Portable API Reference

Build a display control once in C# with the Portable API and use it in both the Windows clients and the browser: one source file, compiled into a WPF build and an HTML5 build.

A portable control is a WPF `UserControl` that implements `T.Toolkit.IPortableControl` and is compiled twice from the same source: once for the WPF clients and once, through OpenSilver, for the HTML5 (browser) client. After you copy the two DLLs into the FrameworX installation, the Designer lists the control in the Components panel like a built-in control, with its own configuration panel.

This page covers **building** the control, **deploying** it to FrameworX and **finding it** in the Designer, using a downloadable template.

> A portable control is not placed with the WPF Control element. For a control that only needs to run on WPF clients, see WPF Control API Reference. For one that only runs in the browser, see Blazor Control API Reference.

Download the template
How a portable control works
Requirements
Rules
Build, deploy and use the template
- 1. Build
- 2. Deploy to FrameworX
- 3. Find it in the Designer
- 4. Place it and link a tag
- 5. Make it your own control

How the code works
- One source, two platforms
- Linking a tag
- Symbols, copy and paste, and saving
- Icon and configuration panel
- The browser side

The IPortableControl members
Troubleshooting
Source code
Related
- In this section...

Download the template

Download PortableControl-SampleGauge.zip. It contains a complete portable control, `TSampleGauge`: a circular gauge linked to a tag, with a minimum, a maximum, a header and a Designer configuration panel.

File	Compiled into	Role
`SampleGauge\TSampleGauge.cs`	Both (linked)	The control: properties, user interface, runtime tag binding and the `IPortableControl` members.
`SampleGauge\TSampleGaugeConfig.xaml` / `.xaml.cs`	WPF only	The configuration panel the Designer opens when you double-click the control.
`SampleGauge\TSampleGauge.png`	WPF only	The icon shown in the Components panel.
`SampleGauge.HTML5\SfCircularGauge.cs`	HTML5 only	The wrapper: a C# class with the same members as the WPF gauge, implemented on the JavaScript gauge.
`nuget.config`		Adds the OpenSilver package feed the HTML5 project needs.
`SampleGauge.sln`		Opens both projects.

Validated with FrameworX 10.1.5, in the WPF client and in the browser.

How a portable control works

One control source file is linked into two projects. Each project produces a DLL with the **same file name**, and each DLL goes to its own folder of the FrameworX installation:

	WPF build	HTML5 build
Project	`SampleGauge\TSampleGauge.csproj`	`SampleGauge.HTML5\TSampleGauge.HTML5.csproj`
Target	.NET Framework 4.8	.NET Standard 2.0 + OpenSilver
User interface	WPF components (the template uses the Syncfusion WPF gauge shipped with FrameworX)	JavaScript components driven from C# through a wrapper class (the template uses the Syncfusion JavaScript gauge from the web client's `Libraries` folder)
Output	`T.Portable.Controls.TSampleGauge.dll`	`T.Portable.Controls.TSampleGauge.dll`

Deploy to	`C:\Program Files\Tatsoft\FrameworX\fx-10\WpfControls\`	`C:\Program Files\Tatsoft\FrameworX\fx-10\HTML5\ExtensionControls\`
Used by	The Designer, the RichClient and the SmartClient	The HTML5 client in the browser, which downloads it by name

There is no registration step. When the Designer starts, it scans the `WpfControls` folder and lists every control it finds there. If a DLL with the same name is also in `HTML5\ExtensionControls`, the control is portable; if not, it is listed with a `(WPF)` suffix and works only on WPF clients.

Requirements

- Windows and Visual Studio with the **.NET desktop development** workload, and a current .NET SDK (the template was built with .NET SDK 10).
- A FrameworX installation. Both projects reference the FrameworX assemblies (`T.Library`, `T.Toolkit` and, for WPF, `T.Library.Wpf`, `T.Toolkit.Wpf`, `T.Kernel`, `T.Wpf` and the Syncfusion gauge) from `C:\Program Files\Tatsoft\FrameworX\fx-10\`.
- Access to nuget.org and to the OpenSilver package feed `https://www.myget.org/F/opensilver/api/v3/index.json`, declared in the template's `nuget.config`.
- Place the control on a **Portable** display to use it in the WPF clients (Windows, RichClient, SmartClient) and in the browser.

Rules

Follow these rules for every portable control. Breaking one usually produces no error: the control is missing from the Components panel, shows as `(WPF)`, or does not load in the browser.

1. **Name the assembly `T.Portable.Controls.T<Name>`, with `RootNamespace` `T.Portable.Controls`.** The Designer only loads DLLs from `WpfControls` whose file name contains `T.Portable.`.
2. **Both builds have the same file name.** The WPF build goes to `fx-10\WpfControls\`, the HTML5 build to `fx-10\HTML5\ExtensionControls\`. The browser downloads the HTML5 build by that name.
3. **The control class is public and implements `IPortableControl`** (namespace `T.Toolkit`). The Designer registers every public class in the DLL that implements it.
4. **The static `GetIconImage()` returns the icon.** Its `pack://` URI must name your own assembly and `.png` file. If it returns nothing, the Designer skips the control without an error.
5. **The class name decides where the control appears.** The Components panel label is the class name without the leading `T`. The section is chosen by a case-sensitive text match on the class name: `Chart` goes to **Charts**, `Gauge` to **Gauges**, `View` to **Viewer**; any other name goes to **Extensions**.
6. **Do not reuse a product control name.** Your DLL would replace the product's. Four retired names are also never listed in the Components panel: `TCircularGauge`, `TCenterValueCircularGauge`, `TRangeCircularGauge` and `TSemiCircleCircularGauge`.
7. **The control source is linked into both projects, never copied.** Put platform differences behind `#if OPENSILVER`. The configuration panel and the icon belong to the WPF project only.
8. **The OpenSilver package version matches the FrameworX web client.** For FrameworX 10.1.5 it is `3.4.0-preview-2026-07-03-134452-d65dd77f`, with `System.Formats.Asn1` and `System.Security.Cryptography.Xml` 10.0.10. A different version builds, but the HTML5 build fails to load in the browser.
9. **FrameworX assemblies are referenced, not copied.** Every FrameworX and third-party reference has `<Private>False</Private>`, and you deploy only the control DLL from each output folder.
10. **Close the Designer before replacing the DLLs, and restart it to see a new control.** The Designer keeps the DLLs loaded, and it scans `WpfControls` only when it starts.

Build, deploy and use the template

1. Build

Open `SampleGauge.sln` and build in Release. If FrameworX is not installed in `C:\Program Files\Tatsoft\FrameworX`, change `FrameworXPath` in both project files to your installation's `fx-10` folder, keeping the trailing backslash. The two builds are:

- WPF: `SampleGauge\bin\Release\T.Portable.Controls.TSampleGauge.dll`
- HTML5: `SampleGauge.HTML5\bin\T.Portable.Controls.TSampleGauge.dll`

Both output folders also contain other assemblies. Copy only the control DLL (rule 9).

2. Deploy to FrameworX

1. Close the Designer (rule 10).
2. Copy the WPF build to `C:\Program Files\Tatsoft\FrameworX\fx-10\WpfControls\`.
3. Copy the HTML5 build to `C:\Program Files\Tatsoft\FrameworX\fx-10\HTML5\ExtensionControls\`.

Both folders are under Program Files, so copying may require administrator rights.

3. Find it in the Designer

Open the Designer and a Portable display in **Displays / Draw**. In the Components panel, open the **Gauges** section: the control is listed as **SampleGauge**, without a `(WPF)` suffix. A control whose class name contains none of Chart, Gauge or View is listed in an **Extensions** section instead (rule 5).

4. Place it and link a tag

Drag **SampleGauge** onto the display and double-click it to open its configuration panel. Enter a tag in **Linked Value** (for example `Tag.Level`) and, if needed, the **Minimum Value**, **Maximum Value** and **Header**. Run the solution: the gauge follows the tag in the WPF client and in the browser.

5. Make it your own control

Choose a class name that follows rules 1, 5 and 6, then change it everywhere below. `SfCircularGauge` is the name of the JavaScript gauge wrapper, not of the control, and can stay.

File	What to change
File and folder names	`TSampleGauge.*`, `TSampleGaugeConfig.*`, and the `SampleGauge` / `SampleGauge.HTML5` folders.
`TSampleGauge.csproj` and `TSampleGauge.HTML5.csproj`	`AssemblyName`, the `Compile`, `Page` and `Resource` items, and the linked source path in the HTML5 project.
`TSampleGauge.cs`	The class and constructor names, the configuration class in `GetConfigControl()`, and the assembly and file name in the `pack://` URI inside `GetIconImage()`.
`TSampleGaugeConfig.xaml` / `.xaml.cs`	`x:Class`, the class name, the control type it casts to, and the design-time `DataContext` type.
`Properties\AssemblyInfo.cs`, `SampleGauge.sln`	The title, product and project names.

How the code works

One source, two platforms

The shared file switches between the WPF gauge and the JavaScript wrapper with a conditional `using`. The OpenSilver package defines the `OPENSILVER` symbol in the HTML5 project.

TSampleGauge.cs

```
#if OPENSILVER
using T.Portable.Components.Gauges;
#else
using Syncfusion.UI.Xaml.Gauges;
#endif
```

Linking a tag

Every bindable property is a `string` that holds a link: a constant, a tag or an expression. The setter applies constants immediately so the Designer preview updates. Tag and expression values arrive at runtime.

TSampleGauge.cs: a bindable property

```
public ObjectReference linkedValueRun = null;
private string _linkedValue = "";
[DefaultValue("")]
public string LinkedValue
{
    get { return _linkedValue; }
    set
    {
        if (this._linkedValue != value)
        {
            this._linkedValue = value;
            NotifyDesignPropertyChanged();
            if (IsStaticValue(value))
            {
                double x = TConvert.ToDouble(value);
                gPointer.Value = x;
                if (gPointer2 != null)
                {
                    gPointer2.Value = x;
                }
            }
        }
    }
}
```

`StartRuntime()` runs when the display opens. It turns each link into an `ObjectReference` and registers `HandleTagEvent`, which runs on every change and pushes the new value into the gauge. `Dispose()` releases each reference with `WK.Dispose` when the display closes.

TSampleGauge.cs: StartRuntime (start and end)

```
public void StartRuntime()
{
    UpdateColors();

    this.headerRun = ObjectReference.ParseString(WK.UntokenizeObjectName(this.HeaderLink).Trim('"'));
    if (!WK.IsUpdatingReport && this.headerRun != null)
        this.headerRun.RegisterEvent(this.HandleTagEvent);

    this.linkedValueRun = ObjectReference.ParseExpression(this.LinkedValue.Trim(), null);
    if (!WK.IsUpdatingReport && this.linkedValueRun != null)
        this.linkedValueRun.RegisterEvent(this.HandleTagEvent);

    ObjectReference.AddEventToExecute(this.HandleTagEvent);
}
```

TSampleGauge.cs: inside HandleTagEvent

```
if (this.linkedValueRun != null)
    gScale.Pointers[0].Value = TK.To<double>(await this.linkedValueRun.GetValueAsync());
```

Symbols, copy and paste, and saving

Every link must also be listed in the token, label, clipboard and save members, one line per link. Leaving them empty compiles and seems to work, then breaks symbol parameters, renames, copy and paste, and cross references.

TSampleGauge.cs: GetTokens and OnSave

```
public void GetTokens(object symbol)
{
    WK.GetTokens(symbol, MinValueLink, eFieldTypeTk.NumberOrObject);
    WK.GetTokens(symbol, MaxValueLink, eFieldTypeTk.NumberOrObject);
    WK.GetTokens(symbol, LinkedValue, eFieldTypeTk.Expression);
    WK.GetTokens(symbol, HeaderLink, eFieldTypeTk.ObjectOrString);
    WK.GetTokens(symbol, RimThicknessLink, eFieldTypeTk.NumberOrObject);
    WK.GetTokens(symbol, PointerThicknessLink, eFieldTypeTk.NumberOrObject);
    WK.GetTokens(symbol, MajorTicksQuantityLink, eFieldTypeTk.NumberOrObject);
    WK.GetTokens(symbol, MinorTicksPerIntervalLink, eFieldTypeTk.NumberOrObject);
}

public void OnSave(int displayId, object el)
{
    OnStartClipboardCopy();
    OnFinishClipboardCopy();

    WK.SaveHelper.SaveStart(displayId, el);

    WK.SaveHelper.SaveLink(LinkedValue, "LinkedValue");
    WK.SaveHelper.SaveLink(MinValueLink, "MinValue");
    WK.SaveHelper.SaveLink(MaxValueLink, "MaxValue");
    WK.SaveHelper.SaveLink(HeaderLink, "HeaderLink");
    WK.SaveHelper.SaveLink(RimThicknessLink, "RimThickness");
    WK.SaveHelper.SaveLink(PointerThicknessLink, "PointerThickness");
    WK.SaveHelper.SaveLink(MajorTicksQuantityLink, "MajorTicksQuantity");
    WK.SaveHelper.SaveLink(MinorTicksPerIntervalLink, "MinorTicksPerInterval");
```

Icon and configuration panel

TSampleGauge.cs

```
        static public byte[] GetIconImage()
        {
            string _imageUri = @"pack://application:,,,/T.Portable.Controls.TSampleGauge;component
/TSampleGauge.png";
            return WK.GetImageFromUri(_imageUri);
        }

        public IControlConfig GetConfigControl()
        {
#if OPENSILVER
            return null;
#else
            return new TSampleGaugeConfig();
#endif
        }
```

The browser side

The HTML5 build declares the JavaScript and CSS files the web client loads before the control renders, through `IHTML5ControlDependencies`. Paths are relative to the web client root or absolute URLs.

TSampleGauge.cs: IHTML5ControlDependencies

```
public string[] ControlDataJS
{
    get
    {
        return new string[] {
                @"Libraries/ej2-base/dist/global/ej2-base.min.js",
                @"Libraries/ej2-circulargauge/dist/global/ej2-circulargauge.min.js",
            };
    }
}

public string[] ControlDataCSS
{
    get
    {
        return new string[] {
                "Libraries/ej2-popups/styles/material.css",
                "https://fonts.googleapis.com/css?family=Roboto:400,500"
           };
    }
}
```

The wrapper exposes the same members as the WPF gauge. Each property that affects rendering must push its value to the JavaScript object in its setter. A property that is only stored in C# compiles, but a tag change never reaches the browser.

SfCircularGauge.cs: a property that pushes to JavaScript

```
public double Value
{
    get => objValue;
    set
    {
        objValue = value;
        if (this.CircularGaugeWebAssemblyObject == null || this.circularPointer == null)
            return;

        OpenSilver.Interop.ExecuteJavaScript("$0.value = $1", this.circularPointer, objValue);
    }
```

The IPortableControl members

Member	When FrameworX calls it	What the control does
`GetConfigControl()`	The Designer opens the configuration panel	Returns the configuration panel on WPF, `null` under `OPENSILVER`.
`StartRuntime()`	The display opens at runtime	Turns links into `ObjectReference` objects and registers the change handler.
`Dispose()`	The display closes	Releases every `ObjectReference` and event it attached.
`GetTokens`, `ApplyTokens`	The control is used in a symbol	Exposes every link so symbol parameters are replaced.
`GetLabels`, `ResolveLabels`	Symbol labels are renamed	Same links, so renames reach the control.
`GetStrings`, `ApplyStrings`	Translation	Exposes fixed texts for localization.
`OnStartClipboardCopy`, `OnFinishClipboardCopy`, `OnFinishClipboardPaste`	Copy and paste	Keeps every link valid, one line per link.

`OnSave(int displayId, object el)`	The display is saved	Calls `WK.SaveHelper.SaveStart`, then `SaveLink` for each link. This feeds cross references.
`GetIconImage()`	The Designer lists the control	Static method, not part of the interface. Returns the icon.

Optional interfaces used by the template: `IHTML5ControlDependencies` (JavaScript and CSS for the browser), `IWKUpdateColors` (theme changes) and `INotifyPropertyChanged` (refreshes the Designer preview from the configuration panel).

Troubleshooting

Symptom	Cause	Fix
The control is not in the Components panel.	The DLL is not in `WpfControls`, its name does not contain `T.Portable.`, the Designer was not restarted, `GetIconImage()` returns nothing, or the class uses a retired name.	Check rules 1, 3, 4, 6 and 10.
The control appears in a different section than expected.	The section comes from the class name, with a case-sensitive match.	See rule 5.
The control shows `(WPF)`.	No DLL with the same name in `HTML5\ExtensionControls`.	Deploy the HTML5 build with the exact same file name (rule 2).
It works in the WPF client but not in the browser.	OpenSilver version mismatch, a missing JavaScript or CSS dependency, or a wrapper property that does not push to JavaScript.	Check rule 8, `IHTML5ControlDependencies`, and the wrapper setters.
Restore fails: package not found.	The OpenSilver preview packages are not on nuget.org.	Keep the template's `nuget.config` with the OpenSilver feed.
Build error NU1605, package downgrade of `System.Security.Cryptography.Xml`.	An older version is referenced than the one OpenSilver requires.	Use version 10.0.10 (rule 8).
The WPF output folder is full of FrameworX DLLs.	A reference lacks `<Private>False</Private>`.	Set it on every FrameworX reference, and deploy only the control DLL (rule 9).
Copying fails: "being used by another process".	The Designer or a client has the DLL loaded.	Close them and copy again (rule 10).

Source code

The complete source of this template is in `PortableControl-SampleGauge.zip`, available under **Download the template** at the top of this page. That zip is the single, authoritative copy of the code. The code on this page is excerpts only, to explain how the template works.

Related

- Custom Control API Reference: choose between WPF, Blazor and Portable controls.
- WPF Control API Reference: a WPF-only control placed with the WPF Control element.
- Blazor Control API Reference: a browser-only control built with Razor components.

In this section...