JavaScript API & Plugins

Audience

This documentation is intended for users who are familiar with JavaScript programming and are experienced with creating dashboards using Sisense.

Use this Getting Started guide to familiarize yourself with the concepts of Sisense's Javascript API and start enhancing Sisense with custom functionality.

Who can use the API?

Any Sisense designer can extend Sisense dashboards and widgets with the Javascript API. Limiting API usage to administrators can only be achieved using the user role customization feature.

Deploying Javascript plugins to the Sisense installation folder can be performed by anyone with access to the physical drive.

What can I do with the API?

The Sisense web application was built to be easily extensible via the Javascript API. Javascript can be implemented using the following two methods:

- **Dashboard and Widget extensions** - Apply Javascript code via Sisense's built in Javascript editor for specific widgets and dashboards. This will apply your code selectively to the dashboards and widgets that you choose. See Adding Custom JavaScript to a Dashboard and Adding Custom JavaScript to a Widget below.
- **Plugins** - Deploy Javascript code as plugins into the Sisense environment. This will apply your Javascript code to the entire application. See Creating Plugins with JavaScript below.

Do things such as:

- Build new types of widgets
- Tweak out-of-the-box visualizations to match the exact look and feel you are looking for
- Fine tune the application behavior for specific needs
- Apply custom formatting

Security and Access Control

All scripts are activated in the context of a user session. Any Sisense API calls are fully authenticated and authorized using the application authentication mechanism.
Scripts are subject to the same strict API governance and internal safeguards as the application.
While JavaScript add-ons can modify the look and feel of the application, they do not allow access to any information not available to the currently authenticated user.

Adding Custom JavaScript to a Dashboard

Enhance Dashboard functionality by adding JavaScript to it in the Sisense web app.

To add custom JavaScript to a dashboard:

1. Click the Dashboard menu button in the top-right corner of the Dashboard and select **Edit Script**.
2. In the JavaScript editor that opens, you can add and edit your JavaScript code.
3. Click Save and refresh your dashboard.

Adding Custom JavaScript to a Widget

Enhance widget functionality by adding JavaScript to it in the Sisense web app.

To add custom JavaScript to a Widget:

1. Hover over the top-right of the widget and click to edit the widget.
2. Click the Widget menu in the top-right corner of the widget and select **Edit Script**.
3. In the JavaScript editor that opens, you can add and edit your JavaScript code.

JavaScript Implementation Examples

- Funnel Chart Plugin
- Accordion Plugin
- Jump-to-Dashboard Plugin
- Custom Styling of Pivots

Visit our forums for more examples

- Javascript API user forum
- Javascript Plugins user forums

For a full list of Sisense Plugins, click here.

Creating Plugins with JavaScript

You can edit the JavaScript file for an existing plugin, or create a new plugin using JavaScript.

To add custom JavaScript to a plugin:

1. Browse to your Plugins folder: `C:\Program Files\Sisense\PrismWeb\plugins`, and open the specific plugin folder. If you are creating a new plugin, create a new plugin folder under thePlugins folder.
2. Open the widget.js file in an editor, and update the code.

For Sisense 7.2 Users

In Sisense V7.2, the location has been changed to `C:\Program Files\Sisense\app\plugins`.

Your existing plugins will be migrated to the new location when upgrading to v7.2.

For more information, please see the v7.2 Developer Release Notes
Note

If an extra JavaScript library is required, you can add a reference to it from the JSON file in the same plugin folder. You can use a relative path to the same folder.

See the following section to see how JavaScript is used to create a new widget.

Creating New Widgets

The easiest way to create a new widget in Sisense is to modify an existing widget, however you can create widgets from scratch. This procedure describes how to create a new widget, but the principles described here can be applied to modifying existing widgets as well.

After you have created a widget, it is displayed in your Widget's list in the Sisense Web Application.

To create a new widget plugin:

1. Create or locate your Plugins folder: `C:\Program Files\Sisense\PrismWeb\plugins`. If the plugins folder does not yet exist, create it under PrismWeb.
2. In the plugin folder, create the following files:
    - **plugin.json**: This file defines your widget's name and refers to the source files that define your widget's styling and functionality. See Building a Widget for more information.
    - **widget.css**: This CSS file contains information about the style of your widget. See Styling Your Widget for more information.
    - **widget.js**: This file defines your widget's functionality. See Defining a Widget's Functionality for more information.
3. Log into a dashboard, and create a new widget. Open the **Advanced** configuration, and you should see the name of your new plugin in the list of available chart types.

Building a Widget

Each widget you create must contain a plugin.json file. This file should be in JSON format with all quotes and without trailing comma.

The plugin.json file can contains the following keys:

name: Contains a string that represents the name of your widget as it will be displayed in the Sisense Web Application.

pluginInfraVersion: An integer that defines which version of the Sisense JS API infrastructure you are using for your widget. From 6.6.1 onwards, new widgets should use the latest infrastructure. There are two possible values:

1: (Default) Plugins prior to Sisense V6.6.1.

2: Plugins created from Sisense V6.6.1 and later.

source: An array that contains string references to JavaScript libraries that define your widget's functionality. See Defining a Widget's Functionality for more information.

style: An array that contains string references to CSS files that define your widget's appearance. See Styling Your Widgets for more information.

Source and style files should be saved in the same folder as your plugin.json file or referenced in the plugin.json file with a relative path.

Below is an example of a plugin.json:

```
{
 "name": "pluginName",
 "pluginInfraVersion": 1,
   "source": [
     "index.js",
       "functions.js"
 ],
   "style": [
     "main.css",
       "main-ie.css"
 ],
}
```

Defining a Widget's Functionality

The JavaScript files that you include in your plugins folder and refer to in the plugins.json file run when the dashboard is loaded. As your dashboard is loaded, your widget is registered and is displayed in your list of widgets.

For information about Sisense objects, events, and functions that you can use in your widgets, see the JavaScript API Reference.

Styling Your Widget

You can define the the look and appearance of your widgets by including CSS files in your plugin folder and then referring to these files when relevant.

To include this file in your widget, it should be referred to in the source array of the plugins.json file.

If your widget includes any graphics or images, these can be included in an Images folder in your plugin's folder. To access images in the folder, provide a relative path to the Images folder.

Below is an example of a widget that refers to an image saved in the Images folder:

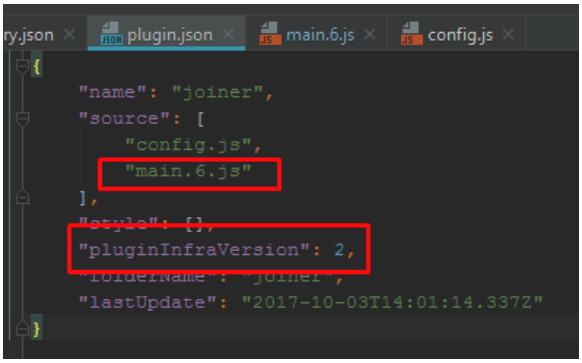
```
body {
 background: url('/plugins/pluginName/images/example.png');
}
```

Implementing ES6

Sisense supports ES6 in JavaScript plugins.

To implement ES6 in your plugins:

1. In plugin.json add: "pluginInfraVersion" : 2
2. The files that are using ES6 must contain the digit "6" in the file name
   For example: main.6.js. The marked files will go through babel.



Debugging Plugins

Sisense enables you to debug your plugins while they are in production. Inside your **plugins** folder located in the directory C:\Program Files\Sisense\PrismWeb\plugins is the entry.json file, which determines if your plugin files are condensed by Sisense in production and which version of your plugin to use.

The entry.json file has the following structure:

```
{
   "isProd": true,
   "watchIgnore": [
       ".hg$",
       ".git$",
       "node_modules"
   ]
}
```

There are two objects you can define:

isProd: When true, Sisense minimizes your CSS and JavaScript files. If you need to debug your plugin, you should set this value to false.

watchIgnore: This array defines a list of plugin files and folders that are to be ignored by Sisense. When you modify your plugin files, it triggers a new plugin build cycle. By adding your files and folders to this array, Sisense ignores any changes to your plugins and continues using the plugin that was available during the last plugin build cycle. Strings in this array are in listed as regular expressions so you define exactly which file and folder to ignore.