Workspace Desktop Edition Developer's Guide

Adding web content to Workspace plugins using Microsoft Edge WebView2

3/31/2025

Contents

Adding web content to Workspace plugins using Microsoft Edge WebView2

To display web content inside Workspace Desktop Edition, developers of Workspace custom modules are often required to integrate a Web browser control into a custom Workspace view.

In the past, developers could leverage the built in .NET Framework WebBowser; however, this embedded web technology based on Microsoft Internet Explorer became out of pace for modern business web application ecosystem, forcing the developer community to employ third-party solutions. Most of these solutions were based on the Chromium Open Source project. This approach came with the constraint of evaluating and adopting a third-party library which sometimes generated compatibility issues for multiple concurrent plugins used the same Chromium-based library.

Workspace Desktop Edition now embeds a WebView2 SDK, which allows Chromium-based rendering customizations with WebView2 control supported by Microsoft Edge WebView2.

Microsoft Edge WebView2 is:

- A lightweight wrapper DLL-set for developers (embedded in Workspace) and a runtime (installed independently and shared with other applications).
- A solution to run multiple web-rendering plugins running side-by-side with full separation (if following code recommendations).
- An API supported by Microsoft.

Microsoft Edge WebView2 is NOT:

- A wrapper on top of Microsoft Edge. Microsoft Edge WebView2 runtime is installed independently from Microsoft Edge browser.
- The unique approach to render Web content in Workspace. Developers can continue to work with their favorite WebBrowser control.

Installation

For WebView2 to operate correctly with a WebView2-based plugin in Workspace, Genesys requires WebView2 to be installed in the **Evergreen** installation mode.

Here are some useful links for getting started with WebView2:

- Download the runtime here.
- Documentation starts here.

Pre-requisites for Microsoft Edge WebView2

Beginning with Workspace 8.5.147.05 or higher, you can access a WebView2 sample that you can use to create your own WebView2 app for Workspace.

Microsoft reference

https://docs.microsoft.com/en-us/microsoft-edge/webview2/

Build pre-requisites (for customization developers)

- .NET Framework 4.6.2 or higher.
- Visual Studio with .NET SDK 4.6.2 or above available (natively in the Visual Studio version or installed as an add-on).
- WebView2 Runtime installed though any Evergreen installation mode.
- WebView2 SDK is a part of WDE 8.5.147.05 or above. **Only DLLs located in the Workspace application should be used for developing.**
  - These can be updated in subsequent versions of Workspace.
  - Plugins based on older versions of WebView2 are supported.
  - WebView2 SDK is in the Workspace Library.

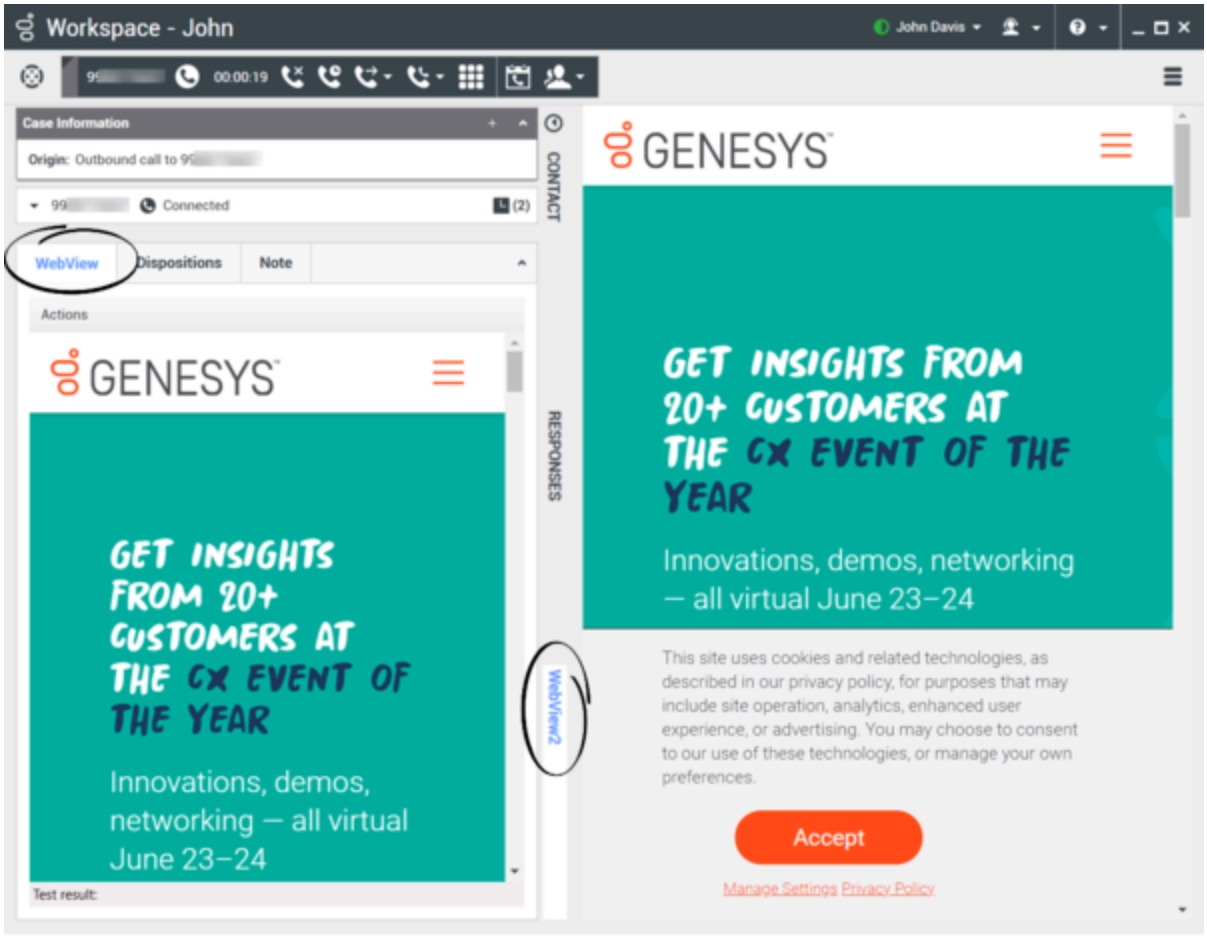
Runtime pre-requisites (for end-users of Workspace)

- .NET Framework 4.6.2 or higher.
- WebView2 Runtime installed though any Evergreen installation mode. **You must install this, it is not provided with Workspace**.
- WDE 8.5.147.05 or above.

Sample Workspace plugin based on Microsoft Edge WebView2

Workspace 8.5.147.05 or higher includes a sample Workspace WebView2 plugin that you can use to help you develop your own app to embed web content in Workspace.

This image demonstrates some of the locations where you can implement a WebView2 plugin.



Installing the sample

1. Download Workspace 8.5.147.05 or higher.
2. Install Workspace. Choose "Install Workspace Desktop Edition Developer Toolkit".
3. Find the folder with the WebView2 plugin sample in the **Samples** folder.
4. Open the solution in Visual Studio and Build it.
5. Run the Workspace application from the **Debug** folder.

This sample demonstrates some best practices to implement web content integration through the API of Microsoft Edge WebView2:

Communication between host and web content

The host and web content can communicate with each other by using `postMessage` as follows:

- Web content in a WebView2 control can post a message to the host using `window.chrome.webview.postMessage`. The host handles the message using any registered `WebMessageReceived` on the host.

- The Host can post messages to the web content in a WebView2 control using `CoreWebView2.PostWebMessageAsString` or `CoreWebView2.PostWebMessageAsJSON`. These messages are caught by handlers and are added to `window.chrome.webview.addEventListener`.

Navigation events

During webpage navigation, the WebView2 control raises events. The application that hosts WebView2 controls listens for the following events:

- `NavigationStarting`
- `SourceChanged`
- `ContentLoading`
- `HistoryChanged`
- `NavigationCompleted`

Process model

For information about how to configure your environment, refer to the WebView2 Process Model documentation. It is possible to use different configurations for each plugin you develop or use.

A single browser process is specified for each user data folder in a user session that serves WebView2 process requests specifying that user data folder. Therefore, one browser process can serve multiple process requests, and one requesting process can use multiple browser processes.

Each browser process has some number of associated renderer processes. Browser processes are created as needed to service (potentially) multiple frames in different instances of WebView2. The number of renderer processes varies based on the site isolation browser feature and the number of distinct disconnected origins rendered in associated instances of WebView2.

Recommendation: to fully separate the runtime environment of your WebView2-based plugin from the runtime of other WebView2-based plugins from other vendors, define a unique data folder name for your plugin. A simple way to do that is to assign the namespace of your plugin as the name of this folder.

Example with two different environments:

```
CoreWebView2EnvironmentOptions opt = new CoreWebView2EnvironmentOptions();

opt.AdditionalBrowserArguments = "--enable-logging --log-
file=C:\\MyData\\tests\\WebView1.log";

CoreWebView2Environment env = await CoreWebView2Environment.CreateAsync(null,
"WebViewFolder1", opt);

await webView.EnsureCoreWebView2Async(env);

CoreWebView2EnvironmentOptions opt2 = new CoreWebView2EnvironmentOptions();

opt2.AdditionalBrowserArguments = "--enable-logging --log-
file=C:\\MyData\\tests\\WebView2.log";

CoreWebView2Environment env2 = await CoreWebView2Environment.CreateAsync(null,
"WebViewFolder2", opt2);
```

```
await webView2.EnsureCoreWebView2Async(env2);
```