

SIEMENS

Ingenuity for life



Integrating User-Defined Controls into WinCC Unified (Custom Web Controls)

Custom Web Controls for WinCC Unified Systems

<https://support.industry.siemens.com/cs/ww/en/view/109779176>

Siemens
Industry
Online
Support



Legal information

Use of application examples

Application examples illustrate the solution of automation tasks through an interaction of several components in the form of text, graphics and/or software modules. The application examples are a free service by Siemens AG and/or a subsidiary of Siemens AG ("Siemens"). They are non-binding and make no claim to completeness or functionality regarding configuration and equipment. The application examples merely offer help with typical tasks; they do not constitute customer-specific solutions. You yourself are responsible for the proper and safe operation of the products in accordance with applicable regulations and must also check the function of the respective application example and customize it for your system.

Siemens grants you the non-exclusive, non-sublicensable and non-transferable right to have the application examples used by technically trained personnel. Any change to the application examples is your responsibility. Sharing the application examples with third parties or copying the application examples or excerpts thereof is permitted only in combination with your own products. The application examples are not required to undergo the customary tests and quality inspections of a chargeable product; they may have functional and performance defects as well as errors. It is your responsibility to use them in such a manner that any malfunctions that may occur do not result in property damage or injury to persons.

Disclaimer of liability

Siemens shall not assume any liability, for any legal reason whatsoever, including, without limitation, liability for the usability, availability, completeness and freedom from defects of the application examples as well as for related information, configuration and performance data and any damage caused thereby. This shall not apply in cases of mandatory liability, for example under the German Product Liability Act, or in cases of intent, gross negligence, or culpable loss of life, bodily injury or damage to health, non-compliance with a guarantee, fraudulent non-disclosure of a defect, or culpable breach of material contractual obligations. Claims for damages arising from a breach of material contractual obligations shall however be limited to the foreseeable damage typical of the type of agreement, unless liability arises from intent or gross negligence or is based on loss of life, bodily injury or damage to health. The foregoing provisions do not imply any change in the burden of proof to your detriment. You shall indemnify Siemens against existing or future claims of third parties in this connection except where Siemens is mandatorily liable.

By using the application examples you acknowledge that Siemens cannot be held liable for any damage beyond the liability provisions described.

Other information

Siemens reserves the right to make changes to the application examples at any time without notice. In case of discrepancies between the suggestions in the application examples and other Siemens publications such as catalogs, the content of the other documentation shall have precedence.

The Siemens terms of use (<https://support.industry.siemens.com>) shall also apply.

Security information

Siemens provides products and solutions with Industrial Security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the Internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place. For additional information on industrial security measures that may be implemented, please visit <https://www.siemens.com/industrialsecurity>.

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed at: <https://www.siemens.com/industrialsecurity>.

Table of Contents

| | | |
|----------|---|-----------|
| | Legal information | 2 |
| 1 | General Information about Custom Web Controls | 4 |
| 2 | "Gauge" Custom Web Control | 5 |
| 2.1 | Introduction | 5 |
| 2.1.1 | Overview | 5 |
| 2.1.2 | Principle of Operation | 5 |
| 2.1.3 | Components Used | 6 |
| 2.2 | Engineering | 7 |
| 2.2.1 | Configuration and Implementation of the Custom Web Control | 7 |
| 2.2.2 | Installation and Integration into the User Project | 11 |
| 2.2.3 | Debugging | 12 |
| 3 | Custom Web Control "Table" | 14 |
| 3.1 | Introduction | 14 |
| 3.1.1 | Overview | 14 |
| 3.1.2 | Range of Functions | 14 |
| 3.1.3 | Components Used | 14 |
| 3.1.4 | Use Cases | 15 |
| 3.2 | Engineering | 16 |
| 3.2.1 | Configuration and Implementation of the Custom Web Control | 16 |
| 3.2.2 | Installation and Integration into the User Project | 18 |
| 3.2.3 | Debugging | 18 |
| 3.2.4 | Necessary Data | 18 |
| 4 | Sample Project Operation | 20 |
| 4.1 | Custom Web Control "Gauge" Operation | 20 |
| 4.2 | Custom Web Control "Table" Operation | 21 |
| 4.2.1 | Transferring Table Contents via a Script | 21 |
| 4.2.2 | Transferring Table Contents via a Screen Object | 22 |
| 4.2.3 | Transferring Table Contents via a CSV File | 23 |
| 5 | Frequent Error Constellations | 25 |
| 5.1 | Custom Web Control does not appear in the TIA Portal Toolbox | 25 |
| 5.2 | TIA Portal Does not Display the Custom Web Control Correctly in the Screen | 26 |
| 5.3 | Custom Web Control Remains Empty in Runtime | 27 |
| 5.4 | Custom Web Control Contains no WinCC Data | 27 |
| 5.5 | Custom Web Control Cannot Write WinCC Data | 28 |
| 6 | Useful Information | 30 |
| 6.1 | Tips & Tricks | 30 |
| 6.2 | Alternative Solutions | 31 |
| 7 | Appendix | 32 |
| 7.1 | Service and support | 32 |
| 7.2 | Industry Mall | 33 |
| 7.3 | Links and literature | 33 |
| 7.4 | Change documentation | 33 |

1 General Information about Custom Web Controls

WinCC Unified offers the option of Custom Web Controls (or CWC for short). This means that your visualization is not limited to the standard Controls of WinCC Unified.

You have the opportunity to create your customer-specific controls with web technology or to use already existing Controls and to apply them to further WinCC Unified Runtime Stations.

The aim of this application example is to show you how to integrate "Custom Web Control" created with Visual Studio Code into WinCC Unified.

Note

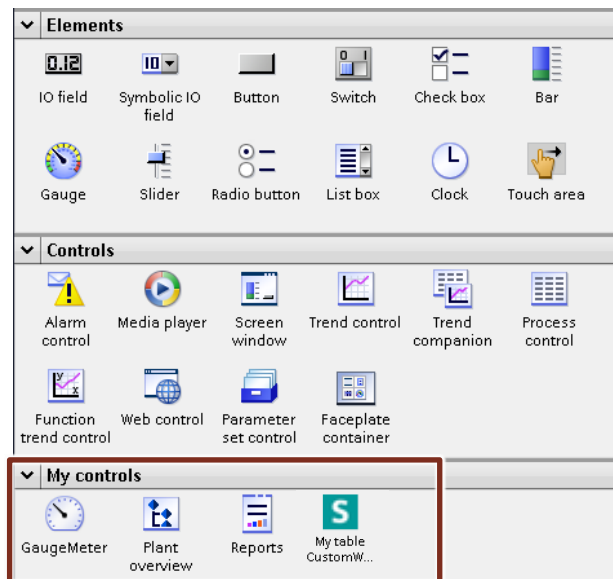
Custom Web Controls and the following configuration steps work with the WinCC Unified PC Runtime and the Unified Comfort Panels.

Please note the power limit of Unified Comfort Panels. Custom Web Controls with 3D representations are not recommended for Unified Comfort Panels, for example.

WinCC Unified Controls and elements

The following figure shows the elements, standard controls, and custom controls ("My controls") in WinCC Unified. The section "My controls" contains the Custom Web Controls.

Figure 1-1



2 "Gauge" Custom Web Control

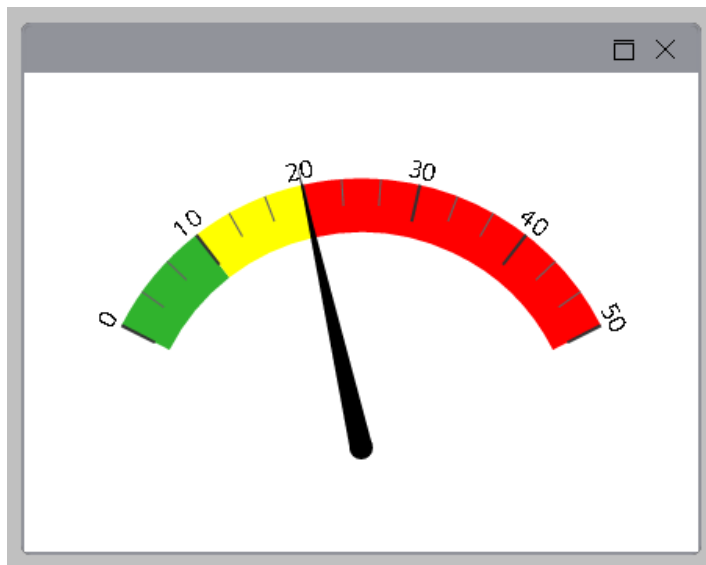
2.1 Introduction

2.1.1 Overview

[Figure 2-1](#) shows the Custom Web Control "Gauge", which is used as an example to demonstrate the individual configuration steps for integration in WinCC Unified.

The example Control is from an OpenSource third-party library "Gauge.js" and will be expanded with code for its use in WinCC Unified.

Figure 2-1



2.1.2 Principle of Operation

The complete Custom Web Control has the following functions:

- Display a WinCC Unified tag value on a gauge control
- Define upper and lower limits
- Define various stylistic parameters for better appearance
- Define value zones with different colors
- Method for blinking the zone where the value currently is
- Event that occurs at the transition from one zone to another

Required knowledge

The application example requires the following basic knowledge:

- Configuring with WinCC Unified
Fundamentals are taught in the SITRAIN course "WinCC Unified & Unified Comfort Panels". See article ID [109773211](#).
- Microsoft Visual Studio Code
- Web page programming with HTML5 and JavaScript

2.1.3 Components Used

The following hardware and software components were used to create this application example:

Table 2-1

| Components | Quantity | Article number | Note |
|-----------------------------------|----------|-----------------------------------|------------------|
| WinCC Unified V16 | 1 | 6AV2102-0AA06-0AA7 | Or later version |
| Microsoft Visual Studio Code 1.44 | 1 | Refer to Internet | Or later version |
| Gauge.js 1.3.7 | 1 | See Internet \5\ | |

Note

Product training on WinCC Unified and Unified Comfort Panels is available via these courses:

- SITRAIN System Course "WinCC Unified & Unified Comfort Panels"
<https://support.industry.siemens.com/cs/ww/en/view/109773211>
- SITRAIN advanced course: SIMATIC WinCC Unified for PC systems
<https://support.industry.siemens.com/cs/ww/en/view/109781323>
- SITRAIN access: WinCC Unified Scripting (JavaScript)
<https://support.industry.siemens.com/cs/ww/en/view/109782872>

2.2 Engineering

2.2.1 Configuration and Implementation of the Custom Web Control

Using an example, this section will show how a Custom Web Control is created with Visual Studio Code.

Note

You may use other text-based development environments.

If you have little experience with the topic and do not know the software tool "Visual Studio Code", for example, you can find a short introduction in Section [6](#).

Creating a Custom Web Control

1. Creating folder structure

(See also the manual referenced in: "General Configuration and Folder Structure".)

- a. Open Windows Explorer and create the following elements in a working folder of your choosing:
- b. File named "manifest.json"
- c. Folder named "assets":
Put an icon here in any image format. It will be shown in the TIA Portal for this Control.

2. Folder named "control":

Here, create a file, "index.html", "styles.css" and a folder "js" in which you place the file "webcc.min.js" from the attached files in this application example

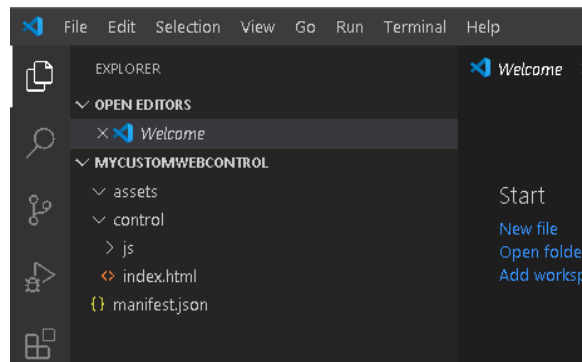
Place the file "gauge.min.js" with the downloaded code here.

(Source: <https://bernii.github.io/gauge.js/> \5\)

Open Visual Studio Code

- a. Open Visual Studio Code (or a different editor).
- b. Use "File" → "Open Folder" to open the folder where the folder structure you just created is located.

Figure 2-2



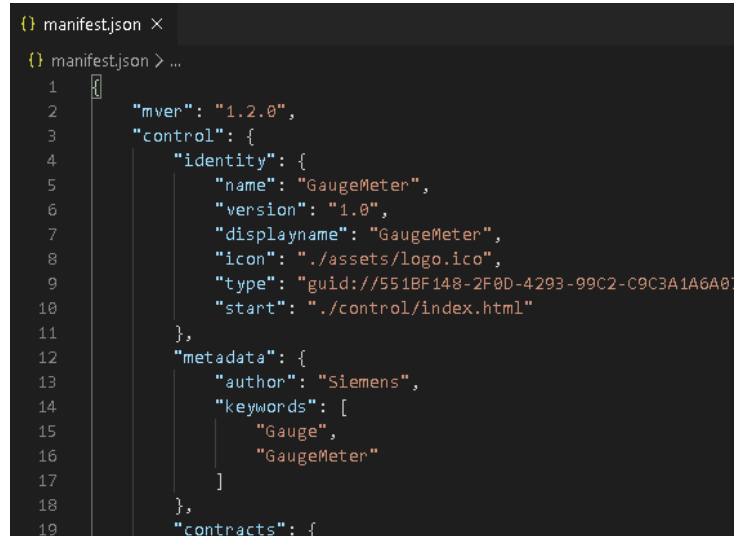
3. Create the Manifest.json file

(See also the manual referenced in: "Contract-Based Interaction and the Manifest File")

- a. Open the file Manifest.json in Visual Studio Code (or a different editor)
This file necessitates a certain structure, which is explained in more detail in the manual. For this application example, it will be sufficient to copy an existing file and make a couple modifications.

- b. For a new Custom Web Control to be created, copy the contents of the Gauge example's manifest.json file into your manifest.json.

Figure 2-3



```
{
  "mver": "1.2.0",
  "control": {
    "identity": {
      "name": "GaugeMeter",
      "version": "1.0",
      "displayname": "GaugeMeter",
      "icon": "./assets/logo.ico",
      "type": "guid://551BF148-2F0D-4293-99C2-C9C3A1A6A07",
      "start": "./control/index.html"
    },
    "metadata": {
      "author": "Siemens",
      "keywords": [
        "Gauge",
        "GaugeMeter"
      ]
    },
    "contracts": {
```

- c. Give a custom name for your Custom Web Control under the "name" attribute in lines 5 and 7.
- d. Adjust the name of your logo in line 8 (all common image formats will work, such as JPG, PNG, BMP, etc.).
- e. In line 9, assign a new, custom GUID. You can create one with an online generator, such as: <https://www.guidgenerator.com/>.
- f. Where necessary, modify the metadata starting on line 12.
- g. Where necessary, change the interface of your Custom Web Control starting on line 19. Some possible data types include: "boolean", "number", "string", "array", or one of your own defined data types, starting on line 79.
(For usage, see the included Manifest.json file.)

Note

In order to be sure that you created a valid JSON file, refer Visual Studio Code to it or copy the file's contents to an online validation tool (such as <https://jsonlint.com>: Insert content and click "Validate JSON" in the lower left.)

4. Index.html file

(See also the manual referenced in "Interaction Between Control and Container via the API".)

- a. Open the file Index.html in Visual Studio Code (or a different editor). This file is the portal to your web page. Here you will also establish a connection to the WinCC Unified Runtime server in order to exchange data.
- b. The connection data for WinCC Unified are in the attached file "webcc.min.js". Place it in the "js" folder and reference the file in index.html as follows:

```
<script src='./js/webcc.min.js'></script>
```

This reference is best made in your <Head> tag (also see line 11 of the attached example).

- c. The connection is established with the function `WebCC.start()` (in line 226). For this, it is important that the connection is established right when the web page is visited. You can best achieve this if the function (like in the example) is directly in the `<Script>` tag and not in a more deeply nested function, which may only be retrieved at a later point in time.

5. Data exchange between Custom Web Control and WinCC

(See also the manual referenced in "Using the Control via WinCC")

- a. From anywhere in your application, you can now access the data that are defined in the Manifest.json file.
- b. Access is facilitated with the API object `"WebCC"`. You already used it to establish the connection. You now have additional options.
- c. If you wish to read or write properties (in the Manifest.json file under "properties" starting on line 41), you can for example access the property "GaugeValue" with write access in the following manner:
`WebCC.Properties.GaugeValue = 5` to set the value to 5. The value of the linked WinCC tag will also be set to 5.
- d. If you are interested in changing the connected WinCC tags (possibly a PLC tag), use the function `"WebCC.onPropertyChanged.subscribe()"`. You should call this function directly after the connection is successfully established in order to receive all changes from the beginning onward (see line 230 and 244). For handoff parameter, use a function that you defined (callback function). In the attached example, this is the function "setProperty" in line 143.

For any data change at all, the function "setProperty" is called and passes a "data" object which contains a "key" and a "value". The "key" is the name of the property which has changed, while "value" is the new value. Therefore, it is recommended to program switch-case branch point in order to be able to process the new value appropriately.

Figure 2-4

```
139 // This is a callback function that is called every ti
140 // other functions so you can see the new value in the
141 // - data: object containing a key and a value propert
142 //       the "value" contains the new value.
143 function setProperty(data) {
144     // console.log('onPropertyChanged ' + data.key);
145     switch (data.key) {
146         case 'GaugeValue':
147             updateValue(data.value);
148             break;
149         case 'GaugeBackColor':
150             document.body.style.backgroundColor = toCo
```

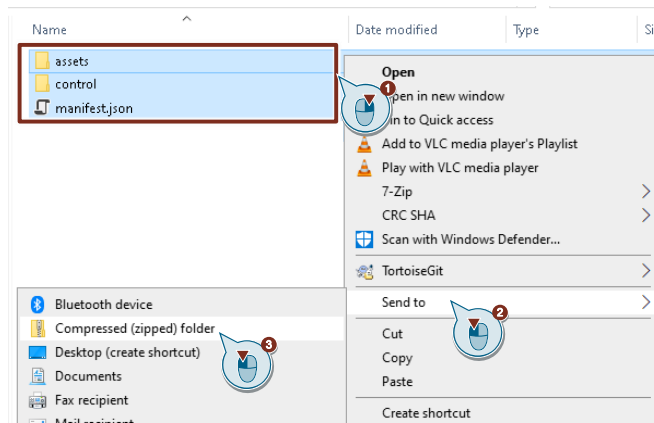
- e. If you have declared methods in the Manifest.json file (line 21 in the "manifest.json" file), WinCC can call these methods and you can react to them in the Custom Web Control. WinCC can call these methods at any time, meaning that you must define what should happen in each case before the connection is established. You will define a function to the exact name that you named in the Manifest.json file, and the same parameters. You can find an example in the attached index.html file starting on line 254.

- f. If you have specified an event in the Manifest.json file (see line 31), you can trigger this event at any point in your code with `"WebCC.Events.fire()"` so that WinCC will be notified. The first handoff parameter in such case is always the name of the event that you wish to trigger, followed by all handoff parameters in the proper order as you specified in the Manifest.json file. You can find an example in the attached index.html file in line 87.

6. Deploy process for using the Custom Web Control in the TIA Portal
(See also the manual referenced in "Creating the ZIP File")

- a. When you are finished programming, you still have to package the code in such a way that TIA Portal correctly recognizes your Custom Web Control.
- b. To do this, open the Windows Explorer and go to your project folder. Select the folders that you created originally, "assets", "control" and the file "manifest.json" and archive them with right-click → "Send to" → "Compressed (zipped) folder".

Figure 2-5

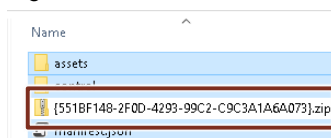


- c. Use your GUID file from the Manifest.json file in order to modify the name of your generated .zip file as follows (the Xs stand for your GUID):

`{XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX}.zip`

Please note the curly braces before and after the GUID.

Figure 2-6



- d. Your Custom Web Control is now finished and you can use it in the TIA Portal.

2.2.2 Installation and Integration into the User Project

Installation in the TIA Portal

Before you can use the Custom Web Control in the TIA Portal, you have two ways to install it (also refer to the official manual under "Installing a Custom Web Control"):

1. Only make available for a specific project
Place your Custom Web Control in your project folder:
"...Project_1\UserFiles\CustomControls\ {xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}.zip"
2. Make available for all projects
Place your Custom Web Control in the installation path of the TIA Portal:
"C:\Program Files\Siemens\Automation\Portal Vxx\Data\Hmi\CustomControls\ {xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}.zip"

Note If you copy the project to another PC, the Custom Web Controls are not transferred and a compilation error occurs in the TIA Portal project. You must also store the Custom Web Controls on the new computer in the directory given above.

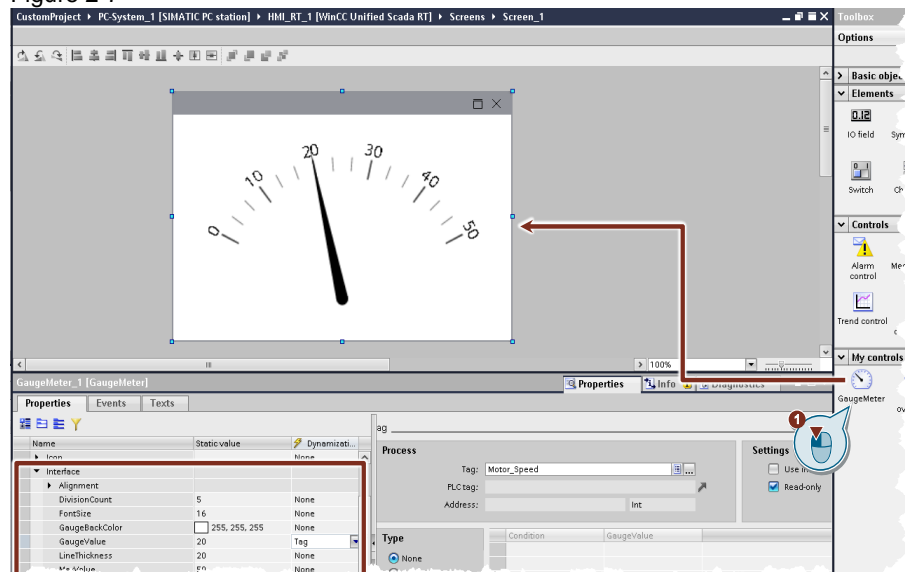
Note When loading a project to the Operator Panel, the TIA Portal also transfers your Custom Web Control. There is no installation to Runtime server.

Integration into the user project

To configure your Custom Web Control in the TIA Portal project, proceed as follows:

1. Open an screen for your Unified Comfort Panel or PC station.
2. Click on your Custom Web Control under "Tools > My controls" and place it in the screen using drag & drop.

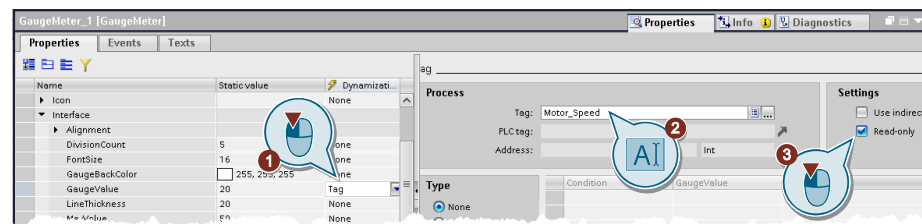
Figure 2-7



In the properties of the Custom Web Control under "Interface", you will find all the properties that you have defined in the Manifest.json file. You can assign a static value to these, as with all other properties of screen objects, or define a dynamization.

3. Dynamize Custom Web Control
 - a. Create a dynamization "Tag" for "GaugeValue".
 - b. Select a suitable PLC or HMI tag.
 - c. If your Custom Web Control has read-only access to the tag, leave the check mark. However, if your implementation entails that the Custom Web Control will modify your tags, then uncheck the checkbox. In this case, the "GaugeMeter" only displays the tag (read).

Figure 2-8



2.2.3 Debugging

In this section, you will learn to debug Custom Web Controls in Runtime.

Debugging is not possible in the TIA Portal. If you cannot transfer the Custom Web Control to Runtime because you encounter problems beforehand, see [Section 4 Possible Solutions](#).

Debugging in Google Chrome

WinCC Unified Runtime sends the Custom Web Control directly to the Web Client when in use. This way, you can also debug it on the Web Client using the standard developer console of the browser.

Open the Unified Runtime screen containing the Custom Web Control in Google Chrome and press F12 to open the developer console.

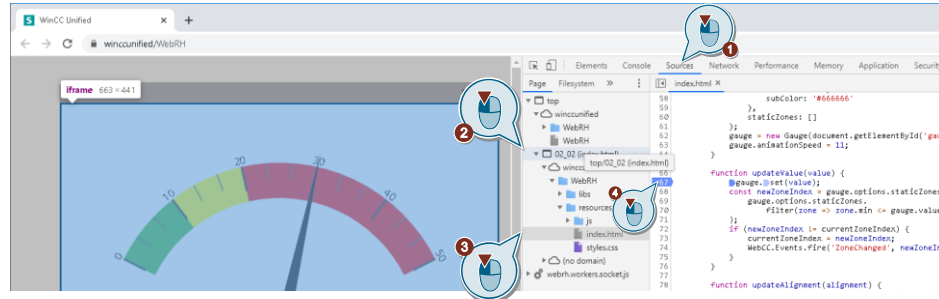
Step-by-Step Guide

In [Figure 2-9](#), you can see how debugging works after opening the developer console using F12.

1. Select the "Sources" tab in the developer console.
2. Slowly move the mouse cursor to the left side of the developer console over the folders under the "top" node. Your Custom Web Control has no explicit name here, but the browser will highlight it in blue once you have selected it. (Here you will find a separate folder for each Custom Web Control instance.)
3. Open the respective folder and navigate to the file that you want to debug. (Here, all the code will be in the index.html file.)

4. In the right-hand window, scroll to the corresponding position and click the line number to insert a breakpoint. Once the code comes upon this position again, it will stop and you will receive detailed information about the program sequence. In the image, there is a breakpoint at the beginning of the function "updateValue". The script will now always stop here if you update the linked value. Remove the breakpoint by clicking the line number again.

Figure 2-9



Note For more information on working with the developer console and which information you can glean from it, consider familiarizing yourself with the official documentation from Google: <https://developers.google.com/web/tools/chrome-devtools/javascript#sources-ui>

Note You can also debug with any other browser; to do this, simply refer to the documentation of the respective browser.

3 Custom Web Control "Table"

3.1 Introduction

3.1.1 Overview

The following is another application example for creating Custom Web Controls. The Custom Web Control "Table" is basically a tabular display of the values. In addition to displaying the values, you can also format them graphically (see the image below).

Figure 3-1

| Name | Salary | Intern | OnVacation | Rating |
|------------------|--|------------------|------------|--------|
| filter column... | | filter column... | | ★★★★★ |
| Worker hall 1 | <div style="width: 20%; background-color: green;"></div> | yes | ✗ | ★★★★★ |
| Worker hall 2 | <div style="width: 15%; background-color: green;"></div> | no | ✗ | ★★★★☆ |
| Line coordinator | <div style="width: 40%; background-color: green;"></div> | yes | ✓ | ★★★★☆ |
| Forklift driver | <div style="width: 25%; background-color: green;"></div> | no | ✓ | ★★★☆☆ |
| Operator | <div style="width: 20%; background-color: green;"></div> | no | ✗ | ★★★☆☆ |
| Test user | <div style="width: 10%; background-color: green;"></div> | no | ✓ | ★★★★★ |
| Administrator | <div style="width: 30%; background-color: green;"></div> | yes | ✗ | ★★★★☆ |

For the example, the JavaScript library "Tabulator" (see <http://tabulator.info>) was used and supplemented with code so that it can be used in WinCC Unified.

3.1.2 Range of Functions

With the Custom Web Control "Table", you have the following options:

- Creating a table based on transferred WinCC Unified tag values (e.g., process values, parameter set values)
- Saving all table values in a string tag
- Individual adjustment of columns (width, display formats of values, filtering of values)

3.1.3 Components Used

This application example was created using the following library in addition to the hardware and software components already shown:

Table 3-1

| Components | Quantity | Article number | Note |
|-----------------------------------|----------|-----------------------------------|------------------|
| WinCC Unified V16 | 1 | 6AV2102-0AA06-0AA7 | Or later version |
| Microsoft Visual Studio Code 1.44 | 1 | Refer to Internet | Or later version |
| Tabulator v4.9 | 1 | See Internet \5\ | |

Note

Product training on WinCC Unified and Unified Comfort Panels is provided in these two courses

- SITRAIN System Course "WinCC Unified & Unified Comfort Panels"
<https://support.industry.siemens.com/cs/ww/en/view/109773211>
- SITRAIN advanced course: SIMATIC WinCC Unified for PC systems
<https://support.industry.siemens.com/cs/ww/en/view/109781323>
- SITRAIN access: WinCC Unified Scripting (JavaScript)
<https://support.industry.siemens.com/cs/ww/en/view/109782872>

3.1.4 Use Cases

The Custom Web Control tables are suitable for the following purposes, among others:

- Output values of a CSV file as table in WinCC Unified Runtime.
- Graphical representation of process values within a table.
- Export process values as CSV.

3.2 Engineering

3.2.1 Configuration and Implementation of the Custom Web Control

This section shows you how to create and implement the Custom Web Control "Table" using Visual Studio Code.

Creating a Custom Web Control

1. Create a folder structure (See also the manual referenced in: "General Configuration and Folder Structure".)

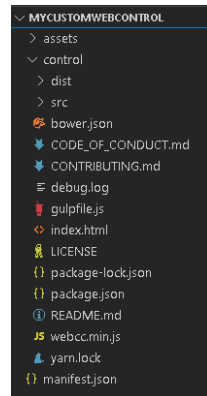
Open Windows Explorer and create the following elements in a working folder of your choosing:

- File named "manifest.json"
(This file necessitates a certain structure, which is explained in more detail in the manual. For this example, it is sufficient to copy an existing file and make adjustments as necessary.)
- Folder named "assets":
Put an icon here in any image format. It will be shown in the TIA Portal for this Control.
- Folder named "control":
Create a file "index.html" and add the file "webcc.min.js" from the attached files to this directory. Unzip the downloaded "Tabulator" library (source: <https://github.com/olifolkerd/tabulator>) into the "control" folder as well.

2. Visual Studio Code

- a. Open Visual Studio Code or another editor of your choice.
- b. Use "File" → "Open Folder" to open the folder where the folder structure you just created is located.

Figure 3-2



3. Create the Manifest.json file

- a. Open the manifest.json file.
- b. Set up your manifest as already described in Section 2.2.1
- c. Leave the property "TableDataString" unchanged. The contents of the table are written back here.

4. Configure Index.html

- a. Reference the necessary scripts.

Figure 3-3

```
<script src="webcc.min.js"></script> <!-- mandatory dependency -->
<script type="text/javascript" src="./dist/js/tabulator.min.js"></script>
<link href="./dist/css/tabulator.css" rel="stylesheet">
```

- b. Start the connection setup with the function "WebCC.start();".
Immediately after the connection, the current value hidden behind "TableDataString" is written to the array "ArrayData".
- c. Finally, you will see the "subscribe" function. It checks if the value of "TableDataString" will change, which happens automatically when the page is loaded. When the page is reloaded, the value is set to the "Static" value from the TIA Portal project. As soon as this happens, the value is overwritten again with the current values from "ArrayData".

5. Work with "Tabulator"

- a. The field "ArrayData" now contains the data points for the table. You can use these to create tables using the "Tabulator" functionality. (For more information, see <http://tabulator.info/>)
- b. The interface tag "ColumnStyleString" defines the properties of the table columns.

6. Deploy process and installation

- a. After programming is complete, create a *.zip file as shown in the previous "Gauge" example.

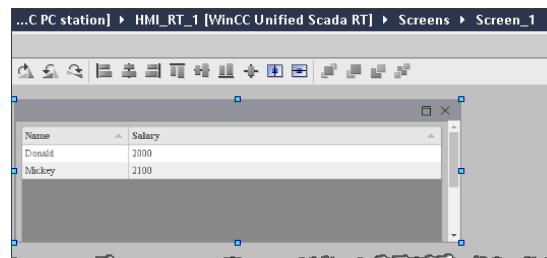
The .zip file should contain the folders "assets", "control", and the file "manifest.json". Give the *.zip file its own GUID name ({xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}.zip).

- b. Now make this file available for your user project by copying it to the "CustomControls" folder of your project.

...Project_1\UserFiles\
CustomControls\{xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}.zip

Open your TIA Portal project and drag your Custom Web Control to your desired screen.

Figure 3-4



- c. Then pass the appropriate data to the "ColumnStyleString" and "TableDataString" properties of the Custom Web Control by script.
For additional information, refer to Section [3.2.4](#), as well as Section [4.2](#).

3.2.2 Installation and Integration into the User Project

How to install the Custom Web Control in the TIA Portal and integrate it into your user program is described in Section [2.2.2](#).

3.2.3 Debugging

Which possibilities you have for error diagnosis and debugging can be found in Section [2.2.3](#).

3.2.4 Necessary Data

To create a table, two strings must be passed to the following two properties of the "Table" Custom Web Control:

- "ColumnStyleString": String for formatting the columns
- "TableDataString": String with data records (table contents)

ColumnStyleString

The "ColumnStyleString" (string in JSON format) defines the layout of the table columns.

An exemplary structure of the "ColumnStyleString" for the column "Name" is shown in the following figure.

Figure 3-5

```
[{"title": "Name", "field": "name", "sorter": "string", "width": 150, "headerFilter": "input"},
```

Table 3-2

| No. | Parameter | Description |
|-----|--------------|---|
| 1. | title | Specifies the name of the column in the Custom Web Control. |
| 2. | field | Serves to assign the values (from the "TableDataString") when filling the table. Note: Ensure that the values are written correctly; otherwise, no assignment to the "title" column can be made. |
| 3. | sorter | Specifies what attribute the list is sorted by. |
| 4. | width | Specifies the width of the column. |
| 5. | headerFilter | Optional: Creates a filter at the top of the column. |

In addition to the simple display of values, you can also format column contents graphically. To do this, specify this via additional parameters, e.g., "hozAlign", "formatter", and "formatterParams".

Figure 3-6

| Name | Salary | onvacation | Rating |
|----------|--------|------------|--------|
| Donald | █ | ✓ | ★★★★☆ |
| Mickey | █ | ✗ | ★★☆☆☆ |
| Goofey | █ | ✓ | ★★★★☆ |
| Pluto | █ | ✓ | ★★☆☆☆ |
| Dagobert | █ | ✗ | ★★★★★ |

Note A detailed description of the formatting options can be found under the following link:

<http://tabulator.info/docs/4.9/format>

TableDataString

The "TableDataString" (string in JSON format) contains the values with which the individual columns are to be filled. The structure consists of the "field" name of the "ColumnDataString" and the value to be entered in the respective row of the table.

Note In Section [4.1](#), the structure of the "ColumnStyleString" and "TableDataString" is shown again in the application, based on the sample project.

You can find more information about the properties at <http://tabulator.info/>.

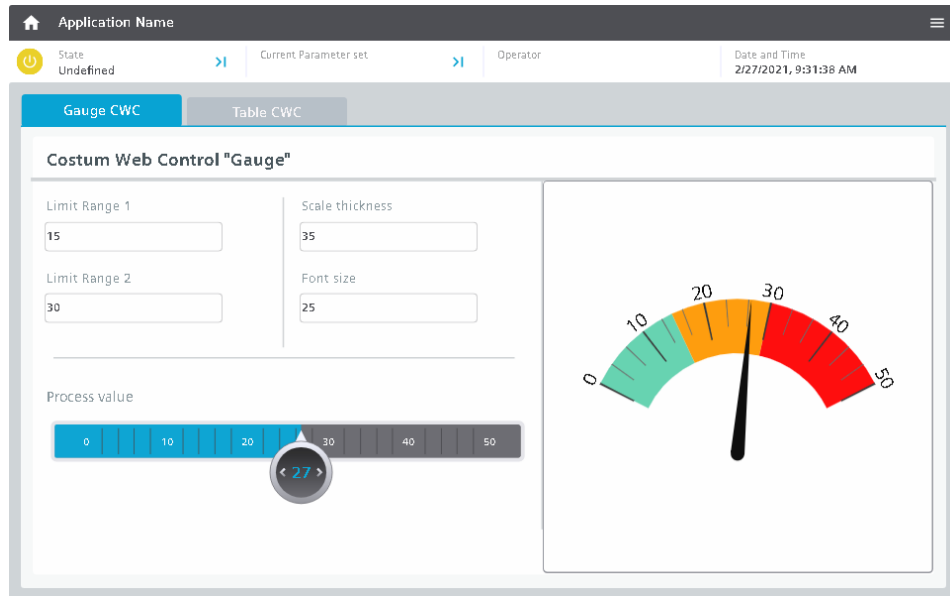
Note If a parameter is empty or the Control is not connected correctly, the Custom Web Control does not display the table correctly (see [Figure 5-3](#)).

4 Sample Project Operation

4.1 Custom Web Control "Gauge" Operation

The sample project consists of two tabs. In the first tab, you will find the CWC "Gauge", where four properties (via I/O fields), as well as the process value (via sliders), can be dynamized as an example.

Figure 4-1



4.2 Custom Web Control "Table" Operation

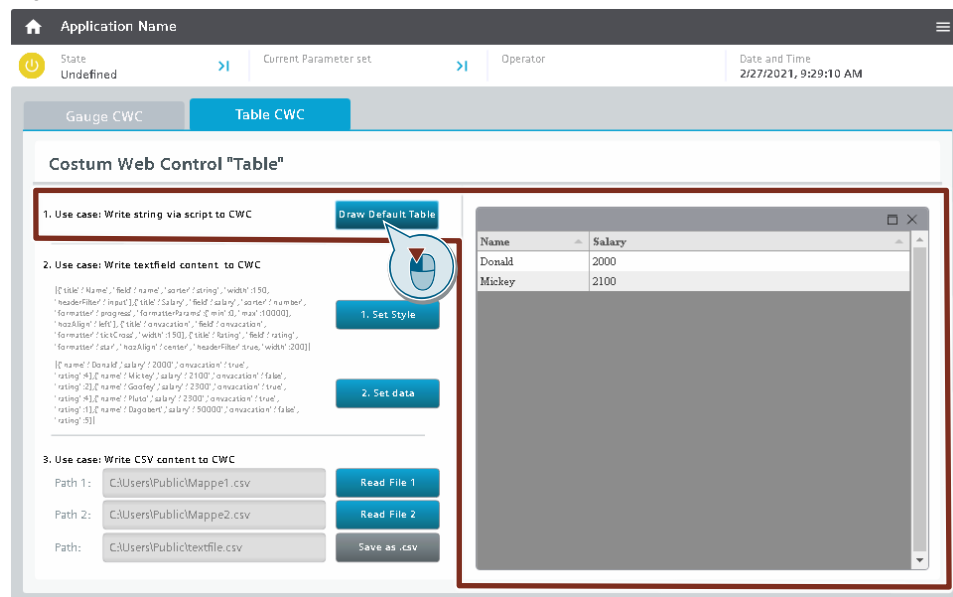
In the second tab you will find the Custom Web Control "Table". The screen shows you three ways in which you can supply the Custom Web Control with the required values (see Section 3.2.4).

1. Script: Table contents are stored in a script
2. Screen object: Table contents are stored in a screen object (e.g., text field)
3. CSV file: Table contents are contained in a CSV file or are to be exported

4.2.1 Transferring Table Contents via a Script

In the first example, the values are simply displayed in two columns. The two strings ("ColumnTableString" and "TableDataString") are passed to the CWC via the "Draw Default Table" button.

Figure 4-2



ColumnStyleString

The "ColumnStyleString" for this example is composed as follows:

```
' [{"title": "Name", "field": "name", "sorter": "string", "width": 150},
  {"title": "Salary", "field": "salary", "sorter": "number", "hozAlign": "left"} ]'
```

TableDataString

The "TableStyleString" for this example is composed as follows:

```
' [{"name": "Donald", "salary": "2000"},
  {"name": "Mickey", "salary": "2100"} ]'
```

Note

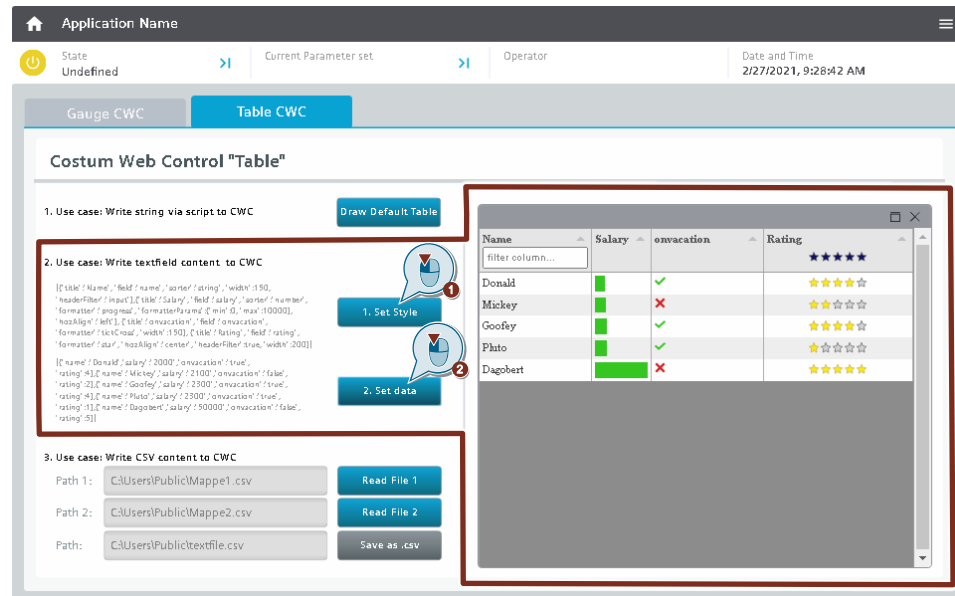
For better readability, line breaks have been added to the two strings in the documentation.

4.2.2 Transferring Table Contents via a Screen Object

In the second example, the table contents are displayed in a graphically formatted manner. In addition, a filter is added in the "Name" and "Rating" columns to filter the entries. The two strings ("ColumnTableString" and "TableDataString") are also stored in a screen object (in this example, a text field).

Via the button "1. Set Style", the format ("ColumnStyleString") is first passed to the CWC. Subsequently, the table content ("TableDataString") is transferred via the "2. set data" button.

Figure 4-3



ColumnStyleString

The "ColumnStyleString" for this example is composed as follows:

```
[{"title":"Name", "field":"name", "sorter":"string", "width":150,
"headerFilter":"input"},
{"title":"Salary", "field":"salary", "sorter":"number",
"formatter":"progress", "formatterParams":{"min":0, "max":10000},
"hozAlign":"left"},
{"title":"onvacation", "field":"onvacation", "formatter":"tickCross",
"width":150},
{"title":"Rating", "field":"rating", "formatter":"star",
"hozAlign":"center", "headerFilter":true, "width":200}]
```

TableDataString

The "TableStyleString" for this example is composed as follows:

```
[{"name":"Donald", "salary":"2000", "onvacation":"true", "rating":4},
{"name":"Mickey", "salary":"2100", "onvacation":"false", "rating":2},
{"name":"Goofey", "salary":"2300", "onvacation":"true", "rating":4},
{"name":"Pluto", "salary":"2300", "onvacation":"true", "rating":1},
{"name":"Dagobert", "salary":"50000", "onvacation":"false", "rating":5}]
```

Note

For better readability, line breaks have been added to the two strings in the documentation.

4.2.3 Transferring Table Contents via a CSV File

In the third use case, the values are displayed both unformatted ("Read file 1") and formatted ("Read file 2").

Note

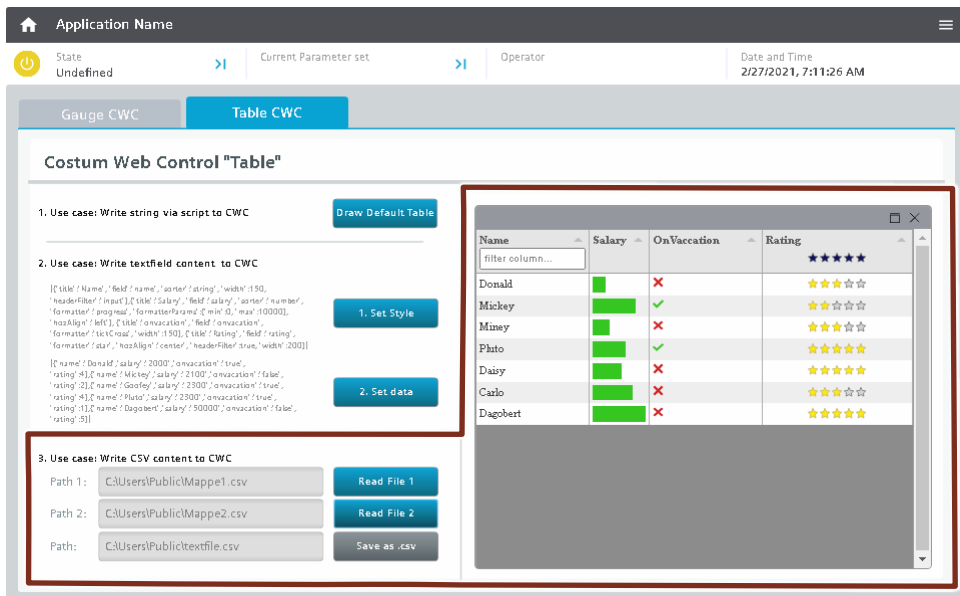
In the project directory of the sample project ("109779176_CustomWebControl_V10_PROJ\UserFiles"), there are already two preconfigured CSV files. If you want to use this with the sample project, you must store both files (folder1 and folder 2) to the following location of your Operator Panel:

- Unified PC Runtime: "C:\Users\Public"
- Unified Comfort Panel: "\home\industrial\"

The string "ColumnTableString" is stored in the script of the respective button and defines the unformatted or formatted display form.

The table data "TableDataString" is read from a CSV file and is converted by a script into the JSON format before it is passed to the CWC.

Figure 4-4



© Siemens AG 2021 All rights reserved

ColumnStyleString

The "ColumnStyleString" for the unformatted example is composed as follows:

```
'[{"title": "Name", "field": "name", "sorter": "string", "width": 150},
{"title": "Salary", "field": "salary", "sorter": "number", "hozAlign": "left"},
{"title": "Intern", "field": "intern", "sorter": "boolean",
"hozAlign": "left"}]
```

The "ColumnStyleString" for the formatted example is composed as follows:

```
'[{"title":"Name", "field":"name", "sorter":"string", "width":150, "headerFilter":"input"}, {"title":"Salary", "field":"salary", "sorter":"number", "formatter":"progress", "formatterParams":{"min":0, "max":10000}, "hozAlign":"left"}, {"title":"OnVaccation", "field":"onvaccation", "formatter":"tickCross", "width":150}, {"title":"Rating", "field":"rating", "formatter":"star", "hozAlign":"center", "headerFilter":true, "width":200}]'
```

TableDataString

The "TableStyleString" for this example is as follows:

```
HMIRuntime.FileSystem.ReadFile("C:\\Users\\Public\\Mappel.csv", "utf8").then( function(text) { //read file and convert to a string in JSON format var lines = text.split("\n"); var result = []; var headers; headers = lines[0].split(";"); for (var i = 1; i < lines.length; i++) { var obj = {}; if(lines[i] == undefined || lines[i].trim() == "") { continue; } var words = lines[i].split(";"); for(var j = 0; j < words.length; j++) { obj[headers[j].trim()] = words[j]; } result.push(obj); } //write to Interface Screen.Items('MyCWC').Properties.TableDataString = JSON.stringify(result);
```


5 Frequent Error Constellations

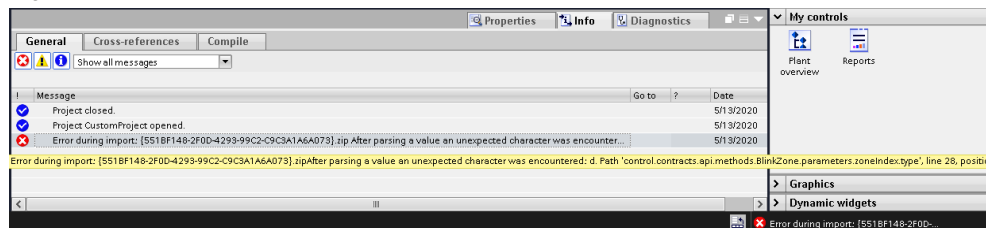
This section describes common error patterns and their solutions that can occur during the creation and integration of a Custom Web Control.

5.1 Custom Web Control does not appear in the TIA Portal Toolbox

Error description

If you open a screen in the TIA Portal and your Custom Web Control does not appear on the right in the Toolbox, then your Manifest.json file has an error. In addition, an error message appears in the info box, as shown in [Figure 5-1](#).

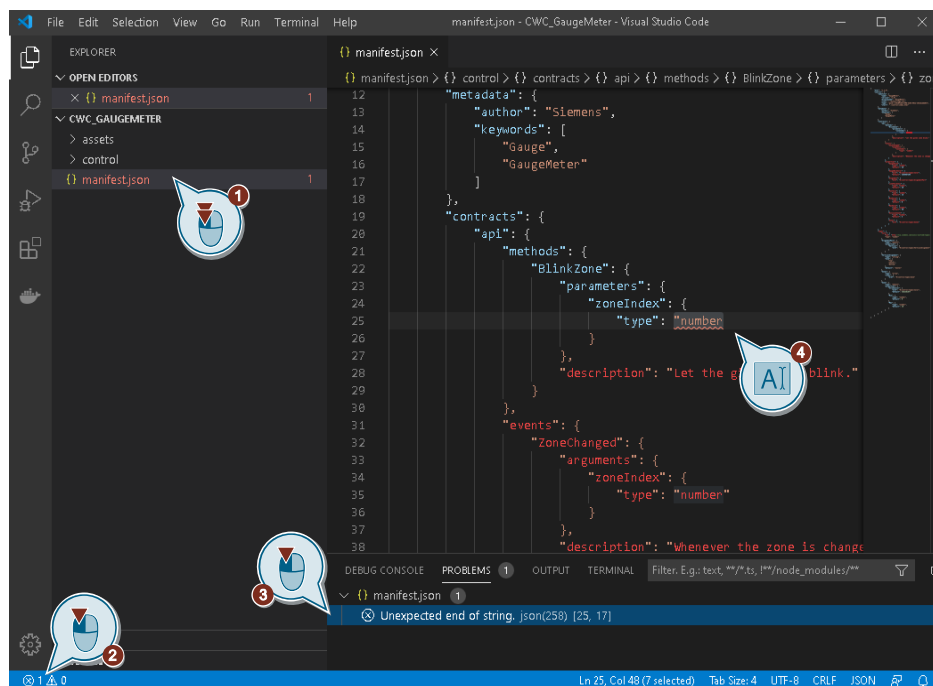
Figure 5-1



Solution

To resolve the issue, open your Manifest.json file with Visual Studio Code and troubleshoot as shown in [Figure 5-2](#).

Figure 5-2



1. Double-click to open the Manifest.json file. Visual Studio Code will then automatically validate it and show the file name in red, as well as the number of errors to the right.
2. You will again see the number of errors to the bottom left. Click on it to open a window on the right with detailed descriptions of all errors.

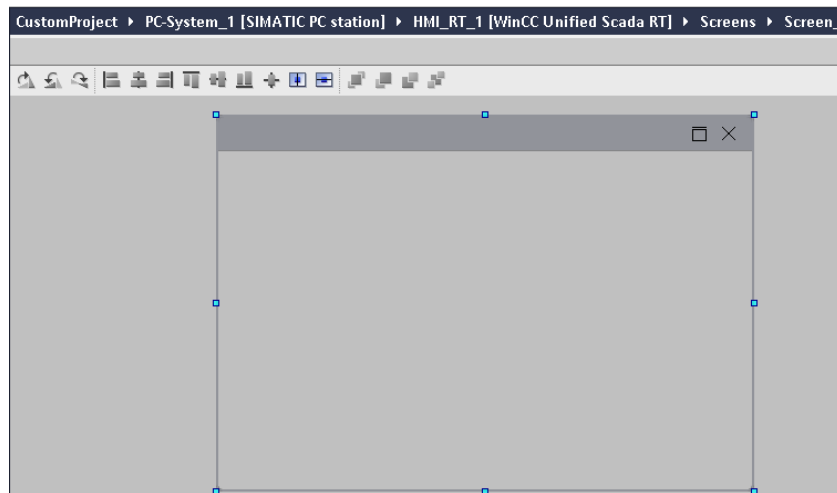
3. Click on an error. Visual Studio Code will jump directly to the point in the document where it occurs.
4. Correct the error with the help of the error description.

5.2 TIA Portal Does not Display the Custom Web Control Correctly in the Screen

Error description

You can successfully insert your Custom Web Control into the screen and interconnect the properties. This also works perfectly in Runtime, but the TIA Portal screen editor displays an empty or faulty Control, such as, for example, in Figure 5-3.

Figure 5-3



Explanation: TIA Portal will try to try to display the web contents but for security reasons it will deactivate some functions, with the result that your Custom Web Control is displayed incorrectly or not at all.

Solution

As a solution, you can program a preview in your Custom Web Control that only displays the TIA Portal screen editor. In Runtime, however, it will function as expected.

In order to achieve this, you must define in JavaScript after a successful connection how the code is to proceed. Using an IF query of the property "isDesignMode" at the API object "WebCC", you can find out whether your Custom Web Control is currently in the TIA Portal or in Runtime.

Your code could resemble [Figure 5-4](#), where you have defined a new function "showDemoData()", in which you program your code to show a preview for the TIA Portal.

Figure 5-4

```

18 ////////////////////////////////////////////////////
19 // Initialize the custom control
20 WebCC.start(
21 // callback
22 function (result) {
23     if (result) {
24         console.log('connected successfully');
25         if (WebCC.isDesignMode) {
26             // do not subscribe, only show some dummy data
27             showDemoData();
28         } else {
29             // Subscribe for value changes
30             WebCC.onPropertyChanged.subscribe(setProperty);

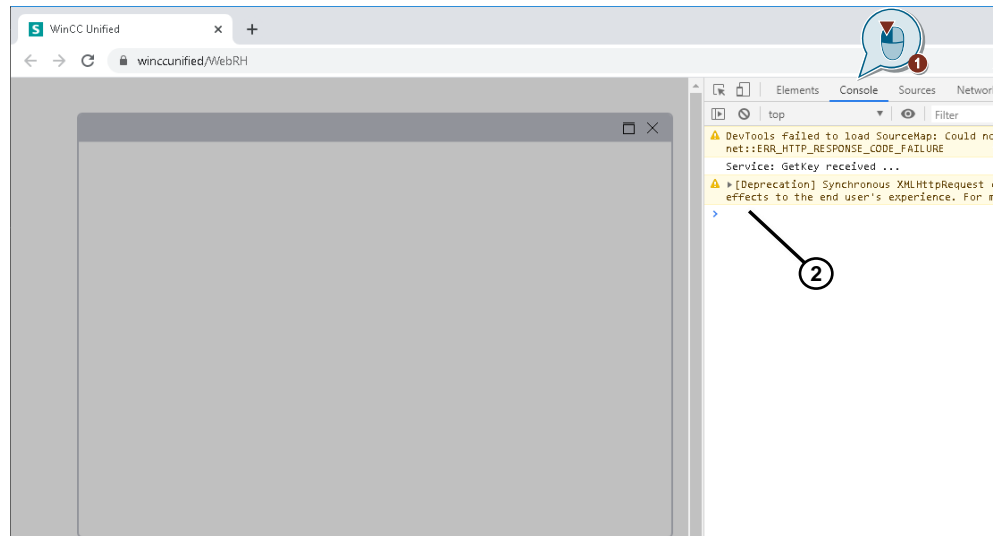
```

5.3 Custom Web Control Remains Empty in Runtime

Error description

You were able to successfully load your Custom Web Control into Runtime from TIA Portal. When you open the screen, you see the border but the contents are missing.

Figure 5-5



Solution

First check whether a connection was established between your Custom Web Control and WinCC Unified. To do this, open the developer console with F12 and then, as in [Figure 5-5](#) click "Console". If you do not see the "connected successfully" success message as shown in the figure, then no connection was established.

In this case, check that the function "`WebCC.Start()`" is called directly when the Custom Web Control is started.

If a connection is established but still no screen appears, the error is in the code that you wrote. Debug your code to fix the error (see [2.2.3 Debugging](#)).

5.4 Custom Web Control Contains no WinCC Data

Error description

The browser is displaying your Custom Web Control correctly. But when you modify the linked tags, you don't see the values change in the Custom Web Control.

Solution

First check that the tags are written correctly everywhere they are used in your code and the Manifest.json file. Pay special attention to equal upper and lower case.

Next, verify that you have linked the correct tag that you are modifying in the TIA Portal. If necessary, output the tag next to the Custom Web Control in an I/O field.

If the I/O field is updated and all tags are correctly connected in the TIA Portal, but the value is still not visible in the Custom Web Control, then the error is in the code.

Check whether you have called the function `WebCC.onPropertyChanged.subscribe()` after a successful connection, and that you have passed a callback function (see [2.2.1 Configuration and Implementation of the Custom Web Control](#), step 5).

Use the debugging function (see [2.2.3 Debugging](#)) to set a breakpoint at the beginning of the callback function (in line 145 in this example).

Now change the linked value and pay attention to whether the breakpoint is reached.

If the browser does not stop at the breakpoint, check if you have correctly passed the property with the correct name when calling `WebCC.Start()` (in the example, see line 263).

Another valid example of a correct call of the function can be found here:

```
WebCC.Start(function(result) {
  if (result) {
    console.log('connected successfully');
    WebCC.onPropertyChanged.subscribe((data) => {
      console.log(data);
    });
  }
}, {
  properties: {MyIntProperty: 0, MyStringProperty: 'test'}
}, [], 10000 // timeout
);
```

The Custom Web Control has requested the properties "MyIntProperty" and "MyStringProperty" from WinCC Unified. They must also be present in the Manifest.json file. Pay attention to the spelling and capitalization.

5.5 Custom Web Control Cannot Write WinCC Data

Error description

The browser is displaying your Custom Web Control correctly.

Your Custom Web Control is programmed to write to properties. The properties are linked to WinCC tags in the TIA Portal. The Custom Web Control should change WinCC tags directly.

When your Custom Web Control writes the property, you will not see any change in the WinCC tags.

Solution

First check that the tags are written correctly everywhere they are used in your code and the Manifest.json file. Pay special attention to equal upper and lower case.

Next, verify that you have linked the correct tag that you are modifying in the TIA Portal. If necessary, output the tag next to the Custom Web Control in an I/O field.

5 Frequent Error Constellations

If the I/O field does not update, the error is in the code. Check whether you are reaching the code line where you write the value.

Using the debugging function (see [2.2.3 Debugging](#)), set a breakpoint in the line where you write the property. Monitor whether the breakpoint is reached.

If the browser stops at the breakpoint and the tag value does not change in WinCC, check that you have used the correct write property again.

Another valid example of correct writing:

```
WebCC.Properties.MyIntProperty = 5;
```

The Custom Web Control writes the property "MyIntProperty". They must also be present in the Manifest.json file. Pay attention to the spelling and capitalization.

6 Useful Information

6.1 Tips & Tricks

Version management

Version management offers some advantages, one of which is that your code is saved again. Another is tracking of changes, in the event that errors can't be traced and you wish to go back to a stable version.

Platforms for version management include <http://github.com/>.

Updating the Custom Web Control while creating it

Section [2.2.2](#) explains how to load the Custom Web Control into Runtime. If you need to experiment a lot during the creation process, you will have to go through the steps frequently. The following alternative procedure can save you time.

There is an alternative method when programming a Custom Web Control:

1. Create the Manifest.json file as completely as you can.
2. Create an empty "index.html" file.
3. Create a Custom Web Control from these two files and integrate it (see [2.2.2 Installation and Integration into the User Project](#)).
4. Launch your browser and view your empty Custom Web Control.
5. Navigate to the code of the online instance of the Custom Web Control. In Windows, it is located here:
"C:\Users\Public\Documents\Siemens\WebUX_ResourceCache_CustomWeb Controls".
6. Modify the code.
7. Update the Unified Runtime web page in the browser with F5.
8. The Custom Web Control has now been updated without needing to download it again.
9. Repeat steps 6 and 7 until you are finished programming.

CAUTION

With this process, your code can be lost!

Once you are finished programming, save the code of the online instance of the Custom Web Control separately.

When Runtime restarts, WinCC Unified will overwrite your code with the original code.

If you have to modify the Manifest.json file after the fact, also save your code separately, as you would have to start over from step 3. Downloading the Custom Web Control again overwrites your online code in the same manner.

6.2 Alternative Solutions

Runtime ODK and OpenPipe

Custom Web Controls are used to exchange data between the operator and the data in WinCC Unified (such as PLC tags).

If you use the Custom Web Control to utilize data from other web services, ask yourself whether you only need the data in the display for a specific operator or if the data are relevant for all operators.

If the data are intended for a specific duration and WinCC Unified must show that operator the data with a Custom Web Control directly, then you made the right choice with the Custom Web Control.

On the other hand, if you query external data with the Custom Web Control and write back tags in WinCC with properties in order to provide the data to multiple users or all users, then you may have the problem of multiple users requesting data and writing tags in WinCC. This is an unnecessary load on your other web service as well as for the WinCC Unified Runtime server.

There is a shorter way that yields higher performance: Write an application that runs on the WinCC Unified Runtime server and fetches the data. This application can then write the data via the OpenPipe or Runtime ODK interface and write tags in WinCC.

If you still wish to have a specific display form that WinCC Unified does not currently offer, create a Custom Web Control that uses solely these WinCC tags.

In this way, you leave the issues of authentication, authorization and future redundancy to the WinCC Unified server and have less development work to do in the Custom Web Control.

7 Appendix

7.1 Service and support

Industry Online Support

Do you have any questions or need assistance?

Siemens Industry Online Support offers round the clock access to our entire service and support know-how and portfolio.

The Industry Online Support is the central address for information about our products, solutions and services.

Product information, manuals, downloads, FAQs, application examples and videos – all information is accessible with just a few mouse clicks:

support.industry.siemens.com

Technical Support

The Technical Support of Siemens Industry provides you fast and competent support regarding all technical queries with numerous tailor-made offers – ranging from basic support to individual support contracts.

Please send queries to Technical Support via Web form:

support.industry.siemens.com/cs/my/src

SITRAIN – Digital Industry Academy

We support you with our globally available training courses for industry with practical experience, innovative learning methods and a concept that's tailored to the customer's specific needs.

For more information on our offered trainings and courses, as well as their locations and dates, refer to our web page:

siemens.com/sitrain

Service offer

Our range of services includes the following:

- Plant data services
- Spare parts services
- Repair services
- On-site and maintenance services
- Retrofitting and modernization services
- Service programs and contracts

You can find detailed information on our range of services in the service catalog web page:

support.industry.siemens.com/cs/sc

Industry Online Support app

You will receive optimum support wherever you are with the "Siemens Industry Online Support" app. The app is available for iOS and Android:

support.industry.siemens.com/cs/ww/en/sc/2067

7.2 Industry Mall



The Siemens Industry Mall is the platform on which the entire Siemens Industry product portfolio is accessible. From the selection of products to the order and the delivery tracking, the Industry Mall enables the complete purchasing processing – directly and independently of time and location:

mall.industry.siemens.com

7.3 Links and literature

Table 7-1

| No. | Subject |
|-----|---|
| \1\ | Siemens Industry Online Support https://support.industry.siemens.com |
| \2\ | Link to the entry page of the application example https://support.industry.siemens.com/cs/ww/en/view/109779176 |
| \3\ | Microsoft Visual Studio Code 1.44 https://code.visualstudio.com/ |
| \4\ | Gauge.js 1.3.7 https://bernii.github.io/gauge.js/ |
| \5\ | Tabulator v4.9 http://tabulator.info/ |
| \6\ | SITRAIN System Course "WinCC Unified & Unified Comfort Panels" https://support.industry.siemens.com/cs/ww/en/view/109773211 |
| \7\ | SITRAIN advanced course: SIMATIC WinCC Unified for PC systems https://support.industry.siemens.com/cs/ww/en/view/109781323 |
| \8\ | SITRAIN: You can find out more about the training and courses as well as their locations and dates at: https://www.siemens.com/sitrain |

7.4 Change documentation

Table 7-2

| Version | Date | Change |
|---------|---------|--|
| V1.0 | 06/2020 | First version |
| V2.0 | 04/2021 | Custom Web Control "Table" added and document restructured |