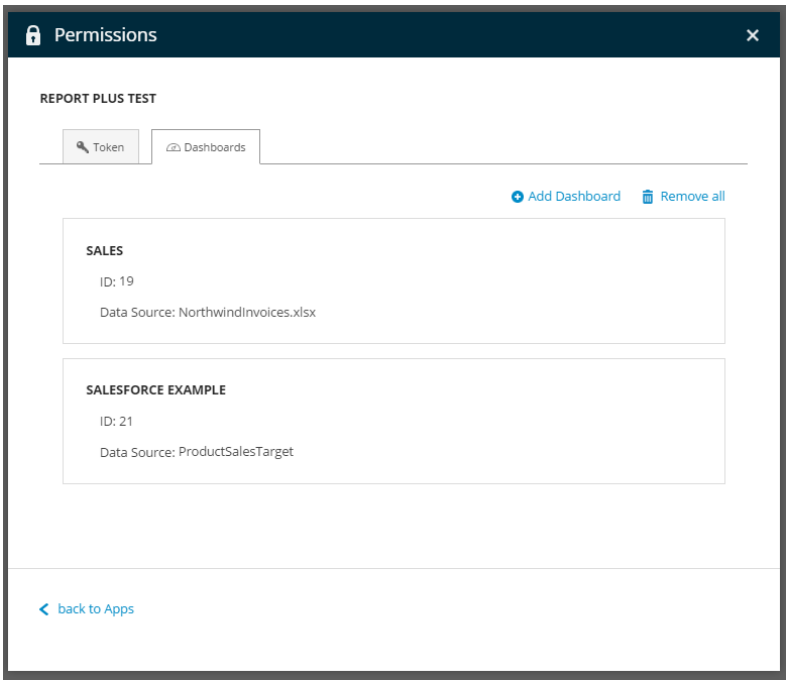
To generate a security key, a description and host URL must be supplied. Multiple keys can be generated for the same application; the only requirement is that each URL must be unique. When the SSO token is sent to ReportPlus, it will verify the calling host URL matches the one associated with the security key.



Dashboard Access and Managing Data Source Credentials

When using SSO authentication, dashboard security is independent from the normal repository folder and dashboard access. To grant an application user access to a dashboard, the ReportPlus administrator must configure both the dashboard and the necessary credentials for the application user through the Manage Dashboard Screens. This screen can be displayed by selecting an application user from the permission dialog.

Using this screen, the administrator can configure any dashboards that require username/password or OAuth-based credentials. For dashboards that authenticate using Kerberos delegation, the configured user of the ReportPlus ASP.NET worker process will be used and must have the necessary access to retrieve data from the data source (e.g. read-only access to SQL Server or SQL Analysis Server - OLAP).


Permissions
REPORT PLUS TEST
Token
Dashboards
Add Dashboard
Remove all
SALES
ID: 19
Data Source: NorthwindInvoices.xlsx
SALESFORCE EXAMPLE
ID: 21
Data Source: ProductSalesTarget
back to Apps

Chapter 2
ReportPlus Server

Requirements

To install ReportPlus Server, the following is required:

- A Windows Server with 4 GB or more of memory.
- Access to a MS SQL Server. (`Repository`, `SecureStore`, and a few other databases used by the dashboard server will be created).
- The Windows server should:
  - Have IIS Installed.
  - Have .NET Framework 4.6 and ASP.NET 4.0 registered in IIS.
  - Have Windows authentication enabled for IIS.
  - Be part of a network domain with access to users in an Active Directory. Dashboards' repository security is integrated with an Active Directory module in this version of ReportPlus Web.
  - Depending on the data sources that will be used for the dashboards, the Windows server may need to have some additional components installed to work properly with those data sources (e.g. to use Analysis Services or SharePoint the relevant client libraries must be installed). There are no other dependencies to just use SQL Server or Excel.
- Supported web browsers:
  - Internet Explorer 11
  - Microsoft Edge
  - Chrome (two most recent major versions)
  - Firefox (two most recent major versions)
  - Safari (only when using JSON Web Token SSO authentication)

ReportPlus Server Installer

The ReportPlus Web installer can be used to install a complete version of ReportPlus Web App, including the Web Repository.

The installation guide is out of this document' scope, but, to have a simple version of ReportPlus Web installed, following the default configuration settings should be enough.

> **Note:** If using SSO Authentication, the ReportPlus embed web application needs to be installed as a separate IIS Application with anonymous access enabled. Use the ReportPlus Server installer to deploy the embed web application.

Principal configuration settings

During the installation process, some features of ReportPlus Web are configured and others are left with default values to be configured later if necessary.

The following are the main configuration settings, which can be modified in the `web.config` file of the ReportPlus server application.

Data Providers Configuration

In the ReportPlus Web App's `web.config` there a section named `infragistics.reportplus/extensions/dataProviders`. This section is used to configure the data sources used by ReportPlus. **For ReportPlus Server, consumer keys and secrets for any OAuth2 data providers must be configured when embedding**. Below is an example of what this section may look like.

```
<!--web.config snippet-->
<configuration>
    <infragistics.reportplus>
        <extensions>
            <dataProvider providerId="DROPBOXPROVIDER">
                <providerProperties>
                    <property name="ConsumerKey" value="consumer_key"/>
                    <property name="ConsumerSecret" value="consumer_secret"/>
                </providerProperties>
            </dataProvider>
            <dataProvider providerId="FACEBOOK">
                <providerProperties>
                    <property name="ConsumerKey" value="consumer_key"/>
                    <property name="ConsumerSecret" value=" consumer_key "/>
                </providerProperties>
            </dataProvider>
            <dataProvider providerId="GOOGLE_ANALYTICS">
                <providerProperties>
                    <property name="ConsumerKey" value="consumer_key"/>
                    <property name="ConsumerSecret" value=" consumer_key"/>
                </providerProperties>
           </dataProvider>
        </extensions>
    </infragistics.reportplus>
</configuration>
```

Note: Any settings modified in this section will need to be changed in both the standard ReportPlus Server `web.config` along with the ReportPlus Embed Server `web.config`.

Geolocalization Bing Key

ReportPlus Web App renders the maps using Bing Maps; the Bing Maps' key is set in ReportPlus Web's `web.config` file.

```
<!--web.config snippet-->
<infragistics.reportplus>
    <geolocation geolocationBingKey="bing_key"/>
</infragistics.reportplus>
```

Logging

ReportPlus Server uses Log4Net to log dashboard server errors, warnings and helpful troubleshooting information. The quality of the information will depend on the configured logging level.

```
<log4net>
  <root>
    <level value="INFO"/>
    <appender-ref ref="RollingFileAppender"/>
    <appender-ref ref="TraceAppender"/>
  </root>
  <appender name="TraceAppender" type="log4net.Appender.TraceAppender">
    <layout type="log4net.Layout.PatternLayout">
      <conversionPattern value="%date [%5thread] [%username] %-5level - %message%newline%exception%newline"/>
    </layout>
  </appender>
  <appender name="RollingFileAppender" type="log4net.Appender.RollingFileAppender">
    <file value="_log\app.log"/>
    <appendToFile value="true"/>
    <rollingStyle value="Size"/>
    <maxSizeRollBackups value="10"/>
    <maximumFileSize value="10MB"/>
    <staticLogFileName value="true"/>
    <layout type="log4net.Layout.PatternLayout">
      <conversionPattern value="%date [%5thread] [%username] %-5level - %message%newline%exception%newline"/>
    </layout>
  </appender>
</log4net>
```

Local Files Folder

In some cases, dashboards are designed using local file data sources, with files stored in the device in which the dashboard was designed. To load a dashboard dependent on local files in ReportPlus Web App, ensure those local files are also accessible to the ReportPlus Web App server. The path to those files is specified in ReportPlus Web's `web.config` file.

```
<!--web.config snippet-->
<configuration>
    <infragistics.reportplus>
        <extensions>
            <dataProvider providerId="LOCALFILE">
                <providerProperties>
                    <property name="RootFolder" value="path_to_local_file_folder"/>
                </providerProperties>
            </dataProvider>
        </extensions>
    </infragistics.reportplus>
</configuration>
```

Databases

If installing on the default installation path, then these entries are all mapped to the same database (if necessary, they can be separated).

- `repository`: The tables that persist dashboards, folders, and permissions for the dashboard repository.

```
<!--web.config snippet-->
<configuration>
    <infragistics.reportplus>
        <repository repositoryConnectionString="connection_string_for_repository"/>
    </infragistics.reportplus>
</configuration>
```

- `security`: Maintains users and passwords configured to have access to different data sources. This information is encrypted.

```
<!--web.config snippet-->
<configuration>
    <infragistics.reportplus>
        <security
            secureStorageConnectionString="connection_string_for_secure_storage"
        />
    </infragistics.reportplus>
</configuration>
```

- `usageMetrics`: If enabled, this database stores usage information including visualizations and data sources.

```
<!--web.config snippet-->
<configuration>
    <infragistics.reportplus>
        <usageMetrics
            usageMetricsConnectionString="connection_string_for_usage_metrics"
        />
    </infragistics.reportplus>
</configuration>
```

Chapter 3
ReportPlus Web App

Embedding the ReportPlus Dashboard Viewer widget in an application

The web application referred to as ReportPlus Web App is an ASP.NET MVC web application. The main webpage hosts a jQuery widget named `igDashboardViewer`, which is responsible for interacting with the backend and rendering a requested dashboard.

This same widget can be hosted by other web applications in the same way it's hosted by the ReportPlus Web App's main webpage. The following sections will explain how to accomplish this.

Dependencies

To use the `igDashboardViewer` widget, the host web application must reference:

- jQuery 1.9.1 or greater
- jQuery UI 1.10.2 or greater
- Modernizr 2.6.2 or greater
- IgniteUI 16.2 or greater
    - Data Visualizations library is not needed. Only reference infragistics.core.js and infragistics.lob.js

Note: Its recommend that you copy these files from the R+ Server – /dist/Vendor/igniteui, /dist/Vendor/jqueryui and /dist/Vendor/modernizr folders.

Steps

The ReportPlus Dashboard Viewer code has dependencies to controls which render the visualization for dashboards. Some of these controls include different types of charts, grids, and gauges.

To embed ReportPlus Dashboard Viewer inside a host web application, follow these steps:

1. In the host application project, create a folder called `rplus`. This folder is where all the required assets (CSS, JavaScript, etc.) will be copied over from the ReportPlus Server.
2. On the ReportPlus Server, look for the folder named `rplus-embedded`. This folder contains the minified and bundled version of the ReportPlus Dashboard viewer. The following files must be **copied** from the `rplus-embedded` folder to the `rplus` folder created in the host application.
    - embedded.bundle.css
    - rplus.embedded.bundle.js
    - 33cdea540128fd7ccaf774542b012c0e.png
    - 565d414f3f06792667d5fa4659acbc1c.png
    - (Optional) Source map files to facilitate debugging the code in the browser's console
        - embedded.bundle.css.map
        - rplus.embedded.bundle.js.map
3. Copy the **themes** folder from ReportPlus Server to your local web project. Any custom themes used will need be too copied to this folder. If a custom theme is missing the default (The Blues Theme) will be used as a fallback.

4. **Reference** the following **stylesheets and scripts** on the webpage where ReportPlus will be embedded:

```
<link href="dist/Vendor/igniteui/infragistics.css" rel="stylesheet" />
<link href="dist/Vendor/igniteui/infragistics.theme.css" rel="stylesheet" />
<link href="rplus-embedded/embedded.bundle.css" rel="stylesheet" />
```

```
<script src="dist/Vendor/jqueryui/jquery.min.js"></script>
<script src="dist/Vendor/jqueryui/jquery-ui.min.js"></script>
<script src="dist/Vendor/modernizr/modernizr.min.js"></script>
<script src="dist/Vendor/jqueryui/jquery.touchSwipe.min.js"></script>
<script src="dist/Vendor/igniteui/infragistics.core.js"></script>
<script src="dist/Vendor/igniteui/infragistics.lob.js"></script>
<script src="rplus-embedded/rplus.embedded.bundle.js"></script>
```

Instantiate the `igDashboardViewer` jQuery widget.

```
$(function() {
  var defaultDashboard = 1;
  $("#dashboard").igDashboardViewer({
     dashboardSettings: {
        definitionUri: "[repository]" + defaultDashboard,
        serviceEndpointUri: serverUrl,
    });
});
```

Widget parameters

The widget can be instantiated with the follow parameters:

- `definitionUri`: the URI that specifies the dashboard to be rendered. The URI includes two parts:
  - The `[repository]`, which means the browser is trying to open a dashboard saved in ReportPlus server repository.
  - The `[Id]` of the dashboard to load from the server repository.

- `serviceEndpointUri`: The URL to the server when the server is configured to accept HTTP requests.

- `httpsServiceEndpointUri`: The URL to the server when the server is configured to accept HTTPS requests.

- `hideAppHeader`: Optional flag (defaults to `false`) to show or hide the header bar for the ReportPlus Dashboard viewer.

- `hideDashboardParametersUI`: Optional flag (defaults to `false`) to show or hide the global filter control.

- `hideSourceIcon`: Optional flag (defaults to `false`) to show or hide the data source icon. When visible, the end user will be allowed to change their data source configuration for the dashboard. Any changes will be saved to the ReportPlus Server.

- `hideFlipGridIcon`: Optional flag (defaults to `false`) to show or hide the flip grid icon. When visible, the end user will be able to switch between the normal state of a visualization and its grid view.

- `dashboardParameters`: Optional (defaults to empty) multi-level array of values that corresponds to one or more global filters and the default values to filter data by.

```
"SingleSelection": ["Sales"],
"MultiNoUnsel": [2, 4, 9, 1]
```

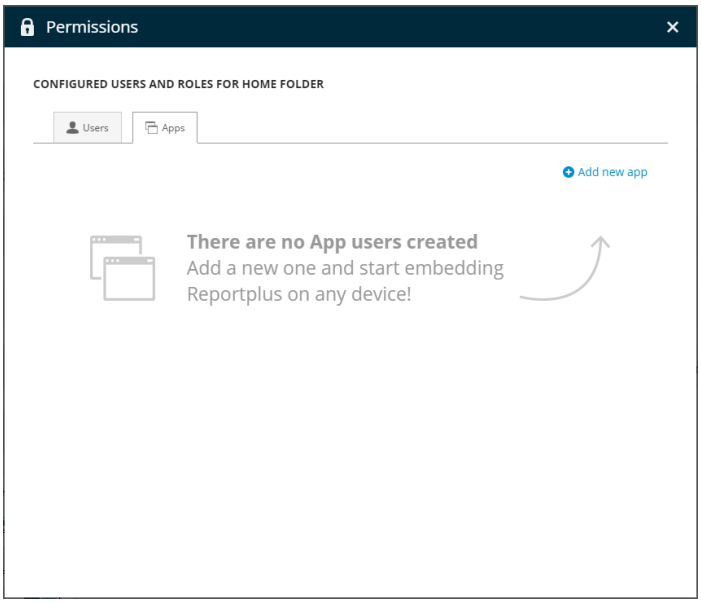
- `securityToken`: Optional string (defaults to `null`) that is only required when using SSO-based authentication. The token must be serialized as a JSON Web Token (JWT) with the correct supported claims for ReportPlus.
- `showExportButton`: Optional boolean (defaults to `false`) to show or hide the export data button. When visible, the end user will be able to export the dashboard data to a csv file.

Generating the Security Token

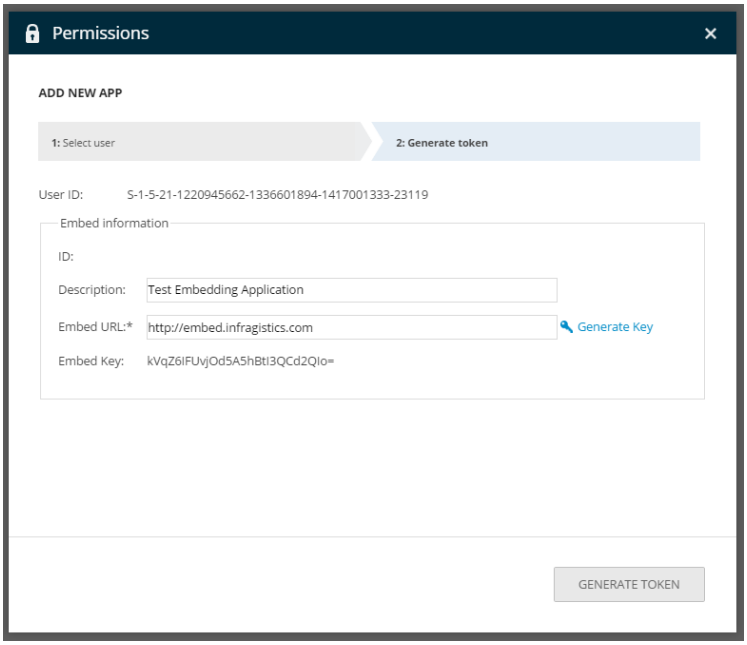
To use Single Sign On (SSO) token authentication when embedding ReportPlus, a JSON Web Token (JWT) must be generated first. For more information on the JSON Web Token (JWT) see https://jwt.io/introduction/.

Creating a Security Token

To create the required JSON Web Token (JWT), create an application user using the standard ReportPlus user Interface. Go to the root folder in the enterprise repository and click on permissions. When the dialog is finished loading, select the apps tab and click on create a new app user.



A security token key must be generated which is used to encrypt the JSON Web Token (JWT). To generate the token, a description and URL must be provided. To authenticate the security token, ReportPlus will verify the token was encrypted with the correct key and the specific URL matches the incoming request.



Once the key is generated, the JSON Web Token (JWT) must be constructed with the following claims included:

Claim	Description
`version`	The version of the app token, 1 is the current version
`userIdentifier`	The User GUID provided when ReportPlus generated the token
`applicationName`	The hosted (embedding) application name
`hostUri`	The URL for the hosted (embedding) application.

Sample Server Code

The provided embed sample is an ASP.NET MVC application that includes the custom helper class `TokenFactory` that makes it easier to generate the JSON Web Token (JWT), this class has a dependency on the `System.IdentityModel.Tokens.Jwt` NuGet package provided by Microsoft. If preferred, any third party library to generate JSON Web Tokens. See https://jwt.io/ for more details about available libraries for generating security tokens.

```
public static class TokenFactory

    public static string CreateToken(TokenParameter parameter)
    {
        var token = new JwtTokenService(parameter.Secret);
        token.CreateSigningCredentials();
        token.AddValue("version", 1);
        token.AddValue("userIdentifier", parameter.UserIdentifier);
        token.AddValue("applicationName", parameter.ApplicationName);
        token.AddValue("HostUri", parameter.HostUri);
        return token.SerializeToken();
    }
```

Appendices

Appendix 1: Document Changelog

Appendix 1: Document Changelog

Version	Chapter	Description
1.4	ReportPlus Web App	Updated dependencies, steps, and added new widget parameter (`showExportButton`)
1.3	ReportPlus Web	Updated Security chapter.
	ReportPlus Web App	Updated widget parameters with `hideAppHeader` and `pathToFiles.`
1.2	All chapters	Updated document to include new SSO-based security authentication model for embedding.
1.1	Introduction	Data Source Authentication added to security levels.
1.0	All chapters	Separate document created for install, Kerberos Single Sign-On, and embedding guide.