

# Reference (WebView2 Win32 C++)

Article • 02/26/2024

The Microsoft Edge WebView2 control enables you to host web content in your application using [Microsoft Edge \(Chromium\)](#) as the rendering engine. For more information, see [Overview of Microsoft Edge WebView2](#)) and [Getting Started with WebView2](#). [ICoreWebView2](#) is a great place to start learning the details of the API.

## Globals

- [Globals](#)

## Interfaces

- [ICoreWebView2](#)
- [ICoreWebView2\\_10](#)
- [ICoreWebView2\\_11](#)
- [ICoreWebView2\\_12](#)
- [ICoreWebView2\\_13](#)
- [ICoreWebView2\\_14](#)
- [ICoreWebView2\\_15](#)
- [ICoreWebView2\\_16](#)
- [ICoreWebView2\\_17](#)
- [ICoreWebView2\\_18](#)
- [ICoreWebView2\\_19](#)
- [ICoreWebView2\\_2](#)
- [ICoreWebView2\\_20](#)
- [ICoreWebView2\\_21](#)
- [ICoreWebView2\\_22](#)
- [ICoreWebView2\\_3](#)
- [ICoreWebView2\\_4](#)
- [ICoreWebView2\\_5](#)
- [ICoreWebView2\\_6](#)
- [ICoreWebView2\\_7](#)
- [ICoreWebView2\\_8](#)
- [ICoreWebView2\\_9](#)
- [ICoreWebView2AcceleratorKeyPressedEventArgs](#)
- [ICoreWebView2AcceleratorKeyPressedEventArgs2](#)
- [ICoreWebView2BasicAuthenticationRequestedEventArgs](#)

- [ICoreWebView2BasicAuthenticationResponse](#)
- [ICoreWebView2BrowserExtension](#)
- [ICoreWebView2BrowserExtensionList](#)
- [ICoreWebView2BrowserProcessExitedEventArgs](#)
- [ICoreWebView2Certificate](#)
- [ICoreWebView2ClientCertificate](#)
- [ICoreWebView2ClientCertificateCollection](#)
- [ICoreWebView2ClientCertificateRequestedEventArgs](#)
- [ICoreWebView2CompositionController](#)
- [ICoreWebView2CompositionController2](#)
- [ICoreWebView2CompositionController3](#)
- [ICoreWebView2ContentLoadingEventArgs](#)
- [ICoreWebView2ContextMenuItem](#)
- [ICoreWebView2ContextMenuItemCollection](#)
- [ICoreWebView2ContextMenuRequestedEventArgs](#)
- [ICoreWebView2ContextMenuTarget](#)
- [ICoreWebView2Controller](#)
- [ICoreWebView2Controller2](#)
- [ICoreWebView2Controller3](#)
- [ICoreWebView2Controller4](#)
- [ICoreWebView2ControllerOptions](#)
- [ICoreWebView2ControllerOptions2](#)
- [ICoreWebView2Cookie](#)
- [ICoreWebView2CookieList](#)
- [ICoreWebView2CookieManager](#)
- [ICoreWebView2CustomSchemeRegistration](#)
- [ICoreWebView2Deferral](#)
- [ICoreWebView2DevToolsProtocolEventReceivedEventArgs](#)
- [ICoreWebView2DevToolsProtocolEventReceivedEventArgs2](#)
- [ICoreWebView2DevToolsProtocolEventReceiver](#)
- [ICoreWebView2DOMContentLoadedEventArgs](#)
- [ICoreWebView2DownloadOperation](#)
- [ICoreWebView2DownloadStartingEventArgs](#)
- [ICoreWebView2Environment](#)
- [ICoreWebView2Environment10](#)
- [ICoreWebView2Environment11](#)
- [ICoreWebView2Environment12](#)
- [ICoreWebView2Environment13](#)
- [ICoreWebView2Environment2](#)
- [ICoreWebView2Environment3](#)

- [ICoreWebView2Environment4](#)
- [ICoreWebView2Environment5](#)
- [ICoreWebView2Environment6](#)
- [ICoreWebView2Environment7](#)
- [ICoreWebView2Environment8](#)
- [ICoreWebView2Environment9](#)
- [ICoreWebView2EnvironmentOptions](#)
- [ICoreWebView2EnvironmentOptions2](#)
- [ICoreWebView2EnvironmentOptions3](#)
- [ICoreWebView2EnvironmentOptions4](#)
- [ICoreWebView2EnvironmentOptions5](#)
- [ICoreWebView2EnvironmentOptions6](#)
- [ICoreWebView2ExecuteScriptResult](#)
- [ICoreWebView2File](#)
- [ICoreWebView2Frame](#)
- [ICoreWebView2Frame2](#)
- [ICoreWebView2Frame3](#)
- [ICoreWebView2Frame4](#)
- [ICoreWebView2Frame5](#)
- [ICoreWebView2FrameCreatedEventArgs](#)
- [ICoreWebView2FrameInfo](#)
- [ICoreWebView2FrameInfo2](#)
- [ICoreWebView2FrameInfoCollection](#)
- [ICoreWebView2FrameInfoCollectionIterator](#)
- [ICoreWebView2HttpHeadersCollectionIterator](#)
- [ICoreWebView2HttpRequestHeaders](#)
- [ICoreWebView2HttpResponseHeaders](#)
- [ICoreWebView2LaunchingExternalUriSchemeEventArgs](#)
- [ICoreWebView2MoveFocusRequestedEventArgs](#)
- [ICoreWebView2NavigationCompletedEventArgs](#)
- [ICoreWebView2NavigationCompletedEventArgs2](#)
- [ICoreWebView2NavigationStartingEventArgs](#)
- [ICoreWebView2NavigationStartingEventArgs2](#)
- [ICoreWebView2NavigationStartingEventArgs3](#)
- [ICoreWebView2NewWindowRequestedEventArgs](#)
- [ICoreWebView2NewWindowRequestedEventArgs2](#)
- [ICoreWebView2NewWindowRequestedEventArgs3](#)
- [ICoreWebView2ObjectCollectionView](#)
- [ICoreWebView2PermissionRequestedEventArgs](#)
- [ICoreWebView2PermissionRequestedEventArgs2](#)

- [ICoreWebView2PermissionRequestedEventArgs3](#)
- [ICoreWebView2PermissionSetting](#)
- [ICoreWebView2PermissionSettingCollectionView](#)
- [ICoreWebView2PointerInfo](#)
- [ICoreWebView2PrintSettings](#)
- [ICoreWebView2PrintSettings2](#)
- [ICoreWebView2ProcessExtendedInfo](#)
- [ICoreWebView2ProcessExtendedInfoCollection](#)
- [ICoreWebView2ProcessFailedEventArgs](#)
- [ICoreWebView2ProcessFailedEventArgs2](#)
- [ICoreWebView2ProcessInfo](#)
- [ICoreWebView2ProcessInfoCollection](#)
- [ICoreWebView2Profile](#)
- [ICoreWebView2Profile2](#)
- [ICoreWebView2Profile3](#)
- [ICoreWebView2Profile4](#)
- [ICoreWebView2Profile5](#)
- [ICoreWebView2Profile6](#)
- [ICoreWebView2Profile7](#)
- [ICoreWebView2Profile8](#)
- [ICoreWebView2ScriptDialogOpeningEventArgs](#)
- [ICoreWebView2ScriptException](#)
- [ICoreWebView2ServerCertificateErrorDetectedEventArgs](#)
- [ICoreWebView2Settings](#)
- [ICoreWebView2Settings2](#)
- [ICoreWebView2Settings3](#)
- [ICoreWebView2Settings4](#)
- [ICoreWebView2Settings5](#)
- [ICoreWebView2Settings6](#)
- [ICoreWebView2Settings7](#)
- [ICoreWebView2Settings8](#)
- [ICoreWebView2SharedBuffer](#)
- [ICoreWebView2SourceChangedEventArgs](#)
- [ICoreWebView2StringCollection](#)
- [ICoreWebView2WebMessageReceivedEventArgs](#)
- [ICoreWebView2WebMessageReceivedEventArgs2](#)
- [ICoreWebView2WebResourceRequest](#)
- [ICoreWebView2WebResourceRequestedEventArgs](#)
- [ICoreWebView2WebResourceRequestedEventArgs2](#)
- [ICoreWebView2WebResourceResponse](#)

- [ICoreWebView2WebResourceResponseReceivedEventArgs](#)
- [ICoreWebView2WebResourceResponseView](#)
- [ICoreWebView2WindowFeatures](#)

## Delegates

- [ICoreWebView2AcceleratorKeyPressedEventHandler](#)
- [ICoreWebView2AddScriptToExecuteOnDocumentCreatedCompletedHandler](#)
- [ICoreWebView2BasicAuthenticationRequestedEventHandler](#)
- [ICoreWebView2BrowserExtensionEnableCompletedHandler](#)
- [ICoreWebView2BrowserExtensionRemoveCompletedHandler](#)
- [ICoreWebView2BrowserProcessExitedEventHandler](#)
- [ICoreWebView2BytesReceivedChangedEventHandler](#)
- [ICoreWebView2CallDevToolsProtocolMethodCompletedHandler](#)
- [ICoreWebView2CapturePreviewCompletedHandler](#)
- [ICoreWebView2ClearBrowsingDataCompletedHandler](#)
- [ICoreWebView2ClearServerCertificateErrorActionsCompletedHandler](#)
- [ICoreWebView2ClientCertificateRequestedEventHandler](#)
- [ICoreWebView2ContainsFullScreenElementChangedEventHandler](#)
- [ICoreWebView2ContentLoadingEventHandler](#)
- [ICoreWebView2ContextMenuRequestedEventHandler](#)
- [ICoreWebView2CreateCoreWebView2CompositionControllerCompletedHandler](#)
- [ICoreWebView2CreateCoreWebView2ControllerCompletedHandler](#)
- [ICoreWebView2CreateCoreWebView2EnvironmentCompletedHandler](#)
- [ICoreWebView2CursorChangedEventHandler](#)
- [ICoreWebView2CustomItemSelectedEventHandler](#)
- [ICoreWebView2DevToolsProtocolEventReceivedEventHandler](#)
- [ICoreWebView2DocumentTitleChangedEventHandler](#)
- [ICoreWebView2DOMContentLoadedEventHandler](#)
- [ICoreWebView2DownloadStartingEventHandler](#)
- [ICoreWebView2EstimatedEndTimeChangedEventHandler](#)
- [ICoreWebView2ExecuteScriptCompletedHandler](#)
- [ICoreWebView2ExecuteScriptWithResultCompletedHandler](#)
- [ICoreWebView2FaviconChangedEventHandler](#)
- [ICoreWebView2FocusChangedEventHandler](#)
- [ICoreWebView2FrameContentLoadingEventHandler](#)
- [ICoreWebView2FrameCreatedEventHandler](#)
- [ICoreWebView2FrameDestroyedEventHandler](#)
- [ICoreWebView2FrameDOMContentLoadedContentLoadedEventHandler](#)
- [ICoreWebView2FrameNameChangedEventHandler](#)

- [ICoreWebView2FrameNavigationCompletedEventHandler](#)
- [ICoreWebView2FrameNavigationStartingEventHandler](#)
- [ICoreWebView2FramePermissionRequestedEventHandler](#)
- [ICoreWebView2FrameWebMessageReceivedEventHandler](#)
- [ICoreWebView2GetCookiesCompletedHandler](#)
- [ICoreWebView2GetFaviconCompletedHandler](#)
- [ICoreWebView2GetNonDefaultPermissionSettingsCompletedHandler](#)
- [ICoreWebView2GetProcessExtendedInfosCompletedHandler](#)
- [ICoreWebView2HistoryChangedEventHandler](#)
- [ICoreWebView2IsDefaultDownloadDialogOpenChangedEventHandler](#)
- [ICoreWebView2IsDocumentPlayingAudioChangedEventHandler](#)
- [ICoreWebView2IsMutedChangedEventHandler](#)
- [ICoreWebView2LaunchingExternalUriSchemeEventHandler](#)
- [ICoreWebView2MoveFocusRequestedEventHandler](#)
- [ICoreWebView2NavigationCompletedEventHandler](#)
- [ICoreWebView2NavigationStartingEventHandler](#)
- [ICoreWebView2NewBrowserVersionAvailableEventHandler](#)
- [ICoreWebView2NewWindowRequestedEventHandler](#)
- [ICoreWebView2PermissionRequestedEventHandler](#)
- [ICoreWebView2PrintCompletedHandler](#)
- [ICoreWebView2PrintToPdfCompletedHandler](#)
- [ICoreWebView2PrintToPdfStreamCompletedHandler](#)
- [ICoreWebView2ProcessFailedEventHandler](#)
- [ICoreWebView2ProcessInfosChangedEventHandler](#)
- [ICoreWebView2ProfileAddBrowserExtensionCompletedHandler](#)
- [ICoreWebView2ProfileDeletedEventHandler](#)
- [ICoreWebView2ProfileGetBrowserExtensionsCompletedHandler](#)
- [ICoreWebView2RasterizationScaleChangedEventHandler](#)
- [ICoreWebView2ScriptDialogOpeningEventHandler](#)
- [ICoreWebView2ServerCertificateErrorDetectedEventHandler](#)
- [ICoreWebView2SetPermissionStateCompletedHandler](#)
- [ICoreWebView2SourceChangedEventHandler](#)
- [ICoreWebView2StateChangedEventHandler](#)
- [ICoreWebView2StatusBarTextChangedEventHandler](#)
- [ICoreWebView2TrySuspendCompletedHandler](#)
- [ICoreWebView2WebMessageReceivedEventHandler](#)
- [ICoreWebView2WebResourceRequestedEventHandler](#)
- [ICoreWebView2WebResourceResponseReceivedEventHandler](#)
- [ICoreWebView2WebResourceResponseViewGetContentCompletedHandler](#)
- [ICoreWebView2WindowCloseRequestedEventHandler](#)

- ICoreWebView2ZoomFactorChangedEventHandler
- 

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2

Article • 02/26/2024

```
interface ICoreWebView2
: public IUnknown
```

WebView2 enables you to host web content using the latest Microsoft Edge browser and web technology.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">add_ContainsFullScreenElementChanged</a>	Add an event handler for the <code>ContainsFullScreenElementChanged</code> event.
<a href="#">add_ContentLoading</a>	Add an event handler for the <code>ContentLoading</code> event.
<a href="#">add_DocumentTitleChanged</a>	Add an event handler for the <code>DocumentTitleChanged</code> event.
<a href="#">add_FrameNavigationCompleted</a>	Add an event handler for the <code>FrameNavigationCompleted</code> event.
<a href="#">add_FrameNavigationStarting</a>	Add an event handler for the <code>FrameNavigationStarting</code> event.
<a href="#">add_HistoryChanged</a>	Add an event handler for the <code>HistoryChanged</code> event.
<a href="#">add_NavigationCompleted</a>	Add an event handler for the <code>NavigationCompleted</code> event.
<a href="#">add_NavigationStarting</a>	Add an event handler for the <code>NavigationStarting</code> event.
<a href="#">add_NewWindowRequested</a>	Add an event handler for the <code>NewWindowRequested</code> event.
<a href="#">add_PermissionRequested</a>	Add an event handler for the <code>PermissionRequested</code> event.

Members	Descriptions
add_ProcessFailed	Add an event handler for the <code>ProcessFailed</code> event.
add_ScriptDialogOpening	Add an event handler for the <code>ScriptDialogOpening</code> event.
add_SourceChanged	Add an event handler for the <code>SourceChanged</code> event.
add_WebMessageReceived	Add an event handler for the <code>WebMessageReceived</code> event.
add_WebResourceRequested	Add an event handler for the <code>WebResourceRequested</code> event.
add_WindowCloseRequested	Add an event handler for the <code>WindowCloseRequested</code> event.
AddHostObjectToScript	Add the provided host object to script running in the WebView with the specified name.
AddScriptToExecuteOnDocumentCreated	Add the provided JavaScript to a list of scripts that should be run after the global object has been created, but before the HTML document has been parsed and before any other script included by the HTML document is run.
AddWebResourceRequestedFilter	Adds a URI and resource context filter for the <code>WebResourceRequested</code> event.
CallDevToolsProtocolMethod	Runs an asynchronous <code>DevToolsProtocol</code> method.
CapturePreview	Capture an image of what WebView is displaying.
ExecuteScript	Run JavaScript code from the <code>javascript</code> parameter in the current top-level document rendered in the WebView.
get_BrowserProcessId	The process ID of the browser process that hosts the WebView.
get_CanGoBack	<code>TRUE</code> if the WebView is able to navigate to a previous page in the navigation history.
get_CanGoForward	<code>TRUE</code> if the WebView is able to navigate to a next page in the navigation history.
get_ContainsFullScreenElement	Indicates if the WebView contains a fullscreen HTML element.
get_DocumentTitle	The title for the current top-level document.

Members	Descriptions
<a href="#">get_Settings</a>	The ICoreWebView2Settings object contains various modifiable settings for the running WebView.
<a href="#">get_Source</a>	The URI of the current top level document.
<a href="#">GetDevToolsProtocolEventReceiver</a>	Get a DevTools Protocol event receiver that allows you to subscribe to a DevTools Protocol event.
<a href="#">GoBack</a>	Navigates the WebView to the previous page in the navigation history.
<a href="#">GoForward</a>	Navigates the WebView to the next page in the navigation history.
<a href="#">Navigate</a>	Cause a navigation of the top-level document to run to the specified URI.
<a href="#">NavigateToString</a>	Initiates a navigation to htmlContent as source HTML of a new document.
<a href="#">OpenDevToolsWindow</a>	Opens the DevTools window for the current document in the WebView.
<a href="#">PostWebMessageAsJson</a>	Post the specified webMessage to the top level document in this WebView.
<a href="#">PostWebMessageAsString</a>	Posts a message that is a simple string rather than a JSON string representation of a JavaScript object.
<a href="#">Reload</a>	Reload the current page.
<a href="#">remove_ContainsFullScreenElementChanged</a>	Remove an event handler previously added with <code>add_ContainsFullScreenElementChanged</code> .
<a href="#">remove_ContentLoading</a>	Remove an event handler previously added with <code>add_ContentLoading</code> .
<a href="#">remove_DocumentTitleChanged</a>	Remove an event handler previously added with <code>add_DocumentTitleChanged</code> .
<a href="#">remove_FrameNavigationCompleted</a>	Remove an event handler previously added with <code>add_FrameNavigationCompleted</code> .
<a href="#">remove_FrameNavigationStarting</a>	Remove an event handler previously added with <code>add_FrameNavigationStarting</code> .
<a href="#">remove_HistoryChanged</a>	Remove an event handler previously added with <code>add_HistoryChanged</code> .

Members	Descriptions
<code>remove_NavigationCompleted</code>	Remove an event handler previously added with <code>add_NavigationCompleted</code> .
<code>remove_NavigationStarting</code>	Remove an event handler previously added with <code>add_NavigationStarting</code> .
<code>remove_NewWindowRequested</code>	Remove an event handler previously added with <code>add_NewWindowRequested</code> .
<code>remove_PermissionRequested</code>	Remove an event handler previously added with <code>add_PermissionRequested</code> .
<code>remove_ProcessFailed</code>	Remove an event handler previously added with <code>add_ProcessFailed</code> .
<code>remove_ScriptDialogOpening</code>	Remove an event handler previously added with <code>add_ScriptDialogOpening</code> .
<code>remove_SourceChanged</code>	Remove an event handler previously added with <code>add_SourceChanged</code> .
<code>remove_WebMessageReceived</code>	Remove an event handler previously added with <code>add_WebMessageReceived</code> .
<code>remove_WebResourceRequested</code>	Remove an event handler previously added with <code>add_WebResourceRequested</code> .
<code>remove_WindowCloseRequested</code>	Remove an event handler previously added with <code>add_WindowCloseRequested</code> .
<code>RemoveHostObjectFromScript</code>	Remove the host object specified by the name so that it is no longer accessible from JavaScript code in the WebView.
<code>RemoveScriptToExecuteOnDocumentCreated</code>	Remove the corresponding JavaScript added using <code>AddScriptToExecuteOnDocumentCreated</code> with the specified script ID.
<code>RemoveWebResourceRequestedFilter</code>	Removes a matching WebResource filter that was previously added for the <code>WebResourceRequested</code> event.
<code>Stop</code>	Stop all navigations and pending resource fetches. Does not stop scripts.

## Applies to

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### add\_ContainsFullScreenElementChanged

Add an event handler for the `ContainsFullScreenElementChanged` event.

```
public HRESULT
add_ContainsFullScreenElementChanged(ICoreWebView2ContainsFullScreenElement
ChangedEventHandler * eventHandler, EventRegistrationToken * token)
```

`ContainsFullScreenElementChanged` triggers when the `ContainsFullScreenElement` property changes. An HTML element inside the WebView may enter fullscreen to the size of the WebView or leave fullscreen. This event is useful when, for example, a video element requests to go fullscreen. The listener of `ContainsFullScreenElementChanged` may resize the WebView in response.

C++

```
// Register a handler for the ContainsFullScreenChanged event.
CHECK_FAILURE(m_webView->add_ContainsFullScreenElementChanged(
    Callback<ICoreWebView2ContainsFullScreenElementChangedEventHandler>(
        [this](ICoreWebView2* sender, IUnknown* args) -> HRESULT
    {
        CHECK_FAILURE(
            sender-
        >get_ContainsFullScreenElement(&m_containsFullscreen));
        if (m_containsFullscreen)
        {
            EnterFullScreen();
        }
        else
        {
            ExitFullScreen();
        }
        return S_OK;
    })
})
```

```
.Get(),
nullptr));
```

## add\_ContentLoading

Add an event handler for the `ContentLoading` event.

```
public HRESULT add_ContentLoading(ICoreWebView2ContentLoadingEventHandler *
eventHandler, EventRegistrationToken * token)
```

`ContentLoading` triggers before any content is loaded, including scripts added with `AddScriptToExecuteOnDocumentCreated`. `ContentLoading` does not trigger if a same page navigation occurs (such as through `fragment` navigations or `history.pushState` navigations). This operation follows the `NavigationStarting` and `SourceChanged` events and precedes the `HistoryChanged` and `NavigationCompleted` events.

## add\_DocumentTitleChanged

Add an event handler for the `DocumentTitleChanged` event.

```
public HRESULT
add_DocumentTitleChanged(ICoreWebView2DocumentTitleChangedEventHandler *
eventHandler, EventRegistrationToken * token)
```

`DocumentTitleChanged` runs when the `DocumentTitle` property of the WebView changes and may run before or after the `NavigationCompleted` event.

C++

```
// Register a handler for the DocumentTitleChanged event.
// This handler just announces the new title on the window's title bar.
CHECK_FAILURE(m_webView->add_DocumentTitleChanged(
    Callback<ICoreWebView2DocumentTitleChangedEventHandler>(
        [this](ICoreWebView2* sender, IUnknown* args) -> HRESULT {
            wil::unique_cotaskmem_string title;
            CHECK_FAILURE(sender->get_DocumentTitle(&title));
            m_appWindow->SetDocumentTitle(title.get());
            return S_OK;
    })
    .Get(),
&m_documentTitleChangedToken));
```

## add\_FrameNavigationCompleted

Add an event handler for the `FrameNavigationCompleted` event.

```
public HRESULT  
add_FrameNavigationCompleted(ICoreWebView2NavigationCompletedEventHandler  
r * eventHandler, EventRegistrationToken * token)
```

`FrameNavigationCompleted` triggers when a child frame has completely loaded (concurrently when `body.onload` has triggered) or loading stopped with error.

C++

```
// Register a handler for the FrameNavigationCompleted event.  
// Check whether the navigation succeeded, and if not, do something.  
CHECK_FAILURE(m_webView->add_FrameNavigationCompleted(  
    Callback<ICoreWebView2NavigationCompletedEventHandler>(  
        [this](ICoreWebView2* sender,  
ICoreWebView2NavigationCompletedEventArgs* args)  
        -> HRESULT {  
            BOOL success;  
            CHECK_FAILURE(args->get_IsSuccess(&success));  
            if (!success)  
            {  
                COREWEBVIEW2_WEB_ERROR_STATUS webErrorStatus;  
                CHECK_FAILURE(args->  
>get_WebErrorStatus(&webErrorStatus));  
                // The web page can cancel its own iframe loads, so  
we'll ignore that.  
                if (webErrorStatus !=  
COREWEBVIEW2_WEB_ERROR_STATUS_OPERATION_CANCELED)  
                {  
                    m_appWindow->AsyncMessageBox(  
                        L"Iframe navigation failed: "  
                        + WebErrorStatusToString(webErrorStatus),  
                        L"Navigation Failure");  
                }  
            }  
            return S_OK;  
        })  
        .Get(),  
        &m_frameNavigationCompletedToken));
```

## add\_FrameNavigationStarting

Add an event handler for the `FrameNavigationStarting` event.

```
public HRESULT  
add_FrameNavigationStarting(ICoreWebView2NavigationStartingEventHandler *  
eventHandler, EventRegistrationToken * token)
```

`FrameNavigationStarting` triggers when a child frame in the WebView requests permission to navigate to a different URI. Redirects trigger this operation as well, and the navigation id is the same as the original one.

Navigations will be blocked until all `FrameNavigationStarting` event handlers return.

C++

```
// Register a handler for the FrameNavigationStarting event.  
// This handler will prevent a frame from navigating to a blocked  
domain.  
CHECK_FAILURE(m_webView->add_FrameNavigationStarting(  
    Callback<ICoreWebView2NavigationStartingEventHandler>(  
        [this](ICoreWebView2* sender,  
        ICoreWebView2NavigationStartingEventArgs* args)  
        -> HRESULT  
    {  
        wil::unique_cotaskmem_string uri;  
        CHECK_FAILURE(args->get_Uri(&uri));  
  
        if (ShouldBlockUri(uri.get()))  
        {  
            CHECK_FAILURE(args->put_Cancel(true));  
        }  
        return S_OK;  
    })  
    .Get(),  
    &m_frameNavigationStartingToken));
```

## add\_HistoryChanged

Add an event handler for the `HistoryChanged` event.

```
public HRESULT add_HistoryChanged(ICoreWebView2HistoryChangedEventHandler  
* eventHandler, EventRegistrationToken * token)
```

`HistoryChanged` is raised for changes to joint session history, which consists of top-level and manual frame navigations. Use `HistoryChanged` to verify that the `CanGoBack` or `CanGoForward` value has changed. `HistoryChanged` also runs for using `GoBack` or `GoForward`. `HistoryChanged` runs after `SourceChanged` and `ContentLoading`. `CanGoBack` is

false for navigations initiated through [ICoreWebView2Frame](#) APIs if there has not yet been a user gesture.

C++

```
// Register a handler for the HistoryChanged event.  
// Update the Back, Forward buttons.  
CHECK_FAILURE(m_webView->add_HistoryChanged(  
    Callback<ICoreWebView2HistoryChangedEventHandler>(  
        [this](ICoreWebView2* sender, IUnknown* args) -> HRESULT {  
            BOOL canGoBack;  
            BOOL canGoForward;  
            sender->get_CanGoBack(&canGoBack);  
            sender->get_CanGoForward(&canGoForward);  
            m_toolbar->SetItemEnabled(Toolbar::Item_BackButton,  
                canGoBack);  
            m_toolbar->SetItemEnabled(Toolbar::Item_ForwardButton,  
                canGoForward);  
  
            return S_OK;  
        })  
        .Get(),  
        &m_historyChangedToken));
```

## add\_NavigationCompleted

Add an event handler for the `NavigationCompleted` event.

```
public HRESULT  
add_NavigationCompleted(ICoreWebView2NavigationCompletedEventHandler *  
eventHandler, EventRegistrationToken * token)
```

`NavigationCompleted` runs when the WebView has completely loaded (concurrently when `body.onload` runs) or loading stopped with error.

C++

```
// Register a handler for the NavigationCompleted event.  
// Check whether the navigation succeeded, and if not, do something.  
// Also update the Cancel buttons.  
CHECK_FAILURE(m_webView->add_NavigationCompleted(  
    Callback<ICoreWebView2NavigationCompletedEventHandler>(  
        [this](ICoreWebView2* sender,  
        ICoreWebView2NavigationCompletedEventArgs* args)  
        -> HRESULT {  
            BOOL success;  
            CHECK_FAILURE(args->get_IsSuccess(&success));  
            if (!success)
```

```

    {
        COREWEBVIEW2_WEB_ERROR_STATUS webErrorStatus;
        CHECK_FAILURE(args-
>get_WebErrorStatus(&webErrorStatus));
        if (webErrorStatus ==
COREWEBVIEW2_WEB_ERROR_STATUS_DISCONNECTED)
        {
            // Do something here if you want to handle a
specific error case.
            // In most cases this isn't necessary, because the
WebView will
            // display its own error page automatically.
        }
    }
    m_toolbar->SetItemEnabled(Toolbar::Item_CancelButton,
false);
    m_toolbar->SetItemEnabled(Toolbar::Item_ReloadButton, true);
    return S_OK;
}
.Get(),
&m_navigationCompletedToken));

```

## add\_NavigationStarting

Add an event handler for the `NavigationStarting` event.

```

public HRESULT
add_NavigationStarting(ICoreWebView2NavigationStartingEventHandler *
eventHandler, EventRegistrationToken * token)

```

`NavigationStarting` runs when the WebView main frame is requesting permission to navigate to a different URI. Redirects trigger this operation as well, and the navigation id is the same as the original one.

Navigations will be blocked until all `NavigationStarting` event handlers return.

C++

```

// Register a handler for the NavigationStarting event.
// This handler will check the domain being navigated to, and if the
domain
// matches a list of blocked sites, it will cancel the navigation and
// possibly display a warning page. It will also disable JavaScript on
// selected websites.
CHECK_FAILURE(m_webView->add_NavigationStarting(
    Callback<ICoreWebView2NavigationStartingEventHandler>(
        [this](ICoreWebView2* sender,
ICoreWebView2NavigationStartingEventArgs* args)
        -> HRESULT

```

```

    {
        wil::unique_cotaskmem_string uri;
        CHECK_FAILURE(args->get_Uri(&uri));

        if (ShouldBlockUri(uri.get()))
        {
            CHECK_FAILURE(args->put_Cancel(true));

            // If the user clicked a link to navigate, show a
warning page.
            BOOL userInitiated;
            CHECK_FAILURE(args-
>get_IsUserInitiated(&userInitiated));
            static const PCWSTR htmlContent =
                L"<h1>Domain Blocked</h1>"
                L"<p>You've attempted to navigate to a domain in the
blocked "
                L"sites list. Press back to return to the previous
page.</p>";
            CHECK_FAILURE(sender->NavigateToString(htmlContent));
        }
        // Changes to settings will apply at the next navigation,
which includes the
        // navigation after a NavigationStarting event. We can use
this to change
        // settings according to what site we're visiting.
        if (ShouldBlockScriptForUri(uri.get()))
        {
            m_settings->put_IsScriptEnabled(FALSE);
        }
        else
        {
            m_settings->put_IsScriptEnabled(m_isScriptEnabled);
        }
        if (m_settings2)
        {
            static const PCWSTR url_compare_example =
L"fourthcoffee.com";
            wil::unique_bstr domain = GetDomainOfUri(uri.get());
            const wchar_t* domains = domain.get();

            if (wcscmp(url_compare_example, domains) == 0)
            {
                SetUserAgent(L"example_navigation_ua");
            }
        }
        // [NavigationKind]
        wil::com_ptr<ICoreWebView2NavigationStartingEventArgs3>
args3;
        if (SUCCEEDED(args->QueryInterface(IID_PPV_ARGS(&args3)))
{
            COREWEBVIEW2_NAVIGATION_KIND kind =
                COREWEBVIEW2_NAVIGATION_KIND_NEW_DOCUMENT;
            CHECK_FAILURE(args3->get_NavigationKind(&kind));
}

```

```
// ! [NavigationKind]
    return S_OK;
}
.Get(),
&m_navigationStartingToken);
```

## add\_NewWindowRequested

Add an event handler for the `NewWindowRequested` event.

```
public HRESULT
add_NewWindowRequested(ICoreWebView2NewWindowRequestedEventHandler *
eventHandler, EventRegistrationToken * token)
```

`NewWindowRequested` runs when content inside the WebView requests to open a new window, such as through `window.open`. The app can pass a target WebView that is considered the opened window or mark the event as `Handled`, in which case WebView2 does not open a window. If either `Handled` or `NewWindow` properties are not set, the target content will be opened on a popup window.

If a deferral is not taken on the event args, scripts that resulted in the new window that are requested are blocked until the event handler returns. If a deferral is taken, then scripts are blocked until the deferral is completed or new window is set.

For more details and considerations on the target WebView to be supplied at the opened window, see

[ICoreWebView2NewWindowRequestedEventArgs::put\\_NewWindow](#).

C++

```
// Register a handler for the NewWindowRequested event.
// This handler will defer the event, create a new app window, and then
once the
// new window is ready, it'll provide that new window's WebView as the
response to
// the request.
CHECK_FAILURE(m_webView->add_NewWindowRequested(
    Callback<ICoreWebView2NewWindowRequestedEventHandler>(
        [this](ICoreWebView2* sender,
        ICoreWebView2NewWindowRequestedEventArgs* args)
    {
        if (!m_shouldHandleNewWindowRequest)
        {
            args->put_Handled(FALSE);
            return S_OK;
        }
    }
))
```

```

        wil::com_ptr<ICoreWebView2NewWindowRequestedEventArgs>
args_as_comptr = args;
        auto args3 =
    
```

```

args_as_comptr.try_query<ICoreWebView2NewWindowRequestedEventArgs3>();
        if (args3)
    {
        wil::com_ptr<ICoreWebView2FrameInfo> frame_info;
        CHECK_FAILURE(args3-
>get_OriginalSourceFrameInfo(&frame_info));
        wil::unique_cotaskmem_string source;
        CHECK_FAILURE(frame_info->get_Source(&source));
        // The host can decide how to open based on source frame
info,
        // such as URI.
        static const wchar_t* browser_launching_domain =
L"www.example.com";
        wil::unique_bstr source_domain =
GetDomainOfUri(source.get());
        const wchar_t* source_domain_as_wchar =
source_domain.get();
        if (wcscmp(browser_launching_domain,
source_domain_as_wchar) == 0)
    {
        // Open the URI in the default browser.
        wil::unique_cotaskmem_string target_uri;
        CHECK_FAILURE(args->get_Uri(&target_uri));
        ShellExecute(
            nullptr, L"open", target_uri.get(), nullptr,
nullptr,
            SW_SHOWNORMAL);
        CHECK_FAILURE(args->put_Handled(TRUE));
        return S_OK;
    }
}

```

```

wil::com_ptr<ICoreWebView2Deferral> deferral;
CHECK_FAILURE(args->GetDeferral(&deferral));
AppWindow* newAppWindow;

wil::com_ptr<ICoreWebView2WindowFeatures> windowFeatures;
CHECK_FAILURE(args->get_WindowFeatures(&windowFeatures));

RECT windowRect = {0};
UINT32 left = 0;
UINT32 top = 0;
UINT32 height = 0;
UINT32 width = 0;
BOOL shouldHaveToolbar = true;

BOOL hasPosition = FALSE;
BOOL hasSize = FALSE;
CHECK_FAILURE(windowFeatures-
>get_HasPosition(&hasPosition));
CHECK_FAILURE(windowFeatures->get_HasSize(&hasSize));

```

```

        bool useDefaultWindow = true;

        if (!hasPosition && !hasSize)
        {
            CHECK_FAILURE(windowFeatures->get_Left(&left));
            CHECK_FAILURE(windowFeatures->get_Top(&top));
            CHECK_FAILURE(windowFeatures->get_Height(&height));
            CHECK_FAILURE(windowFeatures->get_Width(&width));
            useDefaultWindow = false;
        }
        CHECK_FAILURE(windowFeatures-
>get_ShouldDisplayToolbar(&shouldHaveToolbar));

        windowRect.left = left;
        windowRect.right =
            left + (width < s_minNewWindowSize ? s_minNewWindowSize
: width);
        windowRect.top = top;
        windowRect.bottom =
            top + (height < s_minNewWindowSize ? s_minNewWindowSize
: height);

        // passing "none" as uri as its a noinitialnavigation
        if (!useDefaultWindow)
        {
            newAppWindow = new AppWindow(
                m_creationModeId, GetWebViewOption(), L"none",
m_userDataFolder, false,
                nullptr, true, windowRect, !shouldHaveToolbar);
        }
        else
        {
            newAppWindow = new AppWindow(m_creationModeId,
GetWebViewOption(), L"none");
        }
        newAppWindow->m_isPopupWindow = true;
        newAppWindow->m_onWebViewFirstInitialized = [args, deferral,
newAppWindow]()
        {
            CHECK_FAILURE(args->put_NewWindow(newAppWindow-
>m_webView.get()));
            CHECK_FAILURE(args->put_Handled(TRUE));
            CHECK_FAILURE(deferral->Complete());
        };
        return S_OK;
    })
    .Get(),
    nullptr));

```

## add\_PermissionRequested

Add an event handler for the `PermissionRequested` event.

```
public HRESULT  
add_PermissionRequested(ICoreWebView2PermissionRequestedEventHandler *  
eventHandler, EventRegistrationToken * token)
```

`PermissionRequested` runs when content in a WebView requests permission to access some privileged resources.

If a deferral is not taken on the event args, the subsequent scripts are blocked until the event handler returns. If a deferral is taken, the scripts are blocked until the deferral is completed.

C++

```
// Register a handler for the PermissionRequested event.  
// This handler prompts the user to allow or deny the request, and  
remembers  
// the user's choice for later.  
CHECK_FAILURE(m_webView->add_PermissionRequested(  
    Callback<ICoreWebView2PermissionRequestedEventHandler>(<br>  
        this, &SettingsComponent::OnPermissionRequested)<br>  
        .Get(),<br>  
        &m_permissionRequestedToken));
```

C++

```
HRESULT SettingsComponent::OnPermissionRequested(  
    ICoreWebView2* sender, ICoreWebView2PermissionRequestedEventArgs* args)  
{  
    // Obtain a deferral for the event so that the CoreWebView2  
    // doesn't examine the properties we set on the event args until  
    // after we call the Complete method asynchronously later.  
    wil::com_ptr<ICoreWebView2Deferral> deferral;  
    CHECK_FAILURE(args->GetDeferral(&deferral));  
  
    // Do not save state to the profile so that the PermissionRequested  
    // event is  
    // always raised and the app is in control of all permission requests.  
    In  
    // this example, the app listens to all requests and caches permission  
    on  
    // its own to decide whether to show custom UI to the user.  
    wil::com_ptr<ICoreWebView2PermissionRequestedEventArgs3> extended_args;  
    CHECK_FAILURE(args->QueryInterface(IID_PPV_ARGS(&extended_args)));  
    CHECK_FAILURE(extended_args->put_SavesInProfile(FALSE));  
  
    // Do the rest asynchronously, to avoid calling MessageBox in an event  
    // handler.
```

```

m_appWindow->RunAsync(
    [this, deferral, args =
wil::com_ptr<ICoreWebView2PermissionRequestedEventArgs>(args)]
{
    wil::unique_cotaskmem_string uri;
    COREWEBVIEW2_PERMISSION_KIND kind =
COREWEBVIEW2_PERMISSION_KIND_UNKNOWN_PERMISSION;
    BOOL userInitiated = FALSE;
    CHECK_FAILURE(args->get_Uri(&uri));
    CHECK_FAILURE(args->get_PermissionKind(&kind));
    CHECK_FAILURE(args->get_IsUserInitiated(&userInitiated));

    COREWEBVIEW2_PERMISSION_STATE state =
COREWEBVIEW2_PERMISSION_STATE_DEFAULT;

        auto cached_key = std::make_tuple(std::wstring(uri.get()), kind,
userInitiated);
        auto cached_permission = m_cached_permissions.find(cached_key);
        if (cached_permission != m_cached_permissions.end())
        {
            state =
                (cached_permission->second ?
COREWEBVIEW2_PERMISSION_STATE_ALLOW
:
COREWEBVIEW2_PERMISSION_STATE_DENY);
        }
        else
        {
            std::wstring message = L"An iframe has requested device
permission for ";
            message += PermissionKindToString(kind);
            message += L" to the website at ";
            message += uri.get();
            message += L"\n\n";
            message += L"Do you want to grant permission?\n";
            message +=
                (userInitiated ? L"This request came from a user
gesture."
:
L"This request did not come from a user
gesture.");
        }

        int response = MessageBox(
            nullptr, message.c_str(), L"Permission Request",
            MB_YESNOCANCEL | MB_ICONWARNING);
        switch (response)
        {
        case IDYES:
            m_cached_permissions[cached_key] = true;
            state = COREWEBVIEW2_PERMISSION_STATE_ALLOW;
            break;
        case IDNO:
            m_cached_permissions[cached_key] = false;
            state = COREWEBVIEW2_PERMISSION_STATE_DENY;
            break;
        default:

```

```

        state = COREWEBVIEW2_PERMISSION_STATE_DEFAULT;
        break;
    }
}
CHECK_FAILURE(args->put_State(state));
CHECK_FAILURE(deferral->Complete());
});
return S_OK;
}

```

## add\_ProcessFailed

Add an event handler for the `ProcessFailed` event.

```
public HRESULT add_ProcessFailed(ICoreWebView2ProcessFailedEventHandler *  
eventHandler, EventRegistrationToken * token)
```

`ProcessFailed` runs when any of the processes in the [WebView2 Process Group](#) encounters one of the following conditions:

[+] [Expand table](#)

Condition	Details
Unexpected exit	The process indicated by the event args has exited unexpectedly (usually due to a crash). The failure might or might not be recoverable and some failures are auto-recoverable.
Unresponsiveness	The process indicated by the event args has become unresponsive to user input. This is only reported for renderer processes, and will run every few seconds until the process becomes responsive again.

### (!) Note

When the failing process is the browser process, a `ICoreWebView2Environment5::BrowserProcessExited` event will run too.

Your application can use `ICoreWebView2ProcessFailedEventArgs` and `ICoreWebView2ProcessFailedEventArgs2` to identify which condition and process the event is for, and to collect diagnostics and handle recovery if necessary. For more details about which cases need to be handled by your application, see `COREWEBVIEW2_PROCESS_FAILED_KIND`.

C++

```

// Register a handler for the ProcessFailed event.
// This handler checks the failure kind and tries to:
//   * Recreate the webview for browser failure and render unresponsive.
//   * Reload the webview for render failure.
//   * Reload the webview for frame-only render failure impacting app
content.
//   * Log information about the failure for other failures.
CHECK_FAILURE(m_webView->add_ProcessFailed(
    Callback<ICoreWebView2ProcessFailedEventHandler>(
        [this](ICoreWebView2* sender,
ICoreWebView2ProcessFailedEventArgs* argsRaw)
            -> HRESULT {
                wil::com_ptr<ICoreWebView2ProcessFailedEventArgs> args =
argsRaw;
                COREWEBVIEW2_PROCESS_FAILED_KIND kind;
                CHECK_FAILURE(args->get_ProcessFailedKind(&kind));
                if (kind ==
COREWEBVIEW2_PROCESS_FAILED_KIND_BROWSER_PROCESS_EXITED)
{
                    // Do not run a message loop from within the event
handler
                    // as that could lead to reentrancy and leave the event
                    // handler in stack indefinitely. Instead, schedule the
                    // appropriate work to take place after completion of
the
                    // event handler.
                    ScheduleReinitIfSelectedByUser(
                        L"Browser process exited unexpectedly. Recreate
webview?",
                        L"Browser process exited");
}
                else if (kind ==
COREWEBVIEW2_PROCESS_FAILED_KIND_RENDER_PROCESS_UNRESPONSIVE)
{
                    ScheduleReinitIfSelectedByUser(
                        L"Browser render process has stopped responding.
Recreate webview?",
                        L"Web page unresponsive");
}
                else if (kind ==
COREWEBVIEW2_PROCESS_FAILED_KIND_RENDER_PROCESS_EXITED)
{
                    // Reloading the page will start a new render process if
                    // needed.
                    ScheduleReloadIfSelectedByUser(
                        L"Browser render process exited unexpectedly. Reload
page?",
                        L"Render process exited");
}
                // Check the runtime event args implements the newer
interface.
                auto args2 =
args.try_query<ICoreWebView2ProcessFailedEventArgs2>();
                if (!args2)

```

```

    {
        return S_OK;
    }
    if (kind ==
COREWEBVIEW2_PROCESS_FAILED_KIND_FRAME_RENDER_PROCESS_EXITED)
{
    // A frame-only renderer has exited unexpectedly. Check
if
    // reload is needed.
    wil::com_ptr<ICoreWebView2FrameInfoCollection>
frameInfos;
    wil::com_ptr<ICoreWebView2FrameInfoCollectionIterator>
iterator;
    CHECK_FAILURE(args2-
>get_FrameInfosForFailedProcess(&frameInfos));
    CHECK_FAILURE(frameInfos->GetIterator(&iterator));

    BOOL hasCurrent = FALSE;
    while (SUCCEEDED(iterator->get_HasCurrent(&hasCurrent))
&& hasCurrent)
{
    wil::com_ptr<ICoreWebView2FrameInfo> frameInfo;
    CHECK_FAILURE(iterator->GetCurrent(&frameInfo));

    wil::unique_cotaskmem_string nameRaw;
    wil::unique_cotaskmem_string sourceRaw;
    CHECK_FAILURE(frameInfo->get_Name(&nameRaw));
    CHECK_FAILURE(frameInfo->get_Source(&sourceRaw));
    if (IsAppContentUri(sourceRaw.get()))
    {
        ScheduleReloadIfSelectedByUser(
            L"Browser render process for app frame
exited unexpectedly. "
            L"Reload page?",
            L"App content frame unresponsive");
        break;
    }

    BOOL hasNext = FALSE;
    CHECK_FAILURE(iterator->MoveNext(&hasNext));
}
}
else
{
    // Show the process failure details. Apps can collect
info for their logging
    // purposes.
    COREWEBVIEW2_PROCESS_FAILED_REASON reason;
    wil::unique_cotaskmem_string processDescription;
    int exitCode;
    wil::unique_cotaskmem_string failedModule;

    CHECK_FAILURE(args2->get_Reason(&reason));
    CHECK_FAILURE(args2-
>get_ProcessDescription(&processDescription));
}

```

```

        CHECK_FAILURE(args2->get_ExitCode(&exitCode));

        auto argFailedModule =
args.try_query<ICoreWebView2ExperimentalProcessFailedEventArgs>();
        if (argFailedModule)
{
        CHECK_FAILURE(
            argFailedModule-
>get_FailureSourceModulePath(&failedModule));
    }

        std::wstringstream message;
        message << L"Kind: " << ProcessFailedKindToString(kind)
<< L"\n"
                << L"Reason: " <<
ProcessFailedReasonToString(reason) << L"\n"
                << L"Exit code: " << exitCode << L"\n"
                << L"Process description: " <<
processDescription.get() << std::endl
                << (failedModule ? L"Failed module: " : L"")
                << (failedModule ? failedModule.get() : L"");
        m_appWindow->AsyncMessageBox( std::move(message.str()),
L"Child process failed");
    }
    return S_OK;
}
.Get(),
&m_processFailedToken);

```

## add\_ScriptDialogOpening

Add an event handler for the `ScriptDialogOpening` event.

```

public HRESULT
add_ScriptDialogOpening(ICoreWebView2ScriptDialogOpeningEventHandler *
eventHandler, EventRegistrationToken * token)

```

`ScriptDialogOpening` runs when a JavaScript dialog (`alert`, `confirm`, `prompt`, or `beforeunload`) displays for the webview. This event only triggers if the `ICoreWebView2Settings::AreDefaultScriptDialogsEnabled` property is set to `FALSE`. The `ScriptDialogOpening` event suppresses dialogs or replaces default dialogs with custom dialogs.

If a deferral is not taken on the event args, the subsequent scripts are blocked until the event handler returns. If a deferral is taken, the scripts are blocked until the deferral is completed.

C++

```
// Register a handler for the ScriptDialogOpening event.  
// This handler will set up a custom prompt dialog for the user.  
Because  
    // running a message loop inside of an event handler causes problems, we  
    // defer the event and handle it asynchronously.  
    CHECK_FAILURE(m_webView->add_ScriptDialogOpening(  
        Callback<ICoreWebView2ScriptDialogOpeningEventHandler>(  
            [this](ICoreWebView2* sender,  
ICoreWebView2ScriptDialogOpeningEventArgs* args)  
            -> HRESULT  
            {  
                AppWindow* appWindow = m_appWindow;  
                wil::com_ptr<ICoreWebView2ScriptDialogOpeningEventArgs>  
eventArgs = args;  
                wil::com_ptr<ICoreWebView2Deferral> deferral;  
                CHECK_FAILURE(args->GetDeferral(&deferral));  
                appWindow->RunAsync(  
                    [appWindow, eventArgs, deferral]  
                    {  
                        wil::unique_cotaskmem_string uri;  
                        COREWEBVIEW2_SCRIPT_DIALOG_KIND type;  
                        wil::unique_cotaskmem_string message;  
                        wil::unique_cotaskmem_string defaultText;  
  
                        CHECK_FAILURE(eventArgs->get_Uri(&uri));  
                        CHECK_FAILURE(eventArgs->get_Kind(&type));  
                        CHECK_FAILURE(eventArgs->get_Message(&message));  
                        CHECK_FAILURE(eventArgs->  
get_DefaultText(&defaultText));  
  
                        std::wstring promptString =  
                            std::wstring(L"The page at '") + uri.get() + L"'  
says:";  
                        TextInputDialog dialog(  
                            appWindow->GetMainWindow(), L"Script Dialog",  
promptString.c_str(),  
                            message.get(), defaultText.get(),  
                            /* readonly */ type !=  
COREWEBVIEW2_SCRIPT_DIALOG_KIND_PROMPT);  
                        if (dialog.confirmed)  
                        {  
                            CHECK_FAILURE(eventArgs->  
put_ResultText(dialog.input.c_str()));  
                            CHECK_FAILURE(eventArgs->Accept());  
                        }  
                        deferral->Complete();  
                    });  
                return S_OK;  
            })  
            .Get(),  
            &m_scriptDialogOpeningToken));
```

## add\_SourceChanged

Add an event handler for the `SourceChanged` event.

```
public HRESULT add_SourceChanged(ICoreWebView2SourceChangedEventArgs *  
eventHandler, EventRegistrationToken * token)
```

`SourceChanged` triggers when the `Source` property changes. `SourceChanged` runs when navigating to a different site or fragment navigations. It does not trigger for other types of navigations such as page refreshes or `history.pushState` with the same URL as the current page. `SourceChanged` runs before `ContentLoading` for navigation to a new document.

C++

```
// Register a handler for the SourceChanged event.  
// This handler will read the webview's source URI and update  
// the app's address bar.  
CHECK_FAILURE(m_webView->add_SourceChanged(  
    Callback<ICoreWebView2SourceChangedEventArgs>(  
        [this](ICoreWebView2* sender,  
        ICoreWebView2SourceChangedEventArgs* args)  
            -> HRESULT {  
                wil::unique_cotaskmem_string uri;  
                sender->get_Source(&uri);  
                if (wcscmp(uri.get(), L"about:blank") == 0)  
                {  
                    uri = wil::make_cotaskmem_string(L"");  
                }  
                SetWindowText(GetAddressBar(), uri.get());  
  
                return S_OK;  
            })  
        .Get(),  
    &m_sourceChangedToken));
```

## add\_WebMessageReceived

Add an event handler for the `WebMessageReceived` event.

```
public HRESULT  
add_WebMessageReceived(ICoreWebView2WebMessageReceivedEventHandler *  
handler, EventRegistrationToken * token)
```

`WebMessageReceived` runs when the `ICoreWebView2Settings::IsWebMessageEnabled` setting is set and the top-level document of the WebView runs `window.chrome.webview.postMessage`. The `postMessage` function is `void postMessage(object)` where object is any object supported by JSON conversion.

#### HTML

```
window.chrome.webview.addEventListener('message', arg => {
    if ("SetColor" in arg.data) {
        document.getElementById("colorable").style.color =
arg.data.SetColor;
    }
    if ("WindowBounds" in arg.data) {
        document.getElementById("window-bounds").value =
arg.data.WindowBounds;
    }
});

function SetTitleText() {
    let titleText = document.getElementById("title-text");
    window.chrome.webview.postMessage(`SetTitleText
${titleText.value}`);
}
function GetWindowBounds() {
    window.chrome.webview.postMessage("GetWindowBounds");
}
```

When the page calls `postMessage`, the object parameter is converted to a JSON string and is posted asynchronously to the host process. This will result in the handler's `Invoke` method being called with the JSON string as a parameter.

#### C++

```
// Setup the web message received event handler before navigating to
// ensure we don't miss any messages.
CHECK_FAILURE(m_webView->add_WebMessageReceived(
    Microsoft::WRL::Callback<ICoreWebView2WebMessageReceivedEventHandler>(
        [this](ICoreWebView2* sender,
        ICoreWebView2WebMessageReceivedEventArgs* args)
    {
        wil::unique_cotaskmem_string uri;
        CHECK_FAILURE(args->get_Source(&uri));

        // Always validate that the origin of the message is what you
expect.
        if (uri.get() != m_sampleUri)
        {
            // Ignore messages from untrusted sources.
            return S_OK;
        }
    }
));
```

```

    }

    wil::unique_cotaskmem_string messageRaw;
    HRESULT hr = args->TryGetWebMessageAsString(&messageRaw);
    if (hr == E_INVALIDARG)
    {
        // Was not a string message. Ignore.
        return S_OK;
    }
    // Any other problems are fatal.
    CHECK_FAILURE(hr);
    std::wstring message = messageRaw.get();

    if (message.compare(0, 13, L"SetTitleText ") == 0)
    {
        m_appWindow->SetDocumentTitle(message.substr(13).c_str());
    }
    else if (message.compare(L"GetWindowBounds") == 0)
    {
        RECT bounds = m_appWindow->GetWindowBounds();
        std::wstring reply =
            L"\"WindowBounds\":" + std::to_wstring(bounds.left)
            + L"\nTop:" + std::to_wstring(bounds.top)
            + L"\nRight:" + std::to_wstring(bounds.right)
            + L"\nBottom:" + std::to_wstring(bounds.bottom)
            + L"\}";
        CHECK_FAILURE(sender->PostWebMessageAsJson(reply.c_str()));
    }
    else
    {
        // Ignore unrecognized messages, but log for further
        investigation
        // since it suggests a mismatch between the web content and the
        host.
        OutputDebugString(
            std::wstring(L"Unexpected message from main page:" +
message).c_str());
    }
    return S_OK;
}).Get(), &m_webMessageReceivedToken));

```

If the same page calls `postMessage` multiple times, the corresponding `WebMessageReceived` events are guaranteed to be fired in the same order. However, if multiple frames call `postMessage`, there is no guaranteed order. In addition, `WebMessageReceived` events caused by calls to `postMessage` are not guaranteed to be sequenced with events caused by DOM APIs. For example, if the page runs

JavaScript

```

chrome.webview.postMessage("message");
window.open();

```

then the `NewWindowRequested` event might be fired before the `WebMessageReceived` event. If you need the `WebMessageReceived` event to happen before anything else, then in the `WebMessageReceived` handler you can post a message back to the page and have the page wait until it receives that message before continuing.

## add\_WebResourceRequested

Add an event handler for the `WebResourceRequested` event.

```
public HRESULT  
add_WebResourceRequested(ICoreWebView2WebResourceRequestedEventHandler  
* eventHandler, EventRegistrationToken * token)
```

`WebResourceRequested` runs when the WebView is performing a URL request to a matching URL and resource context filter that was added with `AddWebResourceRequestedFilter`. At least one filter must be added for the event to run.

The web resource requested may be blocked until the event handler returns if a deferral is not taken on the event args. If a deferral is taken, then the web resource requested is blocked until the deferral is completed.

If this event is subscribed in the `add_NewWindowRequested` handler it should be called after the new window is set. For more details see [ICoreWebView2NewWindowRequestedEventArgs::put\\_NewWindow](#).

This event is by default raised for file, http, and https URI schemes. This is also raised for registered custom URI schemes. For more details see [ICoreWebView2CustomSchemeRegistration](#).

C++

```
if (m_blockImages)  
{  
    m_webView->AddWebResourceRequestedFilter(  
        L"*", COREWEBVIEW2_WEB_RESOURCE_CONTEXT_IMAGE);  
    CHECK_FAILURE(m_webView->add_WebResourceRequested(  
        Callback<ICoreWebView2WebResourceRequestedEventHandler>(  
            [this](  
                ICoreWebView2* sender,  
                ICoreWebView2WebResourceRequestedEventArgs* args)  
            {  
                COREWEBVIEW2_WEB_RESOURCE_CONTEXT resourceContext;  
                CHECK_FAILURE(args->  
                    get_ResourceContext(&resourceContext));  
                // Ensure that the type is image
```

```

        if (resourceContext != COREWEBVIEW2_WEB_RESOURCE_CONTEXT_IMAGE)
        {
            return E_INVALIDARG;
        }
        // Override the response with an empty one to block
the image.
        // If put_Response is not called, the request will
        // continue as normal.
        wil::com_ptr<ICoreWebView2WebResourceResponse>
response;
        wil::com_ptr<ICoreWebView2Environment> environment;
        wil::com_ptr<ICoreWebView2_2> webview2;
        CHECK_FAILURE(m_webView-
>QueryInterface(IID_PPV_ARGS(&webview2)));
        CHECK_FAILURE(webview2-
>get_Environment(&environment));
        CHECK_FAILURE(environment-
>CreateWebResourceResponse(
            nullptr, 403 /*NoContent*/, L"Blocked",
L"Content-Type: image/jpeg",
            &response));
        CHECK_FAILURE(args->put_Response(response.get()));
        return S_OK;
    })
    .Get(),
    &m_webResourceRequestedTokenForImageBlocking));
}
else
{
    CHECK_FAILURE(m_webView->remove_WebResourceRequested(
        m_webResourceRequestedTokenForImageBlocking));
}

```

C++

```

if (m_replaceImages)
{
    m_webView->AddWebResourceRequestedFilter(
        L"*", COREWEBVIEW2_WEB_RESOURCE_CONTEXT_IMAGE);
    CHECK_FAILURE(m_webView->add_WebResourceRequested(
        Callback<ICoreWebView2WebResourceRequestedEventHandler>(
            [this](
                ICoreWebView2* sender,
                ICoreWebView2WebResourceRequestedEventArgs* args)
        {
            COREWEBVIEW2_WEB_RESOURCE_CONTEXT resourceContext;
            CHECK_FAILURE(args-
>get_ResourceContext(&resourceContext));
            // Ensure that the type is image
            if (resourceContext != COREWEBVIEW2_WEB_RESOURCE_CONTEXT_IMAGE)
            {

```

```

        return E_INVALIDARG;
    }
    // Override the response with an another image.
    // If put_Response is not called, the request will
    // continue as normal.
    // It's not required for this scenario, but
generally you should examine
    // relevant HTTP request headers just like an HTTP
server would do when
    // producing a response stream.
wil::com_ptr<IStream> stream;
CHECK_FAILURE(SHCreateStreamOnFileEx(
    L"assets/EdgeWebView2-80.jpg", STGM_READ,
FILE_ATTRIBUTE_NORMAL,
    FALSE, nullptr, &stream));
wil::com_ptr<ICoreWebView2WebResourceResponse>
response;
    wil::com_ptr<ICoreWebView2Environment> environment;
    wil::com_ptr<ICoreWebView2_2> webview2;
    CHECK_FAILURE(m_webView-
>QueryInterface(IID_PPV_ARGS(&webview2)));
    CHECK_FAILURE(webview2-
>get_Environment(&environment));
    CHECK_FAILURE(environment-
>CreateWebResourceResponse(
        stream.get(), 200, L"OK", L"Content-Type:
image/jpeg", &response));
    CHECK_FAILURE(args->put_Response(response.get()));
    return S_OK;
})
.Get(),
&m_webResourceRequestedTokenForImageReplacing));
}
else
{
    CHECK_FAILURE(m_webView->remove_WebResourceRequested(
        m_webResourceRequestedTokenForImageReplacing));
}

```

## add\_WindowCloseRequested

Add an event handler for the `WindowCloseRequested` event.

```

public HRESULT
add_WindowCloseRequested(ICoreWebView2WindowCloseRequestedEventHandler
* eventHandler, EventRegistrationToken * token)

```

`WindowCloseRequested` triggers when content inside the WebView requested to close the window, such as after `window.close` is run. The app should close the WebView and

related app window if that makes sense to the app.

C++

```
// Register a handler for the WindowCloseRequested event.  
// This handler will close the app window if it is not the main window.  
CHECK_FAILURE(m_webView->add_WindowCloseRequested(  
    Callback<ICoreWebView2WindowCloseRequestedEventHandler>(  
        [this](ICoreWebView2* sender, IUnknown* args)  
    {  
        if (m_isPopupWindow)  
        {  
            CloseAppWindow();  
        }  
        return S_OK;  
    })  
    .Get(),  
    nullptr));
```

## AddHostObjectToScript

Add the provided host object to script running in the WebView with the specified name.

```
public HRESULT AddHostObjectToScript(LPCWSTR name, VARIANT * object)
```

Host objects are exposed as host object proxies using

`window.chrome.webview.hostObjects.{name}`. Host object proxies are promises and resolves to an object representing the host object. The promise is rejected if the app has not added an object with the name. When JavaScript code access a property or method of the object, a promise is return, which resolves to the value returned from the host for the property or method, or rejected in case of error, for example, no property or method on the object or parameters are not valid.

### ⓘ Note

While simple types, `IDispatch` and array are supported, and `IUnknown` objects that also implement `IDispatch` are treated as `IDispatch`, generic `IUnknown`, `VT_DECIMAL`, or `VT_RECORD` variant is not supported. Remote JavaScript objects like callback functions are represented as an `VT_DISPATCH` `VARIANT` with the object implementing `IDispatch`. The JavaScript callback method may be invoked using `DISPID_VALUE` for the `DISPID`. Such callback method invocations will return immediately and will not wait for the JavaScript function to run and so will not provide the return value of the JavaScript function. Nested arrays are supported up

to a depth of 3. Arrays of by reference types are not supported. `VT_EMPTY` and `VT_NULL` are mapped into JavaScript as `null`. In JavaScript, `null` and `undefined` are mapped to `VT_EMPTY`.

Additionally, all host objects are exposed as `window.chrome.webview.hostObjects.sync.{name}`. Here the host objects are exposed as synchronous host object proxies. These are not promises and function runtimes or property access synchronously block running script waiting to communicate cross process for the host code to run. Accordingly the result may have reliability issues and it is recommended that you use the promise-based asynchronous `window.chrome.webview.hostObjects.{name}` API.

Synchronous host object proxies and asynchronous host object proxies may both use a proxy to the same host object. Remote changes made by one proxy propagates to any other proxy of that same host object whether the other proxies are synchronous or asynchronous.

While JavaScript is blocked on a synchronous run to native code, that native code is unable to run back to JavaScript. Attempts to do so fail with

`HRESULT_FROM_WIN32(ERROR_POSSIBLE_DEADLOCK)`.

Host object proxies are JavaScript Proxy objects that intercept all property get, property set, and method invocations. Properties or methods that are a part of the Function or Object prototype are run locally. Additionally any property or method in the `chrome.webview.hostObjects.options.forceLocalProperties` array are also run locally. This defaults to including optional methods that have meaning in JavaScript like `toJSON` and `Symbol.toPrimitive`. Add more to the array as required.

The `chrome.webview.hostObjects.cleanupSome` method performs a best effort garbage collection on host object proxies.

The `chrome.webview.hostObjects.options` object provides the ability to change some functionality of host objects.

[+] Expand table

Options property	Details
<code>forceLocalProperties</code>	This is an array of host object property names that will be run locally, instead of being called on the native host object. This defaults to <code>then</code> , <code>toJSON</code> , <code>Symbol.toString</code> , and <code>Symbol.toPrimitive</code> . You can add other properties to specify that they should be run locally on the javascript host object proxy.

Options property	Details
<code>log</code>	This is a callback that will be called with debug information. For example, you can set this to <code>console.log.bind(console)</code> to have it print debug information to the console to help when troubleshooting host object usage. By default this is null.
<code>shouldSerializeDates</code>	By default this is false, and javascript Date objects will be sent to host objects as a string using <code>JSON.stringify</code> . You can set this property to true to have Date objects properly serialize as a <code>VT_DATE</code> when sending to the native host object, and have <code>VT_DATE</code> properties and return values create a javascript Date object.
<code>defaultSyncProxy</code>	When calling a method on a synchronous proxy, the result should also be a synchronous proxy. But in some cases, the sync/async context is lost (for example, when providing to native code a reference to a function, and then calling that function in native code). In these cases, the proxy will be asynchronous, unless this property is set.
<code>forceAsyncMethodMatches</code>	This is an array of regular expressions. When calling a method on a synchronous proxy, the method call will be performed asynchronously if the method name matches a string or regular expression in this array. Setting this value to <code>Async</code> will make any method that ends with <code>Async</code> be an asynchronous method call. If an <code>async</code> method doesn't match here and isn't forced to be asynchronous, the method will be invoked synchronously, blocking execution of the calling JavaScript and then returning the resolution of the promise, rather than returning a promise.
<code>ignoreMemberNotFoundError</code>	By default, an exception is thrown when attempting to get the value of a proxy property that doesn't exist on the corresponding native class. Setting this property to <code>true</code> switches the behavior to match Chakra WinRT projection (and general JavaScript) behavior of returning <code>undefined</code> with no error.

Host object proxies additionally have the following methods which run locally.

[ ] Expand table

Method name	Details
<code>applyHostFunction</code> , <code>getHostProperty</code> , <code>setHostProperty</code>	Perform a method invocation, property get, or property set on the host object. Use the methods to explicitly force a method or property to run remotely if a conflicting local method or property exists. For instance, <code>proxy.toString()</code> runs the local <code>toString</code> method on the proxy object. But <code>proxy.applyHostFunction('toString')</code> runs <code>toString</code> on the host proxied object instead.

Method name	Details
<code>getLocalProperty</code> , <code>setLocalProperty</code>	Perform property get, or property set locally. Use the methods to force getting or setting a property on the host object proxy rather than on the host object it represents. For instance, <code>proxy.unknownProperty</code> gets the property named <code>unknownProperty</code> from the host proxied object. But <code>proxy.getLocalProperty('unknownProperty')</code> gets the value of the property <code>unknownProperty</code> on the proxy object.
<code>sync</code>	Asynchronous host object proxies expose a sync method which returns a promise for a synchronous host object proxy for the same host object. For example, <code>chrome.webview.hostObjects.sample.methodCall()</code> returns an asynchronous host object proxy. Use the <code>sync</code> method to obtain a synchronous host object proxy instead: <code>const syncProxy = await chrome.webview.hostObjects.sample.methodCall().sync()</code> .
<code>async</code>	Synchronous host object proxies expose an async method which blocks and returns an asynchronous host object proxy for the same host object. For example, <code>chrome.webview.hostObjects.sync.sample.methodCall()</code> returns a synchronous host object proxy. Running the <code>async</code> method on this blocks and then returns an asynchronous host object proxy for the same host object: <code>const asyncProxy = chrome.webview.hostObjects.sync.sample.methodCall().async()</code> .
<code>then</code>	Asynchronous host object proxies have a <code>then</code> method. Allows proxies to be awaitable. <code>then</code> returns a promise that resolves with a representation of the host object. If the proxy represents a JavaScript literal, a copy of that is returned locally. If the proxy represents a function, a non-awaitable proxy is returned. If the proxy represents a JavaScript object with a mix of literal properties and function properties, the a copy of the object is returned with some properties as host object proxies.

All other property and method invocations (other than the above Remote object proxy methods, `forceLocalProperties` list, and properties on Function and Object prototypes) are run remotely. Asynchronous host object proxies return a promise representing asynchronous completion of remotely invoking the method, or getting the property. The promise resolves after the remote operations complete and the promises resolve to the resulting value of the operation. Synchronous host object proxies work similarly, but block running JavaScript and wait for the remote operation to complete.

Setting a property on an asynchronous host object proxy works slightly differently. The set returns immediately and the return value is the value that is set. This is a requirement of the JavaScript Proxy object. If you need to asynchronously wait for the property set to complete, use the `setHostProperty` method which returns a promise as described above. Synchronous object property set property synchronously blocks until the property is set.

For example, suppose you have a COM object with the following interface.

idl

```
[uuid(3a14c9c0-bc3e-453f-a314-4ce4a0ec81d8), object, local]
interface IHostObjectSample : IUnknown
{
    // Demonstrate basic method call with some parameters and a return
    value.
    HRESULT MethodWithParametersAndReturnValue([in] BSTR
stringParameter, [in] INT integerParameter, [out, retval] BSTR*
stringResult);

    // Demonstrate getting and setting a property.
    [propget] HRESULT Property([out, retval] BSTR* stringResult);
    [propput] HRESULT Property([in] BSTR stringValue);

    [propget] HRESULT IndexedProperty(INT index, [out, retval] BSTR *
stringResult);
    [propput] HRESULT IndexedProperty(INT index, [in] BSTR stringValue);

    // Demonstrate native calling back into JavaScript.
    HRESULT CallCallbackAsynchronously([in] IDispatch*
callbackParameter);

    // Demonstrate a property which uses Date types
    [propget] HRESULT DateProperty([out, retval] DATE * dateResult);
    [propput] HRESULT DateProperty([in] DATE dateValue);

    // Creates a date object on the native side and sets the
    DateProperty to it.
    HRESULT CreateNativeDate();

};

}
```

Add an instance of this interface into your JavaScript with `AddHostObjectToScript`. In this case, name it `sample`.

C++

```
VARIANT remoteObjectAsVariant = {};
m_hostObject.query_to<IDispatch>
(&remoteObjectAsVariant.pdispVal);
remoteObjectAsVariant.vt = VT_DISPATCH;

// We can call AddHostObjectToScript multiple times in a row
without
// calling RemoveHostObject first. This will replace the
previous object
// with the new object. In our case this is the same object and
everything
// is fine.
CHECK_FAILURE(
    m_webView->AddHostObjectToScript(L"sample",
```

```
&remoteObjectAsVariant));
    remoteObjectAsVariant.pdispVal->Release();
```

In the HTML document, use the COM object using `chrome.webview.hostObjects.sample`. Note that `CoreWebView2.AddHostObjectToScript` only applies to the top-level document and not to frames. To add host objects to frames use `CoreWebView2Frame.AddHostObjectToScript`.

#### HTML

```
document.getElementById("getPropertyAsyncButton").addEventListener("click",
async () => {
    const PropertyValue = await
chrome.webview.hostObjects.sample.property;
    document.getElementById("getPropertyAsyncOutput").textContent =
PropertyValue;
});

document.getElementById("getPropertySyncButton").addEventListener("click",
() => {
    const PropertyValue =
chrome.webview.hostObjects.sync.sample.property;
    document.getElementById("getPropertySyncOutput").textContent =
PropertyValue;
});

document.getElementById("setPropertyAsyncButton").addEventListener("click",
async () => {
    const PropertyValue =
document.getElementById("setPropertyAsyncInput").value;
        // The following line will work but it will return immediately
before the property value has actually been set.
        // If you need to set the property and wait for the property to
change value, use the setHostProperty function.
    chrome.webview.hostObjects.sample.property = PropertyValue;
    document.getElementById("setPropertyAsyncOutput").textContent =
"Set";
});

document.getElementById("setPropertyExplicitAsyncButton").addEventListener("click",
async () => {
    const PropertyValue =
document.getElementById("setPropertyExplicitAsyncInput").value;
        // If you care about waiting until the property has actually
changed value use the setHostProperty function.
    await
chrome.webview.hostObjects.sample.setHostProperty("property",
PropertyValue);
```

```
document.getElementById("setPropertyExplicitAsyncOutput").textContent =
"Set";
});

document.getElementById("setPropertySyncButton").addEventListener("click",
() => {
    const propertyValue =
document.getElementById("setPropertySyncInput").value;
    chrome.webview.hostObjects.sync.sample.property = propertyValue;
    document.getElementById("setPropertySyncOutput").textContent =
"Set";
});

document.getElementById("getIndexedPropertyAsyncButton").addEventListener("c
lick", async () => {
    const index =
parseInt(document.getElementById("getIndexedPropertyAsyncParam").value);
    const resultValue = await
chrome.webview.hostObjects.sample.IndexedProperty[index];

document.getElementById("getIndexedPropertyAsyncOutput").textContent =
resultValue;
});

document.getElementById("setIndexedPropertyAsyncButton").addEventListener("c
lick", async () => {
    const index =
parseInt(document.getElementById("setIndexedPropertyAsyncParam1").value);
    const value =
document.getElementById("setIndexedPropertyAsyncParam2").value;;
    chrome.webview.hostObjects.sample.IndexedProperty[index] =
value;

document.getElementById("setIndexedPropertyAsyncOutput").textContent =
"Set";
});

document.getElementById("invokeMethodAsyncButton").addEventListener("click",
async () => {
    const paramValue1 =
document.getElementById("invokeMethodAsyncParam1").value;
    const paramValue2 =
parseInt(document.getElementById("invokeMethodAsyncParam2").value);
    const resultValue = await
chrome.webview.hostObjects.sample.MethodWithParametersAndReturnValue(paramVa
lue1, paramValue2);
    document.getElementById("invokeMethodAsyncOutput").textContent =
resultValue;
});

document.getElementById("invokeMethodSyncButton").addEventListener("click",
```

```

() => {
    const paramValue1 =
document.getElementById("invokeMethodSyncParam1").value;
    const paramValue2 =
parseInt(document.getElementById("invokeMethodSyncParam2").value);
    const resultValue =
chrome.webview.hostObjects.sync.sample.MethodWithParametersAndReturnValue(pa
ralue1, paramValue2);
    document.getElementById("invokeMethodSyncOutput").textContent =
resultValue;
};

let callbackCount = 0;

document.getElementById("invokeCallbackButton").addEventListener("click",
async () => {
    chrome.webview.hostObjects.sample.CallCallbackAsynchronously(() => {
        document.getElementById("invokeCallbackOutput").textContent =
"Native object called the callback " + (++callbackCount) + " time(s).";
    });
});

// Date property
document.getElementById("setDateButton").addEventListener("click",
() => {
    chrome.webview.hostObjects.options.shouldSerializeDates = true;
    chrome.webview.hostObjects.sync.sample.dateProperty = new
Date();
    document.getElementById("dateOutput").textContent =
"sample.dateProperty: " +
chrome.webview.hostObjects.sync.sample.dateProperty;
});

document.getElementById("createRemoteDateButton").addEventListener("click",
() => {
    chrome.webview.hostObjects.sync.sample.createNativeDate();
    document.getElementById("dateOutput").textContent =
"sample.dateProperty: " +
chrome.webview.hostObjects.sync.sample.dateProperty;
});

```

Exposing host objects to script has security risk. For more information about best practices, navigate to [Best practices for developing secure WebView2 applications](#).

## AddScriptToExecuteOnDocumentCreated

Add the provided JavaScript to a list of scripts that should be run after the global object has been created, but before the HTML document has been parsed and before any other script included by the HTML document is run.

```
public HRESULT AddScriptToExecuteOnDocumentCreated(LPCWSTR javaScript,  
ICoreWebView2AddScriptToExecuteOnDocumentCreatedCompletedHandler *  
handler)
```

This method injects a script that runs on all top-level document and child frame page navigations. This method runs asynchronously, and you must wait for the completion handler to finish before the injected script is ready to run. When this method completes, the `Invoke` method of the handler is run with the `id` of the injected script. `id` is a string. To remove the injected script, use `RemoveScriptToExecuteOnDocumentCreated`.

If the method is run in `add_NewWindowRequested` handler it should be called before the new window is set. If called after setting the `NewWindow` property, the initial script may or may not apply to the initial navigation and may only apply to the subsequent navigation. For more details see

`ICoreWebView2NewWindowRequestedEventArgs::put_NewWindow`.

#### ⓘ Note

If an HTML document is running in a sandbox of some kind using `sandbox` properties or the `Content-Security-Policy` HTTP header affects the script that runs. For example, if the `allow-modals` keyword is not set then requests to run the `alert` function are ignored.

C++

```
// Prompt the user for some script and register it to execute whenever a new  
// page loads.  
void ScriptComponent::AddInitializeScript()  
{  
    TextInputDialog dialog(  
        m_appWindow->GetMainWindow(),  
        L"Add Initialize Script",  
        L"Initialization Script:",  
        L"Enter the JavaScript code to run as the initialization script that  
"  
        L"runs before any script in the HTML document.",  
        // This example script stops child frames from opening new windows.  
        Because  
        // the initialization script runs before any script in the HTML  
        // document, we  
        // can trust the results of our checks on window.parent and window.top.  
        L;if (window.parent !== window.top) {\r\n            L"    delete window.open;\r\n        L"});  
        if (dialog.confirmed)  
    {
```

```
m_webView->AddScriptToExecuteOnDocumentCreated(
    dialog.input.c_str(),

Callback<ICoreWebView2AddScriptToExecuteOnDocumentCreatedCompletedHandler>(
    [this](HRESULT error, PCWSTR id) -> HRESULT
{
    m_lastInitializescriptId = id;
    m_appWindow->AsyncMessageBox(
        m_lastInitializescriptId,
L"AddScriptToExecuteOnDocumentCreated Id");
    return S_OK;
}).Get()));

}

}
```

## AddWebResourceRequestedFilter

Adds a URI and resource context filter for the `WebResourceRequested` event.

```
public HRESULT AddWebResourceRequestedFilter(LPCWSTR const uri,
COREWEBVIEW2_WEB_RESOURCE_CONTEXT const resourceContext)
```

A web resource request with a resource context that matches this filter's resource context and a URI that matches this filter's URI wildcard string will be raised via the `WebResourceRequested` event.

The `uri` parameter value is a wildcard string matched against the URI of the web resource request. This is a glob style wildcard string in which a `*` matches zero or more characters and a `?` matches exactly one character. These wildcard characters can be escaped using a backslash just before the wildcard character in order to represent the literal `*` or `?`.

The matching occurs over the URI as a whole string and not limiting wildcard matches to particular parts of the URI. The wildcard filter is compared to the URI after the URI has been normalized, any URI fragment has been removed, and non-ASCII hostnames have been converted to punycode.

Specifying a `nullptr` for the `uri` is equivalent to an empty string which matches no URIs.

For more information about resource context filters, navigate to [COREWEBVIEW2\\_WEB\\_RESOURCE\\_CONTEXT](#).

[+] Expand table

URI Filter String	Request URI	Match	Notes
*	https://contoso.com/a/b/c	Yes	A single * will match all URLs
*://contoso.com/*	https://contoso.com/a/b/c	Yes	Matches everything in contoso.com across all schemes
*://contoso.com/*	https://example.com/? https://contoso.com/	Yes	But also matches a URI with just the same text anywhere in the URI
example	https://contoso.com/example	No	The filter does not perform partial matches
*example	https://contoso.com/example	Yes	The filter matches across URI parts
*example	https://contoso.com/path/? example	Yes	The filter matches across URI parts
*example	https://contoso.com/path/? query#example	No	The filter is matched against the URI with no fragment
*example	https://example	No	The URI is normalized before filter matching so the actual URI used for comparison is <a href="https://example/">https://example/</a>
*example/	https://example	Yes	Just like above, but this time the filter ends with a / just like the normalized URI
https://xn--qei.example/	https://xn--qei.example/	Yes	Non-ASCII hostnames are normalized to punycode before wildcard comparison
https://xn--qei.example/	https://xn--qei.example/	No	Non-ASCII hostnames are normalized to punycode before wildcard comparison

## CallDevToolsProtocolMethod

Runs an asynchronous `DevToolsProtocol` method.

```
public HRESULT CallDevToolsProtocolMethod(LPCWSTR methodName, LPCWSTR
parametersAsJson, ICoreWebView2CallDevToolsProtocolMethodCompletedHandler
* handler)
```

For more information about available methods, navigate to [DevTools Protocol Viewer](#). The `methodName` parameter is the full name of the method in the `{domain}.{method}` format. The `parametersAsJson` parameter is a JSON formatted string containing the parameters for the corresponding method. The `Invoke` method of the `handler` is run when the method asynchronously completes. `Invoke` is run with the return object of the method as a JSON string. This function returns `E_INVALIDARG` if the `methodName` is unknown or the `parametersAsJson` has an error. In the case of such an error, the `returnObjectAsJson` parameter of the handler will include information about the error. Note even though WebView2 dispatches the CDP messages in the order called, CDP method calls may be processed out of order. If you require CDP methods to run in a particular order, you should wait for the previous method's completed handler to run before calling the next method.

C++

```
// Prompt the user for the name and parameters of a CDP method, then call
it.
void ScriptComponent::CallCdpMethod()
{
    TextInputDialog dialog(
        m_appWindow->GetMainWindow(),
        L"Call CDP Method",
        L"CDP method name:",
        L"Enter the CDP method name to call, followed by a space,\r\n"
        L"followed by the parameters in JSON format.",
        L"Runtime.evaluate {\"expression\":\"\\\"alert(\\\\\"test\\\\\")\\\"}");
    if (dialog.confirmed)
    {
        size_t delimiterPos = dialog.input.find(L' ');
        std::wstring methodName = dialog.input.substr(0, delimiterPos);
        std::wstring methodParams =
            (delimiterPos < dialog.input.size())
            ? dialog.input.substr(delimiterPos + 1)
            : L"{}";

        m_webView->CallDevToolsProtocolMethod(
            methodName.c_str(), methodParams.c_str(),

Callback<ICoreWebView2CallDevToolsProtocolMethodCompletedHandler>(
            this, &ScriptComponent::CDPMethodCallback)
            .Get());
    }
}
```

## CapturePreview

Capture an image of what WebView is displaying.

```
public HRESULT  
CapturePreview(COREWEBVIEW2_CAPTURE_PREVIEW_IMAGE_FORMAT  
imageFormat, IStream * imageStream,  
ICoreWebView2CapturePreviewCompletedHandler * handler)
```

Specify the format of the image with the `imageFormat` parameter. The resulting image binary data is written to the provided `imageStream` parameter. When `CapturePreview` finishes writing to the stream, the `Invoke` method on the provided `handler` parameter is run. This method fails if called before the first ContentLoading event. For example if this is called in the NavigationStarting event for the first navigation it will fail. For subsequent navigations, the method may not fail, but will not capture an image of a given webpage until the ContentLoading event has been fired for it. Any call to this method prior to that will result in a capture of the page being navigated away from.

C++

```
// Show the user a file selection dialog, then save a screenshot of the  
// WebView  
// to the selected file.  
void FileComponent::SaveScreenshot()  
{  
    WCHAR defaultName[MAX_PATH] = L"WebView2_Screenshot.png";  
    OPENFILENAME openFileName = CreateOpenFileName(defaultName, L"PNG  
File\0*.png\0");  
    if (GetSaveFileName(&openFileName))  
    {  
        wil::com_ptr<IStream> stream;  
        CHECK_FAILURE(SHCreateStreamOnFileEx(  
            defaultName, STGM_READWRITE | STGM_CREATE,  
            FILE_ATTRIBUTE_NORMAL, TRUE, nullptr,  
            &stream));  
  
        CHECK_FAILURE(m_webView->CapturePreview(  
            COREWEBVIEW2_CAPTURE_PREVIEW_IMAGE_FORMAT_PNG, stream.get(),  
            Callback<ICoreWebView2CapturePreviewCompletedHandler>(  
                [appWindow{m_appWindow}](HRESULT error_code) -> HRESULT {  
                    CHECK_FAILURE(error_code);  
                    appWindow->AsyncMessageBox(L"Preview Captured",  
L"Preview Captured");  
                    return S_OK;  
                })  
            .Get()));  
    }  
}
```

## ExecuteScript

Run JavaScript code from the javascript parameter in the current top-level document rendered in the WebView.

```
public HRESULT ExecuteScript(LPCWSTR javaScript,  
ICoreWebView2ExecuteScriptCompletedHandler * handler)
```

The result of evaluating the provided JavaScript is used in this parameter. The result value is a JSON encoded string. If the result is undefined, contains a reference cycle, or otherwise is not able to be encoded into JSON, then the result is considered to be null, which is encoded in JSON as the string "null".

### ① Note

A function that has no explicit return value returns undefined. If the script that was run throws an unhandled exception, then the result is also "null". This method is applied asynchronously. If the method is run after the `NavigationStarting` event during a navigation, the script runs in the new document when loading it, around the time `ContentLoading` is run. This operation executes the script even if `ICoreWebView2Settings::IsScriptEnabled` is set to `FALSE`.

C++

```
// Prompt the user for some script and then execute it.  
void ScriptComponent::InjectScript()  
{  
    TextInputDialog dialog(  
        m_appWindow->GetMainWindow(),  
        L"Inject Script",  
        L"Enter script code:",  
        L"Enter the JavaScript code to run in the webview.",  
        L"window.getComputedStyle(document.body).backgroundColor");  
    if (dialog.confirmed)  
    {  
        m_webView->ExecuteScript(dialog.input.c_str(),  
            Callback<ICoreWebView2ExecuteScriptCompletedHandler>(  
                [appWindow = m_appWindow](HRESULT error, PCWSTR result) ->  
HRESULT  
            {  
                if (error != S_OK) {  
                    ShowFailure(error, L"ExecuteScript failed");  
                }  
                appWindow->AsyncMessageBox(result, L"ExecuteScript Result");  
                return S_OK;  
            }).Get());  
    }  
}
```

```
    }  
}
```

## get\_BrowserProcessId

The process ID of the browser process that hosts the WebView.

```
public HRESULT get_BrowserProcessId(UINT32 * value)
```

## get\_CanGoBack

`TRUE` if the WebView is able to navigate to a previous page in the navigation history.

```
public HRESULT get_CanGoBack(BOOL * canGoBack)
```

If `CanGoBack` changes value, the `HistoryChanged` event runs.

## get\_CanGoForward

`TRUE` if the WebView is able to navigate to a next page in the navigation history.

```
public HRESULT get_CanGoForward(BOOL * canGoForward)
```

If `CanGoForward` changes value, the `HistoryChanged` event runs.

## get\_ContainsFullScreenElement

Indicates if the WebView contains a fullscreen HTML element.

```
public HRESULT get_ContainsFullScreenElement(BOOL * containsFullScreenElement)
```

## get\_DocumentTitle

The title for the current top-level document.

```
public HRESULT get_DocumentTitle(LPWSTR * title)
```

If the document has no explicit title or is otherwise empty, a default that may or may not match the URI of the document is used.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_Settings

The `ICoreWebView2Settings` object contains various modifiable settings for the running `WebView`.

```
| public HRESULT get_Settings(ICoreWebView2Settings ** settings)
```

## get\_Source

The URI of the current top level document.

```
| public HRESULT get_Source(LPWSTR * uri)
```

This value potentially changes as a part of the `SourceChanged` event that runs for some cases such as navigating to a different site or fragment navigations. It remains the same for other types of navigations such as page refreshes or `history.pushState` with the same URL as the current page.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

C++

```
// Register a handler for the SourceChanged event.  
// This handler will read the webview's source URI and update  
// the app's address bar.  
CHECK_FAILURE(m_webView->add_SourceChanged(  
    Callback<ICoreWebView2SourceChangedEventArgs>(  
        [this](ICoreWebView2* sender,  
        ICoreWebView2SourceChangedEventArgs* args)  
            -> HRESULT {  
                wil::unique_cotaskmem_string uri;  
                sender->get_Source(&uri);  
                if (wcscmp(uri.get(), L"about:blank") == 0)  
                {  
                    uri = wil::make_cotaskmem_string(L"");  
                }  
                SetWindowText(GetAddressBar(), uri.get());  
  
                return S_OK;  
            })  
        .Get(),  
        &m_sourceChangedToken));
```

## GetDevToolsProtocolEventReceiver

Get a DevTools Protocol event receiver that allows you to subscribe to a DevTools Protocol event.

```
public HRESULT GetDevToolsProtocolEventReceiver(LPCWSTR eventName,  
ICoreWebView2DevToolsProtocolEventReceiver ** receiver)
```

The `eventName` parameter is the full name of the event in the format `{domain}.{event}`. For more information about DevTools Protocol events description and event args, navigate to [DevTools Protocol Viewer](#).

C++

```
// Prompt the user to name a CDP event, and then subscribe to that event.  
void ScriptComponent::SubscribeToCdpEvent()  
{  
    TextInputDialog dialog(  
        m_appWindow->GetMainWindow(),  
        L"Subscribe to CDP Event",  
        L"CDP event name:",  
        L"Enter the name of the CDP event to subscribe to.\r\n"  
        L"You may also have to call the \"enable\" method of the\r\n"  
        L"event's domain to receive events (for example  
        \\\"Log.enable\\\").\r\n",  
        L"Log.entryAdded");  
    if (dialog.confirmed)  
    {  
        std::wstring eventName = dialog.input;  
        wil::com_ptr<ICoreWebView2DevToolsProtocolEventReceiver> receiver;  
        CHECK_FAILURE(  
            m_webView->GetDevToolsProtocolEventReceiver(eventName.c_str(),  
&receiver));  
  
        // If we are already subscribed to this event, unsubscribe first.  
        auto preexistingToken =  
            m_devToolsProtocolEventReceivedTokenMap.find(eventName);  
        if (preexistingToken !=  
            m_devToolsProtocolEventReceivedTokenMap.end())  
        {  
            CHECK_FAILURE(receiver->remove_DevToolsProtocolEventReceived(  
                preexistingToken->second));  
        }  
  
        CHECK_FAILURE(receiver->add_DevToolsProtocolEventReceived(  
            Callback<ICoreWebView2DevToolsProtocolEventReceivedEventHandler>  
            (  
                [this, eventName](  
                    ICoreWebView2* sender,  
                    ICoreWebView2DevToolsProtocolEventReceivedEventArgs*  
args) -> HRESULT
```

```

    {
        wil::unique_cotaskmem_string parameterObjectAsJson;
        CHECK_FAILURE(args-
>get_ParameterObjectAsJson(&parameterObjectAsJson));
        std::wstring title = eventName;
        std::wstring details = parameterObjectAsJson.get();

wil::com_ptr<ICoreWebView2DevToolsProtocolEventArgs2> args2;
        if (SUCCEEDED(args-
>QueryInterface(IID_PPV_ARGS(&args2)))
{
        wil::unique_cotaskmem_string sessionId;
        CHECK_FAILURE(args2->get_SessionId(&sessionId));
        if (sessionId.get() && *sessionId.get())
{
            title = eventName + L" (session:" +
sessionId.get() + L")";
            std::wstring targetId =
m_devToolsSessionMap[sessionId.get()];
            std::wstring targetLabel =
m_devToolsTargetLabelMap[targetId];
            details = L"From " + targetLabel + L" (session:"
+ sessionId.get() +
L")\r\n" + details;
}
}
        m_appWindow->AsyncMessageBox(details, L"CDP Event Fired:
" + title);
        return S_OK;
})
.Get(),
&m_devToolsProtocolEventArgsReceivedTokenMap[eventName]));
}
}

```

## GoBack

Navigates the WebView to the previous page in the navigation history.

```
public HRESULT GoBack()
```

## GoForward

Navigates the WebView to the next page in the navigation history.

```
public HRESULT GoForward()
```

## Navigate

Cause a navigation of the top-level document to run to the specified URI.

```
public HRESULT Navigate(LPCWSTR uri)
```

For more information, navigate to [Navigation events](#).

 **Note**

This operation starts a navigation and the corresponding `NavigationStarting` event triggers sometime after `Navigate` runs.

C++

```
void ControlComponent::NavigateToAddressBar()
{
    int length = GetWindowTextLength(GetAddressBar());
    std::wstring uri(length, 0);
    PWSTR buffer = const_cast<PWSTR>(uri.data());
    GetWindowText(GetAddressBar(), buffer, length + 1);

    HRESULT hr = m_webView->Navigate(uri.c_str());
    if (hr == E_INVALIDARG)
    {
        // An invalid URI was provided.
        if (uri.find(L' ') == std::wstring::npos
            && uri.find(L'.') != std::wstring::npos)
        {
            // If it contains a dot and no spaces, try tacking http:// on
            // the front.
            hr = m_webView->Navigate((L"http://" + uri).c_str());
        }
        else
        {
            // Otherwise treat it as a web search.
            std::wstring urlEscaped(2048, ' ');
            DWORD dwEscaped = (DWORD)urlEscaped.length();
            UrlEscapeW(uri.c_str(), &urlEscaped[0], &dwEscaped,
URL_ESCAPE_ASCII_URI_COMPONENT);
            hr = m_webView->Navigate(
                (L"https://bing.com/search?q=" +
                 std::regex_replace(urlEscaped, std::wregex(L"(?:%20)+"),
L"+"))
                .c_str());
        }
    }
    if (hr != E_INVALIDARG) {
        CHECK_FAILURE(hr);
    }
}
```

## **NavigateToString**

Initiates a navigation to htmlContent as source HTML of a new document.

```
public HRESULT NavigateToString(LPCWSTR htmlContent)
```

The `htmlContent` parameter may not be larger than 2 MB (2 \* 1024 \* 1024 bytes) in total size. The origin of the new page is `about:blank`.

C++

```
SetVirtualHostNameToFolderMapping(
    L"appassets.example", L"assets",
    COREWEBVIEW2_HOST_RESOURCE_ACCESS_KIND_DENY);

WCHAR c_navString[] = LR"
<head><link rel='stylesheet' href ='http://appassets.example/wv2.css' />
</head>
<body>
    <img src='http://appassets.example/wv2.png' />
    <p><a href='http://appassets.example/winrt_test.txt'>Click me</a></p>
</body>";
m_webView->NavigateToString(c_navString);
```

C++

```
static const PCWSTR htmlContent =
L"<h1>Domain Blocked</h1>"
L"<p>You've attempted to navigate to a domain in the
blocked "
L"sites list. Press back to return to the previous
page.</p>";
CHECK_FAILURE(sender->NavigateToString(htmlContent));
```

## **OpenDevToolsWindow**

Opens the DevTools window for the current document in the WebView.

```
public HRESULT OpenDevToolsWindow()
```

Does nothing if run when the DevTools window is already open.

## **PostWebMessageAsJson**

Post the specified webMessage to the top level document in this WebView.

```
public HRESULT PostWebMessageAsJson(LPCWSTR webMessageAsJson)
```

The main page receives the message by subscribing to the `message` event of the `window.chrome.webview` of the page document.

C++

```
window.chrome.webview.addEventListener('message', handler)  
window.chrome.webview.removeEventListener('message', handler)
```

The event args is an instance of `MessageEvent`. The `ICoreWebView2Settings::IsWebMessageEnabled` setting must be `TRUE` or the web message will not be sent. The `data` property of the event arg is the `webMessage` string parameter parsed as a JSON string into a JavaScript object. The `source` property of the event arg is a reference to the `window.chrome.webview` object. For information about sending messages from the HTML document in the WebView to the host, navigate to [add\\_WebMessageReceived](#). The message is delivered asynchronously. If a navigation occurs before the message is posted to the page, the message is discarded.

C++

```
// Setup the web message received event handler before navigating to  
// ensure we don't miss any messages.  
CHECK_FAILURE(m_webView->add_WebMessageReceived(  
  
Microsoft::WRL::Callback<ICoreWebView2WebMessageReceivedEventHandler>(  
    [this](ICoreWebView2* sender,  
    ICoreWebView2WebMessageReceivedEventArgs* args)  
{  
    wil::unique_cotaskmem_string uri;  
    CHECK_FAILURE(args->get_Source(&uri));  
  
    // Always validate that the origin of the message is what you  
    expect.  
    if (uri.get() != m_sampleUri)  
    {  
        // Ignore messages from untrusted sources.  
        return S_OK;  
    }  
    wil::unique_cotaskmem_string messageRaw;  
    HRESULT hr = args->TryGetWebMessageAsString(&messageRaw);  
    if (hr == E_INVALIDARG)  
    {  
        // Was not a string message. Ignore.  
        return S_OK;  
    }  
    // Any other problems are fatal.  
    CHECK_FAILURE(hr);
```

```

    std::wstring message = messageRaw.get();

    if (message.compare(0, 13, L"SetTitleText ") == 0)
    {
        m_appWindow->SetDocumentTitle(message.substr(13).c_str());
    }
    else if (message.compare(L"GetWindowBounds") == 0)
    {
        RECT bounds = m_appWindow->GetWindowBounds();
        std::wstring reply =
            L"{" L"\"WindowBounds\":" L"\\"Left:" + std::to_wstring(bounds.left)
            + L"\\"nTop:" + std::to_wstring(bounds.top)
            + L"\\"nRight:" + std::to_wstring(bounds.right)
            + L"\\"nBottom:" + std::to_wstring(bounds.bottom)
            + L"\\"}";
        CHECK_FAILURE(sender->PostWebMessageAsJson(reply.c_str()));
    }
    else
    {
        // Ignore unrecognized messages, but log for further
        investigation
        // since it suggests a mismatch between the web content and the
        host.
        OutputDebugString(
            std::wstring(L"Unexpected message from main page:" +
message).c_str());
    }
    return S_OK;
}).Get(), &m_webMessageReceivedToken);

```

## PostWebMessageAsString

Posts a message that is a simple string rather than a JSON string representation of a JavaScript object.

```
public HRESULT PostWebMessageAsString(LPCWSTR webMessageAsString)
```

This behaves in exactly the same manner as `PostWebMessageAsJson`, but the `data` property of the event arg of the `window.chrome.webview` message is a string with the same value as `webMessageAsString`. Use this instead of `PostWebMessageAsJson` if you want to communicate using simple strings rather than JSON objects.

## Reload

Reload the current page.

```
public HRESULT Reload()
```

This is similar to navigating to the URI of current top level document including all navigation events firing and respecting any entries in the HTTP cache. But, the back or forward history are not modified.

## **remove\_ContainsFullScreenElementChanged**

Remove an event handler previously added with `add_ContainsFullScreenElementChanged`.

```
public HRESULT  
remove_ContainsFullScreenElementChanged(EventRegistrationToken token)
```

## **remove\_ContentLoading**

Remove an event handler previously added with `add_ContentLoading`.

```
public HRESULT remove_ContentLoading(EventRegistrationToken token)
```

## **remove\_DocumentTitleChanged**

Remove an event handler previously added with `add_DocumentTitleChanged`.

```
public HRESULT remove_DocumentTitleChanged(EventRegistrationToken token)
```

## **remove\_FrameNavigationCompleted**

Remove an event handler previously added with `add_FrameNavigationCompleted`.

```
public HRESULT remove_FrameNavigationCompleted(EventRegistrationToken token)
```

## **remove\_FrameNavigationStarting**

Remove an event handler previously added with `add_FrameNavigationStarting`.

```
public HRESULT remove_FrameNavigationStarting(EventRegistrationToken token)
```

## **remove\_HistoryChanged**

Remove an event handler previously added with `add_HistoryChanged`.

```
public HRESULT remove_HistoryChanged(EventRegistrationToken token)
```

## **remove\_NavigationCompleted**

Remove an event handler previously added with `add_NavigationCompleted`.

```
public HRESULT remove_NavigationCompleted(EventRegistrationToken token)
```

## **remove\_NavigationStarting**

Remove an event handler previously added with `add_NavigationStarting`.

```
public HRESULT remove_NavigationStarting(EventRegistrationToken token)
```

## **remove\_NewWindowRequested**

Remove an event handler previously added with `add_NewWindowRequested`.

```
public HRESULT remove_NewWindowRequested(EventRegistrationToken token)
```

## **remove\_PermissionRequested**

Remove an event handler previously added with `add_PermissionRequested`.

```
public HRESULT remove_PermissionRequested(EventRegistrationToken token)
```

## **remove\_ProcessFailed**

Remove an event handler previously added with `add_ProcessFailed`.

```
public HRESULT remove_ProcessFailed(EventRegistrationToken token)
```

## **remove\_ScriptDialogOpening**

Remove an event handler previously added with `add_ScriptDialogOpening`.

```
public HRESULT remove_ScriptDialogOpening(EventRegistrationToken token)
```

## **remove\_SourceChanged**

Remove an event handler previously added with `add_SourceChanged`.

```
public HRESULT remove_SourceChanged(EventRegistrationToken token)
```

## **remove\_WebMessageReceived**

Remove an event handler previously added with `add_WebMessageReceived`.

```
public HRESULT remove_WebMessageReceived(EventRegistrationToken token)
```

## **remove\_WebResourceRequested**

Remove an event handler previously added with `add_WebResourceRequested`.

```
public HRESULT remove_WebResourceRequested(EventRegistrationToken token)
```

## **remove\_WindowCloseRequested**

Remove an event handler previously added with `add_WindowCloseRequested`.

```
public HRESULT remove_WindowCloseRequested(EventRegistrationToken token)
```

## **RemoveHostObjectFromScript**

Remove the host object specified by the name so that it is no longer accessible from JavaScript code in the WebView.

```
public HRESULT RemoveHostObjectFromScript(LPCWSTR name)
```

While new access attempts are denied, if the object is already obtained by JavaScript code in the WebView, the JavaScript code continues to have access to that object. Run this method for a name that is already removed or never added fails.

## **RemoveScriptToExecuteOnDocumentCreated**

Remove the corresponding JavaScript added using

`AddScriptToExecuteOnDocumentCreated` with the specified script ID.

```
public HRESULT RemoveScriptToExecuteOnDocumentCreated(LPCWSTR id)
```

The script ID should be the one returned by the `AddScriptToExecuteOnDocumentCreated`.

Both use `AddScriptToExecuteOnDocumentCreated` and this method in `NewWindowRequested` event handler at the same time sometimes causes trouble. Since invalid scripts will be ignored, the script IDs you got may not be valid anymore.

## RemoveWebResourceRequestedFilter

Removes a matching WebResource filter that was previously added for the `WebResourceRequested` event.

```
public HRESULT RemoveWebResourceRequestedFilter(LPCWSTR const uri,  
COREWEBVIEW2_WEB_RESOURCE_CONTEXT const resourceContext)
```

If the same filter was added multiple times, then it must be removed as many times as it was added for the removal to be effective. Returns `E_INVALIDARG` for a filter that was never added.

## Stop

Stop all navigations and pending resource fetches. Does not stop scripts.

```
public HRESULT Stop()
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2\_10

Article • 02/26/2024

```
interface ICoreWebView2_10
: public ICoreWebView2_9
```

This interface is an extension of [ICoreWebView2\\_9](#) that supports BasicAuthenticationRequested event.

## Summary

[+] [Expand table](#)

Members	Descriptions
<a href="#">add_BasicAuthenticationRequested</a>	Add an event handler for the BasicAuthenticationRequested event.
<a href="#">remove_BasicAuthenticationRequested</a>	Remove an event handler previously added with add_BasicAuthenticationRequested.

## Applies to

[+] [Expand table](#)

Product	Introduced
WebView2 Win32	1.0.1150.38
WebView2 Win32 Prerelease	1.0.1133

## Members

### [add\\_BasicAuthenticationRequested](#)

Add an event handler for the BasicAuthenticationRequested event.

```
public HRESULT  
add_BasicAuthenticationRequested(ICoreWebView2BasicAuthenticationRequestedEventHandler * eventHandler, EventRegistrationToken * token)
```

BasicAuthenticationRequested event is raised when WebView encounters a Basic HTTP Authentication request as described in

<https://developer.mozilla.org/docs/Web/HTTP/Authentication>, a Digest HTTP Authentication request as described in <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Authorization#digest>, an NTLM authentication or a Proxy Authentication request.

The host can provide a response with credentials for the authentication or cancel the request. If the host sets the Cancel property to false but does not provide either UserName or Password properties on the Response property, then WebView2 will show the default authentication challenge dialog prompt to the user.

C++

```
if (auto webView10 = m_webView.try_query<ICoreWebView2_10>())  
{  
    CHECK_FAILURE(webView10->add_BasicAuthenticationRequested(  
        Callback<ICoreWebView2BasicAuthenticationRequestedEventHandler>(  
            [this](  
                ICoreWebView2* sender,  
                ICoreWebView2BasicAuthenticationRequestedEventArgs*  
args) {  
            wil::com_ptr<ICoreWebView2BasicAuthenticationResponse>  
basicAuthenticationResponse;  
            CHECK_FAILURE(args->get_Response(&basicAuthenticationResponse));  
            CHECK_FAILURE(basicAuthenticationResponse->put_UserName(L"user"));  
            CHECK_FAILURE(basicAuthenticationResponse->put_Password(L"pass"));  
  
            return S_OK;  
        })  
        .Get(),  
        &m_basicAuthenticationRequestedToken));  
}  
else {  
    FeatureNotAvailable();  
}
```

## remove\_BasicAuthenticationRequested

Remove an event handler previously added with add\_BasicAuthenticationRequested.

```
public HRESULT remove_BasicAuthenticationRequested(EventRegistrationToken  
token)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2\_11

Article • 02/26/2024

```
interface ICoreWebView2_11
: public ICoreWebView2_10
```

This interface is an extension of [ICoreWebView2\\_10](#) that supports sessionId for CDP method calls and ContextMenuRequested event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">add_ContextMenuRequested</a>	Add an event handler for the <code>ContextMenuRequested</code> event.
<a href="#">CallDevToolsProtocolMethodForSession</a>	Runs an asynchronous <code>DevToolsProtocol</code> method for a specific session of an attached target.
<a href="#">remove_ContextMenuRequested</a>	Remove an event handler previously added with <code>add_ContextMenuRequested</code> .

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1185.39
WebView2 Win32 Prerelease	1.0.1189

## Members

### [add\\_ContextMenuRequested](#)

Add an event handler for the `ContextMenuRequested` event.

```
public HRESULT  
add_ContextMenuRequested(ICoreWebView2ContextMenuRequestedEventHandler  
* eventHandler, EventRegistrationToken * token)
```

`ContextMenuRequested` event is raised when a context menu is requested by the user and the content inside WebView hasn't disabled context menus. The host has the option to create their own context menu with the information provided in the event or can add items to or remove items from WebView context menu. If the host doesn't handle the event, WebView will display the default context menu.

C++

```
if (m_allowCustomMenus)  
{  
    m_webView2_11->add_ContextMenuRequested(  
        Callback<ICoreWebView2ContextMenuRequestedEventHandler>(  
            [this](  
                ICoreWebView2* sender,  
                ICoreWebView2ContextMenuRequestedEventArgs*  
eventArgs)  
    {  
  
        wil::com_ptr<ICoreWebView2ContextMenuRequestedEventArgs> args =  
            eventArgs;  
  
        wil::com_ptr<ICoreWebView2ContextMenuCollection> items;  
        CHECK_FAILURE(args->get_MenuItems(&items));  
  
        UINT32 itemCount;  
        CHECK_FAILURE(items->get_Count(&itemCount));  
  
        wil::com_ptr<ICoreWebView2ContextMenuTarget>  
target;  
        CHECK_FAILURE(args->get_ContextMenuTarget(&target));  
  
        COREWEBVIEW2_CONTEXT_MENU_TARGET_KIND  
targetKind;  
        CHECK_FAILURE(target->get_Kind(&targetKind));  
        // Custom UI Context Menu rendered if invoked on  
selected text  
        if (targetKind ==  
  
COREWEBVIEW2_CONTEXT_MENU_TARGET_KIND_SELECTED_TEXT)  
        {  
            auto showMenu = [this, args, itemCount,  
items, target]  
            {
```

```

                CHECK_FAILURE(args->put_Handled(true));
                HMENU hPopupMenu = CreatePopupMenu();
                AddMenuItem(hPopupMenu, items);
                HWND hWnd;
                m_controller->get_ParentWindow(&hWnd);
                SetForegroundWindow(hWnd);
                RECT rct;
                GetClientRect(hWnd, &rct);
                POINT topLeft;
                topLeft.x = rct.left;
                topLeft.y = rct.top;
                ClientToScreen(hWnd, &topLeft);
                POINT p;
                CHECK_FAILURE(args->get_Location(&p));
                RECT bounds;
                CHECK_FAILURE(m_controller-
                >get_Bounds(&bounds));
                double scale;
                m_controller3-
                >get_RasterizationScale(&scale);
                INT32 selectedCommandId =
                TrackPopupMenu(
                    hWnd,
                    TPM_RETURNCMD,
                    * scale),
                    bounds.left + topLeft.x + ((int)(p.x
                    * scale)), bounds.top + topLeft.y + ((int)(p.y
                    * scale)), 0, hWnd,
                    NULL);
                CHECK_FAILURE(
                    args-
                >put_SelectedCommandId(selectedCommandId));
                };
                wil::com_ptr<ICoreWebView2Deferral>
deferral;
                CHECK_FAILURE(args->GetDeferral(&deferral));
                m_appWindow->RunAsync(
                    [deferral, showMenu]()
                {
                    showMenu();
                    CHECK_FAILURE(deferral->Complete());
                });
            }
            // Removing the 'Save image as' context menu
item for image context
            // selections.
            else if (targetKind ==
COREWEBVIEW2_CONTEXT_MENU_TARGET_KIND_IMAGE)
{
                UINT32 removeIndex = itemCount;
                wil::com_ptr<ICoreWebView2ContextMenuItem>
current;
                for (UINT32 i = 0; i < itemCount; i++)
{

```

```

                CHECK_FAILURE(items->GetValueAtIndex(i,
&current));
                wil::unique_cotaskmem_string name;
                CHECK_FAILURE(current->get_Name(&name));
                if (std::wstring(L"saveImageAs") ==
name.get())
{
    removeIndex = i;
}
}
if (removeIndex < itemsCount)
{
    CHECK_FAILURE(items-
>RemoveValueAtIndex(removeIndex));
}
}
// Adding a custom context menu item for the
page that will display
// the page's URI.
else if (targetKind ==
COREWEBVIEW2_CONTEXT_MENU_TARGET_KIND_PAGE)
{
    // Custom items should be reused whenever
possible.
    if (!m_displayPageUrlContextMenuItem)
    {
        wil::com_ptr<ICoreWebView2Environment9>
webviewEnvironment;
        CHECK_FAILURE(
            m_appWindow-
>GetWebViewEnvironment()->QueryInterface(
IID_PPV_ARGS(&webviewEnvironment)));
    }
    wil::com_ptr<ICoreWebView2ContextMenuMenuItem>
        displayPageUrlItem;
    wil::com_ptr<IStream> iconStream;
    CHECK_FAILURE(SHCreateStreamOnFileEx(
        L"small.ico", STGM_READ,
FILE_ATTRIBUTE_NORMAL, FALSE,
        nullptr, &iconStream));
    CHECK_FAILURE(webviewEnvironment-
>CreateContextMenuItem(
        L"Display Page Url",
iconStream.get(),
COREWEBVIEW2_CONTEXT_MENU_ITEM_KIND_COMMAND,
        &displayPageUrlItem));
    CHECK_FAILURE(displayPageUrlItem-
>add_CustomItemSelected(
Callback<ICoreWebView2CustomItemSelectedEventHandler>(
        [appWindow = m_appWindow,
target](

```

```

ICoreWebView2ContextMenuItem* sender,
                                IUnknown* args)
{
    wil::unique_cotaskmem_string
pageUri;
                                CHECK_FAILURE(target-
>get_PageUri(&pageUri));
                                appWindow->AsyncMessageBox(
pageUri.get(), L"Display
Page Uri");
                                return S_OK;
}
                                .Get(),
    nullptr));
                                CHECK_FAILURE(webviewEnvironment-
>CreateContextMenuItem(
L"New Submenu", nullptr,
COREWEBVIEW2_CONTEXT_MENU_ITEM_KIND_SUBMENU,
&m_displayPageUrlContextSubMenuItem));
wil::com_ptr<ICoreWebView2ContextMenuItemCollection>
m_displayPageUrlContextSubMenuItemChildren;
                                CHECK_FAILURE(
m_displayPageUrlContextSubMenuItem-
>get_Children(
&m_displayPageUrlContextSubMenuItemChildren));
m_displayPageUrlContextSubMenuItemChildren
                                ->InsertValueAtIndex(0,
displayPageUrlItem.get());
}

                                CHECK_FAILURE(items->InsertValueAtIndex(
itemsCount,
m_displayPageUrlContextSubMenuItem.get()));
}
// If type is not an image, video, or page, the
regular Edge context
                                // menu will be displayed
                                return S_OK;
})
                                .Get(),
&m_contextMenuItemRequestedToken);

                                MessageBox(
nullptr, L"Custom Context menus are now enabled.",
L"Settings change",
MB_OK);
}
else
{

```

```
m_webView2_11-
>remove_ContextMenuRequested(m_contextMenuRequestedToken);
    MessageBox(
        nullptr, L"Custom Context menus are now disabled.",
L"Settings change",
        MB_OK);
}
```

## CallDevToolsProtocolMethodForSession

Runs an asynchronous `DevToolsProtocol` method for a specific session of an attached target.

```
public HRESULT CallDevToolsProtocolMethodForSession(LPCWSTR sessionId,
LPCWSTR methodName, LPCWSTR parametersAsJson,
ICoreWebView2CallDevToolsProtocolMethodCompletedHandler * handler)
```

There could be multiple `DevToolsProtocol` targets in a WebView. Besides the top level page, iframes from different origin and web workers are also separate targets. Attaching to these targets allows interaction with them. When the DevToolsProtocol is attached to a target, the connection is identified by a sessionId. To use this API, you must set the `flatten` parameter to true when calling `Target.attachToTarget` or `Target.setAutoAttach``DevToolsProtocol` method. Using `Target.setAutoAttach` is recommended as that would allow you to attach to dedicated worker targets, which are not discoverable via other APIs like `Target.getTargets`. For more information about targets and sessions, navigate to [DevTools Protocol Viewer](#). For more information about available methods, navigate to [DevTools Protocol Viewer](#). The `sessionId` parameter is the sessionId for an attached target. `nullptr` or empty string is treated as the session for the default target for the top page. The `methodName` parameter is the full name of the method in the `{domain}.{method}` format. The `parametersAsJson` parameter is a JSON formatted string containing the parameters for the corresponding method. The `Invoke` method of the `handler` is run when the method asynchronously completes. `Invoke` is run with the return object of the method as a JSON string. This function returns `E_INVALIDARG` if the `methodName` is unknown or the `parametersAsJson` has an error. In the case of such an error, the `returnObjectAsJson` parameter of the handler will include information about the error.

C++

```
void ScriptComponent::HandleCDPTargets()
{
    wil::com_ptr<ICoreWebView2DevToolsProtocolEventReceiver> receiver;
```

```

// Enable Runtime events to receive Runtime.consoleAPICalled events.
m_webView->CallDevToolsProtocolMethod(L"Runtime.enable", L"{}", nullptr);
CHECK_FAILURE(
    m_webView-
>GetDevToolsProtocolEventReceiver(L"Runtime.consoleAPICalled", &receiver));
CHECK_FAILURE(receiver->add_DevToolsProtocolEventReceived(
    Callback<ICoreWebView2DevToolsProtocolEventReceivedEventHandler>(
        [this](
            ICoreWebView2* sender,
            ICoreWebView2DevToolsProtocolEventReceivedEventArgs* args) -
> HRESULT
{
    // Get console.log message details and which target it comes
from.
    wil::unique_cotaskmem_string parameterObjectAsJson;
    CHECK_FAILURE(args-
>get_ParameterObjectAsJson(&parameterObjectAsJson));
    std::wstring eventSourceLabel;
    std::wstring eventDetails = parameterObjectAsJson.get();

    wil::com_ptr<ICoreWebView2DevToolsProtocolEventReceivedEventArgs2> args2;
    if (SUCCEEDED(args->QueryInterface(IID_PPV_ARGS(&args2))))
    {
        wil::unique_cotaskmem_string sessionId;
        CHECK_FAILURE(args2->get_SessionId(&sessionId));
        if (sessionId.get() && *sessionId.get())
        {
            std::wstring targetId =
m_devToolsSessionMap[sessionId.get()];
            eventSourceLabel =
m_devToolsTargetLabelMap[targetId];
        }
    }
    // else, leave eventSourceLabel as empty string for the
default target of top
    // page.

    // Log events to debug output, not using dialog as there
could be a lot of
    // console.log events.
    std::wstring message = L"console.log Event: ";
    if (!eventSourceLabel.empty())
    {
        message = message + L"(from " + eventSourceLabel + L")";
    }
    message += eventDetails + L"\n";
    OutputDebugString(message.c_str());
    return S_OK;
})
    .Get(),
    &m_consoleAPICalledToken));
receiver.reset();
// Track Target and session info via CDP events.
CHECK_FAILURE(

```

```

    m_webView-
>GetDevToolsProtocolEventReceiver(L"Target.attachedToTarget", &receiver));
    CHECK_FAILURE(receiver->add_DevToolsProtocolEventReceived(
        Callback<ICoreWebView2DevToolsProtocolEventReceivedEventHandler>(
            [this]{
                ICoreWebView2* sender,
                ICoreWebView2DevToolsProtocolEventReceivedEventArgs* args) -
> HRESULT
{
    // A new target is attached, add its info to maps.
    wil::unique_cotaskmem_string jsonMessage;
    CHECK_FAILURE(args-
>get_ParameterObjectAsJson(&jsonMessage));
    std::wstring sessionId =
GetJSONStringField(jsonMessage.get(), L"sessionId");
    std::wstring targetId =
GetJSONStringField(jsonMessage.get(), L"targetId");
    m_devToolsSessionMap[sessionId] = targetId;
    std::wstring type = GetJSONStringField(jsonMessage.get(),
L"type");
    std::wstring url = GetJSONStringField(jsonMessage.get(),
L"url");
    m_devToolsTargetLabelMap.insert_or_assign(targetId, type +
L"," + url);
    wil::com_ptr<ICoreWebView2_11> webview2 =
        m_webView.try_query<ICoreWebView2_11>();
    if (webview2)
    {
        // Auto-attach to targets further created from this
target (identified by
        // its session ID), like dedicated worker target created
in the iframe.
        webview2->CallDevToolsProtocolMethodForSession(
            sessionId.c_str(), L"Target.setAutoAttach",
            LR"
({ "autoAttach": true, "waitForDebuggerOnStart": false, "flatten": true } ),
            nullptr);
        // Also enable Runtime events to receive
Runtime.consoleAPICalled from the
        // target.
        webview2->CallDevToolsProtocolMethodForSession(
            sessionId.c_str(), L"Runtime.enable", L"{}",
            nullptr);
    }
    return S_OK;
})
    .Get(),
    &m_targetAttachedToken));
receiver.reset();
CHECK_FAILURE(
    m_webView-
>GetDevToolsProtocolEventReceiver(L"Target.detachedFromTarget", &receiver));
CHECK_FAILURE(receiver->add_DevToolsProtocolEventReceived(
    Callback<ICoreWebView2DevToolsProtocolEventReceivedEventHandler>(
        [this](

```

```

    ICoreWebView2* sender,
    ICoreWebView2DevToolsProtocolEventArgs* args) -
> HRESULT
{
    // A target is detached, remove it from the maps.
    wil::unique_cotaskmem_string jsonMessage;
    CHECK_FAILURE(args-
>get_ParameterObjectAsJson(&jsonMessage));
    std::wstring sessionId =
GetJSONStringField(jsonMessage.get(), L"sessionId");
    auto session = m_devToolsSessionMap.find(sessionId);
    if (session != m_devToolsSessionMap.end())
    {
        m_devToolsTargetLabelMap.erase(session->second);
        m_devToolsSessionMap.erase(session);
    }
    return S_OK;
}
.Get(),
&m_targetDetachedToken));
receiver.reset();
CHECK_FAILURE(
    m_webView->GetDevToolsProtocolEventReceiver(L"Target.targetCreated",
&receiver));
CHECK_FAILURE(receiver->add_DevToolsProtocolEventReceived(
    Callback<ICoreWebView2DevToolsProtocolEventArgs>(
        [this](
            ICoreWebView2* sender,
            ICoreWebView2DevToolsProtocolEventArgs* args) -
> HRESULT
{
    // Shared worker targets are not auto attached. Have to
attach it explicitly.
    wil::unique_cotaskmem_string jsonMessage;
    CHECK_FAILURE(args-
>get_ParameterObjectAsJson(&jsonMessage));
    std::wstring type = GetJSONStringField(jsonMessage.get(),
L"type");
    if (type == L"shared_worker")
    {
        std::wstring targetId =
GetJSONStringField(jsonMessage.get(), L"targetId");
        std::wstring parameters =
L"\"targetId\":\"" + targetId + L"\",\"flatten\":
true}";
        // Call Target.attachToTarget and ignore returned value,
let
        // Target.attachedToTarget to handle the result.
        m_webView->CallDevToolsProtocolMethod(
            L"Target.attachToTarget", parameters.c_str(),
nullptr);
    }
    return S_OK;
})
.Get(),

```

```

        &m_targetCreatedToken));
    receiver.reset();
    CHECK_FAILURE(
        m_webView-
>GetDevToolsProtocolEventReceiver(L"Target.targetInfoChanged", &receiver));
    CHECK_FAILURE(receiver->add_DevToolsProtocolEventReceived(
        Callback<ICoreWebView2DevToolsProtocolEventReceivedEventHandler>(
            [this](
                ICoreWebView2* sender,
                ICoreWebView2DevToolsProtocolEventReceivedEventArgs* args) -
> HRESULT
{
    // A target's info (such as its URL) has changed, so update
    its label in the
        // target label map.
    wil::unique_cotaskmem_string jsonMessage;
    CHECK_FAILURE(args-
>get_ParameterObjectAsJson(&jsonMessage));
    std::wstring targetId =
GetJSONStringField(jsonMessage.get(), L"targetId");
    if (m_devToolsTargetLabelMap.find(targetId) !=
        m_devToolsTargetLabelMap.end())
    {
        // This is a target that we are interested in, update
label.
        std::wstring type =
GetJSONStringField(jsonMessage.get(), L"type");
        std::wstring url = GetJSONStringField(jsonMessage.get(),
L"url");
        m_devToolsTargetLabelMap[targetId] = type + L"," + url;
    }
    return S_OK;
})
    .Get(),
    &m_targetInfoChangedToken));
// Setup CDP targets operation mode.
// Set auto attach to attach to dedicated worker target.
m_webView->CallDevToolsProtocolMethod(
    L"Target.setAutoAttach",
    LR"
({\"autoAttach\":true,\"waitForDebuggerOnStart\":false,\"flatten\":true}"),
nullptr);
// Set setDiscoverTargets to get targetCreated event for shared worker
target.
m_webView->CallDevToolsProtocolMethod(
    L"Target.setDiscoverTargets", LR"({\"discover\":true})", nullptr);
}

```

C++

```

// Prompt the user for the sessionid, name and parameters of a CDP method,
then call it.
void ScriptComponent::CallCdpMethodForSession()

```

```

{
    wil::com_ptr<ICoreWebView2_11> webview2 =
    m_webView.try_query<ICoreWebView2_11>();
    CHECK_FEATURE_RETURN_EMPTY(webview2);
    std::wstring sessionList = L"Sessions:";
    for (auto& target : m_devToolsSessionMap)
    {
        sessionList += L"\r\n";
        sessionList += target.first;
        sessionList += L":";
        sessionList += m_devToolsTargetLabelMap[target.second];
    }
    std::wstring description =
        L"Enter the sessionId, CDP method name to call, and parameters in
JSON format, "
        L"separated by space,\r\n" +
        sessionList;
    TextInputDialog dialog(
        m_appWindow->GetMainWindow(), L"Call CDP Method For Session",
        L"Parameters:",
        description.c_str(),
        L"<sessionId> Runtime.getHeapUsage {}");
    if (dialog.confirmed)
    {
        size_t delimiter1Pos = dialog.input.find(L' ');
        std::wstring sessionId = dialog.input.substr(0, delimiter1Pos);
        size_t delimiter2Pos = dialog.input.find(L' ', delimiter1Pos+1);
        std::wstring methodName =
            dialog.input.substr(delimiter1Pos+1, delimiter2Pos -
delimiter1Pos - 1);
        std::wstring methodParams =
            (delimiter2Pos < dialog.input.size() ?
        dialog.input.substr(delimiter2Pos + 1)
            : L"{}");

        webview2->CallDevToolsProtocolMethodForSession(
            sessionId.c_str(), methodName.c_str(), methodParams.c_str(),
            Callback<ICoreWebView2CallDevToolsProtocolMethodCompletedHandler>(
                this, &ScriptComponent::CDPMethodCallback)
            .Get());
    }
}

```

## remove\_ContextMenuItemRequested

Remove an event handler previously added with `add_ContextMenuItemRequested`.

```
public HRESULT remove_ContextMenuItemRequested(EventRegistrationToken token)
```

# Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2\_12

Article • 02/26/2024

```
interface ICoreWebView2_12
: public ICoreWebView2_11
```

This interface is an extension of [ICoreWebView2\\_11](#) that supports `StatusBarTextChanged` event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">add_StatusBarTextChanged</a>	Add an event handler for the <code>StatusBarTextChanged</code> event.
<a href="#">get_StatusBarText</a>	The status message text.
<a href="#">remove_StatusBarTextChanged</a>	Remove an event handler previously added with <code>add_StatusBarTextChanged</code> .

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1185.39
WebView2 Win32 Prerelease	1.0.1189

## Members

### `add_StatusBarTextChanged`

Add an event handler for the `StatusBarTextChanged` event.

```
public HRESULT  
add_StatusBarTextChanged(ICoreWebView2StatusBarTextChangedEventHandler *  
eventHandler, EventRegistrationToken * token)
```

`StatusbarTextchanged` fires when the WebView is showing a status message, a URL, or an empty string (an indication to hide the status bar).

C++

```
m_StatusBar.Initialize(appWindow);  
// Registering a listener for status bar message changes  
CHECK_FAILURE(m_webView2_12->add_StatusBarTextChanged(  
  
Microsoft::WRL::Callback<ICoreWebView2StatusBarTextChangedEventHandler>(  
    [this](ICoreWebView2* sender, IUnknown* args) -> HRESULT  
{  
    if (m_customStatusBar)  
    {  
        wil::unique_cotaskmem_string value;  
        Microsoft::WRL::ComPtr<ICoreWebView2_12> wv;  
        CHECK_FAILURE(sender->  
>QueryInterface(IID_PPV_ARGS(&wv)));  
  
        CHECK_FAILURE(wv->get_StatusBarText(&value));  
        std::wstring valueString = value.get();  
        if (valueString.length() != 0)  
        {  
            m_StatusBar.Show(valueString);  
        }  
        else  
        {  
            m_StatusBar.Hide();  
        }  
    }  
  
    return S_OK;  
})  
.Get(),  
&m_StatusBarTextChangedToken));
```

## get\_StatusBarText

The status message text.

```
public HRESULT get_StatusBarText(LPWSTR * value)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## `remove_StatusBarTextChanged`

Remove an event handler previously added with `add_StatusBarTextChanged`.

```
public HRESULT remove_StatusBarTextChanged(EventRegistrationToken token)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2\_13

Article • 02/26/2024

```
interface ICoreWebView2_13
: public ICoreWebView2_12
```

This interface is an extension of ICoreWebView2\_12 that supports Profile API.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_Profile</a>	The associated <a href="#">ICoreWebView2Profile</a> object.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1210.39
WebView2 Win32 Prerelease	1.0.1248

## Members

### get\_Profile

The associated ICoreWebView2Profile object.

```
public HRESULT get_Profile(ICoreWebView2Profile ** value)
```

If this CoreWebView2 was created with a CoreWebView2ControllerOptions, the CoreWebView2Profile will match those specified options. Otherwise if this

CoreWebView2 was created without a CoreWebView2ControllerOptions, then this will be the default CoreWebView2Profile for the corresponding CoreWebView2Environment.

C++

```
auto webView2_13 = coreWebView2.try_query<ICoreWebView2_13>();
if (webView2_13)
{
    wil::com_ptr<ICoreWebView2Profile> profile;
    CHECK_FAILURE(webView2_13->get_Profile(&profile));
    wil::unique_cotaskmem_string profile_name;
    CHECK_FAILURE(profile->get_ProfileName(&profile_name));
    m_profileName = profile_name.get();
    BOOL inPrivate = FALSE;
    CHECK_FAILURE(profile->get_IsInPrivateModeEnabled(&inPrivate));
    if (!m_webviewOption.downloadPath.empty())
    {
        CHECK_FAILURE(profile->put_DefaultDownloadFolderPath(
            m_webviewOption.downloadPath.c_str()));
    }

    // update window title with m_profileName
    UpdateAppTitle();

    // update window icon
    SetAppIcon(inPrivate);
}
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2\_14

Article • 02/26/2024

```
interface ICoreWebView2_14
: public ICoreWebView2_13
```

This interface is an extension of [ICoreWebView2\\_13](#) that adds ServerCertificate support.

## Summary

[+] [Expand table](#)

Members	Descriptions
<a href="#">add_ServerCertificateErrorDetected</a>	Add an event handler for the ServerCertificateErrorDetected event.
<a href="#">ClearServerCertificateErrorActions</a>	Clears all cached decisions to proceed with TLS certificate errors from the ServerCertificateErrorDetected event for all WebView2's sharing the same session.
<a href="#">remove_ServerCertificateErrorDetected</a>	Remove an event handler previously added with add_ServerCertificateErrorDetected.

## Applies to

[+] [Expand table](#)

Product	Introduced
WebView2 Win32	1.0.1245.22
WebView2 Win32 Prerelease	1.0.1248

## Members

### [add\\_ServerCertificateErrorDetected](#)

Add an event handler for the ServerCertificateErrorDetected event.

```
public HRESULT  
add_ServerCertificateErrorDetected(ICoreWebView2ServerCertificateErrorDetectedEventHandler * eventHandler, EventRegistrationToken * token)
```

The ServerCertificateErrorDetected event is raised when the WebView2 cannot verify server's digital certificate while loading a web page.

This event will raise for all web resources and follows the `WebResourceRequested` event.

If you don't handle the event, WebView2 will show the default TLS interstitial error page to the user for navigations, and for non-navigations the web request is cancelled.

Note that WebView2 before raising `ServerCertificateErrorDetected` raises a `NavigationCompleted` event with `IsSuccess` as FALSE and any of the below WebErrorStatuses that indicate a certificate failure.

- COREWEBVIEW2\_WEB\_ERROR\_STATUS\_CERTIFICATE\_COMMON\_NAME\_IS\_INCORRECT
- COREWEBVIEW2\_WEB\_ERROR\_STATUS\_CERTIFICATE\_EXPIRED
- COREWEBVIEW2\_WEB\_ERROR\_STATUS\_CLIENT\_CERTIFICATE\_CONTAINS\_ERRORS
- COREWEBVIEW2\_WEB\_ERROR\_STATUS\_CERTIFICATE\_REVOKED
- COREWEBVIEW2\_WEB\_ERROR\_STATUS\_CERTIFICATE\_IS\_INVALID

For more details see [ICoreWebView2NavigationCompletedEventArgs::get\\_IsSuccess](#) and handle `ServerCertificateErrorDetected` event or show the default TLS interstitial error page to the user according to the app needs.

WebView2 caches the response when action is

`COREWEBVIEW2_SERVER_CERTIFICATE_ERROR_ACTION_ALWAYS_ALLOW` for the RequestUri's host and the server certificate in the session and the `ServerCertificateErrorDetected` event won't be raised again.

To raise the event again you must clear the cache using `ClearServerCertificateErrorActions`.

C++

```
// When WebView2 doesn't trust a TLS certificate but host app does, this example bypasses  
// the default TLS interstitial page using the ServerCertificateErrorDetected event handler and  
// continues the request to a server. Otherwise, cancel the request.
```

```

void SettingsComponent::ToggleCustomServerCertificateSupport()
{
    if (m_webView2_14)
    {
        if (m_ServerCertificateErrorToken.value == 0)
        {
            CHECK_FAILURE(m_webView2_14->add_ServerCertificateErrorDetected(
                Callback<ICoreWebView2ServerCertificateErrorDetectedEventHandler>(
                    [this](
                        ICoreWebView2* sender,
                        ICoreWebView2ServerCertificateErrorDetectedEventArgs* args)
                {
                    COREWEBVIEW2_WEB_ERROR_STATUS errorStatus;
                    CHECK_FAILURE(args->get_ErrorStatus(&errorStatus));

                    wil::com_ptr<ICoreWebView2Certificate> certificate =
                    nullptr;
                    CHECK_FAILURE(args-
                        >get_ServerCertificate(&certificate));

                    // Continues the request to a server with a TLS
                    certificate if the error
                    // status is of type
                    //
`COREWEBVIEW2_WEB_ERROR_STATUS_CERTIFICATE_IS_INVALID` and trusted by
                    // the host app.
                    if (errorStatus ==
                        COREWEBVIEW2_WEB_ERROR_STATUS_CERTIFICATE_IS_INVALID &&
                        ValidateServerCertificate(certificate.get()))
                    {
                        CHECK_FAILURE(args->put_Action(
                            COREWEBVIEW2_SERVER_CERTIFICATE_ERROR_ACTION_ALWAYS_ALLOW));
                    }
                    else
                    {
                        // Cancel the request for other TLS certificate
                        error types or if
                        // untrusted by the host app.
                        CHECK_FAILURE(args->put_Action(
                            COREWEBVIEW2_SERVER_CERTIFICATE_ERROR_ACTION_CANCEL));
                    }
                    return S_OK;
                })
                .Get(),
                &m_ServerCertificateErrorToken));
    }
    else
    {
        CHECK_FAILURE(m_webView2_14-
            >remove_ServerCertificateErrorDetected(

```

```
        m_ServerCertificateErrorToken));
    m_ServerCertificateErrorToken.value = 0;
}
else
{
    FeatureNotAvailable();
}
}
```

## ClearServerCertificateErrorActions

Clears all cached decisions to proceed with TLS certificate errors from the ServerCertificateErrorDetected event for all WebView2's sharing the same session.

```
public HRESULT
ClearServerCertificateErrorActions(ICoreWebView2ClearServerCertificateErrorActions
CompletedHandler * handler)
```

## remove\_ServerCertificateErrorDetected

Remove an event handler previously added with add\_ServerCertificateErrorDetected.

```
public HRESULT remove_ServerCertificateErrorDetected(EventRegistrationToken
token)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2\_15

Article • 02/26/2024

```
interface ICoreWebView2_15
: public ICoreWebView2_14
```

This interface is an extension of [ICoreWebView2\\_14](#) that supports status Favicons.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">add_FaviconChanged</a>	Add an event handler for the <code>FaviconChanged</code> event.
<a href="#">get_FaviconUri</a>	Get the current Uri of the favicon as a string.
<a href="#">GetFavicon</a>	Async function for getting the actual image data of the favicon.
<a href="#">remove_FaviconChanged</a>	Remove the event handler for <code>FaviconChanged</code> event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1293.44
WebView2 Win32 Prerelease	1.0.1305

## Members

### [add\\_FaviconChanged](#)

Add an event handler for the `FaviconChanged` event.

```
public HRESULT add_FaviconChanged(ICoreWebView2FaviconChangedEventHandler  
* eventHandler, EventRegistrationToken * token)
```

The `FaviconChanged` event is raised when the `favicon` had a different URL than the previous URL. The FaviconChanged event will be raised for first navigating to a new document, whether or not a document declares a Favicon in HTML if the favicon is different from the previous fav icon. The event will be raised again if a favicon is declared in its HTML or has script to set its favicon. The favicon information can then be retrieved with `GetFavicon` and `FaviconUri`.

## get\_FaviconUri

Get the current Uri of the favicon as a string.

```
public HRESULT get_FaviconUri(LPWSTR * value)
```

If the value is null, then the return value is `E_POINTER`, otherwise it is `S_OK`. If a page has no favicon then the value is an empty string.

## GetFavicon

Async function for getting the actual image data of the favicon.

```
public HRESULT GetFavicon(COREWEBVIEW2_FAVICON_IMAGE_FORMAT format,  
ICoreWebView2GetFaviconCompletedHandler * completedHandler)
```

The image is copied to the `imageStream` object in `ICoreWebView2GetFaviconCompletedHandler`. If there is no image then no data would be copied into the `imageStream`. The `format` is the file format to return the image stream. `completedHandler` is executed at the end of the operation.

C++

```
// Register a handler for the FaviconUriChanged event.  
// This will provided the current favicon of the page, as well  
// as any changes that occur during the page lifetime  
if (m_webView2_15)  
{  
    Gdiplus::GdiplusStartupInput gdiplusStartupInput;  
  
    // Initialize GDI+.  
    Gdiplus::GdiplusStartup(&gdiplusToken_, &gdiplusStartupInput, NULL);  
    CHECK_FAILURE(m_webView2_15->add_FaviconChanged(
```

```

Callback<ICoreWebView2FaviconChangedEventHandler>(
    [this](ICoreWebView2* sender, IUnknown* args) -> HRESULT
{
    if (m_faviconChanged)
    {
        wil::unique_cotaskmem_string url;
        Microsoft::WRL::ComPtr<ICoreWebView2_15> webview2;
        CHECK_FAILURE(sender-
>QueryInterface(IID_PPV_ARGS(&webview2)));

        CHECK_FAILURE(webview2->get_FaviconUri(&url));
        std::wstring strUrl(url.get());

        webview2->GetFavicon(
            COREWEBVIEW2_FAVICON_IMAGE_FORMAT_PNG,

```

Callback<ICoreWebView2GetFaviconCompletedHandler>(
 [this,
 strUrl](HRESULT errorCode, IStream\*
iconStream) -> HRESULT
{
 CHECK\_FAILURE(errorCode);
 Gdiplus::Bitmap iconBitmap(iconStream);
 wil::unique\_hicon icon;
 if (iconBitmap.GetHICON(&icon) ==
Gdiplus::Status::Ok)
 {
 m\_favicon = std::move(icon);
 SendMessage(
 m\_appWindow->GetMainWindow(),
WM\_SETICON,
 ICON\_SMALL,
(LPARAM)m\_favicon.get());
 m\_statusBar.Show(strUrl);
 }
 else
 {
 SendMessage(
 m\_appWindow->GetMainWindow(),
WM\_SETICON,
 ICON\_SMALL, (LPARAM)IDC\_NO);
 m\_statusBar.Show(L"No Icon");
 }

```

        return S_OK;
    })
    .Get());
}
return S_OK;
})
.Get(),
&m_faviconChangedToken));
}

```

## `remove_FaviconChanged`

Remove the event handler for `FaviconChanged` event.

```
public HRESULT remove_FaviconChanged(EventRegistrationToken token)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2\_16

Article • 02/26/2024

```
interface ICoreWebView2_16
: public ICoreWebView2_15
```

A continuation of the [ICoreWebView2](#) interface to support printing.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Print</a>	Print the current web page asynchronously to the specified printer with the provided settings.
<a href="#">PrintToPdfStream</a>	Provides the Pdf data of current web page asynchronously for the provided settings.
<a href="#">ShowPrintUI</a>	Opens the print dialog to print the current web page.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1518.46
WebView2 Win32 Prerelease	1.0.1549

## Members

### Print

Print the current web page asynchronously to the specified printer with the provided settings.

```
public HRESULT Print(ICoreWebView2PrintSettings * printSettings,  
ICoreWebView2PrintCompletedHandler * handler)
```

See [ICoreWebView2PrintSettings](#) for description of settings. Passing `nullptr` for `printSettings` results in default print settings used.

The handler will return `errorCode` as `S_OK` and `printStatus` as `COREWEBVIEW2_PRINT_STATUS_PRINTER_UNAVAILABLE` if `printerName` doesn't match with the name of any installed printers on the user OS. The handler will return `errorCode` as `E_INVALIDARG` and `printStatus` as `COREWEBVIEW2_PRINT_STATUS_OTHER_ERROR` if the caller provides invalid settings for a given printer.

The async `Print` operation completes when it finishes printing to the printer. At this time the [ICoreWebView2PrintCompletedHandler](#) is invoked. Only one `Printing` operation can be in progress at a time. If `Print` is called while a `Print` or `PrintToPdf` or `PrintToPdfStream` or `ShowPrintUI` job is in progress, the completed handler is immediately invoked with `E_ABORT` and `printStatus` is `COREWEBVIEW2_PRINT_STATUS_OTHER_ERROR`. This is only for printing operation on one webview.

[ ] Expand table

<code>errorCode</code>	<code>printStatus</code>	<code>Notes</code>
<code>S_OK</code>	<code>COREWEBVIEW2_PRINT_STATUS_SUCCEEDED</code>	Print operation succeeded.
<code>S_OK</code>	<code>COREWEBVIEW2_PRINT_STATUS_PRINTER_UNAVAILABLE</code>	If specified printer is not found or printer status is not available, offline or error state.
<code>S_OK</code>	<code>COREWEBVIEW2_PRINT_STATUS_OTHER_ERROR</code>	Print operation is failed.
<code>E_INVALIDARG</code>	<code>COREWEBVIEW2_PRINT_STATUS_OTHER_ERROR</code>	If the caller provides invalid settings for the specified printer.
<code>E_ABORT</code>	<code>COREWEBVIEW2_PRINT_STATUS_OTHER_ERROR</code>	Print operation is failed as printing job already in progress.

```
// This example prints the current web page to a specified printer with the
// settings.
bool AppWindow::PrintToPrinter()
{
    std::wstring printerName = GetPrinterName();
    // Host apps custom print settings based on the user selection.
    SamplePrintSettings samplePrintSettings =
GetSelectedPrinterPrintSettings(printerName);

    wil::com_ptr<ICoreWebView2_16> webView2_16;
    CHECK_FAILURE(m_webView->QueryInterface(IID_PPV_ARGS(&webView2_16)));

    wil::com_ptr<ICoreWebView2Environment6> webviewEnvironment6;
    CHECK_FAILURE(m_webViewEnvironment-
>QueryInterface(IID_PPV_ARGS(&webviewEnvironment6)));

    wil::com_ptr<ICoreWebView2PrintSettings> printSettings;
    CHECK_FAILURE(webviewEnvironment6->CreatePrintSettings(&printSettings));

    wil::com_ptr<ICoreWebView2PrintSettings2> printSettings2;
    CHECK_FAILURE(printSettings-
>QueryInterface(IID_PPV_ARGS(&printSettings2)));

    CHECK_FAILURE(printSettings-
>put_Orientation(samplePrintSettings.Orientation));
    CHECK_FAILURE(printSettings2->put_Copies(samplePrintSettings.Copies));
    CHECK_FAILURE(printSettings2-
>put_PagesPerSide(samplePrintSettings.PagesPerSide));
    CHECK_FAILURE(printSettings2-
>put_PageRanges(samplePrintSettings.Pages.c_str()));
    if (samplePrintSettings.Media == COREWEBVIEW2_PRINT_MEDIA_SIZE_CUSTOM)
    {
        CHECK_FAILURE(printSettings-
>put_PageWidth(samplePrintSettings.PaperWidth));
        CHECK_FAILURE(printSettings-
>put_PageHeight(samplePrintSettings.PaperHeight));
    }
    CHECK_FAILURE(printSettings2-
>put_ColorMode(samplePrintSettings.ColorMode));
    CHECK_FAILURE(printSettings2-
>put_Collation(samplePrintSettings.Collation));
    CHECK_FAILURE(printSettings2->put_Duplex(samplePrintSettings.Duplex));
    CHECK_FAILURE(printSettings-
>put_ScaleFactor(samplePrintSettings.ScaleFactor));
    CHECK_FAILURE(
        printSettings-
>put_ShouldPrintBackgrounds(samplePrintSettings.PrintBackgrounds));
    CHECK_FAILURE(
        printSettings-
>put_ShouldPrintBackgrounds(samplePrintSettings.PrintBackgrounds));
    CHECK_FAILURE(
        printSettings-
>put_ShouldPrintHeaderAndFooter(samplePrintSettings.HeaderAndFooter));
    CHECK_FAILURE(printSettings-
```

```

>put_HeaderTitle(samplePrintSettings.HeaderTitle.c_str()));
    CHECK_FAILURE(printSettings-
>put_FooterUri(samplePrintSettings.FooterUri.c_str()));
    CHECK_FAILURE(printSettings2->put_PrinterName(printerName.c_str()));

    wil::unique_cotaskmem_string title;
    CHECK_FAILURE(m_webView->get_DocumentTitle(&title));

    CHECK_FAILURE(webView2_16->Print(
        printSettings.get(),
        Callback<ICoreWebView2PrintCompletedHandler>(
            [title = std::move(title),
             this](HRESULT errorCode, COREWEBVIEW2_PRINT_STATUS printStatus)
-> HRESULT
{
    std::wstring message = L "";
    if (errorCode == S_OK && printStatus ==
COREWEBVIEW2_PRINT_STATUS_SUCCEEDED)
    {
        message = L "Printing " + std::wstring(title.get()) +
L " document to printer is succeeded";
    }
    else if (
        errorCode == S_OK &&
        printStatus ==
COREWEBVIEW2_PRINT_STATUS_PRINTER_UNAVAILABLE)
    {
        message = L "Selected printer is not found, not
available, offline or "
                    L "error state.";
    }
    else if (errorCode == E_INVALIDARG)
    {
        message = L "Invalid settings provided for the specified
printer";
    }
    else if (errorCode == E_ABORT)
    {
        message = L "Printing " + std::wstring(title.get()) +
L " document already in progress";
    }
    else
    {
        message = L "Printing " + std::wstring(title.get()) +
L " document to printer is failed";
    }
}

    AsyncMessageBox(message, L "Print to printer");

    return S_OK;
})
    .Get()));
return true;
}

```

## PrintToPdfStream

Provides the Pdf data of current web page asynchronously for the provided settings.

```
public HRESULT PrintToPdfStream(ICoreWebView2PrintSettings * printSettings,  
ICoreWebView2PrintToPdfStreamCompletedHandler * handler)
```

Stream will be rewound to the start of the pdf data.

See [ICoreWebView2PrintSettings](#) for description of settings. Passing `nullptr` for `printSettings` results in default print settings used.

The async `PrintToPdfStream` operation completes when it finishes writing to the stream. At this time the [ICoreWebView2PrintToPdfStreamCompletedHandler](#) is invoked. Only one `Printing` operation can be in progress at a time. If `PrintToPdfStream` is called while a `PrintToPdfStream` or `PrintToPdf` or `Print` or `ShowPrintUI` job is in progress, the completed handler is immediately invoked with `E_ABORT`. This is only for printing operation on one webview.

C++

```
// This example prints the Pdf data of the current page to a stream.  
bool AppWindow::PrintToPdfStream()  
{  
    wil::com_ptr<ICoreWebView2_16> webView2_16;  
    CHECK_FAILURE(m_webView->QueryInterface(IID_PPV_ARGS(&webView2_16));  
  
    wil::unique_cotaskmem_string title;  
    CHECK_FAILURE(m_webView->get_DocumentTitle(&title));  
  
    // Passing nullptr for `ICoreWebView2PrintSettings` results in default  
    // print settings used.  
    CHECK_FAILURE(webView2_16->PrintToPdfStream(  
        nullptr,  
        Callback<ICoreWebView2PrintToPdfStreamCompletedHandler>(  
            [title = std::move(title), this](HRESULT errorCode, IStream*  
pdfData) -> HRESULT  
            {  
                DisplayPdfDataInPrintDialog(pdfData);  
  
                std::wstring message =  
                    L"Printing " + std::wstring(title.get()) + L" document  
to PDF Stream " +  
                    ((errorCode == S_OK && pdfData != nullptr) ?  
L"succeded" : L"failed");  
  
                AsyncMessageBox(message, L"Print to PDF Stream");  
  
                return S_OK;  
            }  
    ));  
}
```

```
        })
        .Get()));
    return true;
}
```

## ShowPrintUI

Opens the print dialog to print the current web page.

```
public HRESULT ShowPrintUI(COREWEBVIEW2_PRINT_DIALOG_KIND
printDialogKind)
```

See [COREWEBVIEW2\\_PRINT\\_DIALOG\\_KIND](#) for descriptions of print dialog kinds.

Invoking browser or system print dialog doesn't open new print dialog if it is already open.

C++

```
// Shows the user a print dialog. If `printDialogKind` is
// COREWEBVIEW2_PRINT_DIALOG_KIND_BROWSER, opens a browser print preview
// dialog,
// COREWEBVIEW2_PRINT_DIALOG_KIND_SYSTEM opens a system print dialog.
bool AppWindow::ShowPrintUI(COREWEBVIEW2_PRINT_DIALOG_KIND printDialogKind)
{
    auto webView2_16 = m_webView.try_query<ICoreWebView2_16>();
    CHECK_FEATURE_RETURN(webView2_16);
    CHECK_FAILURE(webView2_16->ShowPrintUI(printDialogKind));
    return true;
}
```

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2\_17

Article • 02/26/2024

```
interface ICoreWebView2_17
: public ICoreWebView2_16
```

This interface is an extension of [ICoreWebView2\\_16](#) that supports shared buffer based on file mapping.

## Summary

[+] [Expand table](#)

Members	Descriptions
<a href="#">PostSharedBufferToScript</a>	Share a shared buffer object with script of the main frame in the WebView.

## Applies to

[+] [Expand table](#)

Product	Introduced
WebView2 Win32	1.0.1661.34
WebView2 Win32 Prerelease	1.0.1671

## Members

### [PostSharedBufferToScript](#)

Share a shared buffer object with script of the main frame in the WebView.

```
public HRESULT PostSharedBufferToScript(ICoreWebView2SharedBuffer *
sharedBuffer, COREWEBVIEW2_SHARED_BUFFER_ACCESS access, LPCWSTR
additionalDataAsJson)
```

The script will receive a `sharedbufferreceived` event from `chrome.webview`. The event arg for that event will have the following methods and properties: `getBuffer()`: return an `ArrayBuffer` object with the backing content from the shared buffer. `additionalData`: an object as the result of parsing `additionalDataAsJson` as JSON string. This property will be `undefined` if `additionalDataAsJson` is `nullptr` or empty string. `source`: with a value set as `chrome.webview` object. If a string is provided as `additionalDataAsJson` but it is not a valid JSON string, the API will fail with `E_INVALIDARG`. If `access` is `COREWEBVIEW2_SHARED_BUFFER_ACCESS_READ_ONLY`, the script will only have read access to the buffer. If the script tries to modify the content in a read only buffer, it will cause an access violation in WebView renderer process and crash the renderer process. If the shared buffer is already closed, the API will fail with `RO_E_CLOSED`.

The script code should call `chrome.webview.releaseBuffer` with the shared buffer as the parameter to release underlying resources as soon as it does not need access to the shared buffer any more.

The application can post the same shared buffer object to multiple web pages or iframes, or post to the same web page or iframe multiple times. Each `PostSharedBufferToScript` will create a separate `ArrayBuffer` object with its own view of the memory and is separately released. The underlying shared memory will be released when all the views are released.

For example, if we want to send data to script for one time read only consumption.

C++

```
wil::com_ptr<ICoreWebView2Environment12> environment;
CHECK_FAILURE(
    m_appWindow->GetWebViewEnvironment()-
>QueryInterface(IID_PPV_ARGS(&environment)));

wil::com_ptr<ICoreWebView2SharedBuffer> sharedBuffer;
CHECK_FAILURE(environment->CreateSharedBuffer(bufferSize,
&sharedBuffer));
// Set data into the shared memory via IStream.
wil::com_ptr<IStream> stream;
CHECK_FAILURE(sharedBuffer->OpenStream(&stream));
CHECK_FAILURE(stream->Write(data, sizeof(data), nullptr));
PCWSTR additionalDataAsJson = L"\"myBufferType\":\"bufferType1\"";
if (fromFrame)
{
    m_webviewFrame4->PostSharedBufferToScript(
        sharedBuffer.get(),
        COREWEBVIEW2_SHARED_BUFFER_ACCESS_READ_ONLY,
        additionalDataAsJson);
}
else
```

```
{  
    m_webView17->PostSharedBufferToScript(  
        sharedBuffer.get(),  
        COREWEBVIEW2_SHARED_BUFFER_ACCESS_READ_ONLY,  
        additionalDataAsJson);  
}  
// Explicitly close the one time shared buffer to ensure that the  
resource is released.  
sharedBuffer->Close();
```

In the HTML document,

HTML

```
window.chrome.webview.addEventListener("sharedbufferreceived", e  
=> {  
    SharedBufferReceived(e);});
```

HTML

```
let readOnlySharedBuffer;  
function ShowReadOnlySharedBuffer() {  
    if (readOnlySharedBuffer) {  
        DisplaySharedBufferData(readOnlySharedBuffer);  
    } else {  
        // Post a web message to ask host to share the one time read  
only buffer.  
        chrome.webview.postMessage("RequestOneTimeShareBuffer");  
    }  
}  
  
function DisplaySharedBufferData(buffer) {  
    document.getElementById("shared-buffer-data").value =  
        new TextDecoder().decode(new Uint8Array(buffer));  
}  
  
function SharedBufferReceived(e) {  
    if (e.additionalData && e.additionalData.myBufferType ==  
"bufferType1") {  
        readOnlySharedBuffer = e.getBuffer();  
    } else {  
        sharedBuffer = e.getBuffer();  
    }  
    DisplaySharedBufferData(e.getBuffer());  
}  
  
function ReleaseBuffer(buffer) {  
    window.chrome.webview.releaseBuffer(buffer);  
}
```

Sharing a buffer to script has security risk. You should only share buffer with trusted site. If a buffer is shared to a untrusted site, possible sensitive information could be leaked. If a buffer is shared as modifiable by the script and the script modifies it in an unexpected way, it could result in corrupted data that might even crash the application.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2\_18

Article • 02/26/2024

```
interface ICoreWebView2_18
: public ICoreWebView2_17
```

This interface is an extension of [ICoreWebView2\\_17](#) that manages navigation requests to URI schemes registered with the OS.

## Summary

[+] [Expand table](#)

Members	Descriptions
<a href="#">add_LaunchingExternalUriScheme</a>	Add an event handler for the <code>LaunchingExternalUriScheme</code> event.
<a href="#">remove_LaunchingExternalUriScheme</a>	Remove an event handler previously added with <code>add_LaunchingExternalUriScheme</code> .

## Applies to

[+] [Expand table](#)

Product	Introduced
WebView2 Win32	1.0.1823.32
WebView2 Win32 Prerelease	1.0.1905

## Members

### [add\\_LaunchingExternalUriScheme](#)

Add an event handler for the `LaunchingExternalUriScheme` event.

```
public HRESULT  
add_LaunchingExternalUriScheme(ICoreWebView2LaunchingExternalUriSchemeEven  
tHandler * eventHandler, EventRegistrationToken * token)
```

The `LaunchingExternalUriScheme` event is raised when a navigation request is made to a URI scheme that is registered with the OS. The `LaunchingExternalUriScheme` event handler may suppress the default dialog or replace the default dialog with a custom dialog.

If a deferral is not taken on the event args, the external URI scheme launch is blocked until the event handler returns. If a deferral is taken, the external URI scheme launch is blocked until the deferral is completed. The host also has the option to cancel the URI scheme launch.

The `NavigationStarting` and `NavigationCompleted` events will be raised, regardless of whether the `Cancel` property is set to `TRUE` or `FALSE`. The `NavigationCompleted` event will be raised with the `IsSuccess` property set to `FALSE` and the `WebErrorStatus` property set to `ConnectionAborted` regardless of whether the host sets the `Cancel` property on the `ICoreWebView2LaunchingExternalUriSchemeEventArgs`. The `SourceChanged`, `ContentLoading`, and `HistoryChanged` events will not be raised for this navigation to the external URI scheme regardless of the `Cancel` property. The `LaunchingExternalUriScheme` event will be raised after the `NavigationStarting` event and before the `NavigationCompleted` event. The default `CoreWebView2Settings` will also be updated upon navigation to an external URI scheme. If a setting on the `CoreWebView2Settings` interface has been changed, navigating to an external URI scheme will trigger the `CoreWebView2Settings` to update.

The WebView2 may not display the default dialog based on user settings, browser settings, and whether the origin is determined as a [trustworthy origin] (<https://w3c.github.io/webappsec-secure-contexts#potentially-trustworthy-origin>); however, the event will still be raised.

If the request is initiated by a cross-origin frame without a user gesture, the request will be blocked and the `LaunchingExternalUriScheme` event will not be raised.

C++

```
m_launchingExternalUriScheme = !m_launchingExternalUriScheme;  
CHECK_FEATURE_RETURN(m_webView2_18);  
if (m_launchingExternalUriScheme)  
{  
    CHECK_FAILURE(m_webView2_18->add_LaunchingExternalUriScheme(
```

```
Callback<ICoreWebView2LaunchingExternalUriSchemeEventHandler>(
    [this](
        ICoreWebView2* sender,
        ICoreWebView2LaunchingExternalUriSchemeEventArgs* args)
    {
        auto showDialog = [this, args]
        {
            wil::unique_cotaskmem_string uri;
            CHECK_FAILURE(args->get_Uri(&uri));
            if (wcscmp(uri.get(), L"calculator://") ==
0)
            {
                // Set the event args to cancel the
                // calculator app. This will always
                // URI scheme launch.
                args->put_Cancel(true);
                std::wstring schemeUrl =
L"calculator://";
                SHELLEXECUTEINFO info = {sizeof(info)};
                info.fMask = SEE_MASK_NOASYNC;
                info.lpVerb = L"open";
                info.lpFile = schemeUrl.c_str();
                info.nShow = SW_SHOWNORMAL;
                ::ShellExecuteEx(&info);
            }
            else if (wcscmp(uri.get(), L"malicious://") ==
0)
            {
                // Always block the request in this case
                // the event.
                args->put_Cancel(true);
            }
            else if (wcscmp(uri.get(), L"contoso://") ==
0)
            {
                // To display a custom dialog we cancel
                // display a custom dialog, and then
                // external URI scheme depending on the
                // selection.
                args->put_Cancel(true);
                wil::unique_cotaskmem_string
initiatingOrigin;
                CHECK_FAILURE(
                    args-
>get_InitiatingOrigin(&initiatingOrigin));
                std::wstring message =
L"Launching External URI Scheme
request";
```

```

        std::wstring initiatingOriginString =
            initiatingOrigin.get();
        if (initiatingOriginString.empty())
        {
            message += L" from ";
            message += initiatingOriginString;
        }
        message += L" to ";
        message += uri.get();
        message += L"?\\n\\n";
        message += L"Do you want to grant
permission?\n";
        int response = MessageBox(
            nullptr, message.c_str(),
            L"Launching External URI Scheme",
            MB_YESNO | MB_ICONWARNING);
        if (response == IDYES)
        {
            std::wstring schemeUrl = uri.get();
            SHELLEXECUTEINFO info =
                {
                    .fMask = SEE_MASK_NOASYNC,
                    .lpVerb = L"open",
                    .lpFile = schemeUrl.c_str(),
                    .nShow = SW_SHOWNORMAL,
                    ::ShellExecuteEx(&info),
                };
        }
        else
        {
            // Do not cancel the event, allowing the
            // request to use
            // the default dialog.
        }
        return S_OK;
    };
    showDialog();
    return S_OK;
    // A deferral may be taken for the event so that
    // we set on the
    // method
    // more time to
    // scheme or not.
    // A deferral doesn't need to be taken in this
    // case, so taking
    // a deferral is commented out here.
    // wil::com_ptr<ICoreWebView2Deferral> deferral;
    // CHECK_FAILURE(args->GetDeferral(&deferral));
    // m_appWindow->RunAsync(

```

```
        //      [deferral, showDialog]()
        //
        //      {
        //          showDialog();
        //      CHECK_FAILURE(deferral->Complete());
        //      });
        // return S_OK;
    })
    .Get(),
    &m_launchingExternalUriSchemeToken));
}
else
{
    m_webView2_18->remove_LaunchingExternalUriScheme(
        m_launchingExternalUriSchemeToken);
}

MessageBox(
    nullptr,
    (std::wstring(L"Launching External URI Scheme support has
been ") +
     (m_launchingExternalUriScheme ? L"enabled." :
L"disabled."))
    .c_str(),
    L"Launching External URI Scheme", MB_OK);
return true;
```

MSOWNERS: [mwinstanley@microsoft.com](mailto:mwinstanley@microsoft.com)

## remove\_LaunchingExternalUriScheme

Remove an event handler previously added with `add_LaunchingExternalUriScheme`.

```
public HRESULT remove_LaunchingExternalUriScheme(EventRegistrationToken
token)
```

MSOWNERS: [mwinstanley@microsoft.com](mailto:mwinstanley@microsoft.com)

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2\_19

Article • 02/26/2024

```
interface ICoreWebView2_19
: public ICoreWebView2_18
```

This interface is an extension of ICoreWebView2\_18 that manages memory usage target level.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_MemoryUsageTargetLevel</a>	<code>MemoryUsageTargetLevel</code> indicates desired memory consumption level of WebView.
<a href="#">put_MemoryUsageTargetLevel</a>	An app may set <code>MemoryUsageTargetLevel</code> to indicate desired memory consumption level of WebView.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1823.32
WebView2 Win32 Prerelease	1.0.1905

## Members

### `get_MemoryUsageTargetLevel`

`MemoryUsageTargetLevel` indicates desired memory consumption level of WebView.

```
public HRESULT  
get_MemoryUsageTargetLevel(COREWEBVIEW2_MEMORY_USAGE_TARGET_LEVEL *  
level)
```

## put\_MemoryUsageTargetLevel

An app may set `MemoryUsageTargetLevel` to indicate desired memory consumption level of WebView.

```
public HRESULT  
put_MemoryUsageTargetLevel(COREWEBVIEW2_MEMORY_USAGE_TARGET_LEVEL  
level)
```

Scripts will not be impacted and continue to run. This is useful for inactive apps that still want to run scripts and/or keep network connections alive and therefore could not call `TrySuspend` and `Resume` to reduce memory consumption. These apps can set memory usage target level to `COREWEBVIEW2_MEMORY_USAGE_TARGET_LEVEL_LOW` when the app becomes inactive, and set back to `COREWEBVIEW2_MEMORY_USAGE_TARGET_LEVEL_NORMAL` when the app becomes active. It is not necessary to set CoreWebView2Controller's `IsVisible` property to false when setting the property. It is a best effort operation to change memory usage level, and the API will return before the operation completes. Setting the level to `COREWEBVIEW2_MEMORY_USAGE_TARGET_LEVEL_LOW` could potentially cause memory for some WebView browser processes to be swapped out to disk in some circumstances. It is a best effort to reduce memory usage as much as possible. If a script runs after its related memory has been swapped out, the memory will be swapped back in to ensure the script can still run, but performance might be impacted. Therefore, the app should set the level back to `COREWEBVIEW2_MEMORY_USAGE_TARGET_LEVEL_NORMAL` when the app becomes active again. Setting memory usage target level back to normal will not happen automatically. An app should choose to use either the combination of `TrySuspend` and `Resume` or the combination of setting `MemoryUsageTargetLevel` to low and normal. It is not advisable to mix them. Trying to set `MemoryUsageTargetLevel` while suspended will be ignored. The `TrySuspend` and `Resume` methods will change the `MemoryUsageTargetLevel`. `TrySuspend` will automatically set `MemoryUsageTargetLevel` to low while `Resume` on suspended WebView will automatically set `MemoryUsageTargetLevel` to normal. Calling `Resume` when the WebView is not suspended would not change `MemoryUsageTargetLevel`.

C++

```
void ViewComponent::ToggleMemoryUsageTargetLevel()
{
    wil::com_ptr<ICoreWebView2_19> webView;
    webView = m_webView.try_query<ICoreWebView2_19>();
    CHECK_FEATURE_RETURN_EMPTY(webView);
    COREWEBVIEW2_MEMORY_USAGE_TARGET_LEVEL memory_target_level =
        COREWEBVIEW2_MEMORY_USAGE_TARGET_LEVEL_NORMAL;
    CHECK_FAILURE(webView-
>get_MemoryUsageTargetLevel(&memory_target_level));
    memory_target_level = (memory_target_level ==
COREWEBVIEW2_MEMORY_USAGE_TARGET_LEVEL_LOW)
    ?
    COREWEBVIEW2_MEMORY_USAGE_TARGET_LEVEL_NORMAL
    : COREWEBVIEW2_MEMORY_USAGE_TARGET_LEVEL_LOW;
    CHECK_FAILURE(webView->put_MemoryUsageTargetLevel(memory_target_level));
    MessageBox(
        nullptr,
        (memory_target_level == COREWEBVIEW2_MEMORY_USAGE_TARGET_LEVEL_LOW)
        ? L"MemoryUsageTargetLevel is set to LOW."
        : L"MemoryUsageTargetLevel is set to NORMAL.",
        L"MemoryUsageTargetLevel change", MB_OK);
}
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2\_2

Article • 02/26/2024

```
interface ICoreWebView2_2
: public ICoreWebView2
```

A continuation of the [ICoreWebView2](#) interface.

## Summary

[ ] [Expand table](#)

Members	Descriptions
<a href="#">add_DOMContentLoaded</a>	Add an event handler for the DOMContentLoaded event.
<a href="#">add_WebResourceResponseReceived</a>	Add an event handler for the WebResourceResponseReceived event.
<a href="#">get_CookieManager</a>	Gets the cookie manager object associated with this <a href="#">ICoreWebView2</a> .
<a href="#">get_Environment</a>	Exposes the CoreWebView2Environment used to create this CoreWebView2.
<a href="#">NavigateWithWebResourceRequest</a>	Navigates using a constructed WebResourceRequest object.
<a href="#">remove_DOMContentLoaded</a>	Remove an event handler previously added with add_DOMContentLoaded.
<a href="#">remove_WebResourceResponseReceived</a>	Remove an event handler previously added with add_WebResourceResponseReceived.

## Applies to

[ ] [Expand table](#)

Product	Introduced
WebView2 Win32	1.0.705.50

Product	Introduced
WebView2 Win32 Prerelease	1.0.721

# Members

## add\_DOMContentLoaded

Add an event handler for the DOMContentLoaded event.

```
public HRESULT
add_DOMContentLoaded(ICoreWebView2DOMContentLoadedEventHandler *
eventHandler, EventRegistrationToken * token)
```

DOMContentLoaded is raised when the initial html document has been parsed. This aligns with the document's DOMContentLoaded event in html.

C++

```
// Register a handler for the DOMContentLoaded event.
// Check whether the DOM content loaded
CHECK_FAILURE(m_appWindow->GetWebView()-
>QueryInterface(IID_PPV_ARGS(&m_webView2)));
CHECK_FAILURE(m_webView2->add_DOMContentLoaded(
    Callback<ICoreWebView2DOMContentLoadedEventHandler>(
        [this](ICoreWebView2* sender,
ICoreWebView2DOMContentLoadedEventArgs* args)
        -> HRESULT {
            m_webView->ExecuteScript(
                LR"~(
                    let content = document.createElement("h2");
                    content.style.color = 'blue';
                    content.textContent = "This text was added by the host
app";
                    document.body.appendChild(content);
                )~",
                Callback<ICoreWebView2ExecuteScriptCompletedHandler>(
                    [](HRESULT error, PCWSTR result) -> HRESULT { return
S_OK; })
                    .Get());
            return S_OK;
        }
        .Get(),
        &m_DOMContentLoadedToken));
    );
```

## add\_WebResourceResponseReceived

Add an event handler for the WebResourceResponseReceived event.

```
public HRESULT  
add_WebResourceResponseReceived(ICoreWebView2WebResourceResponseReceive  
dEventHandler * eventHandler, EventRegistrationToken * token)
```

WebResourceResponseReceived is raised when the WebView receives the response for a request for a web resource (any URI resolution performed by the WebView; such as HTTP/HTTPS, file and data requests from redirects, navigations, declarations in HTML, implicit favicon lookups, and fetch API usage in the document). The host app can use this event to view the actual request and response for a web resource. There is no guarantee about the order in which the WebView processes the response and the host app's handler runs. The app's handler will not block the WebView from processing the response.

C++

```
CHECK_FAILURE(m_webView->add_WebResourceResponseReceived(  
    Callback<ICoreWebView2WebResourceResponseReceivedEventHandler>(  
        [this](  
            ICoreWebView2* sender,  
            ICoreWebView2WebResourceResponseReceivedEventArgs* args) {  
                wil::com_ptr<ICoreWebView2WebResourceRequest> request;  
                CHECK_FAILURE(args->get_Request(&request));  
                wil::unique_cotaskmem_string uri;  
                CHECK_FAILURE(request->get_Uri(&uri));  
                if (wcscmp(uri.get(),  
L"https://authenticationtest.com/HTTPAuth/") == 0)  
                {  
                    wil::com_ptr<ICoreWebView2HttpRequestHeaders>  
requestHeaders;  
                    CHECK_FAILURE(request->get_Headers(&requestHeaders));  
  
                    wil::unique_cotaskmem_string authHeaderValue;  
                    if (requestHeaders->GetHeader(L"Authorization",  
&authHeaderValue) == S_OK)  
                    {  
                        m_appWindow->AsyncMessageBox(  
                            std::wstring(L"Authorization: ") +  
authHeaderValue.get(),  
                            L"Authentication result");  
                        m_appWindow->DeleteComponent(this);  
                    }  
                }  
  
                return S_OK;  
            })
```

```
.Get(),
&m_webResourceResponseReceivedToken));
```

## get\_CookieManager

Gets the cookie manager object associated with this [ICoreWebView2](#).

```
public HRESULT get_CookieManager(ICoreWebView2CookieManager **  
cookieManager)
```

See [ICoreWebView2CookieManager](#).

C++

```
auto webview2_2 = m_webView.try_query<ICoreWebView2_2>();  
CHECK_FEATURE_RETURN_EMPTY(webview2_2);  
CHECK_FAILURE(webview2_2->get_CookieManager(&m_cookieManager));
```

## get\_Environment

Exposes the CoreWebView2Environment used to create this CoreWebView2.

```
public HRESULT get_Environment(ICoreWebView2Environment ** environment)
```

## NavigateWithWebResourceRequest

Navigates using a constructed WebResourceRequest object.

```
public HRESULT  
NavigateWithWebResourceRequest(ICoreWebView2WebResourceRequest * request)
```

This lets you provide post data or additional request headers during navigation. The headers in the WebResourceRequest override headers added by WebView2 runtime except for Cookie headers. Method can only be either "GET" or "POST". Provided post data will only be sent only if the method is "POST" and the uri scheme is HTTP(S).

C++

```
wil::com_ptr<ICoreWebView2Environment2> webviewEnvironment2;  
CHECK_FAILURE(appWindow->GetWebViewEnvironment()->QueryInterface(  
    IID_PPV_ARGS(&webviewEnvironment2)));  
wil::com_ptr<ICoreWebView2WebResourceRequest> webResourceRequest;  
wil::com_ptr<IStream> postDataStream = SHCreateMemStream(
```

```
    reinterpret_cast<const BYTE*>(postDataBytes.get()),  
    sizeNeededForMultiByte);  
  
    // This acts as a form submit to  
    https://www.w3schools.com/action\_page.php  
    CHECK_FAILURE(webviewEnvironment2->CreateWebResourceRequest(  
        L"https://www.w3schools.com/action\_page.php", L"POST",  
        postDataStream.get(),  
        L"Content-Type: application/x-www-form-urlencoded",  
        &webResourceRequest));  
    wil::com_ptr<ICoreWebView2_2> webview2;  
    CHECK_FAILURE(m_appWindow->GetWebView()->  
        QueryInterface(IID_PPV_ARGS(&webview2)));  
    CHECK_FAILURE(webview2->  
        NavigateWithWebResourceRequest(webResourceRequest.get()));
```

## **remove\_DOMContentLoaded**

Remove an event handler previously added with add\_DOMContentLoaded.

```
public HRESULT remove_DOMContentLoaded(EventRegistrationToken token)
```

## **remove\_WebResourceResponseReceived**

Remove an event handler previously added with add\_WebResourceResponseReceived.

```
public HRESULT remove_WebResourceResponseReceived(EventRegistrationToken  
token)
```

---

## **Feedback**

Was this page helpful?

 Yes

 No

# interface ICoreWebView2\_20

Article • 02/26/2024

```
interface ICoreWebView2_20
: public ICoreWebView2_19
```

This interface is an extension of [ICoreWebView2\\_19](#) that provides the `FrameId` property.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_FrameId</a>	The unique identifier of the main frame.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.2210.55
WebView2 Win32 Prerelease	1.0.2357

## Members

### [get\\_FrameId](#)

The unique identifier of the main frame.

```
public HRESULT get\_FrameId(UINT32 * id)
```

It's the same kind of ID as with the `FrameId` in `CoreWebView2Frame` and via `CoreWebView2FrameInfo`. Note that `FrameId` may not be valid if `CoreWebView2` has not

done any navigation. It's safe to get this value during or after the first ContentLoading event. Otherwise, it could return the invalid frame Id 0.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2\_21

Article • 02/26/2024

```
interface ICoreWebView2_21
: public ICoreWebView2_20
```

This is the interface for getting string and exception with ExecuteScriptWithResult.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">ExecuteScriptWithResult</a>	Run JavaScript code from the JavaScript parameter in the current top-level document rendered in the WebView.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.2277.86
WebView2 Win32 Prerelease	1.0.2357

## Members

### ExecuteScriptWithResult

Run JavaScript code from the JavaScript parameter in the current top-level document rendered in the WebView.

```
public HRESULT ExecuteScriptWithResult(LPCWSTR javaScript,
ICoreWebView2ExecuteScriptWithResultCompletedHandler * handler)
```

The result of the execution is returned asynchronously in the CoreWebView2ExecuteScriptResult object which has methods and properties to obtain the successful result of script execution as well as any unhandled JavaScript exceptions. If this method is run after the NavigationStarting event during a navigation, the script runs in the new document when loading it, around the time ContentLoading is run. This operation executes the script even if ICoreWebView2Settings::IsScriptEnabled is set to FALSE.

C++

```
void ScriptComponent::ExecuteScriptWithResult()
{
    TextInputDialog dialog(
        m_appWindow->GetMainWindow(), L"Execute Script With Result", L"Enter
script code:",
        L"Enter the JavaScript code to run in the webview.", L"");

    if (dialog.confirmed)
    {
        wil::com_ptr<ICoreWebView2_21> webView =
m_webView.try_query<ICoreWebView2_21>();

        if (!webView)
        {
            MessageBox(
                nullptr, L"Get webview2 failed!", L"ExecuteScriptWithResult
Result", MB_OK);
            return;
        }

        // The main interface for execute script, the first param is the
string
        // which user want to execute, the second param is the callback to
process
        // the result, here use a lamada to the param.
        webView->ExecuteScriptWithResult(
            dialog.input.c_str(),
            // The callback function has two param, the first one is the
status of call.s
            // it will always be the S_OK for now, and the second is the
result struct.
            Callback<ICoreWebView2ExecuteScriptWithResultCompletedHandler>(
                [this](HRESULT errorCode, ICoreWebView2ExecuteScriptResult*
result) -> HRESULT
            {
                if (errorCode != S_OK || result == nullptr)
                {
                    MessageBox(
                        nullptr, L"Call interface failed!",
                        L"ExecuteScriptWithResult Result", MB_OK);
                    return S_OK;
                }
                else

```

```

        {
            wil::com_ptr<ICoreWebView2ScriptException>
exception;

            BOOL isSuccess;

            // User should always invoke the get_Success firstly
to get if execute
            // success.
            if (result->get_Succeeded(&isSuccess) != S_OK)
{
            MessageBox(
                nullptr, L"Get execute status failed!",
                L"ExecuteScriptWithResult Result", MB_OK);
            return S_OK;
}

            // If execute success, then we can get the raw json
data, and try to get
            // the string.
            if (isSuccess)
{
            wil::unique_cotaskmem_string rawJsonData;
            // Get the raw json.
            if (result->get_ResultAsJson(&rawJsonData) ==
S_OK)
{
            MessageBox(
                nullptr, rawJsonData.get(),
                L"ExecuteScriptWithResult Json Result",
                MB_OK);
}
            else
{
            MessageBox(
                nullptr, L"Get raw json data failed",
                L"ExecuteScriptWithResult Json Result",
                MB_OK);
}
}

            // Get the string, and if the result is not the
string type,
            // it will return the E_INVALIDARG.
            wil::unique_cotaskmem_string stringData;
            BOOL isString = FALSE;
            if (result->TryGetResultAsString(&stringData,
&isString) == S_OK &&
                isString)
{
            MessageBox(
                nullptr, stringData.get(),
                L"ExecuteScriptWithResult String
Result", MB_OK);
}
            else
{
}
}

```

```

        MessageBox(
            nullptr, L"Get string failed",
            L"ExecuteScriptWithResult String
Result", MB_OK);
    }
}
else // If execute failed, then we can get the
exception struct to get
// the reason of failed.
{
    if (result->get_Exception(&exception) == S_OK)
    {
        // Get the exception name, this could return
        // such as `throw 1`.
        wil::unique_cotaskmem_string exceptionName;
        if (exception && exception-
>get_Name(&exceptionName) == S_OK)
        {
            MessageBox(
                nullptr, exceptionName.get(),
                L"ExecuteScriptWithResult Exception
Name", MB_OK);
        }

        // Get the exception message, this could
        // return the empty
        // string, such as `throw 1`.
        wil::unique_cotaskmem_string
exceptionMessage;
        if (exception &&
            exception-
>get_Message(&exceptionMessage) == S_OK)
        {
            MessageBox(
                nullptr, exceptionMessage.get(),
                L"ExecuteScriptWithResult Exception
Message", MB_OK);
        }

        // Get the exception detail, it's a json
        // struct data with all
        // exception infomation , we can parse it
        // and get the detail
        // what we need.
        wil::unique_cotaskmem_string
exceptionDetail;
        if (exception &&
            exception->get_ToJson(&exceptionDetail)
== S_OK)
        {
            MessageBox(
                nullptr, exceptionDetail.get(),
                L"ExecuteScriptWithResult Exception
Detail", MB_OK);
        }
    }
}

```

```

        }

        uint32_t lineNumber = 0;
        uint32_t columnNumber = 0;
        if (exception &&
            exception->get_LineNumber(&lineNumber)
== S_OK &&
            exception-
>get_ColumnNumber(&columnNumber) == S_OK)
{
    auto exceptionLocationInfo =
        L"LineNumber:" +
        std::wstring(lineNumber) +
        L", ColumnNumber:" +
        std::wstring(columnNumber);
    MessageBox(
        nullptr,
        exceptionLocationInfo.c_str(),
        L"ExecuteScriptWithResult Exception
Location", MB_OK);
}
else
{
    MessageBox(
        nullptr, L"Get exception failed",
        L"ExecuteScriptWithResult Result",
        MB_OK);
}
return S_OK;
})
.Get());
}
}

```

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2\_22

Article • 02/26/2024

```
interface ICoreWebView2_22
: public ICoreWebView2_21
```

This interface is an extension of [ICoreWebView2](#) that allows to set filters in order to receive `WebResourceRequested` events for service workers, shared workers and different origin iframes.

## Summary

[+] [Expand table](#)

Members	Descriptions
<a href="#">AddWebResourceRequestedFilterWithRequestSourceKinds</a>	A web resource request with a resource context that matches this filter's resource context and a URI that matches this filter's URI wildcard string for corresponding request sources will be raised via the <code>WebResourceRequested</code> event.
<a href="#">RemoveWebResourceRequestedFilterWithRequestSourceKinds</a>	Removes a matching <code>WebResource</code> filter that was previously added for the <code>WebResourceRequested</code> event.

## Applies to

[+] [Expand table](#)

Product	Introduced
WebView2 Win32	N/A
WebView2 Win32 Prerelease	1.0.2357

# Members

## AddWebResourceRequestedFilterWithRequestSourceKinds

A web resource request with a resource context that matches this filter's resource context and a URI that matches this filter's URI wildcard string for corresponding request sources will be raised via the `WebResourceRequested` event.

```
public HRESULT  
AddWebResourceRequestedFilterWithRequestSourceKinds(LPCWSTR const uri,  
COREWEBVIEW2_WEB_RESOURCE_CONTEXT const resourceContext,  
COREWEBVIEW2_WEB_RESOURCE_REQUEST_SOURCE_KINDS const  
requestSourceKinds)
```

To receive all raised events filters have to be added before main page navigation.

The `uri` parameter value is a wildcard string matched against the URI of the web resource request. This is a glob style wildcard string in which a `*` matches zero or more characters and a `?` matches exactly one character. These wildcard characters can be escaped using a backslash just before the wildcard character in order to represent the literal `*` or `?`.

The matching occurs over the URI as a whole string and not limiting wildcard matches to particular parts of the URI. The wildcard filter is compared to the URI after the URI has been normalized, any URI fragment has been removed, and non-ASCII hostnames have been converted to punycode.

Specifying a `nullptr` for the `uri` is equivalent to an empty string which matches no URIs.

For more information about resource context filters, navigate to [COREWEBVIEW2\\_WEB\\_RESOURCE\\_CONTEXT](#).

The `requestSourceKinds` is a mask of one or more `COREWEBVIEW2_WEB_RESOURCE_REQUEST_SOURCE_KINDS`. OR operation(s) can be applied to multiple `COREWEBVIEW2_WEB_RESOURCE_REQUEST_SOURCE_KINDS` to create a mask representing those data types. API returns `E_INVALIDARG` if `requestSourceKinds` equals to zero. For more information about request source kinds, navigate to [COREWEBVIEW2\\_WEB\\_RESOURCE\\_REQUEST\\_SOURCE\\_KINDS](#).

Because service workers and shared workers run separately from any one HTML document their `WebResourceRequested` will be raised for all CoreWebView2s that have

appropriate filters added in the corresponding CoreWebView2Environment. You should only add a WebResourceRequested filter for COREWEBVIEW2\_WEB\_RESOURCE\_REQUEST\_SOURCE\_KINDS\_SERVICE\_WORKER or COREWEBVIEW2\_WEB\_RESOURCE\_REQUEST\_SOURCE\_KINDS\_SHARED\_WORKER on one CoreWebView2 to avoid handling the same WebResourceRequested event multiple times.

[Expand table](#)

URI Filter String	Request URI	Match	Notes
*	https://contoso.com/a/b/c	Yes	A single * will match all URLs
*://contoso.com/*	https://contoso.com/a/b/c	Yes	Matches everything in contoso.com across all schemes
*://contoso.com/*	https://example.com/? https://contoso.com/	Yes	But also matches a URI with just the same text anywhere in the URI
example	https://contoso.com/example	No	The filter does not perform partial matches
*example	https://contoso.com/example	Yes	The filter matches across URI parts
*example	https://contoso.com/path/? example	Yes	The filter matches across URI parts
*example	https://contoso.com/path/? query#example	No	The filter is matched against the URI with no fragment
*example	https://example	No	The URI is normalized before filter matching so the actual URI used for comparison is https://example/
*example/	https://example	Yes	Just like above, but this time the filter ends with a / just like the normalized URI
https://xn--qei.example/	https://�����.example/	Yes	Non-ASCII hostnames are normalized to

URI Filter String	Request URI	Match	Notes
			punycode before wildcard comparison
https://&#x2764;.example/	https://xn--qei.example/	No	Non-ASCII hostnames are normalized to punycode before wildcard comparison

C++

```
wil::com_ptr<ICoreWebView2_22> webView =
m_webView.try_query<ICoreWebView2_22>();
if (webView)
{
    // Filter must be added for application to receive any
    WebResourceRequested event
    CHECK_FAILURE(webView-
>AddWebResourceRequestedFilterWithRequestSourceKinds(
    L"*worker.js", COREWEBVIEW2_WEB_RESOURCE_CONTEXT_ALL,
    COREWEBVIEW2_WEB_RESOURCE_REQUEST_SOURCE_KINDS_ALL));
    CHECK_FAILURE(m_webView->add_WebResourceRequested(
        Callback<ICoreWebView2WebResourceRequestedEventHandler>(
            [this](ICoreWebView2* sender,
ICoreWebView2WebResourceRequestedEventArgs* args)
        {
wil::com_ptr<ICoreWebView2WebResourceRequestedEventArgs2>
    webResourceRequestArgs;
    if (SUCCEEDED(args-
>QueryInterface(IID_PPV_ARGS(&webResourceRequestArgs))))
    {
        COREWEBVIEW2_WEB_RESOURCE_REQUEST_SOURCE_KINDS
requestSourceKind =
COREWEBVIEW2_WEB_RESOURCE_REQUEST_SOURCE_KINDS_ALL;
        CHECK_FAILURE(webResourceRequestArgs-
>get_RequestedSourceKind(
            &requestSourceKind));
        // Ensure that script is from shared worker source
        if (requestSourceKind ==
COREWEBVIEW2_WEB_RESOURCE_REQUEST_SOURCE_KINDS_SHARED_WORKER)
        {
            Microsoft::WRL::ComPtr<IStream> response_stream;
            CHECK_FAILURE(SHCreateStreamOnFileEx(
                L"assets/DemoWorker.js", STGM_READ,
FILE_ATTRIBUTE_NORMAL,
                FALSE, nullptr, &response_stream));
Microsoft::WRL::ComPtr<ICoreWebView2WebResourceResponse> response;
```

```

        // Get the default webview environment
        Microsoft::WRL::ComPtr<ICoreWebView2_2>
    webview2;
        CHECK_FAILURE(sender-
    >QueryInterface(IID_PPV_ARGS(&webview2)));

        Microsoft::WRL::ComPtr<ICoreWebView2Environment>
    environment;
        CHECK_FAILURE(webview2-
    >get_Environment(&environment));
        CHECK_FAILURE(environment-
    >CreateWebResourceResponse(
            response_stream.Get(), 200, L"OK", L"",
    &response));

        CHECK_FAILURE(args-
    >put_Response(response.Get()));
    }
}
return S_OK;
})
.Get(),
&m_webResourceRequestedToken));
}

```

## RemoveWebResourceRequestedFilterWithRequestSourceKinds

Removes a matching WebResource filter that was previously added for the `WebResourceRequested` event.

```

public HRESULT
RemoveWebResourceRequestedFilterWithRequestSourceKinds(LPCWSTR const uri,
COREWEBVIEW2_WEB_RESOURCE_CONTEXT const resourceContext,
COREWEBVIEW2_WEB_RESOURCE_REQUEST_SOURCE_KINDS const
requestSourceKinds)

```

If the same filter was added multiple times, then it must be removed as many times as it was added for the removal to be effective. Returns `E_INVALIDARG` for a filter that was not added or is already removed. If the filter was added for multiple requestSourceKinds and removed just for one of them the filter remains for the non-removed requestSourceKinds.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2\_3

Article • 02/26/2024

```
interface ICoreWebView2_3
: public ICoreWebView2_2
```

A continuation of the [ICoreWebView2\\_2](#) interface.

## Summary

[+] [Expand table](#)

Members	Descriptions
<a href="#">ClearVirtualHostNameToFolderMapping</a>	Clears a host name mapping for local folder that was added by <a href="#">SetVirtualHostNameToFolderMapping</a> .
<a href="#">get_IsSuspended</a>	Whether WebView is suspended.
<a href="#">Resume</a>	Resumes the WebView so that it resumes activities on the web page.
<a href="#">SetVirtualHostNameToFolderMapping</a>	Sets a mapping between a virtual host name and a folder path to make available to web sites via that host name.
<a href="#">TrySuspend</a>	An app may call the <a href="#">TrySuspend</a> API to have the WebView2 consume less memory.

## Applies to

[+] [Expand table](#)

Product	Introduced
WebView2 Win32	1.0.774.44
WebView2 Win32 Prerelease	1.0.790

## Members

## ClearVirtualHostNameToFolderMapping

Clears a host name mapping for local folder that was added by

`SetVirtualHostNameToFolderMapping`.

```
public HRESULT ClearVirtualHostNameToFolderMapping(LPCWSTR hostName)
```

## get\_IsSuspended

Whether WebView is suspended.

```
public HRESULT get_IsSuspended(BOOL * isSuspended)
```

`TRUE` when WebView is suspended, from the time when TrySuspend has completed successfully until WebView is resumed.

## Resume

Resumes the WebView so that it resumes activities on the web page.

```
public HRESULT Resume()
```

This API can be called while the WebView2 controller is invisible. The app can interact with the WebView immediately after `Resume`. WebView will be automatically resumed when it becomes visible.

C++

```
if (message == WM_SIZE)
{
    if (wParam == SIZE_MINIMIZED)
    {
        // Hide the webview when the app window is minimized.
        m_controller->put_IsVisible(FALSE);
        Suspend();
    }
    else if (wParam == SIZE_RESTORED)
    {
        // When the app window is restored, show the webview
        // (unless the user has toggle visibility off).
        if (m_isVisible)
        {
            Resume();
            m_controller->put_IsVisible(TRUE);
        }
    }
}
```

```
    }  
}
```

C++

```
void ViewComponent::Resume()  
{  
    wil::com_ptr<ICoreWebView2_3> webView;  
    webView = m_webView.try_query<ICoreWebView2_3>();  
    CHECK_FEATURE_RETURN_EMPTY(webView);  
    webView->Resume();  
}
```

## SetVirtualHostNameToFolderMapping

Sets a mapping between a virtual host name and a folder path to make available to web sites via that host name.

```
public HRESULT SetVirtualHostNameToFolderMapping(LPCWSTR hostName,  
LPCWSTR folderPath, COREWEBVIEW2_HOST_RESOURCE_ACCESS_KIND accessKind)
```

After setting the mapping, documents loaded in the WebView can use HTTP or HTTPS URLs at the specified host name specified by hostName to access files in the local folder specified by folderPath.

This mapping applies to both top-level document and iframe navigations as well as subresource references from a document. This also applies to web workers including dedicated/shared worker and service worker, for loading either worker scripts or subresources (importScripts(), fetch(), XHR, etc.) issued from within a worker. For virtual host mapping to work with service worker, please keep the virtual host name mappings consistent among all WebViews sharing the same browser instance. As service worker works independently of WebViews, we merge mappings from all WebViews when resolving virtual host name, inconsistent mappings between WebViews would lead unexpected behavior.

Due to a current implementation limitation, media files accessed using virtual host name can be very slow to load. As the resource loaders for the current page might have already been created and running, changes to the mapping might not be applied to the current page and a reload of the page is needed to apply the new mapping.

Both absolute and relative paths are supported for folderPath. Relative paths are interpreted as relative to the folder where the exe of the app is in.

Note that the `FolderPath` length must not exceed the Windows MAX\_PATH limit.

`accessKind` specifies the level of access to resources under the virtual host from other sites.

For example, after calling

C++

```
SetVirtualHostNameToFolderMapping(  
    L"appassets.example", L"assets",  
    COREWEBVIEW2_HOST_RESOURCE_ACCESS_KIND_DENY);
```

navigating to `https://appassets.example/my-local-file.html` will show the content from `my-local-file.html` in the `assets` subfolder located on disk under the same path as the app's executable file.

DOM elements that want to reference local files will have their host reference virtual host in the source. If there are multiple folders being used, define one unique virtual host per folder. For example, you can embed a local image like this

C++

```
WCHAR c_navString[] = L"<img src=\"http://appassets.example/wv2.png\"/>";  
m_webView->NavigateToString(c_navString);
```

The example above shows the image `wv2.png` by resolving the folder mapping above.

You should typically choose virtual host names that are never used by real sites. If you own a domain such as `example.com`, another option is to use a subdomain reserved for the app (like `my-app.example.com`).

[RFC 6761](#) has reserved several special-use domain names that are guaranteed to not be used by real sites (for example, `.example`, `.test`, and `.invalid`.)

Note that using `.local` as the top-level domain name will work but can cause a delay during navigations. You should avoid using `.local` if you can.

Apps should use distinct domain names when mapping folder from different sources that should be isolated from each other. For instance, the app might use `app-file.example` for files that ship as part of the app, and `book1.example` might be used for files containing books from a less trusted source that were previously downloaded and saved to the disk by the app.

The host name used in the APIs is canonicalized using Chromium's host name parsing logic before being used internally. For more information see [HTML5 2.6 URLs](#).

All host names that are canonicalized to the same string are considered identical. For example, `EXAMPLE.COM` and `example.com` are treated as the same host name. An international host name and its Punycode-encoded host name are considered the same host name. There is no DNS resolution for host name and the trailing '.' is not normalized as part of canonicalization.

Therefore `example.com` and `example.com.` are treated as different host names. Similarly, `virtual-host-name` and `virtual-host-name.example.com` are treated as different host names even if the machine has a DNS suffix of `example.com`.

Specify the minimal cross-origin access necessary to run the app. If there is not a need to access local resources from other origins, use `COREWEBVIEW2_HOST_RESOURCE_ACCESS_KIND_DENY`.

C++

```
// Setup host resource mapping for local files.  
m_webView3->SetVirtualHostNameToFolderMapping(  
    L"appassets.example", L"assets",  
    COREWEBVIEW2_HOST_RESOURCE_ACCESS_KIND_DENY_CORS);
```

C++

```
const std::wstring localFileRootUrl = L"https://appassets.example/";  
return localFileRootUrl + regex_replace(relativePath,  
std::wregex(L"\\""/"), L"/");
```

## TrySuspend

An app may call the `TrySuspend` API to have the WebView2 consume less memory.

```
public HRESULT TrySuspend(ICoreWebView2TrySuspendCompletedHandler *  
handler)
```

This is useful when a Win32 app becomes invisible, or when a Universal Windows Platform app is being suspended, during the suspended event handler before completing the suspended event. The CoreWebView2Controller's `IsVisible` property must be false when the API is called. Otherwise, the API fails with

`HRESULT_FROM_WIN32(ERROR_INVALID_STATE)`. Suspending is similar to putting a tab to

sleep in the Edge browser. Suspending pauses WebView script timers and animations, minimizes CPU usage for the associated browser renderer process and allows the operating system to reuse the memory that was used by the renderer process for other processes. Note that Suspend is best effort and considered completed successfully once the request is sent to browser renderer process. If there is a running script, the script will continue to run and the renderer process will be suspended after that script is done. See [Sleeping Tabs FAQ](#) for conditions that might prevent WebView from being suspended. In those situations, the completed handler will be invoked with isSuccessful as false and errorCode as S\_OK. The WebView will be automatically resumed when it becomes visible. Therefore, the app normally does not have to call `Resume` explicitly. The app can call `Resume` and then `TrySuspend` periodically for an invisible WebView so that the invisible WebView can sync up with latest data and the page ready to show fresh content when it becomes visible. All WebView APIs can still be accessed when a WebView is suspended. Some APIs like Navigate will auto resume the WebView. To avoid unexpected auto resume, check `IsSuspended` property before calling APIs that might change WebView state.

C++

```
if (message == WM_SIZE)
{
    if (wParam == SIZE_MINIMIZED)
    {
        // Hide the webview when the app window is minimized.
        m_controller->put_IsVisible(FALSE);
        Suspend();
    }
    else if (wParam == SIZE_RESTORED)
    {
        // When the app window is restored, show the webview
        // (unless the user has toggle visibility off).
        if (m_isVisible)
        {
            Resume();
            m_controller->put_IsVisible(TRUE);
        }
    }
}
```

C++

```
void ViewComponent::Suspend()
{
    wil::com_ptr<ICoreWebView2_3> webView;
    webView = m_webView.try_query<ICoreWebView2_3>();
    CHECK_FEATURE_RETURN_EMPTY(webView);
    HRESULT hr = webView->TrySuspend(
```

```
Callback<ICoreWebView2TrySuspendCompletedHandler>(
    [this](HRESULT errorCode, BOOL.isSuccessful) -> HRESULT {
        if ((errorCode != S_OK) || !isSuccessful)
        {
            std::wstringstream formattedMessage;
            formattedMessage << "TrySuspend result (0x" << std::hex
            << errorCode
                           << ") " << (isSuccessful ? "succeeded"
                           : "failed");
            m_appWindow->AsyncMessageBox(
                std::move(formattedMessage.str()), L"TrySuspend
result");
        }
        return S_OK;
    })
    .Get());
if (FAILED(hr))
    ShowFailure(hr, L"Call to TryFreeze failed");
}
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2\_4

Article • 02/26/2024

```
interface ICoreWebView2_4
: public ICoreWebView2_3
```

A continuation of the [ICoreWebView2\\_3](#) interface to support FrameCreated and DownloadStarting events.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">add_DownloadStarting</a>	Add an event handler for the <code>DownloadStarting</code> event.
<a href="#">add_FrameCreated</a>	Raised when a new iframe is created.
<a href="#">remove_DownloadStarting</a>	Remove an event handler previously added with <code>add_DownloadStarting</code> .
<a href="#">remove_FrameCreated</a>	Remove an event handler previously added with <code>add_FrameCreated</code> .

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.902.49
WebView2 Win32 Prerelease	1.0.902

## Members

### [add\\_DownloadStarting](#)

Add an event handler for the `DownloadStarting` event.

```
public HRESULT  
add_DownloadStarting(ICoreWebView2DownloadStartingEventHandler *  
eventHandler, EventRegistrationToken * token)
```

This event is raised when a download has begun, blocking the default download dialog, but not blocking the progress of the download.

The host can choose to cancel a download, change the result file path, and hide the default download dialog. If the host chooses to cancel the download, the download is not saved, no dialog is shown, and the state is changed to COREWEBVIEW2\_DOWNLOAD\_STATE\_INTERRUPTED with interrupt reason COREWEBVIEW2\_DOWNLOAD\_INTERRUPT\_REASON\_USER\_CANCELED. Otherwise, the download is saved to the default path after the event completes, and default download dialog is shown if the host did not choose to hide it. The host can change the visibility of the download dialog using the `Handled` property. If the event is not handled, downloads complete normally with the default dialog shown.

C++

```
// Register a handler for the `DownloadStarting` event.  
// This example hides the default download dialog and shows a dialog box  
instead.  
// The dialog box displays the default result file path and allows the  
user to specify a different path.  
// Selecting `OK` will save the download to the chosen path.  
// Selecting `CANCEL` will cancel the download.  
m_demoUri = L"https://demo.smartscreen.msft.net/";  
  
m_webView2_4 = m_webView.try_query<ICoreWebView2_4>();  
if (m_webView2_4) {  
    CHECK_FAILURE(m_webView2_4->add_DownloadStarting(  
        Callback<ICoreWebView2DownloadStartingEventHandler>(  
            [this](  
                ICoreWebView2* sender,  
                ICoreWebView2DownloadStartingEventArgs* args) -> HRESULT  
            {  
                // We avoid potential reentrancy from running a message  
                // loop in the download  
                // starting event handler by showing our download dialog  
                // via this lambda run  
                // asynchronously later outside of this event handler.  
                Note that a long running  
                // synchronous UI prompt or other blocking item on the  
                // UI thread can potentially  
                // block the WebView2 from doing anything.  
                auto showDialog = [this, args]  
                {  
                    // Hide the default download dialog.  
                    CHECK_FAILURE(args->put_Handled(TRUE));
```

```

        wil::com_ptr<ICoreWebView2DownloadOperation>
download;
        CHECK_FAILURE(args-
>get_DownloadOperation(&download));

        INT64 totalBytesToReceive = 0;
        CHECK_FAILURE(download-
>get_TotalBytesToReceive(&totalBytesToReceive));

        wil::unique_cotaskmem_string uri;
        CHECK_FAILURE(download->get_Uri(&uri));

        wil::unique_cotaskmem_string mimeType;
        CHECK_FAILURE(download->get_MimeType(&mimeType));

        wil::unique_cotaskmem_string contentDisposition;
        CHECK_FAILURE(download-
>get_ContentDisposition(&contentDisposition));

        // Get the suggested path from the event args.
        wil::unique_cotaskmem_string resultFilePath;
        CHECK_FAILURE(args-
>get_ResultFilePath(&resultFilePath));

        std::wstring prompt =
            std::wstring(
                L"Enter result file path or select `OK` to
use default path. "
                L"Select `Cancel` to cancel the download.");
        std::wstring description = std::wstring(L"URI: ") +
uri.get() + L"\r\n" +
                L"Mime type: " +
mimeType.get() + L"\r\n";
        if (totalBytesToReceive >= 0)
        {
            description = description + L"Total bytes to
receive: " +
std::to_wstring(totalBytesToReceive) + L"\r\n";
        }

        TextInputDialog dialog(
            m_appWindow->GetMainWindow(), L"Download
Starting", prompt.c_str(),
            description.c_str(), resultFilePath.get());
        if (dialog.confirmed)
        {
            // If user selects `OK`, the download will
complete normally.
            // Result file path will be updated if a new one
was provided.
            CHECK_FAILURE(args-
>put_ResultFilePath(dialog.input.c_str()));

```

```

        UpdateProgress(download.get());
    }
    else
    {
        // If user selects `Cancel`, the download will
be canceled.
        CHECK_FAILURE(args->put_Cancel(TRUE));
    }
};

// Obtain a deferral for the event so that the
CoreWebView2
// doesn't examine the properties we set on the event
args until
// after we call the Complete method asynchronously
later.
wil::com_ptr<ICoreWebView2Deferral> deferral;
CHECK_FAILURE(args->GetDeferral(&deferral));

// We avoid potential reentrancy from running a message
loop in the download
// starting event handler by showing our download dialog
later when we
// complete the deferral asynchronously.
m_appWindow->RunAsync([deferral, showDialog]() {
    showDialog();
    CHECK_FAILURE(deferral->Complete());
});

return S_OK;
})
.Get(),
&m_downloadStartingToken));
}
else
{
    // This scenario component is useless without this feature, but
deleting an object in its own
    // constructor is...not advisable, so we'll just let this component
do nothing until it goes away.
    FeatureNotAvailable();
}

```

## **add\_FrameCreated**

Raised when a new iframe is created.

```
public HRESULT add_FrameCreated(ICoreWebView2FrameCreatedEventHandler *
eventHandler, EventRegistrationToken * token)
```

Handle this event to get access to [ICoreWebView2Frame](#) objects. Use [ICoreWebView2Frame.add\\_Destroyed](#) to listen for when this iframe goes away.

## **remove\_DownloadStarting**

Remove an event handler previously added with `add_DownloadStarting`.

```
public HRESULT remove_DownloadStarting(EventRegistrationToken token)
```

## **remove\_FrameCreated**

Remove an event handler previously added with `add_FrameCreated`.

```
public HRESULT remove_FrameCreated(EventRegistrationToken token)
```

---

## **Feedback**

Was this page helpful?

 Yes

 No

# interface ICoreWebView2\_5

Article • 02/26/2024

```
interface ICoreWebView2_5
: public ICoreWebView2_4
```

A continuation of the [ICoreWebView2\\_4](#) interface to support ClientCertificateRequested event.

## Summary

[+] [Expand table](#)

Members	Descriptions
<a href="#">add_ClientCertificateRequested</a>	Add an event handler for the ClientCertificateRequested event.
<a href="#">remove_ClientCertificateRequested</a>	Remove an event handler previously added with add_ClientCertificateRequested.

## Applies to

[+] [Expand table](#)

Product	Introduced
WebView2 Win32	1.0.961.33
WebView2 Win32 Prerelease	1.0.955

## Members

### [add\\_ClientCertificateRequested](#)

Add an event handler for the ClientCertificateRequested event.

```
public HRESULT  
add_ClientCertificateRequested(ICoreWebView2ClientCertificateRequestedEventHandler * eventHandler, EventRegistrationToken * token)
```

The ClientCertificateRequested event is raised when the WebView2 is making a request to an HTTP server that needs a client certificate for HTTP authentication. Read more about HTTP client certificates at [RFC 8446 The Transport Layer Security \(TLS\) Protocol Version 1.3](#).

With this event you have several options for responding to client certificate requests:

[+] Expand table

Scenario	Handled	Cancel	SelectedCertificate
Respond to server with a certificate	True	False	MutuallyTrustedCertificate value
Respond to server without certificate	True	False	null
Display default client certificate selection dialog prompt	False	False	n/a
Cancel the request	n/a	True	n/a

If you don't handle the event, WebView2 will show the default client certificate selection dialog prompt to user.

C++

```
void SettingsComponent::EnableCustomClientCertificateSelection()  
{  
    if (m_webView2_5)  
    {  
        if (m_ClientCertificateRequestedToken.value == 0)  
        {  
            CHECK_FAILURE(m_webView2_5->add_ClientCertificateRequested(  
                Callback<ICoreWebView2ClientCertificateRequestedEventHandler>(  
                    [this](  
                        ICoreWebView2* sender,  
                        ICoreWebView2ClientCertificateRequestedEventArgs*  
args)  
                {  
  
                    wil::com_ptr<ICoreWebView2ClientCertificateCollection>  
                    certificateCollection;  
                    CHECK_FAILURE(  
                        args->
```

```

>get_MutuallyTrustedCertificates(&certificateCollection));

    UINT certificateCollectionCount = 0;
    CHECK_FAILURE(
        certificateCollection-
>get_Count(&certificateCollectionCount));
    wil::com_ptr<ICoreWebView2ClientCertificate>
certificate = nullptr;

    if (certificateCollectionCount > 0)
    {
        // There is no significance to the order,
picking a certificate
        // arbitrarily.
        CHECK_FAILURE(certificateCollection-
>GetValueAtIndex(
            certificateCollectionCount - 1,
&certificate));
        // Continue with the selected certificate to
respond to the server.
        CHECK_FAILURE(args-
>put_SelectedCertificate(certificate.get()));
        CHECK_FAILURE(args->put_Handled(TRUE));
    }
    else
    {
        // Continue without a certificate to respond to
the server if
        // certificate collection is empty.
        CHECK_FAILURE(args->put_Handled(TRUE));
    }
    return S_OK;
}
.Get(),
&m_ClientCertificateRequestedToken));
}
else
{
    CHECK_FAILURE(m_webView2_5->remove_ClientCertificateRequested(
        m_ClientCertificateRequestedToken));
    m_ClientCertificateRequestedToken.value = 0;
}
}
}

```

C++

```

m_webView2_5 = m_webView.try_query<ICoreWebView2_5>();
if (m_webView2_5)
{
    CHECK_FAILURE(
        m_webView2_5->add_ClientCertificateRequested(

```

```
Callback<ICoreWebView2ClientCertificateRequestedEventHandler>(
    [this](
        ICoreWebView2* sender,
        ICoreWebView2ClientCertificateRequestedEventArgs*
args) {
    auto showDialog = [this, args] {

wil::com_ptr<ICoreWebView2ClientCertificateCollection>
    certificateCollection;
    CHECK_FAILURE(args-
>get_MutuallyTrustedCertificates(&certificateCollection));

    wil::unique_cotaskmem_string host;
    CHECK_FAILURE(args->get_Host(&host));

    INT port = FALSE;
    CHECK_FAILURE(args->get_Port(&port));

    UINT certificateCollectionCount;
    CHECK_FAILURE(certificateCollection-
>get_Count(&certificateCollectionCount));

    wil::com_ptr<ICoreWebView2ClientCertificate>
certificate = nullptr;

    if (certificateCollectionCount > 0)
    {
        ClientCertificate clientCertificate;
        for (UINT i = 0; i <
certificateCollectionCount; i++)
        {
            CHECK_FAILURE(
                certificateCollection-
>GetValueAtIndex(i, &certificate));
            CHECK_FAILURE(certificate-
>get_Subject(&clientCertificate.Subject));
            CHECK_FAILURE(certificate-
>get_DisplayName(&clientCertificate.DisplayName));
            CHECK_FAILURE(certificate-
>get_Issuer(&clientCertificate.Issuer)));
            COREWEBVIEW2_CLIENT_CERTIFICATE_KIND
Kind;
            CHECK_FAILURE(
                certificate->get_Kind(&Kind));
            clientCertificate.CertificateKind =
NameOfCertificateKind(Kind);
            CHECK_FAILURE(certificate-
>get_ValidFrom(&clientCertificate.ValidFrom));
            CHECK_FAILURE(certificate-
```

```

>get_ValidTo(&clientCertificate.ValidTo));

clientCertificates_.push_back(clientCertificate);
    }

        // Display custom dialog box for the client
certificate selection.

        ClientCertificateSelectionDialog dialog(
            m_appWindow->GetMainWindow(), L"Select a
Certificate for authentication", host.get(), port, clientCertificates_);

        if (dialog.confirmed)
        {
            int selectedIndex = dialog.selectedItem;
            if (selectedIndex >= 0)
            {
                CHECK_FAILURE(
                    certificateCollection-
>GetValueAtIndex(selectedIndex, &certificate));
                    // Continue with the selected
certificate to respond to the server if `OK` is selected.
                    CHECK_FAILURE(args-
>put_SelectedCertificate(certificate.get()));
                }
            }
            // Continue without a certificate to respond
to the server if `CANCEL` is selected.
            CHECK_FAILURE(args->put_Handled(TRUE));
        }
        else
        {
            // Continue without a certificate to respond
to the server if certificate collection is empty.
            CHECK_FAILURE(args->put_Handled(TRUE));
        }
    };

        // Obtain a deferral for the event so that the
CoreWebView2
            // doesn't examine the properties we set on the
event args and
            // after we call the Complete method asynchronously
later.

        wil::com_ptr<ICoreWebView2Deferral> deferral;
        CHECK_FAILURE(args->GetDeferral(&deferral));

            // complete the deferral asynchronously.
        m_appWindow->RunAsync([deferral, showDialog]() {
            showDialog();
            CHECK_FAILURE(deferral->Complete());
        });

        return S_OK;
    })

```

```
        .Get(),
        &m_ClientCertificateRequestedToken));  
  
    MessageBox(  
        nullptr, L"Custom Client Certificate selection dialog will be  
used next when WebView2 "  
        L"is making a request to an HTTP server that needs a client  
certificate.",  
        L"Client certificate selection", MB_OK);  
    }  
    else  
    {  
        FeatureNotAvailable();  
    }  
}
```

## remove\_ClientCertificateRequested

Remove an event handler previously added with add\_ClientCertificateRequested.

```
public HRESULT remove_ClientCertificateRequested(EventRegistrationToken token)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2\_6

Article • 02/26/2024

```
interface ICoreWebView2_6
: public ICoreWebView2_5
```

This interface is an extension of [ICoreWebView2\\_5](#) that manages opening the browser task manager window.

## Summary

[+] [Expand table](#)

Members	Descriptions
<a href="#">OpenTaskManagerWindow</a>	Opens the Browser Task Manager view as a new window in the foreground.

## Applies to

[+] [Expand table](#)

Product	Introduced
WebView2 Win32	1.0.992.28
WebView2 Win32 Prerelease	1.0.1010

## Members

### [OpenTaskManagerWindow](#)

Opens the Browser Task Manager view as a new window in the foreground.

```
public HRESULT OpenTaskManagerWindow()
```

If the Browser Task Manager is already open, this will bring it into the foreground. WebView2 currently blocks the Shift+Esc shortcut for opening the task manager. An end user can open the browser task manager manually via the `Browser task manager` entry of the DevTools window's title bar's context menu.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2\_7

Article • 02/26/2024

```
interface ICoreWebView2_7
: public ICoreWebView2_6
```

This interface is an extension of [ICoreWebView2\\_6](#) that supports printing to PDF.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">PrintToPdf</a>	Print the current page to PDF asynchronously with the provided settings.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1020.30
WebView2 Win32 Prerelease	1.0.1056

## Members

### [PrintToPdf](#)

Print the current page to PDF asynchronously with the provided settings.

```
public HRESULT PrintToPdf(LPCWSTR resultFilePath, ICoreWebView2PrintSettings *  
printSettings, ICoreWebView2PrintToPdfCompletedHandler * handler)
```

See [ICoreWebView2PrintSettings](#) for description of settings. Passing nullptr for `printSettings` results in default print settings used.

Use `resultFilePath` to specify the path to the PDF file. The host should provide an absolute path, including file name. If the path points to an existing file, the file will be overwritten. If the path is not valid, the method fails with `E_INVALIDARG`.

The async `PrintToPdf` operation completes when the data has been written to the PDF file. At this time the `ICoreWebView2PrintToPdfCompletedHandler` is invoked. If the application exits before printing is complete, the file is not saved. Only one `Printing` operation can be in progress at a time. If `PrintToPdf` is called while a `PrintToPdf` or `PrintToPdfStream` or `Print` or `ShowPrintUI` job is in progress, the completed handler is immediately invoked with `isSuccessful` set to FALSE.

C++

```
// Shows the user a file selection dialog, then uses the selected path when
// printing to PDF. If `enableLandscape` is true, the page is printed
// in landscape mode, otherwise the page is printed in portrait mode.
void FileComponent::PrintToPdf(bool enableLandscape)
{
    WCHAR defaultName[MAX_PATH] = L"WebView2_PrintedPdf.pdf";
    OPENFILENAME openFileName = CreateOpenFileName(defaultName, L"PDF
File\0*.pdf\0");
    if (GetSaveFileName(&openFileName))
    {
        wil::com_ptr<ICoreWebView2PrintSettings> printSettings = nullptr;
        if (enableLandscape)
        {
            wil::com_ptr<ICoreWebView2Environment6> webviewEnvironment6;
            CHECK_FAILURE(m_appWindow->GetWebViewEnvironment()-
>QueryInterface(
                IID_PPV_ARGS(&webviewEnvironment6)));
            if (webviewEnvironment6)
            {
                CHECK_FAILURE(webviewEnvironment6-
>CreatePrintSettings(&printSettings));
                CHECK_FAILURE(
                    printSettings-
>put_Orientation(COREWEBVIEW2_PRINT_ORIENTATION_LANDSCAPE));
            }
        }

        wil::com_ptr<ICoreWebView2_7> webview2_7;
        CHECK_FAILURE(m_webView->QueryInterface(IID_PPV_ARGS(&webview2_7)));
        if (webview2_7)
        {
            m_printToPdfInProgress = true;
            CHECK_FAILURE(webview2_7->PrintToPdf(
                openFileName.lpszFile, printSettings.get(),
                Callback<ICoreWebView2PrintToPdfCompletedHandler>(
                    [this](HRESULT errorCode, BOOL isSuccessful) -> HRESULT
{
                CHECK_FAILURE(errorCode);
            }
        }
    }
}
```

```
    m_printToPdfInProgress = false;
    m_appWindow->AsyncMessageBox(
        (isSuccessful) ? L"Print to PDF succeeded"
                      : L"Print to PDF failed",
        L"Print to PDF Completed");
    return S_OK;
}
}
}
```

---

## Feedback

Was this page helpful?



# interface ICoreWebView2\_8

Article • 02/26/2024

```
interface ICoreWebView2_8
: public ICoreWebView2_7
```

This interface is an extension of [ICoreWebView2\\_7](#) that supports media features.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">add_IsDocumentPlayingAudioChanged</a>	Adds an event handler for the <code>IsDocumentPlayingAudioChanged</code> event.
<a href="#">add_IsMutedChanged</a>	Adds an event handler for the <code>IsMutedChanged</code> event.
<a href="#">get_IsDocumentPlayingAudio</a>	Indicates whether any audio output from this CoreWebView2 is playing.
<a href="#">get_IsMuted</a>	Indicates whether all audio output from this CoreWebView2 is muted or not.
<a href="#">put_IsMuted</a>	Sets the <code>IsMuted</code> property.
<a href="#">remove_IsDocumentPlayingAudioChanged</a>	Remove an event handler previously added with <code>add_IsDocumentPlayingAudioChanged</code> .
<a href="#">remove_IsMutedChanged</a>	Remove an event handler previously added with <code>add_IsMutedChanged</code> .

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1072.54
WebView2 Win32 Prerelease	1.0.1083

# Members

## add\_IsDocumentPlayingAudioChanged

Adds an event handler for the `IsDocumentPlayingAudioChanged` event.

```
public HRESULT  
add_IsDocumentPlayingAudioChanged(ICoreWebView2IsDocumentPlayingAudioCh  
angedEventHandler * eventHandler, EventRegistrationToken * token)
```

`IsDocumentPlayingAudioChanged` is raised when the `IsDocumentPlayingAudio` property changes value.

C++

```
AudioComponent::AudioComponent(AppWindow* appWindow)  
    : m_appWindow(appWindow), m_webView(appWindow->GetWebView())  
{  
    auto webview2_8 = m_webView.try_query<ICoreWebView2_8>();  
    if (webview2_8)  
    {  
        // Register a handler for the IsDocumentPlayingAudioChanged event.  
        CHECK_FAILURE(webview2_8->add_IsDocumentPlayingAudioChanged(  
            Callback<ICoreWebView2IsDocumentPlayingAudioChangedEventHandler>  
(  
                [this, webview2_8](ICoreWebView2* sender, IUnknown* args) ->  
HRESULT  
                {  
                    UpdateTitleWithMuteState(webview2_8);  
                    return S_OK;  
                })  
            .Get(),  
            &m_isDocumentPlayingAudioChangedToken));  
  
        // Register a handler for the IsMutedChanged event.  
        CHECK_FAILURE(webview2_8->add_IsMutedChanged(  
            Callback<ICoreWebView2IsMutedChangedEventHandler>(  
                [this, webview2_8](ICoreWebView2* sender, IUnknown* args) ->  
HRESULT  
                {  
                    UpdateTitleWithMuteState(webview2_8);  
                    return S_OK;  
                })  
            .Get(),  
            &m_isMutedChangedToken));  
    }  
}  
  
bool AudioComponent::HandleWindowMessage(
```

```

    HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam, LRESULT* result)
{
    if (message == WM_COMMAND)
    {
        switch (LOWORD(wParam))
        {
            case IDM_TOGGLE_MUTE_STATE:
                ToggleMuteState();
                return true;
        }
    }
    return false;
}

// Toggle the mute state of the current window and show a mute or unmute
// icon on the title bar
void AudioComponent::ToggleMuteState()
{
    auto webview2_8 = m_webView.try_query<ICoreWebView2_8>();
    if (webview2_8)
    {
        BOOL isMuted;
        CHECK_FAILURE(webview2_8->get_IsMuted(&isMuted));
        CHECK_FAILURE(webview2_8->put_IsMuted(!isMuted));
        std::wstring result = !isMuted ? L"WebView is Now Muted" :
L"WebView is Now Unmuted";
        MessageBox(nullptr, result.c_str(), L"Mute State Changed", MB_OK);
    }
}

void AudioComponent::UpdateTitleWithMuteState(wil::com_ptr<ICoreWebView2_8>
webview2_8)
{
    BOOL isDocumentPlayingAudio;
    CHECK_FAILURE(webview2_8-
>get_IsDocumentPlayingAudio(&isDocumentPlayingAudio));

    BOOL isMuted;
    CHECK_FAILURE(webview2_8->get_IsMuted(&isMuted));

    wil::unique_cotaskmem_string title;
    CHECK_FAILURE(m_webView->get_DocumentTitle(&title));
    std::wstring result = L"";

    if (isDocumentPlayingAudio)
    {
        if (isMuted)
        {
            result = L"???? " + std::wstring(title.get());
        }
        else
        {
            result = L"???? " + std::wstring(title.get());
        }
    }
}

```

```
    else
    {
        result = std::wstring(title.get());
    }

    m_appWindow->SetDocumentTitle(result.c_str());
}
```

## add\_IsMutedChanged

Adds an event handler for the `IsMutedChanged` event.

```
public HRESULT
add_IsMutedChanged(ICoreWebView2IsMutedChangedEventHandler *
eventHandler, EventRegistrationToken * token)
```

`IsMutedChanged` is raised when the `IsMuted` property changes value.

C++

```
AudioComponent::AudioComponent(AppWindow* appWindow)
    : m_appWindow(appWindow), m_webView(appWindow->GetWebView())
{
    auto webview2_8 = m_webView.try_query<ICoreWebView2_8>();
    if (webview2_8)
    {
        // Register a handler for the IsDocumentPlayingAudioChanged event.
        CHECK_FAILURE(webview2_8->add_IsDocumentPlayingAudioChanged(
            Callback<ICoreWebView2IsDocumentPlayingAudioChangedEventHandler>
(
                [this, webview2_8](ICoreWebView2* sender, IUnknown* args) ->
HRESULT
{
            UpdateTitleWithMuteState(webview2_8);
            return S_OK;
        })
        .Get(),
        &m_isDocumentPlayingAudioChangedToken));

        // Register a handler for the IsMutedChanged event.
        CHECK_FAILURE(webview2_8->add_IsMutedChanged(
            Callback<ICoreWebView2IsMutedChangedEventHandler>(
                [this, webview2_8](ICoreWebView2* sender, IUnknown* args) ->
HRESULT
{
            UpdateTitleWithMuteState(webview2_8);
            return S_OK;
        })
        .Get(),
        &m_isMutedChangedToken));
    }
}
```

```

        }

    }

    bool AudioComponent::HandleWindowMessage(
        HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam, LRESULT* result)
    {
        if (message == WM_COMMAND)
        {
            switch (LOWORD(wParam))
            {
                case IDM_TOGGLE_MUTE_STATE:
                    ToggleMuteState();
                    return true;
            }
        }
        return false;
    }

    // Toggle the mute state of the current window and show a mute or unmute
    // icon on the title bar
    void AudioComponent::ToggleMuteState()
    {
        auto webview2_8 = m_webView.try_query<ICoreWebView2_8>();
        if (webview2_8)
        {
            BOOL isMuted;
            CHECK_FAILURE(webview2_8->get_IsMuted(&isMuted));
            CHECK_FAILURE(webview2_8->put_IsMuted(!isMuted));
            std::wstring result = !isMuted ? L"WebView is Now Muted" :
L"WebView is Now Unmuted";
            MessageBox(nullptr, result.c_str(), L"Mute State Changed", MB_OK);
        }
    }

    void AudioComponent::UpdateTitleWithMuteState(wil::com_ptr<ICoreWebView2_8>
webview2_8)
    {
        BOOL isDocumentPlayingAudio;
        CHECK_FAILURE(webview2_8-
>get_IsDocumentPlayingAudio(&isDocumentPlayingAudio));

        BOOL isMuted;
        CHECK_FAILURE(webview2_8->get_IsMuted(&isMuted));

        wil::unique_cotaskmem_string title;
        CHECK_FAILURE(m_webView->get_DocumentTitle(&title));
        std::wstring result = L"";

        if (isDocumentPlayingAudio)
        {
            if (isMuted)
            {
                result = L"???? " + std::wstring(title.get());
            }
            else
        }
    }
}

```

```

    {
        result = L"???? " + std::wstring(title.get());
    }
} else
{
    result = std::wstring(title.get());
}

m_appWindow->SetDocumentTitle(result.c_str());
}

```

## get\_IsDocumentPlayingAudio

Indicates whether any audio output from this CoreWebView2 is playing.

```
public HRESULT get_IsDocumentPlayingAudio(BOOL * value)
```

This property will be true if audio is playing even if IsMuted is true.

C++

```

AudioComponent::AudioComponent(AppWindow* appWindow)
    : m_appWindow(appWindow), m_webView(appWindow->GetWebView())
{
    auto webview2_8 = m_webView.try_query<ICoreWebView2_8>();
    if (webview2_8)
    {
        // Register a handler for the IsDocumentPlayingAudioChanged event.
        CHECK_FAILURE(webview2_8->add_IsDocumentPlayingAudioChanged(
            Callback<ICoreWebView2IsDocumentPlayingAudioChangedEventArgs>
(
                [this, webview2_8](ICoreWebView2* sender, IUnknown* args) ->
HRESULT
{
    UpdateTitleWithMuteState(webview2_8);
    return S_OK;
})
        .Get(),
        &m_isDocumentPlayingAudioChangedToken));

        // Register a handler for the IsMutedChanged event.
        CHECK_FAILURE(webview2_8->add_IsMutedChanged(
            Callback<ICoreWebView2IsMutedChangedEventArgs>
(
                [this, webview2_8](ICoreWebView2* sender, IUnknown* args) ->
HRESULT
{
    UpdateTitleWithMuteState(webview2_8);
    return S_OK;
})
        .Get(),
        &m_isMutedChangedToken));
    }
}

```

```

        &m_isMutedChangedToken));
    }
}

bool AudioComponent::HandleWindowMessage(
    HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam, LRESULT* result)
{
    if (message == WM_COMMAND)
    {
        switch (LOWORD(wParam))
        {
            case IDM_TOGGLE_MUTE_STATE:
                ToggleMuteState();
                return true;
        }
    }
    return false;
}

// Toggle the mute state of the current window and show a mute or unmute
// icon on the title bar
void AudioComponent::ToggleMuteState()
{
    auto webview2_8 = m_webView.try_query<ICoreWebView2_8>();
    if (webview2_8)
    {
        BOOL isMuted;
        CHECK_FAILURE(webview2_8->get_IsMuted(&isMuted));
        CHECK_FAILURE(webview2_8->put_IsMuted(!isMuted));
        std::wstring result = !isMuted ? L"WebView is Now Muted" :
L"WebView is Now Unmuted";
        MessageBox(nullptr, result.c_str(), L"Mute State Changed", MB_OK);
    }
}

void AudioComponent::UpdateTitleWithMuteState(wil::com_ptr<ICoreWebView2_8>
webview2_8)
{
    BOOL isDocumentPlayingAudio;
    CHECK_FAILURE(webview2_8-
>get_IsDocumentPlayingAudio(&isDocumentPlayingAudio));

    BOOL isMuted;
    CHECK_FAILURE(webview2_8->get_IsMuted(&isMuted));

    wil::unique_cotaskmem_string title;
    CHECK_FAILURE(m_webView->get_DocumentTitle(&title));
    std::wstring result = L"";

    if (isDocumentPlayingAudio)
    {
        if (isMuted)
        {
            result = L"???? " + std::wstring(title.get());
        }
    }
}

```

```

        else
        {
            result = L"????? " + std::wstring(title.get());
        }
    }
else
{
    result = std::wstring(title.get());
}

m_appWindow->SetDocumentTitle(result.c_str());
}

```

## get\_IsMuted

Indicates whether all audio output from this CoreWebView2 is muted or not.

```
public HRESULT get_IsMuted(BOOL * value)
```

C++

```

AudioComponent::AudioComponent(AppWindow* appWindow)
    : m_appWindow(appWindow), m_webView(appWindow->GetWebView())
{
    auto webview2_8 = m_webView.try_query<ICoreWebView2_8>();
    if (webview2_8)
    {
        // Register a handler for the IsDocumentPlayingAudioChanged event.
        CHECK_FAILURE(webview2_8->add_IsDocumentPlayingAudioChanged(
            Callback<ICoreWebView2IsDocumentPlayingAudioChangedEventHandler>
(
                [this, webview2_8](ICoreWebView2* sender, IUnknown* args) ->
HRESULT
{
    UpdateTitleWithMuteState(webview2_8);
    return S_OK;
})
                .Get(),
                &m_isDocumentPlayingAudioChangedToken));

        // Register a handler for the IsMutedChanged event.
        CHECK_FAILURE(webview2_8->add_IsMutedChanged(
            Callback<ICoreWebView2IsMutedChangedEventHandler>(
                [this, webview2_8](ICoreWebView2* sender, IUnknown* args) ->
HRESULT
{
    UpdateTitleWithMuteState(webview2_8);
    return S_OK;
})
                .Get(),
                &m_isMutedChangedToken));
    }
}

```

```

        }

    }

    bool AudioComponent::HandleWindowMessage(
        HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam, LRESULT* result)
    {
        if (message == WM_COMMAND)
        {
            switch (LOWORD(wParam))
            {
                case IDM_TOGGLE_MUTE_STATE:
                    ToggleMuteState();
                    return true;
            }
        }
        return false;
    }

    // Toggle the mute state of the current window and show a mute or unmute
    // icon on the title bar
    void AudioComponent::ToggleMuteState()
    {
        auto webview2_8 = m_webView.try_query<ICoreWebView2_8>();
        if (webview2_8)
        {
            BOOL isMuted;
            CHECK_FAILURE(webview2_8->get_IsMuted(&isMuted));
            CHECK_FAILURE(webview2_8->put_IsMuted(!isMuted));
            std::wstring result = !isMuted ? L"WebView is Now Muted" :
L"WebView is Now Unmuted";
            MessageBox(nullptr, result.c_str(), L"Mute State Changed", MB_OK);
        }
    }

    void AudioComponent::UpdateTitleWithMuteState(wil::com_ptr<ICoreWebView2_8>
webview2_8)
    {
        BOOL isDocumentPlayingAudio;
        CHECK_FAILURE(webview2_8-
>get_IsDocumentPlayingAudio(&isDocumentPlayingAudio));

        BOOL isMuted;
        CHECK_FAILURE(webview2_8->get_IsMuted(&isMuted));

        wil::unique_cotaskmem_string title;
        CHECK_FAILURE(m_webView->get_DocumentTitle(&title));
        std::wstring result = L"";

        if (isDocumentPlayingAudio)
        {
            if (isMuted)
            {
                result = L"???? " + std::wstring(title.get());
            }
            else
        }
    }
}

```

```

    {
        result = L"???? " + std::wstring(title.get());
    }
} else
{
    result = std::wstring(title.get());
}

m_appWindow->SetDocumentTitle(result.c_str());
}

```

## put\_IsMuted

Sets the `IsMuted` property.

```
public HRESULT put_IsMuted(BOOL value)
```

C++

```

AudioComponent::AudioComponent(AppWindow* appWindow)
    : m_appWindow(appWindow), m_webView(appWindow->GetWebView())
{
    auto webview2_8 = m_webView.try_query<ICoreWebView2_8>();
    if (webview2_8)
    {
        // Register a handler for the IsDocumentPlayingAudioChanged event.
        CHECK_FAILURE(webview2_8->add_IsDocumentPlayingAudioChanged(
            Callback<ICoreWebView2IsDocumentPlayingAudioChangedEventHandler>
(
                [this, webview2_8](ICoreWebView2* sender, IUnknown* args) ->
HRESULT
{
    UpdateTitleWithMuteState(webview2_8);
    return S_OK;
})
            .Get(),
            &m_isDocumentPlayingAudioChangedToken));

        // Register a handler for the IsMutedChanged event.
        CHECK_FAILURE(webview2_8->add_IsMutedChanged(
            Callback<ICoreWebView2IsMutedChangedEventHandler>
(
                [this, webview2_8](ICoreWebView2* sender, IUnknown* args) ->
HRESULT
{
    UpdateTitleWithMuteState(webview2_8);
    return S_OK;
})
            .Get(),
            &m_isMutedChangedToken));
    }
}

```

```
}

bool AudioComponent::HandleWindowMessage(
    HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam, LRESULT* result)
{
    if (message == WM_COMMAND)
    {
        switch (LOWORD(wParam))
        {
            case IDM_TOGGLE_MUTE_STATE:
                ToggleMuteState();
                return true;
        }
    }
    return false;
}

// Toggle the mute state of the current window and show a mute or unmute
// icon on the title bar
void AudioComponent::ToggleMuteState()
{
    auto webview2_8 = m_webView.try_query<ICoreWebView2_8>();
    if (webview2_8)
    {
        BOOL isMuted;
        CHECK_FAILURE(webview2_8->get_IsMuted(&isMuted));
        CHECK_FAILURE(webview2_8->put_IsMuted(!isMuted));
        std::wstring result = !isMuted ? L"WebView is Now Muted" :
L"WebView is Now Unmuted";
        MessageBox(nullptr, result.c_str(), L"Mute State Changed", MB_OK);
    }
}

void AudioComponent::UpdateTitleWithMuteState(wil::com_ptr<ICoreWebView2_8>
webview2_8)
{
    BOOL isDocumentPlayingAudio;
    CHECK_FAILURE(webview2_8-
>get_IsDocumentPlayingAudio(&isDocumentPlayingAudio));

    BOOL isMuted;
    CHECK_FAILURE(webview2_8->get_IsMuted(&isMuted));

    wil::unique_cotaskmem_string title;
    CHECK_FAILURE(m_webView->get_DocumentTitle(&title));
    std::wstring result = L"";

    if (isDocumentPlayingAudio)
    {
        if (isMuted)
        {
            result = L"???? " + std::wstring(title.get());
        }
        else
        {

```

```
        result = L"????? " + std::wstring(title.get());
    }
}
else
{
    result = std::wstring(title.get());
}

m_appWindow->SetDocumentTitle(result.c_str());
}
```

## remove\_IsDocumentPlayingAudioChanged

Remove an event handler previously added with `add_IsDocumentPlayingAudioChanged`.

```
public HRESULT remove_IsDocumentPlayingAudioChanged(EventRegistrationToken  
token)
```

## remove\_IsMutedChanged

Remove an event handler previously added with `add_IsMutedChanged`.

```
public HRESULT remove_IsMutedChanged(EventRegistrationToken token)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2\_9

Article • 02/26/2024

```
interface ICoreWebView2_9
: public ICoreWebView2_8
```

This interface is an extension of [ICoreWebView2\\_8](#) that default download dialog positioning and anchoring.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">add_IsDefaultDownloadDialogOpenChanged</a>	Raised when the <code>IsDefaultDownloadDialogOpen</code> property changes.
<a href="#">CloseDefaultDownloadDialog</a>	Close the default download dialog.
<a href="#">get_DefaultDownloadDialogCornerAlignment</a>	Get the default download dialog corner alignment.
<a href="#">get_DefaultDownloadDialogMargin</a>	Get the default download dialog margin.
<a href="#">get_IsDefaultDownloadDialogOpen</a>	<code>TRUE</code> if the default download dialog is currently open.
<a href="#">OpenDefaultDownloadDialog</a>	Open the default download dialog.
<a href="#">put_DefaultDownloadDialogCornerAlignment</a>	Set the default download dialog corner alignment.
<a href="#">put_DefaultDownloadDialogMargin</a>	Set the default download dialog margin relative to the WebView corner specified by <code>DefaultDownloadDialogCornerAlignment</code> .
<a href="#">remove_IsDefaultDownloadDialogOpenChanged</a>	Remove an event handler previously added with <code>add_IsDefaultDownloadDialogOpenChanged</code> .

## Applies to

Product	Introduced
WebView2 Win32	1.0.1072.54
WebView2 Win32 Prerelease	1.0.1083

## Members

### **add\_IsDefaultDownloadDialogOpenChanged**

Raised when the `IsDefaultDownloadDialogOpen` property changes.

```
public HRESULT
add_IsDefaultDownloadDialogOpenChanged(ICoreWebView2IsDefaultDownloadDial
ogOpenChangedEventHandler * handler, EventRegistrationToken * token)
```

This event comes after the `DownloadStarting` event. Setting the `Handled` property on the `DownloadStartingEventArgs` disables the default download dialog and ensures that this event is never raised.

### **CloseDefaultDownloadDialog**

Close the default download dialog.

```
public HRESULT CloseDefaultDownloadDialog()
```

Calling this method raises the `IsDefaultDownloadDialogOpenChanged` event if the dialog was open. No effect if the dialog is already closed.

### **get\_DefaultDownloadDialogCornerAlignment**

Get the default download dialog corner alignment.

```
public HRESULT
get_DefaultDownloadDialogCornerAlignment(COREWEBVIEW2_DEFAULT_DOWNLO
AD_DIALOG_CORNER_ALIGNMENT * value)
```

### **get\_DefaultDownloadDialogMargin**

Get the default download dialog margin.

```
public HRESULT get_DefaultDownloadDialogMargin(POINT * value)
```

## get\_IsDefaultDownloadDialogOpen

**TRUE** if the default download dialog is currently open.

```
public HRESULT get_IsDefaultDownloadDialogOpen(BOOL * value)
```

The value of this property changes only when the default download dialog is explicitly opened or closed. Hiding the WebView implicitly hides the dialog, but does not change the value of this property.

## OpenDefaultDownloadDialog

Open the default download dialog.

```
public HRESULT OpenDefaultDownloadDialog()
```

If the dialog is opened before there are recent downloads, the dialog shows all past downloads for the current profile. Otherwise, the dialog shows only the recent downloads with a "See more" button for past downloads. Calling this method raises the **IsDefaultDownloadDialogOpenChanged** event if the dialog was closed. No effect if the dialog is already open.

C++

```
void ViewComponent::ToggleDefaultDownloadDialog()
{
    if (m_webView2_9)
    {
        BOOL isOpen;
        m_webView2_9->get_IsDefaultDownloadDialogOpen(&isOpen);
        if (isOpen)
        {
            m_webView2_9->CloseDefaultDownloadDialog();
        }
        else
        {
            m_webView2_9->OpenDefaultDownloadDialog();
        }
    }
}
```

## put\_DefaultDownloadDialogCornerAlignment

Set the default download dialog corner alignment.

```
public HRESULT  
put_DefaultDownloadDialogCornerAlignment(COREWEBVIEW2_DEFAULT_DOWNLOAD_DIALOG_CORNER_ALIGNMENT value)
```

The dialog can be aligned to any of the WebView corners (see COREWEBVIEW2\_DEFAULT\_DOWNLOAD\_DIALOG\_CORNER\_ALIGNMENT). When the WebView or dialog changes size, the dialog keeps its position relative to the corner. The dialog may become partially or completely outside of the WebView bounds if the WebView is small enough. Set the margin relative to the corner with the DefaultDownloadDialogMargin property. The corner alignment and margin should be set during initialization to ensure that they are correctly applied when the layout is first computed, otherwise they will not take effect until the next time the WebView position or size is updated.

C++

```
void ViewComponent::SetDefaultDownloadDialogPosition()  
{  
    COREWEBVIEW2_DEFAULT_DOWNLOAD_DIALOG_CORNER_ALIGNMENT cornerAlignment =  
        COREWEBVIEW2_DEFAULT_DOWNLOAD_DIALOG_CORNER_ALIGNMENT_TOP_LEFT;  
    POINT margin = {m_downloadsButtonMargin,  
        (m_downloadsButtonMargin + m_downloadsButtonHeight)};  
    CHECK_FAILURE(  
        m_webView2_9->put_DefaultDownloadDialogCornerAlignment(  
            cornerAlignment));  
    CHECK_FAILURE(  
        m_webView2_9->put_DefaultDownloadDialogMargin(margin));  
}
```

## put\_DefaultDownloadDialogMargin

Set the default download dialog margin relative to the WebView corner specified by DefaultDownloadDialogCornerAlignment.

```
public HRESULT put_DefaultDownloadDialogMargin(POINT value)
```

The margin is a point that describes the vertical and horizontal distances between the chosen WebView corner and the default download dialog corner nearest to it. Positive values move the dialog towards the center of the WebView from the chosen WebView

corner, and negative values move the dialog away from it. Use (0, 0) to align the dialog to the WebView corner with no margin. The corner alignment and margin should be set during initialization to ensure that they are correctly applied when the layout is first computed, otherwise they will not take effect until the next time the WebView position or size is updated.

## **remove\_IsDefaultDownloadDialogOpenChanged**

Remove an event handler previously added with

`add_IsDefaultDownloadDialogOpenChanged`.

```
public HRESULT  
remove_IsDefaultDownloadDialogOpenChanged(EventRegistrationToken token)
```

---

## **Feedback**

Was this page helpful?

 Yes

 No

# interface ICoreWebView2AcceleratorKeyPressedEventArgs

Article • 02/26/2024

```
interface ICoreWebView2AcceleratorKeyPressedEventArgs
: public IUnknown
```

Event args for the `AcceleratorKeyPressed` event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_Handled</a>	During <code>AcceleratorKeyPressedEvent</code> handler invocation the WebView is blocked waiting for the decision of if the accelerator is handled by the host (or not).
<a href="#">get_KeyEventKind</a>	The key event type that caused the event to run.
<a href="#">get_KeyEventLParam</a>	The <code>LPARAM</code> value that accompanied the window message.
<a href="#">get_PhysicalKeyStatus</a>	A structure representing the information passed in the <code>LPARAM</code> of the window message.
<a href="#">get_VirtualKey</a>	The Win32 virtual key code of the key that was pressed or released.
<a href="#">put_Handled</a>	Sets the <code>Handled</code> property.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430

Product	Introduced
WebView2 Win32 Prerelease	0.9.488

## Members

### get\_Handled

During `AcceleratorKeyPressedEvent` handler invocation the WebView is blocked waiting for the decision of if the accelerator is handled by the host (or not).

```
public HRESULT get_Handled(BOOL * handled)
```

If the `Handled` property is set to `TRUE` then this prevents the WebView from performing the default action for this accelerator key. Otherwise the WebView performs the default action for the accelerator key.

### get\_KeyEventKind

The key event type that caused the event to run.

```
public HRESULT get_KeyEventKind(COREWEBVIEW2_KEY_EVENT_KIND *
keyEventKind)
```

### get\_KeyEventLParam

The `LPARAM` value that accompanied the window message.

```
public HRESULT get_KeyEventLParam(INT * lParam)
```

For more information, navigate to [WM\\_KEYDOWN](#) and [WM\\_KEYUP](#).

### get\_PhysicalKeyStatus

A structure representing the information passed in the `LPARAM` of the window message.

```
public HRESULT get_PhysicalKeyStatus(COREWEBVIEW2_PHYSICAL_KEY_STATUS *
physicalKeyStatus)
```

## get\_VirtualKey

The Win32 virtual key code of the key that was pressed or released.

```
public HRESULT get_VirtualKey(UINT * virtualKey)
```

It is one of the Win32 virtual key constants such as `VK_RETURN` or an (uppercase) ASCII value such as `A`. Verify whether Ctrl or Alt are pressed by running `GetKeyState(VK_CONTROL)` or `GetKeyState(VK_MENU)`.

## put\_Handled

Sets the `Handled` property.

```
public HRESULT put_Handled(BOOL handled)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2AcceleratorKeyPressedEventArgs2

Article • 02/26/2024

```
interface ICoreWebView2AcceleratorKeyPressedEventArgs2
: public ICoreWebView2AcceleratorKeyPressedEventArgs
```

This is This is a continuation of the ICoreWebView2AcceleratorKeyPressedEventArgs interface.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_IsBrowserAcceleratorKeyEnabled</a>	This property allows developers to enable or disable the browser from handling a specific browser accelerator key such as Ctrl+P or F3, etc.
<a href="#">put_IsBrowserAcceleratorKeyEnabled</a>	Sets the <code>IsBrowserAcceleratorKeyEnabled</code> property.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.2210.55
WebView2 Win32 Prerelease	1.0.2357

## Members

### [get\\_IsBrowserAcceleratorKeyEnabled](#)

This property allows developers to enable or disable the browser from handling a specific browser accelerator key such as Ctrl+P or F3, etc.

```
public HRESULT get_IsBrowserAcceleratorKeyEnabled(BOOL * value)
```

Browser accelerator keys are the keys/key combinations that access features specific to a web browser, including but not limited to:

- Ctrl-F and F3 for Find on Page
- Ctrl-P for Print
- Ctrl-R and F5 for Reload
- Ctrl-Plus and Ctrl-Minus for zooming
- Ctrl-Shift-C and F12 for DevTools
- Special keys for browser functions, such as Back, Forward, and Search

This property does not disable accelerator keys related to movement and text editing, such as:

- Home, End, Page Up, and Page Down
- Ctrl-X, Ctrl-C, Ctrl-V
- Ctrl-A for Select All
- Ctrl-Z for Undo

The `CoreWebView2Settings.AreBrowserAcceleratorKeysEnabled` API is a convenient setting for developers to disable all the browser accelerator keys together, and sets the default value for the `IsBrowserAcceleratorKeyEnabled` property. By default,

`CoreWebView2Settings.AreBrowserAcceleratorKeysEnabled` is `TRUE` and `IsBrowserAcceleratorKeyEnabled` is `TRUE`. When developers change `CoreWebView2Settings.AreBrowserAcceleratorKeysEnabled` setting to `FALSE`, this will change default value for `IsBrowserAcceleratorKeyEnabled` to `FALSE`. If developers want specific keys to be handled by the browser after changing the `CoreWebView2Settings.AreBrowserAcceleratorKeysEnabled` setting to `FALSE`, they need to enable these keys by setting `IsBrowserAcceleratorKeyEnabled` to `TRUE`. This API will give the event arg higher priority over the `CoreWebView2Settings.AreBrowserAcceleratorKeysEnabled` setting when we handle the keys.

For browser accelerator keys, when an accelerator key is pressed, the propagation and processing order is:

1. A CoreWebView2Controller.AcceleratorKeyPressed event is raised
2. WebView2 browser feature accelerator key handling
3. Web Content Handling: If the key combination isn't reserved for browser actions, the key event propagates to the web content, where JavaScript event listeners can capture and respond to it.

ICoreWebView2AcceleratorKeyPressedEventArgs has a `Handled` property, that developers can use to mark a key as handled. When the key is marked as handled anywhere along the path, the event propagation stops, and web content will not receive the key. With `IsBrowserAcceleratorKeyEnabled` property, if developers mark `IsBrowserAcceleratorKeyEnabled` as `FALSE`, the browser will skip the WebView2 browser feature accelerator key handling process, but the event propagation continues, and web content will receive the key combination.

C++

```
if (m_settings3)
{
    // Register a handler for the AcceleratorKeyPressed event.
    CHECK_FAILURE(m_controller->add_AcceleratorKeyPressed(
        Callback<ICoreWebView2AcceleratorKeyPressedEventHandler>(
            [this](
                ICoreWebView2Controller* sender,
                ICoreWebView2AcceleratorKeyPressedEventArgs* args) ->
    HRESULT
    {
        COREWEBVIEW2_KEY_EVENT_KIND kind;
        CHECK_FAILURE(args->get_KeyEventKind(&kind));
        // We only care about key down events.
        if (kind == COREWEBVIEW2_KEY_EVENT_KIND_KEY_DOWN ||
            kind == COREWEBVIEW2_KEY_EVENT_KIND_SYSTEM_KEY_DOWN)
        {
            UINT key;
            CHECK_FAILURE(args->get_VirtualKey(&key));

wil::com_ptr<ICoreWebView2AcceleratorKeyPressedEventArgs2> args2;

            args->QueryInterface(IID_PPV_ARGS(&args2));
            if (args2)
            {
                if (key == 'P' && (GetKeyState(VK_CONTROL) < 0))
                {
                    // tell the browser to skip the key
                    CHECK_FAILURE(args2-
```

```
>put_IsBrowserAcceleratorKeyEnabled(FALSE));
    }
    if (key == VK_F7)
    {
        // tell the browser to process the key
        CHECK_FAILURE(args2-
>put_IsBrowserAcceleratorKeyEnabled(TRUE));
    }
}
return S_OK;
})
.Get(),
&m_acceleratorKeyPressedToken);
}
```

Gets the `IsBrowserAcceleratorKeyEnabled` property.

## `put_IsBrowserAcceleratorKeyEnabled`

Sets the `IsBrowserAcceleratorKeyEnabled` property.

```
public HRESULT put_IsBrowserAcceleratorKeyEnabled(BOOL value)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2BasicAuthenticationRequestedEventArgs

Article • 02/26/2024

```
interface ICoreWebView2BasicAuthenticationRequestedEventArgs
: public IUnknown
```

Event args for the BasicAuthenticationRequested event.

## Summary

[ ] Expand table

Members	Descriptions
<a href="#">get_Cancel</a>	Cancel the authentication request.
<a href="#">get_Challenge</a>	The authentication challenge string.
<a href="#">get_Response</a>	Response to the authentication request with credentials.
<a href="#">get_Uri</a>	The URI that led to the authentication challenge.
<a href="#">GetDeferral</a>	Returns an ICoreWebView2Deferral object.
<a href="#">put_Cancel</a>	Set the Cancel property.

Will contain the request that led to the HTTP authorization challenge, the challenge and allows the host to provide authentication response or cancel the request.

## Applies to

[ ] Expand table

Product	Introduced
WebView2 Win32	1.0.1150.38

Product	Introduced
WebView2 Win32 Prerelease	1.0.1133

## Members

### get\_Cancel

Cancel the authentication request.

```
public HRESULT get_Cancel(BOOL * cancel)
```

False by default. If set to true, Response will be ignored.

### get\_Challenge

The authentication challenge string.

```
public HRESULT get_Challenge(LPWSTR * challenge)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

### get\_Response

Response to the authentication request with credentials.

```
public HRESULT get_Response(ICoreWebView2BasicAuthenticationResponse ** response)
```

This object will be populated by the app if the host would like to provide authentication credentials.

### get\_Uri

The URI that led to the authentication challenge.

```
public HRESULT get_Uri(LPWSTR * value)
```

For proxy authentication requests, this will be the URI of the proxy server.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## GetDeferral

Returns an `ICoreWebView2Deferral` object.

```
public HRESULT GetDeferral(ICoreWebView2Deferral ** deferral)
```

Use this deferral to defer the decision to show the Basic Authentication dialog.

## put\_Cancel

Set the `Cancel` property.

```
public HRESULT put_Cancel(BOOL cancel)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2BasicAuthenticationResponse

Article • 02/26/2024

```
interface ICoreWebView2BasicAuthenticationResponse
: public IUnknown
```

Represents a Basic HTTP authentication response that contains a user name and a password as according to RFC7617 (<https://tools.ietf.org/html/rfc7617>)

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_Password</a>	Password provided for authentication.
<a href="#">get_UserName</a>	User name provided for authentication.
<a href="#">put_Password</a>	Set password property.
<a href="#">put_UserName</a>	Set user name property.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1150.38
WebView2 Win32 Prerelease	1.0.1133

## Members

## **get\_Password**

Password provided for authentication.

```
| public HRESULT get_Password(LPWSTR * password)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## **get\_UserName**

User name provided for authentication.

```
| public HRESULT get_UserName(LPWSTR * userName)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## **put\_Password**

Set password property.

```
| public HRESULT put_Password(LPCWSTR password)
```

## **put\_UserName**

Set user name property.

```
| public HRESULT put_UserName(LPCWSTR userName)
```

---

## **Feedback**

Was this page helpful?

 Yes	 No
---	--

# interface ICoreWebView2BrowserExtension

Article • 02/26/2024

```
interface ICoreWebView2BrowserExtension
: public IUnknown
```

Provides a set of properties for managing an Extension, which includes an ID, name, and whether it is enabled or not, and the ability to Remove the Extension, and enable or disable it.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Enable</a>	Sets whether this browser extension is enabled or disabled.
<a href="#">get_Id</a>	This is the browser extension's ID.
<a href="#">get_IsEnabled</a>	If <code>isEnabled</code> is true then the Extension is enabled and running in WebView instances.
<a href="#">get_Name</a>	This is the browser extension's name.
<a href="#">Remove</a>	Removes this browser extension from its WebView2 Profile.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.2210.55
WebView2 Win32 Prerelease	1.0.2194

## Members

## Enable

Sets whether this browser extension is enabled or disabled.

```
public HRESULT Enable(BOOL isEnabled,  
ICoreWebView2BrowserExtensionEnableCompletedHandler * handler)
```

This change applies immediately to the extension in all HTML documents in all WebView2s associated with this profile. After an extension is removed, calling `Enable` will not change the value of `IsEnabled`.

## get\_Id

This is the browser extension's ID.

```
public HRESULT get_Id(LPWSTR * value)
```

This is the same browser extension ID returned by the browser extension API [chrome.runtime.id](#). Please see that documentation for more details on how the ID is generated. After an extension is removed, calling `Id` will return the id of the extension that is removed. The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_IsEnabled

If `isEnabled` is true then the Extension is enabled and running in WebView instances.

```
public HRESULT get_IsEnabled(BOOL * value)
```

If it is false then the Extension is disabled and not running in WebView instances. When a Extension is first installed, `IsEnabled` are default to be `TRUE`. `isEnabled` is persisted per profile. After an extension is removed, calling `isEnabled` will return the value at the time it was removed.

## get\_Name

This is the browser extension's name.

```
public HRESULT get_Name(LPWSTR * value)
```

This value is defined in this browser extension's manifest.json file. If manifest.json define extension's localized name, this value will be the localized version of the name. Please see [Manifest.json name](#) for more details. After an extension is removed, calling `Name` will return the name of the extension that is removed. The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## Remove

Removes this browser extension from its WebView2 Profile.

```
public HRESULT
```

```
Remove(ICoreWebView2BrowserExtensionRemoveCompletedHandler * handler)
```

The browser extension is removed immediately including from all currently running HTML documents associated with this WebView2 Profile. The removal is persisted and future uses of this profile will not have this extension installed. After an extension is removed, calling `Remove` again will cause an exception.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2BrowserExtensionList

Article • 02/26/2024

```
interface ICoreWebView2BrowserExtensionList
: public IUnknown
```

Provides a set of properties for managing browser Extension Lists from user profile.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_Count</a>	The number of browser Extensions in the list.
<a href="#">GetValueAtIndex</a>	Gets the browser Extension located in the browser Extension List at the given index.

This includes the number of browser Extensions in the list, and the ability to get an browser Extension from the list at a particular index.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.2210.55
WebView2 Win32 Prerelease	1.0.2194

## Members

### [get\\_Count](#)

The number of browser Extensions in the list.

```
public HRESULT get_Count(UINT * count)
```

## GetValueAtIndex

Gets the browser Extension located in the browser Extension List at the given index.

```
public HRESULT GetValueAtIndex(UINT index, ICoreWebView2BrowserExtension **  
extension)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2BrowserProcessExitedEventArgs

Article • 02/26/2024

```
interface ICoreWebView2BrowserProcessExitedEventArgs
: public IUnknown
```

Event args for the `BrowserProcessExited` event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_BrowserProcessExitKind</a>	The kind of browser process exit that has occurred.
<a href="#">get_BrowserProcessId</a>	The process ID of the browser process that has exited.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.992.28
WebView2 Win32 Prerelease	1.0.1010

## Members

### `get_BrowserProcessExitKind`

The kind of browser process exit that has occurred.

```
public HRESULT  
get_BrowserProcessExitKind(COREWEBVIEW2_BROWSER_PROCESS_EXIT_KIND *  
browserProcessExitKind)
```

## get\_BrowserProcessId

The process ID of the browser process that has exited.

```
public HRESULT get_BrowserProcessId(UINT32 * value)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Certificate

Article • 02/26/2024

```
interface ICoreWebView2Certificate
: public IUnknown
```

Provides access to the certificate metadata.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_DerEncodedSerialNumber</a>	Base64 encoding of DER encoded serial number of the certificate.
<a href="#">get_DisplayName</a>	Display name for a certificate.
<a href="#">get_Issuer</a>	Name of the certificate authority that issued the certificate.
<a href="#">get_PemEncodedIssuerCertificateChain</a>	Collection of PEM encoded certificate issuer chain.
<a href="#">get_Subject</a>	Subject of the certificate.
<a href="#">get_ValidFrom</a>	The valid start date and time for the certificate as the number of seconds since the UNIX epoch.
<a href="#">get_ValidTo</a>	The valid expiration date and time for the certificate as the number of seconds since the UNIX epoch.
<a href="#">ToPemEncoding</a>	PEM encoded data for the certificate.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1245.22

Product	Introduced
WebView2 Win32 Prerelease	1.0.1158

## Members

### get\_DerEncodedSerialNumber

Base64 encoding of DER encoded serial number of the certificate.

```
public HRESULT get_DerEncodedSerialNumber(LPWSTR * value)
```

Read more about DER at [RFC 7468 DER] (<https://tools.ietf.org/html/rfc7468#appendix-B>).

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

### get\_DisplayName

Display name for a certificate.

```
public HRESULT get_DisplayName(LPWSTR * value)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#)

### get\_Issuer

Name of the certificate authority that issued the certificate.

```
public HRESULT get_Issuer(LPWSTR * value)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

### get\_PemEncodedIssuerCertificateChain

Collection of PEM encoded certificate issuer chain.

```
public HRESULT  
get_PemEncodedIssuerCertificateChain(ICoreWebView2StringCollection ** value)
```

In this collection first element is the current certificate followed by intermediate1, intermediate2...intermediateN-1. Root certificate is the last element in collection.

## get\_Subject

Subject of the certificate.

```
| public HRESULT get_Subject(LPWSTR * value)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_ValidFrom

The valid start date and time for the certificate as the number of seconds since the UNIX epoch.

```
| public HRESULT get_ValidFrom(double * value)
```

## get\_ValidTo

The valid expiration date and time for the certificate as the number of seconds since the UNIX epoch.

```
| public HRESULT get_ValidTo(double * value)
```

## ToPemEncoding

PEM encoded data for the certificate.

```
| public HRESULT ToPemEncoding(LPWSTR * pemEncodedData)
```

Returns Base64 encoding of DER encoded certificate. Read more about PEM at [RFC 1421 Privacy Enhanced Mail] (<https://tools.ietf.org/html/rfc1421>).

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2ClientCertificate

Article • 02/26/2024

```
interface ICoreWebView2ClientCertificate
: public IUnknown
```

Provides access to the client certificate metadata.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_DerEncodedSerialNumber</a>	Base64 encoding of DER encoded serial number of the certificate.
<a href="#">get_DisplayName</a>	Display name for a certificate.
<a href="#">get_Issuer</a>	Name of the certificate authority that issued the certificate.
<a href="#">get_Kind</a>	Kind of a certificate (eg., smart card, pin, other).
<a href="#">get_PemEncodedIssuerCertificateChain</a>	Collection of PEM encoded client certificate issuer chain.
<a href="#">get_Subject</a>	Subject of the certificate.
<a href="#">get_ValidFrom</a>	The valid start date and time for the certificate as the number of seconds since the UNIX epoch.
<a href="#">get_ValidTo</a>	The valid expiration date and time for the certificate as the number of seconds since the UNIX epoch.
<a href="#">ToPemEncoding</a>	PEM encoded data for the certificate.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.961.33
WebView2 Win32 Prerelease	1.0.955

## Members

### get\_DerEncodedSerialNumber

Base64 encoding of DER encoded serial number of the certificate.

```
public HRESULT get_DerEncodedSerialNumber(LPWSTR * value)
```

Read more about DER at [RFC 7468 DER] (<https://tools.ietf.org/html/rfc7468#appendix-B>).

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

### get\_DisplayName

Display name for a certificate.

```
public HRESULT get_DisplayName(LPWSTR * value)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

### get\_Issuer

Name of the certificate authority that issued the certificate.

```
public HRESULT get_Issuer(LPWSTR * value)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

### get\_Kind

Kind of a certificate (eg., smart card, pin, other).

```
public HRESULT get_Kind(COREWEBVIEW2_CLIENT_CERTIFICATE_KIND * value)
```

## get\_PemEncodedIssuerCertificateChain

Collection of PEM encoded client certificate issuer chain.

```
public HRESULT  
get_PemEncodedIssuerCertificateChain(ICoreWebView2StringCollection ** value)
```

In this collection first element is the current certificate followed by intermediate1, intermediate2...intermediateN-1. Root certificate is the last element in collection.

## get\_Subject

Subject of the certificate.

```
public HRESULT get_Subject(LPWSTR * value)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_ValidFrom

The valid start date and time for the certificate as the number of seconds since the UNIX epoch.

```
public HRESULT get_ValidFrom(double * value)
```

## get\_ValidTo

The valid expiration date and time for the certificate as the number of seconds since the UNIX epoch.

```
public HRESULT get_ValidTo(double * value)
```

## ToPemEncoding

PEM encoded data for the certificate.

```
public HRESULT ToPemEncoding(LPWSTR * pemEncodedData)
```

Returns Base64 encoding of DER encoded certificate. Read more about PEM at [RFC 1421 Privacy Enhanced Mail] (<https://tools.ietf.org/html/rfc1421>).

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2ClientCertificateCollection

Article • 02/26/2024

```
interface ICoreWebView2ClientCertificateCollection
: public IUnknown
```

A collection of client certificate object.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_Count</a>	The number of client certificates contained in the ICoreWebView2ClientCertificateCollection.
<a href="#">GetValueAtIndex</a>	Gets the certificate object at the given index.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.961.33
WebView2 Win32 Prerelease	1.0.955

## Members

### [get\\_Count](#)

The number of client certificates contained in the ICoreWebView2ClientCertificateCollection.

```
public HRESULT get_Count(UINT * value)
```

## GetValueAtIndex

Gets the certificate object at the given index.

```
public HRESULT GetValueAtIndex(UINT index, ICoreWebView2ClientCertificate **  
certificate)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2ClientCertificateRequestedEventArgs

Article • 02/26/2024

```
interface ICoreWebView2ClientCertificateRequestedEventArgs
: public IUnknown
```

Event args for the `ClientCertificateRequested` event.

## Summary

[ ] Expand table

Members	Descriptions
<a href="#">get_AllowedCertificateAuthorities</a>	Returns the ICoreWebView2StringCollection.
<a href="#">get_Cancel</a>	You may set this flag to cancel the certificate selection.
<a href="#">get_Handled</a>	You may set this flag to <code>TRUE</code> to respond to the server with or without a certificate.
<a href="#">get_Host</a>	Host name of the server that requested client certificate authentication.
<a href="#">get_IsProxy</a>	Returns true if the server that issued this request is an http proxy.
<a href="#">get_MutuallyTrustedCertificates</a>	Returns the ICoreWebView2ClientCertificateCollection when client certificate authentication is requested.
<a href="#">get_Port</a>	Port of the server that requested client certificate authentication.
<a href="#">get_SelectedCertificate</a>	Returns the selected certificate.
<a href="#">GetDeferral</a>	Returns an ICoreWebView2Deferral object.
<a href="#">put_Cancel</a>	Sets the <code>Cancel</code> property.
<a href="#">put_Handled</a>	Sets the <code>Handled</code> property.

Members	Descriptions
<a href="#">put_SelectedCertificate</a>	Sets the certificate to respond to the server.

## Applies to

[\[+\] Expand table](#)

Product	Introduced
WebView2 Win32	1.0.961.33
WebView2 Win32 Prerelease	1.0.955

## Members

### get\_AllowedCertificateAuthorities

Returns the ICoreWebView2StringCollection.

```
public HRESULT get\_AllowedCertificateAuthorities(ICoreWebView2StringCollection  
** value)
```

The collection contains Base64 encoding of DER encoded distinguished names of certificate authorities allowed by the server.

### get\_Cancel

You may set this flag to cancel the certificate selection.

```
public HRESULT get\_Cancel(BOOL * value)
```

If canceled, the request is aborted regardless of the `Handled` property. By default the value is `FALSE`.

### get\_Handled

You may set this flag to `TRUE` to respond to the server with or without a certificate.

```
public HRESULT get\_Handled(BOOL * value)
```

If this flag is `TRUE` with a `SelectedCertificate` it responds to the server with the selected certificate otherwise respond to the server without a certificate. By default the value of `Handled` and `Cancel` are `FALSE` and display default client certificate selection dialog prompt to allow the user to choose a certificate. The `SelectedCertificate` is ignored unless `Handled` is set `TRUE`.

## get\_Host

Host name of the server that requested client certificate authentication.

```
public HRESULT get_Host(LPWSTR * value)
```

Normalization rules applied to the hostname are:

- Convert to lowercase characters for ascii characters.
- Punycode is used for representing non ascii characters.
- Strip square brackets for IPV6 address.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_IsProxy

Returns true if the server that issued this request is an http proxy.

```
public HRESULT get_IsProxy(BOOL * value)
```

Returns false if the server is the origin server.

## get\_MutuallyTrustedCertificates

Returns the `ICoreWebView2ClientCertificateCollection` when client certificate authentication is requested.

```
public HRESULT  
get_MutuallyTrustedCertificates(ICoreWebView2ClientCertificateCollection ** value)
```

The collection contains mutually trusted CA certificates.

## get\_Port

Port of the server that requested client certificate authentication.

```
public HRESULT get_Port(int * value)
```

## get\_SelectedCertificate

Returns the selected certificate.

```
public HRESULT get_SelectedCertificate(ICoreWebView2ClientCertificate ** value)
```

## GetDeferral

Returns an ICoreWebView2Deferral object.

```
public HRESULT GetDeferral(ICoreWebView2Deferral ** deferral)
```

Use this operation to complete the event at a later time.

## put\_Cancel

Sets the `Cancel` property.

```
public HRESULT put_Cancel(BOOL value)
```

## put\_Handled

Sets the `Handled` property.

```
public HRESULT put_Handled(BOOL value)
```

## put\_SelectedCertificate

Sets the certificate to respond to the server.

```
public HRESULT put_SelectedCertificate(ICoreWebView2ClientCertificate * value)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2CompositionController

Article • 02/26/2024

```
interface ICoreWebView2CompositionController
: public IUnknown
```

This interface is an extension of the ICoreWebView2Controller interface to support visual hosting.

## Summary

[ ] Expand table

Members	Descriptions
<a href="#">add_CursorChanged</a>	Add an event handler for the CursorChanged event.
<a href="#">get_Cursor</a>	The current cursor that WebView thinks it should be.
<a href="#">get_RootVisualTarget</a>	The RootVisualTarget is a visual in the hosting app's visual tree.
<a href="#">get_SystemCursorId</a>	The current system cursor ID reported by the underlying rendering engine for WebView.
<a href="#">put_RootVisualTarget</a>	Set the RootVisualTarget property.
<a href="#">remove_CursorChanged</a>	Remove an event handler previously added with add_CursorChanged.
<a href="#">SendMouseInput</a>	If eventKind is COREWEBVIEW2_MOUSE_EVENT_KIND_HORIZONTAL_WHEEL or COREWEBVIEW2_MOUSE_EVENT_KIND_WHEEL, then mouseData specifies the amount of wheel movement.
<a href="#">SendPointerInput</a>	SendPointerInput accepts touch or pen pointer input of types defined in COREWEBVIEW2_POINTER_EVENT_KIND.

An object implementing the ICoreWebView2CompositionController interface will also implement ICoreWebView2Controller. Callers are expected to use ICoreWebView2Controller for resizing, visibility, focus, and so on, and then use ICoreWebView2CompositionController to connect to a composition tree and provide input meant for the WebView.

# Applies to

[Expand table](#)

Product	Introduced
WebView2 Win32	1.0.774.44
WebView2 Win32 Prerelease	1.0.790

## Members

### add\_CursorChanged

Add an event handler for the CursorChanged event.

```
public HRESULT add_CursorChanged(ICoreWebView2CursorChangedEventArgs *  
eventHandler, EventRegistrationToken * token)
```

The event is raised when WebView thinks the cursor should be changed. For example, when the mouse cursor is currently the default cursor but is then moved over text, it may try to change to the IBeam cursor.

It is expected for the developer to send COREWEBVIEW2\_MOUSE\_EVENT\_KIND\_LEAVE messages (in addition to COREWEBVIEW2\_MOUSE\_EVENT\_KIND\_MOVE messages) through the SendMouseInput API. This is to ensure that the mouse is actually within the WebView that sends out CursorChanged events.

C++

```
// Register a handler for the CursorChanged event.  
CHECK_FAILURE(m_compositionController->add_CursorChanged(  
    Callback<ICoreWebView2CursorChangedEventArgs>(  
        [this](ICoreWebView2CompositionController* sender, IUnknown*  
args)  
        -> HRESULT {  
            HRESULT hr = S_OK;  
            HCURSOR cursor;  
            if (!m_useCursorId)  
            {  
                CHECK_FAILURE(sender->get_Cursor(&cursor));  
            }  
            else  
            {  
                UINT32 cursorId;
```

```
        CHECK_FAILURE(m_compositionController->get_SystemCursorId(&cursorId));
                cursor = ::LoadCursor(nullptr,
MAKEINTRESOURCE(cursorId));
                if (cursor == nullptr)
{
                hr = HRESULT_FROM_WIN32(GetLastError());
}
}

if (SUCCEEDED(hr))
{
    SetClassLongPtr(
        m_appWindow->GetMainWindow(), GCLP_HCURSOR,
(LONG_PTR)cursor);
}
return hr;
})
.Get(),
&m_cursorChangedToken));
```

## get\_Cursor

The current cursor that WebView thinks it should be.

```
public HRESULT get_Cursor(HCURSOR * cursor)
```

The cursor should be set in WM\_SETCURSOR through ::SetCursor or set on the corresponding parent/ancestor HWND of the WebView through ::SetClassLongPtr. The HCURSOR can be freed so CopyCursor/DestroyCursor is recommended to keep your own copy if you are doing more than immediately setting the cursor.

## get\_RootVisualTarget

The RootVisualTarget is a visual in the hosting app's visual tree.

```
public HRESULT get_RootVisualTarget(IUnknown ** target)
```

This visual is where the WebView will connect its visual tree. The app uses this visual to position the WebView within the app. The app still needs to use the Bounds property to size the WebView. The RootVisualTarget property can be an IDCompositionVisual or a Windows::UI::Composition::ContainerVisual. WebView will connect its visual tree to the provided visual before returning from the property setter. The app needs to commit on its device setting the RootVisualTarget property. The RootVisualTarget property supports being set to nullptr to disconnect the WebView from the app's visual tree.

C++

```
// Set the host app visual that the WebView will connect its
visual
// tree to.
BuildDCompTreeUsingVisual();
if (_isDcompTargetMode)
{
    if (!m_dcompTarget)
    {
        m_dcompTarget = Make<DCompTargetImpl>(_this);
    }
    CHECK_FAILURE(
        _compositionController-
>put_RootVisualTarget(m_dcompTarget.get()));
}
else
{
    CHECK_FAILURE(
        _compositionController-
>put_RootVisualTarget(m_dcompWebViewVisual.get()));
}
CHECK_FAILURE(m_dcompDevice->Commit());
```

C++

```
// Create host app visual that the WebView will connect to.
// - Create a IDCompositionTarget for the host window
// - Create a visual and set that as the IDCompositionTarget's root
// - Create another visual and add that to the IDCompositionTarget's root.
// This visual will be the visual root for the WebView.
void ViewComponent::BuildDCompTreeUsingVisual()
{
    CHECK_FAILURE_BOOL(m_dcompDevice != nullptr);

    if (m_dcompWebViewVisual == nullptr)
    {
        CHECK_FAILURE(m_dcompDevice->CreateTargetForHwnd(
            m_appWindow->GetMainWindow(), TRUE, &m_dcompHwndTarget));
        CHECK_FAILURE(m_dcompDevice->CreateVisual(&m_dcompRootVisual));
        CHECK_FAILURE(m_dcompHwndTarget->SetRoot(m_dcompRootVisual.get()));
        CHECK_FAILURE(m_dcompDevice->CreateVisual(&m_dcompWebViewVisual));
        CHECK_FAILURE(m_dcompRootVisual-
>AddVisual(m_dcompWebViewVisual.get(), TRUE, nullptr));
    }
}
```

## get\_SystemCursorId

The current system cursor ID reported by the underlying rendering engine for WebView.

```
public HRESULT get_SystemCursorId(UINT32 * systemCursorId)
```

For example, most of the time, when the cursor is over text, this will return the int value for IDC\_IIBEAM. The systemCursorId is only valid if the rendering engine reports a default Windows cursor resource value. Navigate to [LoadCursorW](#) for more details. Otherwise, if custom CSS cursors are being used, this will return 0. To actually use systemCursorId in LoadCursor or LoadImage, MAKEINTRESOURCE must be called on it first.

C++

```
    UINT32 cursorId;
    CHECK_FAILURE(m_compositionController-
>get_SystemCursorId(&cursorId));
    cursor = ::LoadCursor(nullptr,
MAKEINTRESOURCE(cursorId));
    if (cursor == nullptr)
    {
        hr = HRESULT_FROM_WIN32(GetLastError());
    }
```

## put\_RootVisualTarget

Set the RootVisualTarget property.

```
public HRESULT put_RootVisualTarget(IUnknown * target)
```

## remove\_CursorChanged

Remove an event handler previously added with add\_CursorChanged.

```
public HRESULT remove_CursorChanged(EventRegistrationToken token)
```

## SendMouseInput

If eventKind is COREWEBVIEW2\_MOUSE\_EVENT\_KIND\_HORIZONTAL\_WHEEL or COREWEBVIEW2\_MOUSE\_EVENT\_KIND\_WHEEL, then mouseData specifies the amount of wheel movement.

```
public HRESULT SendMouseInput(COREWEBVIEW2_MOUSE_EVENT_KIND eventKind,
COREWEBVIEW2_MOUSE_EVENT_VIRTUAL_KEYS virtualKeys, UINT32 mouseData,
POINT point)
```

A positive value indicates that the wheel was rotated forward, away from the user; a negative value indicates that the wheel was rotated backward, toward the user. One wheel click is defined as WHEEL\_DELTA, which is 120. If eventKind is COREWEBVIEW2\_MOUSE\_EVENT\_KIND\_X\_BUTTON\_DOUBLE\_CLICK, COREWEBVIEW2\_MOUSE\_EVENT\_KIND\_X\_BUTTON\_DOWN, or COREWEBVIEW2\_MOUSE\_EVENT\_KIND\_X\_BUTTON\_UP, then mouseData specifies which X buttons were pressed or released. This value should be 1 if the first X button is pressed/released and 2 if the second X button is pressed/released. If eventKind is COREWEBVIEW2\_MOUSE\_EVENT\_KIND\_LEAVE, then virtualKeys, mouseData, and point should all be zero. If eventKind is any other value, then mouseData should be zero. Point is expected to be in the client coordinate space of the WebView. To track mouse events that start in the WebView and can potentially move outside of the WebView and host application, calling SetCapture and ReleaseCapture is recommended. To dismiss hover popups, it is also recommended to send COREWEBVIEW2\_MOUSE\_EVENT\_KIND\_LEAVE messages.

C++

```
bool ViewComponent::OnMouseMessage(UINT message, WPARAM wParam, LPARAM lParam)
{
    // Manually relay mouse messages to the WebView
    if (m_dcompDevice || m_wincompCompositor)
    {
        POINT point;
        POINTSTOPOINT(point, lParam);
        if (message == WM_MOUSEWHEEL ||
            message == WM_MOUSEHWHEEL ||
            message == WM_NCRBUTTONDOWN || message == WM_NCRBUTTONUP)
        {
            // Mouse wheel messages are delivered in screen coordinates.
            // SendMouseInput expects client coordinates for the WebView, so
convert
            // the point from screen to client.
            ::ScreenToClient(m_appWindow->GetMainWindow(), &point);
        }
        // Send the message to the WebView if the mouse location is inside
the
        // bounds of the WebView, if the message is telling the WebView the
        // mouse has left the client area, or if we are currently capturing
        // mouse events.
        bool isMouseInWebView = PtInRect(&m_webViewBounds, point);
        if (isMouseInWebView || message == WM_MOUSELEAVE ||
m_isCapturingMouse)
        {
            DWORD mouseData = 0;

            switch (message)
```

```

{
    case WM_MOUSEWHEEL:
    case WM_MOUSEHWHEEL:
        mouseData = GET_WHEEL_DELTA_WPARAM(wParam);
        break;
    case WM_XBUTTONDOWNDBLCLK:
    case WM_XBUTTONDOWN:
    case WM_XBUTTONUP:
        mouseData = GET_XBUTTON_WPARAM(wParam);
        break;
    case WM_MOUSEMOVE:
        if (!m_isTrackingMouse)
        {
            // WebView needs to know when the mouse leaves the
            client area
            // so that it can dismiss hover popups. TrackMouseEvent
            will
            // provide a notification when the mouse leaves the
            client area.
            TrackMouseEvent(TME_LEAVE);
            m_isTrackingMouse = true;
        }
        break;
    case WM_MOUSELEAVE:
        m_isTrackingMouse = false;
        break;
}

// We need to capture the mouse in case the user drags the
// mouse outside of the window bounds and we still need to send
// mouse messages to the WebView process. This is useful for
// scenarios like dragging the scroll bar or panning a map.
// This is very similar to the Pointer Message case where a
// press started inside of the WebView.
if (message == WM_LBUTTONDOWN || message == WM_MBUTTONDOWN ||
    message == WM_RBUTTONDOWN || message == WM_XBUTTONDOWN)
{
    if (isMouseInWebView && ::GetCapture() != m_appWindow-
>GetMainWindow())
    {
        m_isCapturingMouse = true;
        ::SetCapture(m_appWindow->GetMainWindow());
    }
}
else if (message == WM_LBUTTONUP || message == WM_MBUTTONUP ||
    message == WM_RBUTTONUP || message == WM_XBUTTONUP)
{
    if (::GetCapture() == m_appWindow->GetMainWindow())
    {
        m_isCapturingMouse = false;
        ::ReleaseCapture();
    }
}

// Adjust the point from app client coordinates to webview

```

```
client coordinates.  
        // WM_MOUSELEAVE messages don't have a point, so don't adjust  
        the point.  
        if (message != WM_MOUSELEAVE)  
        {  
            point.x -= m_webViewBounds.left;  
            point.y -= m_webViewBounds.top;  
        }  
  
        CHECK_FAILURE(m_compositionController->SendMouseInput(  
            static_cast<COREWEBVIEW2_MOUSE_EVENT_KIND>(message),  
            static_cast<COREWEBVIEW2_MOUSE_EVENT_VIRTUAL_KEYS>  
(GET_KEYSTATE_WPARAM(wParam)),  
            mouseData, point));  
        return true;  
    }  
    else if (message == WM_MOUSEMOVE && m_isTrackingMouse)  
    {  
        // When the mouse moves outside of the WebView, but still inside  
        the app  
        // turn off mouse tracking and send the WebView a leave event.  
        m_isTrackingMouse = false;  
        TrackMouseEvents(TME_LEAVE | TME_CANCEL);  
        OnMouseMessage(WM_MOUSELEAVE, 0, 0);  
    }  
    return false;  
}
```

## SendPointerInput

SendPointerInput accepts touch or pen pointer input of types defined in COREWEBVIEW2\_POINTER\_EVENT\_KIND.

```
public HRESULT SendPointerInput(COREWEBVIEW2_POINTER_EVENT_KIND  
eventKind, ICoreWebView2PointerInfo * pointerInfo)
```

Any pointer input from the system must be converted into an ICoreWebView2PointerInfo first.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2CompositionController2

Article • 02/26/2024

```
interface ICoreWebView2CompositionController2
    : public ICoreWebView2CompositionController
```

A continuation of the [ICoreWebView2CompositionController](#) interface.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_AutomationProvider</a>	Returns the Automation Provider for the WebView.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.774.44
WebView2 Win32 Prerelease	1.0.824

## Members

### [get\\_AutomationProvider](#)

Returns the Automation Provider for the WebView.

```
public HRESULT get_AutomationProvider(IUnknown ** provider)
```

This object implements [IRawElementProviderSimple](#).

---

# Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2CompositionController3

Article • 02/26/2024

```
interface ICoreWebView2CompositionController3
    : public ICoreWebView2CompositionController2
```

This interface is the continuation of the [ICoreWebView2CompositionController2](#) interface to manage drag and drop.

## Summary

[ ] Expand table

Members	Descriptions
<a href="#">DragEnter</a>	This function corresponds to <a href="#">IDropTarget::DragEnter</a> .
<a href="#">DragLeave</a>	This function corresponds to <a href="#">IDropTarget::DragLeave</a> .
<a href="#">DragOver</a>	This function corresponds to <a href="#">IDropTarget::DragOver</a> .
<a href="#">Drop</a>	This function corresponds to <a href="#">IDropTarget::Drop</a> .

## Applies to

[ ] Expand table

Product	Introduced
WebView2 Win32	1.0.1370.28
WebView2 Win32 Prerelease	1.0.1369

## Members

### [DragEnter](#)

This function corresponds to [IDropTarget::DragEnter](#).

```
public HRESULT DragEnter(IDataObject * dataObject, DWORD keyState, POINT point, DWORD * effect)
```

This function has a dependency on AllowExternalDrop property of CoreWebView2Controller and return E\_FAIL to callers to indicate this operation is not allowed if AllowExternalDrop property is set to false.

The hosting application must register as an IDropTarget and implement and forward DragEnter calls to this function.

point parameter must be modified to include the WebView's offset and be in the WebView's client coordinates (Similar to how SendMouseInput works).

C++

```
HRESULT DropTarget::DragEnter(
    IDataObject* dataObject, DWORD keyState, POINTL cursorPosition, DWORD* effect)
{
    POINT point = {cursorPosition.x, cursorPosition.y};
    // Convert the screen point to client coordinates add the WebView's offset.
    m_viewComponent->OffsetPointToWebView(&point);
    return m_webViewCompositionController3->DragEnter(dataObject, keyState,
    point, effect);
}
```

## DragLeave

This function corresponds to [IDropTarget::DragLeave](#).

```
public HRESULT DragLeave()
```

This function has a dependency on AllowExternalDrop property of CoreWebView2Controller and return E\_FAIL to callers to indicate this operation is not allowed if AllowExternalDrop property is set to false.

The hosting application must register as an IDropTarget and implement and forward DragLeave calls to this function.

C++

```
HRESULT DropTarget::DragLeave()
{
    return m_webViewCompositionController3->DragLeave();
}
```

## DragOver

This function corresponds to [IDropTarget::DragOver](#).

```
public HRESULT DragOver(DWORD keyState, POINT point, DWORD * effect)
```

This function has a dependency on AllowExternalDrop property of CoreWebView2Controller and return E\_FAIL to callers to indicate this operation is not allowed if AllowExternalDrop property is set to false.

The hosting application must register as an IDropTarget and implement and forward DragOver calls to this function.

point parameter must be modified to include the WebView's offset and be in the WebView's client coordinates (Similar to how SendMouseInput works).

C++

```
HRESULT DropTarget::DragOver(DWORD keyState, POINTL cursorPosition, DWORD* effect)
{
    POINT point = {cursorPosition.x, cursorPosition.y};
    // Convert the screen point to client coordinates add the WebView's offset.
    // This returns whether the resultant point is over the WebView visual.
    m_viewComponent->OffsetPointToWebView(&point);
    return m_webViewCompositionController3->DragOver(keyState, point,
effect);
```

## Drop

This function corresponds to [IDropTarget::Drop](#).

```
public HRESULT Drop(IDataObject * dataObject, DWORD keyState, POINT point,
DWORD * effect)
```

This function has a dependency on AllowExternalDrop property of CoreWebView2Controller and return E\_FAIL to callers to indicate this operation is not allowed if AllowExternalDrop property is set to false.

The hosting application must register as an IDropTarget and implement and forward Drop calls to this function.

point parameter must be modified to include the WebView's offset and be in the WebView's client coordinates (Similar to how SendMouseInput works).

C++

```
HRESULT DropTarget::Drop(
    IDataObject* dataObject, DWORD keyState, POINTL cursorPosition, DWORD*
effect)
{
    POINT point = {cursorPosition.x, cursorPosition.y};
    // Convert the screen point to client coordinates add the WebView's
    // offset.
    // This returns whether the resultant point is over the WebView visual.
    m_viewComponent->OffsetPointToWebView(&point);
    return m_webViewCompositionController3->Drop(dataObject, keyState,
    point, effect);
}
```

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2ContentLoadingEventArgs

Article • 02/26/2024

```
interface ICoreWebView2ContentLoadingEventArgs
: public IUnknown
```

Event args for the `ContentLoading` event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_IsErrorPage</a>	<code>TRUE</code> if the loaded content is an error page.
<a href="#">get_NavigationId</a>	The ID of the navigation.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### `get_IsErrorPage`

`TRUE` if the loaded content is an error page.

```
public HRESULT get_IsErrorPage(BOOL * isErrorPage)
```

## get\_NavigationId

The ID of the navigation.

```
public HRESULT get_NavigationId(UINT64 * navigationId)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2ContextMenuItem

Article • 02/26/2024

```
interface ICoreWebView2ContextMenuItem
: public IUnknown
```

Represents a context menu item of a context menu displayed by WebView.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">add_CustomItemSelected</a>	Add an event handler for the <code>CustomItemSelected</code> event.
<a href="#">get_Children</a>	Gets the list of children menu items through a <code>ContextMenuCollection</code> if the kind is Submenu.
<a href="#">get_CommandId</a>	Gets the Command ID for the <code>ContextMenu</code> .
<a href="#">get_Icon</a>	Gets the Icon for the <code>ContextMenu</code> in PNG, Bitmap or SVG formats in the form of an IStream.
<a href="#">get_IsChecked</a>	Gets the checked property of the <code>ContextMenu</code> , used if the kind is Check box or Radio.
<a href="#">get_IsEnabled</a>	Gets the enabled property of the <code>ContextMenu</code> .
<a href="#">get_Kind</a>	Gets the <code>ContextMenu</code> kind.
<a href="#">get_Label</a>	Gets the localized label for the <code>ContextMenu</code> .
<a href="#">get_Name</a>	Gets the unlocalized name for the <code>ContextMenu</code> .
<a href="#">get_ShortcutKeyDescription</a>	Gets the localized keyboard shortcut for this ContextMenuItem.
<a href="#">put_IsChecked</a>	Sets the checked property of the <code>ContextMenu</code> .
<a href="#">put_IsEnabled</a>	Sets the enabled property of the <code>ContextMenu</code> .
<a href="#">remove_CustomItemSelected</a>	Remove an event handler previously added with

Members	Descriptions
	add_CustomItemSelected.

## Applies to

[Expand table](#)

Product	Introduced
WebView2 Win32	1.0.1185.39
WebView2 Win32 Prerelease	1.0.1189

## Members

### add\_CustomItemSelected

Add an event handler for the `CustomItemSelected` event.

```
public HRESULT  
add_CustomItemSelected(ICoreWebView2CustomItemSelectedEventHandler *  
eventHandler, EventRegistrationToken * token)
```

`CustomItemSelected` event is raised when the user selects this `ContextMenuItem`. Will only be raised for end developer created context menu items

### get\_Children

Gets the list of children menu items through a `ContextMenuCollection` if the kind is Submenu.

```
public HRESULT get_Children(ICoreWebView2ContextMenuItemCollection ** value)
```

If the kind is not submenu, will return null.

### get\_CommandId

Gets the Command ID for the `ContextMenu`.

```
public HRESULT get_CommandId(INT32 * value)
```

Use this to report the `SelectedCommandId` in `ContextMenuRequested` event.

## get\_Icon

Gets the Icon for the `ContextMenuItem` in PNG, Bitmap or SVG formats in the form of an `IStream`.

```
public HRESULT get_Icon(IStream ** value)
```

Stream will be rewound to the start of the image data.

## get\_IsChecked

Gets the checked property of the `ContextMenuItem`, used if the kind is Check box or Radio.

```
public HRESULT get_IsChecked(BOOL * value)
```

## get\_IsEnabled

Gets the enabled property of the `ContextMenuItem`.

```
public HRESULT get_IsEnabled(BOOL * value)
```

## get\_Kind

Gets the `ContextMenuItem` kind.

```
public HRESULT get_Kind(COREWEBVIEW2_CONTEXT_MENU_ITEM_KIND * value)
```

## get\_Label

Gets the localized label for the `ContextMenuItem`.

```
public HRESULT get_Label(LPWSTR * value)
```

Will contain an ampersand for characters to be used as keyboard accelerator.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_Name

Gets the unlocalized name for the `ContextMenuItem`.

```
public HRESULT get_Name(LPWSTR * value)
```

Use this to distinguish between context menu item types. This will be the English label of the menu item in lower camel case. For example, the "Save as" menu item will be "saveAs". Extension menu items will be "extension", custom menu items will be "custom" and spellcheck items will be "spellCheck". Some example context menu item names are:

- "saveAs"
- "copyImage"
- "openLinkInNewWindow"
- "cut"
- "copy"
- "paste"

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_ShortcutKeyDescription

Gets the localized keyboard shortcut for this `ContextMenuItem`.

```
public HRESULT get_ShortcutKeyDescription(LPWSTR * value)
```

It will be the empty string if there is no keyboard shortcut. This is text intended to be displayed to the end user to show the keyboard shortcut. For example this property is `Ctrl+Shift+I` for the "Inspect" `ContextMenuItem`.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## put\_IsChecked

Sets the checked property of the `ContextMenuItem`.

```
public HRESULT put_IsChecked(BOOL value)
```

Must only be used for custom context menu items that are of kind Check box or Radio.

## **put\_IsEnabled**

Sets the enabled property of the `ContextMenuItem`.

```
public HRESULT put_IsEnabled(BOOL value)
```

Must only be used in the case of a custom context menu item. The default value for this is `TRUE`.

## **remove\_CustomItemSelected**

Remove an event handler previously added with `add_CustomItemSelected`.

```
public HRESULT remove_CustomItemSelected(EventRegistrationToken token)
```

---

## **Feedback**

Was this page helpful?



# interface ICoreWebView2ContextMenuItemCollection

Article • 02/26/2024

```
interface ICoreWebView2ContextMenuItemCollection
: public IUnknown
```

Represents a collection of `ContextMenuItem` objects.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_Count</a>	Gets the number of <code>ContextMenuItem</code> objects contained in the <code>ContextMenuItemCollection</code> .
<a href="#">GetValueAtIndex</a>	Gets the <code>ContextMenuItem</code> at the specified index.
<a href="#">InsertValueAtIndex</a>	Inserts the <code>ContextMenuItem</code> at the specified index.
<a href="#">RemoveValueAtIndex</a>	Removes the <code>ContextMenuItem</code> at the specified index.

Used to get, remove and add `ContextMenuItem` objects at the specified index.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1185.39
WebView2 Win32 Prerelease	1.0.1189

## Members

## get\_Count

Gets the number of `ContextMenuItem` objects contained in the `ContextMenuCollection`.

```
public HRESULT get_Count(UINT32 * value)
```

## GetValueAtIndex

Gets the `ContextMenuItem` at the specified index.

```
public HRESULT GetValueAtIndex(UINT32 index, ICoreWebView2MenuItem ** value)
```

## InsertValueAtIndex

Inserts the `ContextMenuItem` at the specified index.

```
public HRESULT InsertValueAtIndex(UINT32 index, ICoreWebView2MenuItem * value)
```

## RemoveValueAtIndex

Removes the `ContextMenuItem` at the specified index.

```
public HRESULT RemoveValueAtIndex(UINT32 index)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2ContextMenuRequestedEventArgs

Article • 02/26/2024

```
interface ICoreWebView2ContextMenuRequestedEventArgs
: public IUnknown
```

Event args for the `ContextMenuRequested` event.

## Summary

[ ] Expand table

Members	Descriptions
<a href="#">get_ContextMenuItemTarget</a>	Gets the target information associated with the requested context menu.
<a href="#">get_Handled</a>	Gets whether the <code>ContextMenuRequested</code> event is handled by host.
<a href="#">get_Location</a>	Gets the coordinates where the context menu request occurred in relation to the upper left corner of the WebView bounds.
<a href="#">get_MenuItems</a>	Gets the collection of <code>ContextMenuItem</code> objects.
<a href="#">get_SelectedCommandId</a>	Gets the selected CommandId.
<a href="#">GetDeferral</a>	Returns an <code>ICoreWebView2Deferral</code> object.
<a href="#">put_Handled</a>	Sets whether the <code>ContextMenuRequested</code> event is handled by host after the event handler completes or if there is a deferral then after the deferral is completed.
<a href="#">put_SelectedCommandId</a>	Sets the selected context menu item's command ID.

Will contain the selection information and a collection of all of the default context menu items that the WebView would show. Allows the app to draw its own context menu or add/remove from the default context menu.

# Applies to

 Expand table

Product	Introduced
WebView2 Win32	1.0.1185.39
WebView2 Win32 Prerelease	1.0.1189

## Members

### get\_ContextMenuItemTarget

Gets the target information associated with the requested context menu.

```
public HRESULT get_ContextMenuItemTarget(ICoreWebView2ContextMenuTarget ** value)
```

See [ICoreWebView2ContextMenuTarget](#) for more details.

### get\_Handled

Gets whether the [ContextMenuRequested](#) event is handled by host.

```
public HRESULT get_Handled(BOOL * value)
```

### get\_Location

Gets the coordinates where the context menu request occurred in relation to the upper left corner of the WebView bounds.

```
public HRESULT get_Location(POINT * value)
```

### get\_MenuItems

Gets the collection of [ContextMenuItems](#) objects.

```
public HRESULT get_MenuItems(ICoreWebView2ContextMenuCollection ** value)
```

See [ICoreWebView2ContextMenuItemCollection](#) for more details.

## get\_SelectedCommandId

Gets the selected CommandId.

```
public HRESULT get_SelectedCommandId(INT32 * value)
```

## GetDeferral

Returns an [ICoreWebView2Deferral](#) object.

```
public HRESULT GetDeferral(ICoreWebView2Deferral ** deferral)
```

Use this operation to complete the event when the custom context menu is closed.

## put\_Handled

Sets whether the `ContextMenuRequested` event is handled by host after the event handler completes or if there is a deferral then after the deferral is completed.

```
public HRESULT put_Handled(BOOL value)
```

If `Handled` is set to TRUE then WebView will not display a context menu and will instead use the `SelectedCommandId` property to indicate which, if any, context menu item command to invoke. If after the event handler or deferral completes `Handled` is set to FALSE then WebView will display a context menu based on the contents of the `MenuItems` property. The default value is FALSE.

## put\_SelectedCommandId

Sets the selected context menu item's command ID.

```
public HRESULT put_SelectedCommandId(INT32 value)
```

When this is set, WebView will execute the selected command. This value should always be obtained via the selected `ContextMenuItem`'s `CommandId` property. The default value is -1 which means that no selection occurred. The app can also report the selected command ID for a custom context menu item, which will cause the `CustomItemSelected` event to be fired for the custom item, however while command IDs for each custom

context menu item is unique during a ContextMenuRequested event, CoreWebView2 may reassign command ID values of deleted custom ContextMenuItems to new objects and the command ID assigned to the same custom item can be different between each app runtime.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2ContextMenuTarget

Article • 02/26/2024

```
interface ICoreWebView2ContextMenuTarget
: public IUnknown
```

Represents the information regarding the context menu target.

## Summary

[ ] Expand table

Members	Descriptions
<a href="#">get_FrameUri</a>	Gets the uri of the frame.
<a href="#">get_HasLinkText</a>	Returns TRUE if the context menu is requested on text element that contains an anchor tag.
<a href="#">get_HasLinkUri</a>	Returns TRUE if the context menu is requested on HTML containing an anchor tag.
<a href="#">get_HasSelection</a>	Returns TRUE if the context menu is requested on a selection.
<a href="#">get_HasSourceUri</a>	Returns TRUE if the context menu is requested on HTML containing a source uri.
<a href="#">get_IsEditable</a>	Returns TRUE if the context menu is requested on an editable component.
<a href="#">get_IsRequestedForMainFrame</a>	Returns TRUE if the context menu was requested on the main frame and FALSE if invoked on another frame.
<a href="#">get_Kind</a>	Gets the kind of context that the user selected.
<a href="#">get_LinkText</a>	Gets the text of the link (if <code>HasLinkText</code> is TRUE, null otherwise).
<a href="#">get_LinkUri</a>	Gets the uri of the link (if <code>HasLinkUri</code> is TRUE, null otherwise).
<a href="#">get_PageUri</a>	Gets the uri of the page.
<a href="#">get_SelectionText</a>	Gets the selected text (if <code>HasSelection</code> is TRUE, null otherwise).

Members	Descriptions
<a href="#">get_SourceUri</a>	Gets the active source uri of element (if <code>HasSourceUri</code> is TRUE, null otherwise).

Includes the context selected and the appropriate data used for the actions of a context menu.

## Applies to

[Expand table](#)

Product	Introduced
WebView2 Win32	1.0.1185.39
WebView2 Win32 Prerelease	1.0.1189

## Members

### [get\\_FrameUri](#)

Gets the uri of the frame.

```
public HRESULT get_FrameUri(LPWSTR * value)
```

Will match the PageUri if `IsRequestedForMainFrame` is TRUE.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

### [get\\_HasLinkText](#)

Returns TRUE if the context menu is requested on text element that contains an anchor tag.

```
public HRESULT get_HasLinkText(BOOL * value)
```

### [get\\_HasLinkUri](#)

Returns TRUE if the context menu is requested on HTML containing an anchor tag.

```
public HRESULT get_HasLinkUri(BOOL * value)
```

## get\_HasSelection

Returns TRUE if the context menu is requested on a selection.

```
public HRESULT get_HasSelection(BOOL * value)
```

## get\_HasSourceUri

Returns TRUE if the context menu is requested on HTML containing a source uri.

```
public HRESULT get_HasSourceUri(BOOL * value)
```

## get\_IsEditable

Returns TRUE if the context menu is requested on an editable component.

```
public HRESULT get_IsEditable(BOOL * value)
```

## get\_IsRequestedForMainFrame

Returns TRUE if the context menu was requested on the main frame and FALSE if invoked on another frame.

```
public HRESULT get_IsRequestedForMainFrame(BOOL * value)
```

## get\_Kind

Gets the kind of context that the user selected.

```
public HRESULT get_Kind(COREWEBVIEW2_CONTEXT_MENU_TARGET_KIND * value)
```

## get\_LinkText

Gets the text of the link (if `HasLinkText` is TRUE, null otherwise).

```
public HRESULT get_LinkText(LPWSTR * value)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_LinkUri

Gets the uri of the link (if `HasLinkUri` is TRUE, null otherwise).

```
public HRESULT get_LinkUri(LPWSTR * value)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_PageUri

Gets the uri of the page.

```
public HRESULT get_PageUri(LPWSTR * value)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_SelectionText

Gets the selected text (if `HasSelection` is TRUE, null otherwise).

```
public HRESULT get_SelectionText(LPWSTR * value)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_SourceUri

Gets the active source uri of element (if `HasSourceUri` is TRUE, null otherwise).

```
public HRESULT get_SourceUri(LPWSTR * value)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Controller

Article • 02/26/2024

```
interface ICoreWebView2Controller
: public IUnknown
```

The owner of the `CoreWebView2` object that provides support for resizing, showing and hiding, focusing, and other functionality related to windowing and composition.

## Summary

[ ] Expand table

Members	Descriptions
<a href="#">add_AcceleratorKeyPressed</a>	Adds an event handler for the <code>AcceleratorKeyPressed</code> event.
<a href="#">add_GotFocus</a>	Adds an event handler for the <code>GotFocus</code> event.
<a href="#">add_LostFocus</a>	Adds an event handler for the <code>LostFocus</code> event.
<a href="#">add_MoveFocusRequested</a>	Adds an event handler for the <code>MoveFocusRequested</code> event.
<a href="#">add_ZoomFactorChanged</a>	Adds an event handler for the <code>ZoomFactorChanged</code> event.
<a href="#">Close</a>	Closes the WebView and cleans up the underlying browser instance.
<a href="#">get_Bounds</a>	The WebView bounds.
<a href="#">get_CoreWebView2</a>	Gets the <code>CoreWebView2</code> associated with this <code>CoreWebView2Controller</code> .
<a href="#">get_IsVisible</a>	The <code>IsVisible</code> property determines whether to show or hide the WebView2.
<a href="#">get_ParentWindow</a>	The parent window provided by the app that this WebView is using to render content.
<a href="#">get_ZoomFactor</a>	The zoom factor for the WebView.
<a href="#">MoveFocus</a>	Moves focus into WebView.

Members	Descriptions
<a href="#">NotifyParentWindowPositionChanged</a>	This is a notification separate from <code>Bounds</code> that tells <code>WebView</code> that the main <code>WebView</code> parent (or any ancestor) <code>HWND</code> moved.
<a href="#">put_Bounds</a>	Sets the <code>Bounds</code> property.
<a href="#">put_IsVisible</a>	Sets the <code>IsVisible</code> property.
<a href="#">put_ParentWindow</a>	Sets the parent window for the <code>WebView</code> .
<a href="#">put_ZoomFactor</a>	Sets the <code>ZoomFactor</code> property.
<a href="#">remove_AcceleratorKeyPressed</a>	Removes an event handler previously added with <code>add_AcceleratorKeyPressed</code> .
<a href="#">remove_GotFocus</a>	Removes an event handler previously added with <code>add_GotFocus</code> .
<a href="#">remove_LostFocus</a>	Removes an event handler previously added with <code>add_LostFocus</code> .
<a href="#">remove_MoveFocusRequested</a>	Removes an event handler previously added with <code>add_MoveFocusRequested</code> .
<a href="#">remove_ZoomFactorChanged</a>	Remove an event handler previously added with <code>add_ZoomFactorChanged</code> .
<a href="#">SetBoundsAndZoomFactor</a>	Updates <code>Bounds</code> and <code>ZoomFactor</code> properties at the same time.

The `CoreWebView2Controller` owns the `CoreWebView2`, and if all references to the `CoreWebView2Controller` go away, the `WebView` is closed.

## Applies to

[] [Expand table](#)

Product	Introduced
WebView2 Win32	0.9.488
WebView2 Win32 Prerelease	0.9.488

## Members

## `add_AcceleratorKeyPressed`

Adds an event handler for the `AcceleratorKeyPressed` event.

```
public HRESULT  
add_AcceleratorKeyPressed(ICoreWebView2AcceleratorKeyPressedEventHandler *  
eventHandler, EventRegistrationToken * token)
```

`AcceleratorKeyPressed` runs when an accelerator key or key combo is pressed or released while the WebView is focused. A key is considered an accelerator if either of the following conditions are true.

- Ctrl or Alt is currently being held.
- The pressed key does not map to a character.

A few specific keys are never considered accelerators, such as Shift. The `Escape` key is always considered an accelerator.

Auto-repeated key events caused by holding the key down also triggers this event. Filter out the auto-repeated key events by verifying the `KeyEventLParam` or `PhysicalKeyStatus` event args.

In windowed mode, the event handler is run synchronously. Until you run `Handled()` on the event args or the event handler returns, the browser process is blocked and outgoing cross-process COM requests fail with `RPC_E_CANTCALLOUT_ININPUTSYNCCALL`. All `CoreWebView2` API methods work, however.

In windowless mode, the event handler is run asynchronously. Further input do not reach the browser until the event handler returns or `Handled()` is run, but the browser process is not blocked, and outgoing COM requests work normally.

It is recommended to run `Handled(TRUE)` as early as are able to know that you want to handle the accelerator key.

C++

```
// Register a handler for the AcceleratorKeyPressed event.  
CHECK_FAILURE(m_controller->add_AcceleratorKeyPressed(  
    Callback<ICoreWebView2AcceleratorKeyPressedEventHandler>(  
        [this](  
            ICoreWebView2Controller* sender,  
            ICoreWebView2AcceleratorKeyPressedEventArgs* args) ->  
HRESULT {  
    COREWEBVIEW2_KEY_EVENT_KIND kind;  
    CHECK_FAILURE(args->get_KeyEventKind(&kind));
```

```

        // We only care about key down events.
        if (kind == COREWEBVIEW2_KEY_EVENT_KIND_KEY_DOWN ||
            kind == COREWEBVIEW2_KEY_EVENT_KIND_SYSTEM_KEY_DOWN)
    {
        UINT key;
        CHECK_FAILURE(args->get_VirtualKey(&key));
        // Check if the key is one we want to handle.
        std::function<void()> action = m_appWindow-
>GetAcceleratorKeyFunction(key);
        if (action)
        {
            // Keep the browser from handling this key, whether
it's autorepeated or
            // not.
            CHECK_FAILURE(args->put_Handled(TRUE));

            // Filter out autorepeated keys.
            COREWEBVIEW2_PHYSICAL_KEY_STATUS status;
            CHECK_FAILURE(args->get_PhysicalKeyStatus(&status));
            if (!status.WasKeyDown)
            {
                // Perform the action asynchronously to avoid
blocking the
                // browser process's event queue.
                m_appWindow->RunAsync(action);
            }
        }
    }
    return S_OK;
}
.Get(),
&m_acceleratorKeyPressedToken));

```

## **add\_GotFocus**

Adds an event handler for the `GotFocus` event.

```
public HRESULT add_GotFocus(ICoreWebView2FocusChangedEventArgs *  
eventHandler, EventRegistrationToken * token)
```

`GotFocus` runs when WebView has focus.

## **add\_LostFocus**

Adds an event handler for the `LostFocus` event.

```
public HRESULT add_LostFocus(ICoreWebView2FocusChangedEventArgs *  
eventHandler, EventRegistrationToken * token)
```

`LostFocus` runs when WebView loses focus. In the case where `MoveFocusRequested` event is run, the focus is still on WebView when `MoveFocusRequested` event runs. `LostFocus` only runs afterwards when code of the app or default action of `MoveFocusRequested` event set focus away from WebView.

## add\_MoveFocusRequested

Adds an event handler for the `MoveFocusRequested` event.

```
public HRESULT  
add_MoveFocusRequested(ICoreWebView2MoveFocusRequestedEventHandler *  
eventHandler, EventRegistrationToken * token)
```

`MoveFocusRequested` runs when user tries to tab out of the WebView. The focus of the WebView has not changed when this event is run.

C++

```
// Register a handler for the MoveFocusRequested event.  
// This event will be fired when the user tabs out of the webview.  
// The handler will focus another window in the app, depending on which  
// direction the focus is being shifted.  
CHECK_FAILURE(m_controller->add_MoveFocusRequested(  
    Callback<ICoreWebView2MoveFocusRequestedEventHandler>(  
        [this](  
            ICoreWebView2Controller* sender,  
            ICoreWebView2MoveFocusRequestedEventArgs* args) -> HRESULT {  
                if (!g_autoTabHandle)  
                {  
                    COREWEBVIEW2_MOVE_FOCUS_REASON reason;  
                    CHECK_FAILURE(args->get_Reason(&reason));  
  
                    if (reason == COREWEBVIEW2_MOVE_FOCUS_REASON_NEXT)  
                    {  
                        TabForwards(-1);  
                    }  
                    else if (reason ==  
COREWEBVIEW2_MOVE_FOCUS_REASON_PREVIOUS)  
                    {  
                        TabBackwards(m_tabbableWindows.size());  
                    }  
                    CHECK_FAILURE(args->put_Handled(TRUE));  
                }  
                return S_OK;  
            })  
        .Get(),  
        &m_moveFocusRequestedToken));
```

## `add_ZoomFactorChanged`

Adds an event handler for the `ZoomFactorChanged` event.

```
public HRESULT  
add_ZoomFactorChanged(ICoreWebView2ZoomFactorChangedEventHandler *  
eventHandler, EventRegistrationToken * token)
```

`ZoomFactorChanged` runs when the `ZoomFactor` property of the WebView changes. The event may run because the `ZoomFactor` property was modified, or due to the user manually modifying the zoom. When it is modified using the `ZoomFactor` property, the internal zoom factor is updated immediately and no `ZoomFactorChanged` event is triggered. WebView associates the last used zoom factor for each site. It is possible for the zoom factor to change when navigating to a different page. When the zoom factor changes due to a navigation change, the `ZoomFactorChanged` event runs right after the `ContentLoading` event.

C++

```
// Register a handler for the ZoomFactorChanged event.  
// This handler just announces the new level of zoom on the window's  
title bar.  
CHECK_FAILURE(m_controller->add_ZoomFactorChanged(  
    Callback<ICoreWebView2ZoomFactorChangedEventHandler>(  
        [this](ICoreWebView2Controller* sender, IUnknown* args) ->  
HRESULT {  
    double zoomFactor;  
    CHECK_FAILURE(sender->get_ZoomFactor(&zoomFactor));  
  
    UpdateDocumentTitle(m_appWindow, L" (Zoom: ", zoomFactor);  
    return S_OK;  
})  
.Get(),  
&m_zoomFactorChangedToken));
```

## `Close`

Closes the WebView and cleans up the underlying browser instance.

```
public HRESULT Close()
```

Cleaning up the browser instance releases the resources powering the WebView. The browser instance is shut down if no other WebViews are using it.

After running `Close`, most methods will fail and event handlers stop running. Specifically, the WebView releases the associated references to any associated event handlers when `Close` is run.

`Close` is implicitly run when the `CoreWebView2Controller` loses the final reference and is destructed. But it is best practice to explicitly run `close` to avoid any accidental cycle of references between the WebView and the app code. Specifically, if you capture a reference to the WebView in an event handler you create a reference cycle between the WebView and the event handler. Run `Close` to break the cycle by releasing all event handlers. But to avoid the situation, it is best to both explicitly run `Close` on the WebView and to not capture a reference to the WebView to ensure the WebView is cleaned up correctly. `Close` is synchronous and won't trigger the `beforeunload` event.

C++

```
// Close the WebView and deinitialize related state. This doesn't close the
// app window.
bool AppWindow::CloseWebView(bool cleanupUserDataFolder)
{
    if (auto file = GetComponent<FileComponent>())
    {
        if (file->IsPrintToPdfInProgress())
        {
            int selection = MessageBox(
                m_mainWindow, L"Print to PDF is in progress. Continue
closing?", L"Print to PDF", MB_YESNO);
            if (selection == IDNO)
            {
                return false;
            }
        }
    }
    // 1. Delete components.
    DeleteAllComponents();

    // 2. If cleanup needed and BrowserProcessExited event interface
    available,
    // register to cleanup upon browser exit.
    wil::com_ptr<ICoreWebView2Environment5> environment5;
    if (m_webViewEnvironment)
    {
        environment5 =
m_webViewEnvironment.try_query<ICoreWebView2Environment5>();
    }
    if (cleanupUserDataFolder && environment5)
    {
        // Before closing the WebView, register a handler with code to run
        once the
        // browser process and associated processes are terminated.
    }
}
```

```

    CHECK_FAILURE(environment5->add_BrowserProcessExited(
        Callback<ICoreWebView2BrowserProcessExitedEventHandler>(
            [environment5, this](
                ICoreWebView2Environment* sender,
                ICoreWebView2BrowserProcessExitedEventArgs* args)
        {
            COREWEBVIEW2_BROWSER_PROCESS_EXIT_KIND kind;
            UINT32 pid;
            CHECK_FAILURE(args->get_BrowserProcessExitKind(&kind));
            CHECK_FAILURE(args->get_BrowserProcessId(&pid));

                // If a new WebView is created from this
CoreWebView2Environment after
                // the browser has exited but before our handler gets to
run, a new
                // browser process will be created and lock the user
data folder
                // again. Do not attempt to cleanup the user data folder
in these
                // cases. We check the PID of the exited browser process
against the
                // PID of the browser process to which our last
CoreWebView2 attached.
                if (pid == m_newestBrowserPid)
                {
                    // Watch for graceful browser process exit. Let
ProcessFailed event
                    // handler take care of failed browser process
termination.
                    if (kind ==
COREWEBVIEW2_BROWSER_PROCESS_EXIT_KIND_NORMAL)
                    {
                        CHECK_FAILURE(environment5-
>remove_BrowserProcessExited(
                            m_browserExitedEventToken));
                        // Release the environment only after the
handler is invoked.
                        // Otherwise, there will be no environment to
raise the event when
                        // the collection of WebView2 Runtime processes
exit.
                        m_webViewEnvironment = nullptr;
                        RunAsync([this]() { CleanupUserDataFolder(); });
                    }
                }
                else
                {
                    // The exiting process is not the last in use. Do
not attempt cleanup
                    // as we might still have a webview open over the
user data folder.
                    // Do not block from event handler.
                    AsyncMessageBox(
                        L"A new browser process prevented cleanup of the
user data folder.",

```

```

                L"Cleanup User Data Folder");
            }

            return S_OK;
        })
        .Get(),
        &m_browserExitedEventToken));
    }

    // 3. Close the webview.
    if (m_controller)
    {
        m_controller->Close();
        m_controller = nullptr;
        m_webView = nullptr;
        m_webView3 = nullptr;
    }

    // 4. If BrowserProcessExited event interface is not available, release
    // environment and proceed to cleanup immediately. If the interface is
    // available, release environment only if not waiting for the event.
    if (!environment5)
    {
        m_webViewEnvironment = nullptr;
        if (cleanupUserDataFolder)
        {
            CleanupUserDataFolder();
        }
    }
    else if (!cleanupUserDataFolder)
    {
        // Release the environment object here only if no cleanup is needed.
        // If cleanup is needed, the environment object release is deferred
        // until the browser process exits, otherwise the handler for the
        // BrowserProcessExited event will not be called.
        m_webViewEnvironment = nullptr;
    }

    // reset profile name
    m_profileName = L "";
    m_documentTitle = L "";
    return true;
}

```

## get\_Bounds

The WebView bounds.

```
public HRESULT get_Bounds(RECT * bounds)
```

Bounds are relative to the parent `HWND`. The app has two ways to position a WebView.

- Create a child `HWND` that is the WebView parent `HWND`. Position the window where the WebView should be. Use `(0, 0)` for the top-left corner (the offset) of the `Bounds` of the WebView.
- Use the top-most window of the app as the WebView parent `HWND`. For example, to position WebView correctly in the app, set the top-left corner of the Bound of the WebView.

The values of `Bounds` are limited by the coordinate space of the host.

## get\_CoreWebView2

Gets the `CoreWebView2` associated with this `CoreWebView2Controller`.

```
public HRESULT get_CoreWebView2(ICoreWebView2 ** coreWebView2)
```

## get\_IsVisible

The `IsVisible` property determines whether to show or hide the WebView2.

```
public HRESULT get_IsVisible(BOOL * isVisible)
```

If `IsVisible` is set to `FALSE`, the WebView2 is transparent and is not rendered. However, this does not affect the window containing the WebView2 (the `HWND` parameter that was passed to `CreateCoreWebView2Controller`). If you want that window to disappear too, run `ShowWindow` on it directly in addition to modifying the `IsVisible` property. WebView2 as a child window does not get window messages when the top window is minimized or restored. For performance reasons, developers should set the `IsVisible` property of the WebView to `FALSE` when the app window is minimized and back to `TRUE` when the app window is restored. The app window does this by handling `SIZE_MINIMIZED` and `SIZE_RESTORED` command upon receiving `WM_SIZE` message.

There are CPU and memory benefits when the page is hidden. For instance, Chromium has code that throttles activities on the page like animations and some tasks are run less frequently. Similarly, WebView2 will purge some caches to reduce memory usage.

C++

```
void ViewComponent::ToggleVisibility()
{
    BOOL visible;
    m_controller->get_IsVisible(&visible);
```

```
    m_isVisible = !visible;
    m_controller->put_IsVisible(m_isVisible);
}
```

## get\_ParentWindow

The parent window provided by the app that this WebView is using to render content.

```
public HRESULT get_ParentWindow(HWND * parentWindow)
```

This API initially returns the window passed into `CreateCoreWebView2Controller`.

## get\_ZoomFactor

The zoom factor for the WebView.

```
public HRESULT get_ZoomFactor(double * zoomFactor)
```

### ⓘ Note

Changing zoom factor may cause `window.innerWidth`, `window.innerHeight`, both, and page layout to change. A zoom factor that is applied by the host by running `ZoomFactor` becomes the new default zoom for the WebView. The zoom factor applies across navigations and is the zoom factor WebView is returned to when the user chooses Ctrl+0. When the zoom factor is changed by the user (resulting in the app receiving `ZoomFactorChanged`), that zoom applies only for the current page. Any user applied zoom is only for the current page and is reset on a navigation. Specifying a `zoomFactor` less than or equal to `0` is not allowed. WebView also has an internal supported zoom factor range. When a specified zoom factor is out of that range, it is normalized to be within the range, and a `ZoomFactorChanged` event is triggered for the real applied zoom factor. When the range normalization happens, the `ZoomFactor` property reports the zoom factor specified during the previous modification of the `ZoomFactor` property until the `ZoomFactorChanged` event is received after WebView applies the normalized zoom factor.

## MoveFocus

Moves focus into WebView.

```
public HRESULT MoveFocus(COREWEBVIEW2_MOVE_FOCUS_REASON reason)
```

WebView gets focus and focus is set to correspondent element in the page hosted in the WebView. For Programmatic reason, focus is set to previously focused element or the default element if no previously focused element exists. For `Next` reason, focus is set to the first element. For `Previous` reason, focus is set to the last element. WebView changes focus through user interaction including selecting into a WebView or Tab into it. For tabbing, the app runs `MoveFocus` with `Next` or `Previous` to align with Tab and Shift+Tab respectively when it decides the WebView is the next element that may exist in a tab. Or, the app runs `IsDialogMessage` as part of the associated message loop to allow the platform to auto handle tabbing. The platform rotates through all windows with `WS_TABSTOP`. When the WebView gets focus from `IsDialogMessage`, it is internally put the focus on the first or last element for tab and Shift+Tab respectively.

C++

```
while (GetMessage(&msg, nullptr, 0, 0))
{
    if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))
    {
        // Calling IsDialogMessage handles Tab traversal automatically.

If the
        // app wants the platform to auto handle tab, then call
IsDialogMessage
        // before calling TranslateMessage/DispatchMessage. If the app
wants to
        // handle tabbing itself, then skip calling IsDialogMessage and
call
        // TranslateMessage/DispatchMessage directly.
        if (!g_autoTabHandle || !IsDialogMessage(GetAncestor(msg.hwnd,
GA_ROOT), &msg))
        {
            TranslateMessage(&msg);
            DispatchMessage(&msg);
        }
    }
}
```

C++

```
if (wParam == VK_TAB)
{
    // Find out if the window is one we've customized for tab
handling
    for (size_t i = 0; i < m_tabbableWindows.size(); i++)
    {
        if (m_tabbableWindows[i].first == hWnd)
        {
```

```

        if (GetKeyState(VK_SHIFT) < 0)
        {
            TabBackwards(i);
        }
        else
        {
            TabForwards(i);
        }
        return true;
    }
}
}

```

C++

```

void ControlComponent::TabForwards(size_t currentIndex)
{
    // Find first enabled window after the active one
    for (size_t i = currentIndex + 1; i < m_tabbableWindows.size(); i++)
    {
        HWND hwnd = m_tabbableWindows.at(i).first;
        if (IsWindowEnabled(hwnd))
        {
            SetFocus(hwnd);
            return;
        }
    }
    // If this is the last enabled window, tab forwards into the WebView.
    m_controller->MoveFocus(COREWEBVIEW2_MOVE_FOCUS_REASON_NEXT);
}

void ControlComponent::TabBackwards(size_t currentIndex)
{
    // Find first enabled window before the active one
    for (size_t i = currentIndex - 1; i >= 0 && i <
m_tabbableWindows.size(); i--)
    {
        HWND hwnd = m_tabbableWindows.at(i).first;
        if (IsWindowEnabled(hwnd))
        {
            SetFocus(hwnd);
            return;
        }
    }
    // If this is the last enabled window, tab forwards into the WebView.
    CHECK_FAILURE(m_controller-
>MoveFocus(COREWEBVIEW2_MOVE_FOCUS_REASON_PREVIOUS));
}

```

## NotifyParentWindowPositionChanged

This is a notification separate from `Bounds` that tells WebView that the main WebView parent (or any ancestor) `HWND` moved.

```
public HRESULT NotifyParentWindowPositionChanged()
```

This is needed for accessibility and certain dialogs in WebView to work correctly.

C++

```
if (message == WM_MOVE || message == WM_MOVING)
{
    m_controller->NotifyParentWindowPositionChanged();
    return true;
}
```

## put\_Bounds

Sets the `Bounds` property.

```
public HRESULT put_Bounds(RECT bounds)
```

C++

```
// Update the bounds of the WebView window to fit available space.
void ViewComponent::ResizeWebView()
{
    SIZE webViewSize = {
        LONG((m_webViewBounds.right - m_webViewBounds.left) *
m_webViewRatio * m_webViewScale),
        LONG((m_webViewBounds.bottom - m_webViewBounds.top) *
m_webViewRatio * m_webViewScale) };

    RECT desiredBounds = m_webViewBounds;
    desiredBounds.bottom = LONG(
        webViewSize.cy + m_webViewBounds.top);
    desiredBounds.right = LONG(
        webViewSize.cx + m_webViewBounds.left);

    m_controller->put_Bounds(desiredBounds);
    if (m_compositionController)
    {
        POINT webViewOffset = {m_webViewBounds.left, m_webViewBounds.top};

        if (m_dcompDevice)
        {
            CHECK_FAILURE(m_dcompRootVisual-
>SetOffsetX(float(webViewOffset.x)));
            CHECK_FAILURE(m_dcompRootVisual-
```

```
>SetOffsetY(float(webViewOffset.y)));
    CHECK_FAILURE(m_dcompRootVisual->SetClip(
        {0, 0, float(webViewSize.cx), float(webViewSize.cy)}));
    CHECK_FAILURE(m_dcompDevice->Commit());
}
else if (m_wincompCompositor)
{
    if (m_wincompRootVisual != nullptr)
    {
        numerics::float2 size = {static_cast<float>(webViewSize.cx),
                                static_cast<float>
(webViewSize.cy)};
        m_wincompRootVisual.Size(size);

        numerics::float3 offset = {static_cast<float>
(webViewOffset.x),
                                static_cast<float>
(webViewOffset.y), 0.0f};
        m_wincompRootVisual.Offset(offset);

        winrtComp::IInsetClip insetClip =
m_wincompCompositor.CreateInsetClip();

        m_wincompRootVisual.Clip(insetClip.as<winrtComp::CompositionClip>());
    }
}
}
```

## **put\_IsVisible**

Sets the `IsVisible` property.

public HRESULT put\_IsVisible(BOOL isVisible)

C++

```
if (message == WM_SIZE)
{
    if (wParam == SIZE_MINIMIZED)
    {
        // Hide the webview when the app window is minimized.
        m_controller->put_IsVisible(FALSE);
        Suspend();
    }
    else if (wParam == SIZE_RESTORED)
    {
        // When the app window is restored, show the webview
        // (unless the user has toggle visibility off).
        if (m_isVisible)
        {

```

```
        Resume();
        m_controller->put_IsVisible(TRUE);
    }
}
```

## **put\_ParentWindow**

Sets the parent window for the WebView.

```
public HRESULT put_ParentWindow(HWND parentWindow)
```

This causes the WebView to re-parent the main WebView window to the newly provided window.

## **put\_ZoomFactor**

Sets the `ZoomFactor` property.

```
public HRESULT put_ZoomFactor(double zoomFactor)
```

## **remove\_AcceleratorKeyPressed**

Removes an event handler previously added with `add_AcceleratorKeyPressed`.

```
public HRESULT remove_AcceleratorKeyPressed(EventRegistrationToken token)
```

## **remove\_GotFocus**

Removes an event handler previously added with `add_GotFocus`.

```
public HRESULT remove_GotFocus(EventRegistrationToken token)
```

## **remove\_LostFocus**

Removes an event handler previously added with `add_LostFocus`.

```
public HRESULT remove_LostFocus(EventRegistrationToken token)
```

## **remove\_MoveFocusRequested**

Removes an event handler previously added with `add_MoveFocusRequested`.

```
public HRESULT remove_MoveFocusRequested(EventRegistrationToken token)
```

## remove\_ZoomFactorChanged

Remove an event handler previously added with `add_ZoomFactorChanged`.

```
public HRESULT remove_ZoomFactorChanged(EventRegistrationToken token)
```

## SetBoundsAndZoomFactor

Updates `Bounds` and `ZoomFactor` properties at the same time.

```
public HRESULT SetBoundsAndZoomFactor(RECT bounds, double zoomFactor)
```

This operation is atomic from the perspective of the host. After returning from this function, the `Bounds` and `ZoomFactor` properties are both updated if the function is successful, or neither is updated if the function fails. If `Bounds` and `ZoomFactor` are both updated by the same scale (for example, `Bounds` and `ZoomFactor` are both doubled), then the page does not display a change in `window.innerWidth` or `window.innerHeight` and the WebView renders the content at the new size and zoom without intermediate renderings. This function also updates just one of `ZoomFactor` or `Bounds` by passing in the new value for one and the current value for the other.

C++

```
void ViewComponent::SetScale(float scale)
{
    RECT bounds;
    CHECK_FAILURE(m_controller->get_Bounds(&bounds));
    double scaleChange = scale / m_webViewScale;

    bounds.bottom = LONG(
        (bounds.bottom - bounds.top) * scaleChange + bounds.top);
    bounds.right = LONG(
        (bounds.right - bounds.left) * scaleChange + bounds.left);

    m_webViewScale = scale;
    m_controller->SetBoundsAndZoomFactor(bounds, scale);
}
```

# Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Controller2

Article • 02/26/2024

```
interface ICoreWebView2Controller2
    : public ICoreWebView2Controller
```

A continuation of the [ICoreWebView2Controller](#) interface.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_DefaultBackgroundColor</a>	The <code>DefaultBackgroundColor</code> property is the color WebView renders underneath all web content.
<a href="#">put_DefaultBackgroundColor</a>	Sets the <code>DefaultBackgroundColor</code> property.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.774.44
WebView2 Win32 Prerelease	1.0.790

## Members

### [get\\_DefaultBackgroundColor](#)

The `DefaultBackgroundColor` property is the color WebView renders underneath all web content.

```
public HRESULT get_DefaultBackgroundColor(COREWEBVIEW2_COLOR *
backgroundColor)
```

This means WebView renders this color when there is no web content loaded such as before the initial navigation or between navigations. This also means web pages with undefined css background properties or background properties containing transparent pixels will render their contents over this color. Web pages with defined and opaque background properties that span the page will obscure the `DefaultBackgroundColor` and display normally. The default value for this property is white to resemble the native browser experience.

The Color is specified by the `COREWEBVIEW2_COLOR` that represents an RGBA value. The `A` represents an Alpha value, meaning `DefaultBackgroundColor` can be transparent. In the case of a transparent `DefaultBackgroundColor` WebView will render hosting app content as the background. This Alpha value is not supported on Windows 7. Any `A` value other than 255 will result in `E_INVALIDARG` on Windows 7. It is supported on all other WebView compatible platforms.

Semi-transparent colors are not currently supported by this API and setting `DefaultBackgroundColor` to a semi-transparent color will fail with `E_INVALIDARG`. The only supported alpha values are 0 and 255, all other values will result in `E_INVALIDARG`. `DefaultBackgroundColor` can only be an opaque color or transparent.

This value may also be set by using the `WEBVIEW2_DEFAULT_BACKGROUND_COLOR` environment variable. There is a known issue with background color where setting the color by API can still leave the app with a white flicker before the `DefaultBackgroundColor` takes effect. Setting the color via environment variable solves this issue. The value must be a hex value that can optionally prepend a 0x. The value must account for the alpha value which is represented by the first 2 digits. So any hex value fewer than 8 digits will assume a prepended 00 to the hex value and result in a transparent color. `get_DefaultBackgroundColor` will return the result of this environment variable if used. This environment variable can only set the `DefaultBackgroundColor` once. Subsequent updates to background color must be done through API call.

C++

```
void ViewComponent::SetBackgroundColor(COLORREF color, bool transparent)
{
    m_webViewColor.R = GetRValue(color);
    m_webViewColor.G = GetGValue(color);
    m_webViewColor.B = GetBValue(color);
    m_webViewColor.A = transparent ? 0 : 255;
    wil::com_ptr<ICoreWebView2Controller2> controller2 =
        m_controller.query<ICoreWebView2Controller2>();
    controller2->put_DefaultBackgroundColor(m_webViewColor);
}
```

## `put_DefaultBackgroundColor`

Sets the `DefaultBackgroundColor` property.

```
public HRESULT put_DefaultBackgroundColor(COREWEBVIEW2_COLOR  
backgroundColor)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Controller3

Article • 02/26/2024

```
interface ICoreWebView2Controller3
: public ICoreWebView2Controller2
```

A continuation of the [ICoreWebView2Controller2](#) interface.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">add_RasterizationScaleChanged</a>	Add an event handler for the RasterizationScaleChanged event.
<a href="#">get_BoundsMode</a>	BoundsMode affects how setting the Bounds and RasterizationScale properties work.
<a href="#">get_RasterizationScale</a>	The rasterization scale for the WebView.
<a href="#">get_ShouldDetectMonitorScaleChanges</a>	ShouldDetectMonitorScaleChanges property determines whether the WebView attempts to track monitor DPI scale changes.
<a href="#">put_BoundsMode</a>	Set the BoundsMode property.
<a href="#">put_RasterizationScale</a>	Set the rasterization scale property.
<a href="#">put_ShouldDetectMonitorScaleChanges</a>	Set the ShouldDetectMonitorScaleChanges property.
<a href="#">remove_RasterizationScaleChanged</a>	Remove an event handler previously added with add_RasterizationScaleChanged.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.774.44

Product	Introduced
WebView2 Win32 Prerelease	1.0.824

## Members

### add\_RasterizationScaleChanged

Add an event handler for the RasterizationScaleChanged event.

```
public HRESULT  
add_RasterizationScaleChanged(ICoreWebView2RasterizationScaleChangedEventHa  
ndler * eventHandler, EventRegistrationToken * token)
```

The event is raised when the WebView detects that the monitor DPI scale has changed, ShouldDetectMonitorScaleChanges is true, and the WebView has changed the RasterizationScale property.

C++

```
CHECK_FAILURE(m_controller3->add_RasterizationScaleChanged(  
    Callback<ICoreWebView2RasterizationScaleChangedEventHandler>(  
        [this](ICoreWebView2Controller* sender, IUnknown* args) ->  
HRESULT {  
    double rasterizationScale;  
    CHECK_FAILURE(m_controller3-  
>get_RasterizationScale(&rasterizationScale));  
  
    UpdateDocumentTitle(  
        m_appWindow, L" (RasterizationScale: ",  
rasterizationScale);  
    return S_OK;  
})  
.Get(), &m_rasterizationScaleChangedToken));
```

### get\_BoundsMode

BoundsMode affects how setting the Bounds and RasterizationScale properties work.

```
public HRESULT get_BoundsMode(COREWEBVIEW2_BOUNDS_MODE *  
boundsMode)
```

Bounds mode can either be in COREWEBVIEW2\_BOUNDS\_MODE\_USE\_RAW\_PIXELS mode or COREWEBVIEW2\_BOUNDS\_MODE\_USE\_RASTERIZATION\_SCALE mode.

When the mode is in COREWEBVIEW2\_BOUNDS\_MODE\_USE\_RAW\_PIXELS, setting the bounds property will set the size of the WebView in raw screen pixels. Changing the rasterization scale in this mode won't change the raw pixel size of the WebView and will only change the rasterization scale.

When the mode is in COREWEBVIEW2\_BOUNDS\_MODE\_USE\_RASTERIZATION\_SCALE, setting the bounds property will change the logical size of the WebView which can be described by the following equation:

text

```
Logical size * rasterization scale = Raw Pixel size
```

In this case, changing the rasterization scale will keep the logical size the same and change the raw pixel size.

C++

```
void ViewComponent::SetBoundsMode(COREWEBVIEW2_BOUNDS_MODE boundsMode)
{
    if (m_controller3)
    {
        m_boundsMode = boundsMode;
        CHECK_FAILURE(m_controller3->put_BoundsMode(boundsMode));
        ResizeWebView();
    }
}
```

## get\_RasterizationScale

The rasterization scale for the WebView.

```
public HRESULT get_RasterizationScale(double * scale)
```

The rasterization scale is the combination of the monitor DPI scale and text scaling set by the user. This value should be updated when the DPI scale of the app's top level window changes (i.e. monitor DPI scale changes or window changes monitor) or when the text scale factor of the system changes.

C++

```

case WM_DPICHANGED:
{
    m_toolbar.UpdateDpiAndTextScale();
    if (auto view = GetComponent<ViewComponent>())
    {
        view->UpdateDpiAndTextScale();
    }

    RECT* const newWindowSize = reinterpret_cast<RECT*>(lParam);
    SetWindowPos(
        hWnd, nullptr, newWindowSize->left, newWindowSize->top,
        newWindowSize->right - newWindowSize->left,
        newWindowSize->bottom - newWindowSize->top, SWP_NOZORDER |
    SWP_NOACTIVATE);
    return true;
}
break;

```

C++

```

if
(winrt::try_get_activation_factory<winrt::Windows::UI::ViewManagement::UISettings>())
{
    m_uiSettings = winrt::Windows::UI::ViewManagement::UISettings();
    m_uiSettings.TextScaleFactorChanged({this,
&AppWindow::OnTextScaleChanged});
}

```

C++

```

void AppWindow::OnTextScaleChanged(
    winrt::Windows::UI::ViewManagement::UISettings const& settings,
    winrt::Windows::Foundation::Inspectable const& args)
{
    RunAsync(
        [this]
    {
        m_toolbar.UpdateDpiAndTextScale();
        if (auto view = GetComponent<ViewComponent>())
        {
            view->UpdateDpiAndTextScale();
        }
    });
}

```

Rasterization scale applies to the WebView content, as well as popups, context menus, scroll bars, and so on. Normal app scaling scenarios should use the ZoomFactor

property or SetBoundsAndZoomFactor API which only scale the rendered HTML content and not popups, context menus, scroll bars, and so on.

C++

```
void ViewComponent::SetRasterizationScale(float additionalScale)
{
    if (m_controller3)
    {
        CHECK_FAILURE(m_controller3->put_ShouldDetectMonitorScaleChanges(FALSE));
        m_webviewAdditionalRasterizationScale = additionalScale;
        double rasterizationScale =
            additionalScale * m_appWindow->GetDpiScale() * m_appWindow->GetTextScale();
        CHECK_FAILURE(m_controller3->put_RasterizationScale(rasterizationScale));
    }
}
```

## get\_ShouldDetectMonitorScaleChanges

ShouldDetectMonitorScaleChanges property determines whether the WebView attempts to track monitor DPI scale changes.

```
public HRESULT get_ShouldDetectMonitorScaleChanges(BOOL * value)
```

When true, the WebView will track monitor DPI scale changes, update the RasterizationScale property, and raises RasterizationScaleChanged event. When false, the WebView will not track monitor DPI scale changes, and the app must update the RasterizationScale property itself. RasterizationScaleChanged event will never raise when ShouldDetectMonitorScaleChanges is false. Apps that want to set their own rasterization scale should set this property to false to avoid the WebView2 updating the RasterizationScale property to match the monitor DPI scale.

## put\_BoundsMode

Set the BoundsMode property.

```
public HRESULT put_BoundsMode(COREWEBVIEW2_BOUNDS_MODE boundsMode)
```

## put\_RasterizationScale

Set the rasterization scale property.

```
public HRESULT put_RasterizationScale(double scale)
```

## **put\_ShouldDetectMonitorScaleChanges**

Set the ShouldDetectMonitorScaleChanges property.

```
public HRESULT put_ShouldDetectMonitorScaleChanges(BOOL value)
```

## **remove\_RasterizationScaleChanged**

Remove an event handler previously added with add\_RasterizationScaleChanged.

```
public HRESULT remove_RasterizationScaleChanged(EventRegistrationToken token)
```

---

## **Feedback**

Was this page helpful?



# interface ICoreWebView2Controller4

Article • 02/26/2024

```
interface ICoreWebView2Controller4
: public ICoreWebView2Controller3
```

This is the ICoreWebView2Controller4 interface.

## Summary

[+] [Expand table](#)

Members	Descriptions
<a href="#">get_AllowExternalDrop</a>	Gets the <code>AllowExternalDrop</code> property which is used to configure the capability that dragging objects from outside the bounds of webview2 and dropping into webview2 is allowed or disallowed.
<a href="#">put_AllowExternalDrop</a>	Sets the <code>AllowExternalDrop</code> property which is used to configure the capability that dragging objects from outside the bounds of webview2 and dropping into webview2 is allowed or disallowed.

The ICoreWebView2Controller4 provides interface to enable/disable external drop.

## Applies to

[+] [Expand table](#)

Product	Introduced
WebView2 Win32	1.0.1185.39
WebView2 Win32 Prerelease	1.0.1189

## Members

### [get\\_AllowExternalDrop](#)

Gets the `AllowExternalDrop` property which is used to configure the capability that dragging objects from outside the bounds of webview2 and dropping into webview2 is allowed or disallowed.

```
public HRESULT get_AllowExternalDrop(BOOL * value)
```

The default value is TRUE.

C++

```
wil::com_ptr<ICoreWebView2Controller> controller =
    m_appWindow->GetWebViewController();
wil::com_ptr<ICoreWebView2Controller4> controller4 =
    controller.try_query<ICoreWebView2Controller4>();
if (controller4)
{
    BOOL allowExternalDrop;
    CHECK_FAILURE(controller4-
>get_AllowExternalDrop(&allowExternalDrop));
    if (allowExternalDrop)
    {
        CHECK_FAILURE(controller4-
>put_AllowExternalDrop(FALSE));
        MessageBox(
            nullptr, L"WebView disallows dropping files now.",
            L"WebView AllowExternalDrop property changed",
            MB_OK);
    }
    else
    {
        CHECK_FAILURE(controller4->put_AllowExternalDrop(TRUE));
        MessageBox(
            nullptr, L"WebView allows dropping files now.",
            L"WebView AllowExternalDrop property changed",
            MB_OK);
    }
}
```

## put\_AllowExternalDrop

Sets the `AllowExternalDrop` property which is used to configure the capability that dragging objects from outside the bounds of webview2 and dropping into webview2 is allowed or disallowed.

```
public HRESULT put_AllowExternalDrop(BOOL value)
```

C++

```
wil::com_ptr<ICoreWebView2Controller> controller =
    m_appWindow->GetWebViewController();
wil::com_ptr<ICoreWebView2Controller4> controller4 =
    controller.try_query<ICoreWebView2Controller4>();
if (controller4)
{
    BOOL allowExternalDrop;
    CHECK_FAILURE(controller4-
>get_AllowExternalDrop(&allowExternalDrop));
    if (allowExternalDrop)
    {
        CHECK_FAILURE(controller4-
>put_AllowExternalDrop(FALSE));
        MessageBox(
            nullptr, L"WebView disallows dropping files now.",
            L"WebView AllowExternalDrop property changed",
            MB_OK);
    }
    else
    {
        CHECK_FAILURE(controller4->put_AllowExternalDrop(TRUE));
        MessageBox(
            nullptr, L"WebView allows dropping files now.",
            L"WebView AllowExternalDrop property changed",
            MB_OK);
    }
}
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2ControllerOptions

Article • 02/26/2024

```
interface ICoreWebView2ControllerOptions
: public IUnknown
```

This interface is used to manage profile options that created by 'CreateCoreWebView2ControllerOptions'.

## Summary

[ ] Expand table

Members	Descriptions
<a href="#">get_IsInPrivateModeEnabled</a>	<code>IsInPrivateModeEnabled</code> property is to enable/disable InPrivate mode.
<a href="#">get_ProfileName</a>	<code>ProfileName</code> property is to specify a profile name, which is only allowed to contain the following ASCII characters.
<a href="#">put_IsInPrivateModeEnabled</a>	Sets the <code>IsInPrivateModeEnabled</code> property.
<a href="#">put_ProfileName</a>	Sets the <code>ProfileName</code> property.

C++

```
auto webViewEnvironment10 =
m_webViewEnvironment.try_query<ICoreWebView2Environment10>();
if (!webViewEnvironment10)
{
    FeatureNotAvailable();
    return S_OK;
}

wil::com_ptr<ICoreWebView2ControllerOptions> options;
// The validation of parameters occurs when setting the properties.
HRESULT hr = webViewEnvironment10-
>CreateCoreWebView2ControllerOptions(&options);
if (hr == E_INVALIDARG)
{
    ShowFailure(hr, L"Unable to create WebView2 due to an invalid
```

```
profile name.");
    CloseAppWindow();
    return S_OK;
}
CHECK_FAILURE(hr);
```

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1210.39
WebView2 Win32 Prerelease	1.0.1222

## Members

### get\_IsInPrivateModeEnabled

`IsInPrivateModeEnabled` property is to enable/disable InPrivate mode.

```
public HRESULT get_IsInPrivateModeEnabled(BOOL * value)
```

### get\_ProfileName

`ProfileName` property is to specify a profile name, which is only allowed to contain the following ASCII characters.

```
public HRESULT get_ProfileName(LPWSTR * value)
```

It has a maximum length of 64 characters excluding the null-terminator. It is ASCII case insensitive.

- alphabet characters: a-z and A-Z
- digit characters: 0-9
- and '#', '@', '\$', '(', ')', '+', '-', '\_', '~', !, '' (space).

Note: the text must not end with a period '.' or '' (space). And, although upper-case letters are allowed, they're treated just as lower-case counterparts because the profile

name will be mapped to the real profile directory path on disk and Windows file system handles path names in a case-insensitive way.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## **put\_IsInPrivateModeEnabled**

Sets the `IsInPrivateModeEnabled` property.

```
public HRESULT put_IsInPrivateModeEnabled(BOOL value)
```

## **put\_ProfileName**

Sets the `ProfileName` property.

```
public HRESULT put_ProfileName(LPCWSTR value)
```

---

## **Feedback**

Was this page helpful?

 Yes

 No

# interface ICoreWebView2ControllerOptions2

Article • 02/26/2024

```
interface ICoreWebView2ControllerOptions2
    : public ICoreWebView2ControllerOptions
```

This is the interface in ControllerOptions for ScriptLocale.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_ScriptLocale</a>	The default locale for the WebView2.
<a href="#">put_ScriptLocale</a>	Sets the <code>ScriptLocale</code> property.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1661.34
WebView2 Win32 Prerelease	1.0.1671

## Members

### [get\\_ScriptLocale](#)

The default locale for the WebView2.

```
public HRESULT get_ScriptLocale(LPWSTR * locale)
```

It sets the default locale for all Intl JavaScript APIs and other JavaScript APIs that depend on it, namely `Intl.DateTimeFormat()` which affects string formatting like in the time/date formats. Example: `Intl.DateTimeFormat().format(new Date())` The intended locale value is in the format of BCP 47 Language Tags. More information can be found from [IETF BCP47](#).

This property sets the locale for a CoreWebView2Environment used to create the `WebView2ControllerOptions` object, which is passed as a parameter in `CreateCoreWebView2ControllerWithOptions`.

Changes to the `ScriptLocale` property apply to renderer processes created after the change. Any existing renderer processes will continue to use the previous `ScriptLocale` value. To ensure changes are applied to all renderer process, close and restart the `CoreWebView2Environment` and all associated `WebView2` objects.

The default value for `ScriptLocale` will depend on the `WebView2` language and OS region. If the language portions of the `WebView2` language and OS region match, then it will use the OS region. Otherwise, it will use the `WebView2` language.

[+] Expand table

OS Region	WebView2 Language	Default WebView2 ScriptLocale
en-GB	en-US	en-GB
es-MX	en-US	en-US
en-US	en-GB	en-US

You can set the `ScriptLocale` to the empty string to get the default `ScriptLocale` value.

Use OS specific APIs to determine the OS region to use with this property if you want to match the OS. For example:

Win32 C++:

C++

```
wchar_t osLocale[LOCALE_NAME_MAX_LENGTH] = {0};  
 GetUserDefaultLocaleName(osLocale, LOCALE_NAME_MAX_LENGTH);
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

C++

```
wil::com_ptr<ICoreWebView2ControllerOptions2>
webView2ControllerOptions2;
    if (SUCCEEDED(options-
>QueryInterface(IID_PPV_ARGS(&webView2ControllerOptions2))))
    {
        if (m_webviewOption.useOSRegion)
        {
            wchar_t osLocale[LOCALE_NAME_MAX_LENGTH] = {0};
            GetUserDefaultLocaleName(osLocale, LOCALE_NAME_MAX_LENGTH);
            CHECK_FAILURE(webView2ControllerOptions2-
>put_ScriptLocale(osLocale));
        }
        else if (!m_webviewOption.scriptLocale.empty())
        {
            CHECK_FAILURE(webView2ControllerOptions2->put_ScriptLocale(
                m_webviewOption.scriptLocale.c_str()));
        }
    }
}
```

## put\_ScriptLocale

Sets the `ScriptLocale` property.

```
public HRESULT put_ScriptLocale(LPCWSTR locale)
```

---

## Feedback

Was this page helpful?



# interface ICoreWebView2Cookie

Article • 02/26/2024

```
interface ICoreWebView2Cookie
: public IUnknown
```

Provides a set of properties that are used to manage an ICoreWebView2Cookie.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_Domain</a>	The domain for which the cookie is valid.
<a href="#">get_Expires</a>	The expiration date and time for the cookie as the number of seconds since the UNIX epoch.
<a href="#">get_IsHttpOnly</a>	Whether this cookie is http-only.
<a href="#">get_IsSecure</a>	The security level of this cookie.
<a href="#">get_IsSession</a>	Whether this is a session cookie. The default is false.
<a href="#">get_Name</a>	Cookie name.
<a href="#">get_Path</a>	The path for which the cookie is valid.
<a href="#">get_SameSite</a>	SameSite status of the cookie which represents the enforcement mode of the cookie.
<a href="#">get_Value</a>	Cookie value.
<a href="#">put_Expires</a>	Set the Expires property.
<a href="#">put_IsHttpOnly</a>	Set the IsHttpOnly property.
<a href="#">put_IsSecure</a>	Set the IsSecure property.
<a href="#">put_SameSite</a>	Set the SameSite property.
<a href="#">put_Value</a>	Set the cookie value property.

C++

```

wil::unique_cotaskmem_string name;
CHECK_FAILURE(cookie->get_Name(&name));
wil::unique_cotaskmem_string value;
CHECK_FAILURE(cookie->get_Value(&value));
wil::unique_cotaskmem_string domain;
CHECK_FAILURE(cookie->get_Domain(&domain));
wil::unique_cotaskmem_string path;
CHECK_FAILURE(cookie->get_Path(&path));
double expires;
CHECK_FAILURE(cookie->get_Expires(&expires));
BOOL isHttpOnly = FALSE;
CHECK_FAILURE(cookie->get_IsHttpOnly(&isHttpOnly));
COREWEBVIEW2_COOKIE_SAME_SITE_KIND same_site;
std::wstring same_site_as_string;
CHECK_FAILURE(cookie->get_SameSite(&same_site));
switch (same_site)
{
case COREWEBVIEW2_COOKIE_SAME_SITE_KIND_NONE:
    same_site_as_string = L"None";
    break;
case COREWEBVIEW2_COOKIE_SAME_SITE_KIND_LAX:
    same_site_as_string = L"Lax";
    break;
case COREWEBVIEW2_COOKIE_SAME_SITE_KIND_STRICT:
    same_site_as_string = L"Strict";
    break;
}
BOOL isSecure = FALSE;
CHECK_FAILURE(cookie->get_IsSecure(&isSecure));
BOOL isSession = FALSE;
CHECK_FAILURE(cookie->get_IsSession(&isSession));

std::wstring result = L"";
result += L"\\"Name\\": " + EncodeQuote(name.get()) + L", " + L"\\"Value\\":
" +
        EncodeQuote(value.get()) + L", " + L"\\"Domain\\": " +
EncodeQuote(domain.get()) +
        L", " + L"\\"Path\\": " + EncodeQuote(path.get()) + L", " +
L"\\"HttpOnly\\": " +
        BoolToString(isHttpOnly) + L", " + L"\\"Secure\\": " +
BoolToString(isSecure) + L", " +
        L"\\"SameSite\\": " + EncodeQuote(same_site_as_string) + L", " +
L"\\"Expires\\": ";
if (!!isSession)
{
    result += L"This is a session cookie.";
}
else
{
    result += std::to_wstring(expires);
}

return result + L"\\"}";

```

# Applies to

[Expand table](#)

Product	Introduced
WebView2 Win32	1.0.705.50
WebView2 Win32 Prerelease	1.0.721

## Members

### get\_Domain

The domain for which the cookie is valid.

```
public HRESULT get_Domain(LPWSTR * domain)
```

The default is the host that this cookie has been received from. Note that, for instance, ".bing.com", "bing.com", and "www.bing.com" are considered different domains.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

### get\_Expires

The expiration date and time for the cookie as the number of seconds since the UNIX epoch.

```
public HRESULT get_Expires(double * expires)
```

The default is -1.0, which means cookies are session cookies by default.

### get\_IsHttpOnly

Whether this cookie is http-only.

```
public HRESULT get_IsHttpOnly(BOOL * isHttpOnly)
```

True if a page script or other active content cannot access this cookie. The default is false.

## **get\_IsSecure**

The security level of this cookie.

```
public HRESULT get_IsSecure(BOOL * isSecure)
```

True if the client is only to return the cookie in subsequent requests if those requests use HTTPS. The default is false. Note that cookie that requests COREWEBVIEW2\_COOKIE\_SAME\_SITE\_KIND\_NONE but is not marked Secure will be rejected.

## **get\_IsSession**

Whether this is a session cookie. The default is false.

```
public HRESULT get_IsSession(BOOL * isSession)
```

## **get\_Name**

Cookie name.

```
public HRESULT get_Name(LPWSTR * name)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## **get\_Path**

The path for which the cookie is valid.

```
public HRESULT get_Path(LPWSTR * path)
```

The default is "/", which means this cookie will be sent to all pages on the Domain.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## **get\_SameSite**

SameSite status of the cookie which represents the enforcement mode of the cookie.

```
public HRESULT get_SameSite(COREWEBVIEW2_COOKIE_SAME_SITE_KIND * sameSite)
```

The default is COREWEBVIEW2\_COOKIE\_SAME\_SITE\_KIND\_LAX.

## get\_Value

Cookie value.

```
public HRESULT get_Value(LPWSTR * value)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## put\_Expires

Set the Expires property.

```
public HRESULT put_Expires(double expires)
```

Cookies are session cookies and will not be persistent if Expires is set to -1.0. NaN, infinity, and any negative value set other than -1.0 is disallowed.

## put\_IsHttpOnly

Set the IsHttpOnly property.

```
public HRESULT put_IsHttpOnly(BOOL isHttpOnly)
```

## put\_IsSecure

Set the IsSecure property.

```
public HRESULT put_IsSecure(BOOL isSecure)
```

## put\_SameSite

Set the SameSite property.

```
public HRESULT put_SameSite(COREWEBVIEW2_COOKIE_SAME_SITE_KIND sameSite)
```

## put\_Value

Set the cookie value property.

```
public HRESULT put_Value(LPCWSTR value)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2CookieList

Article • 02/26/2024

```
interface ICoreWebView2CookieList
: public IUnknown
```

A list of cookie objects.

## Summary

[\[+\] Expand table](#)

Members	Descriptions
<a href="#">get_Count</a>	The number of cookies contained in the ICoreWebView2CookieList.
<a href="#">GetValueAtIndex</a>	Gets the cookie object at the given index.

See [ICoreWebView2Cookie](#).

C++

```
if (m_cookieManager)
{
    CHECK_FAILURE(m_cookieManager->GetCookies(
        uri.c_str(),
        Callback<ICoreWebView2GetCookiesCompletedHandler>(
            [this, uri](HRESULT error_code, ICoreWebView2CookieList*
list) -> HRESULT {
            CHECK_FAILURE(error_code);

            std::wstring result;
            UINT cookie_list_size;
            CHECK_FAILURE(list->get_Count(&cookie_list_size));

            if (cookie_list_size == 0)
            {
                result += L"No cookies found.";
            }
            else
            {
                result += std::to_wstring(cookie_list_size) + L"
cookie(s) found";
                if (!uri.empty())
                {

```

```

        result += L" on " + uri;
    }
    result += L"\n\n[";
    for (UINT i = 0; i < cookie_list_size; ++i)
    {
        wil::com_ptr<ICoreWebView2Cookie> cookie;
        CHECK_FAILURE(list->GetValueAtIndex(i,
&cookie));

        if (cookie.get())
        {
            result += CookieToString(cookie.get());
            if (i != cookie_list_size - 1)
            {
                result += L",\n";
            }
        }
        result += L"]";
    }
    m_appWindow->AsyncMessageBox(std::move(result),
L"GetCookies Result");
    return S_OK;
}
.Get()));
}

```

## Applies to

  Expand table

Product	Introduced
WebView2 Win32	1.0.705.50
WebView2 Win32 Prerelease	1.0.721

## Members

### get\_Count

The number of cookies contained in the ICoreWebView2CookieList.

```
public HRESULT get_Count(UINT * count)
```

### GetValueAtIndex

Gets the cookie object at the given index.

```
public HRESULT GetValueAtIndex(UINT index, ICoreWebView2Cookie ** cookie)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2CookieManager

Article • 02/26/2024

```
interface ICoreWebView2CookieManager
: public IUnknown
```

Creates, adds or updates, gets, or or view the cookies.

## Summary

[ ] Expand table

Members	Descriptions
<a href="#">AddOrUpdateCookie</a>	Adds or updates a cookie with the given cookie data; may overwrite cookies with matching name, domain, and path if they exist.
<a href="#">CopyCookie</a>	Creates a cookie whose params matches those of the specified cookie.
<a href="#">CreateCookie</a>	Create a cookie object with a specified name, value, domain, and path.
<a href="#">DeleteAllCookies</a>	Deletes all cookies under the same profile.
<a href="#">DeleteCookie</a>	Deletes a cookie whose name and domain/path pair match those of the specified cookie.
<a href="#">DeleteCookies</a>	Deletes cookies with matching name and uri.
<a href="#">DeleteCookiesWithDomainAndPath</a>	Deletes cookies with matching name and domain/path pair.
<a href="#">GetCookies</a>	Gets a list of cookies matching the specific URI.

The changes would apply to the context of the user profile. That is, other WebViews under the same user profile could be affected.

## Applies to

Product	Introduced
WebView2 Win32	1.0.705.50
WebView2 Win32 Prerelease	1.0.721

## Members

### AddOrUpdateCookie

Adds or updates a cookie with the given cookie data; may overwrite cookies with matching name, domain, and path if they exist.

```
public HRESULT AddOrUpdateCookie(ICoreWebView2Cookie * cookie)
```

This method will fail if the domain of the given cookie is not specified.

C++

```
wil::com_ptr<ICoreWebView2Cookie> cookie;
CHECK_FAILURE(m_cookieManager->CreateCookie(
    L"CookieName", L"CookieValue", L".bing.com", L"/",
&cookie));
CHECK_FAILURE(m_cookieManager-
>AddOrUpdateCookie(cookie.get()));
```

### CopyCookie

Creates a cookie whose params matches those of the specified cookie.

```
public HRESULT CopyCookie(ICoreWebView2Cookie * cookieParam,
ICoreWebView2Cookie ** cookie)
```

### CreateCookie

Create a cookie object with a specified name, value, domain, and path.

```
public HRESULT CreateCookie(LPCWSTR name, LPCWSTR value, LPCWSTR domain,
LPCWSTR path, ICoreWebView2Cookie ** cookie)
```

One can set other optional properties after cookie creation. This only creates a cookie object and it is not added to the cookie manager until you call AddOrUpdateCookie. Leading or trailing whitespace(s), empty string, and special characters are not allowed for name. See [ICoreWebView2Cookie](#) for more details.

## DeleteAllCookies

Deletes all cookies under the same profile.

```
public HRESULT DeleteAllCookies()
```

This could affect other WebViews under the same user profile.

## DeleteCookie

Deletes a cookie whose name and domain/path pair match those of the specified cookie.

```
public HRESULT DeleteCookie(ICoreWebView2Cookie * cookie)
```

## DeleteCookies

Deletes cookies with matching name and uri.

```
public HRESULT DeleteCookies(LPCWSTR name, LPCWSTR uri)
```

Cookie name is required. All cookies with the given name where domain and path match provided URI are deleted.

## DeleteCookiesWithDomainAndPath

Deletes cookies with matching name and domain/path pair.

```
public HRESULT DeleteCookiesWithDomainAndPath(LPCWSTR name, LPCWSTR  
domain, LPCWSTR path)
```

Cookie name is required. If domain is specified, deletes only cookies with the exact domain. If path is specified, deletes only cookies with the exact path.

## GetCookies

Gets a list of cookies matching the specific URI.

```
public HRESULT GetCookies(LPCWSTR uri,  
ICoreWebView2GetCookiesCompletedHandler * handler)
```

If uri is empty string or null, all cookies under the same profile are returned. You can modify the cookie objects by calling [ICoreWebView2CookieManager::AddOrUpdateCookie](#), and the changes will be applied to the webview.

C++

```
if (m_cookieManager)  
{  
    CHECK_FAILURE(m_cookieManager->GetCookies(  
        uri.c_str(),  
        Callback<ICoreWebView2GetCookiesCompletedHandler>(  
            [this, uri](HRESULT error_code, ICoreWebView2CookieList*  
list) -> HRESULT {  
                CHECK_FAILURE(error_code);  
  
                std::wstring result;  
                UINT cookie_list_size;  
                CHECK_FAILURE(list->get_Count(&cookie_list_size));  
  
                if (cookie_list_size == 0)  
                {  
                    result += L"No cookies found.";  
                }  
                else  
                {  
                    result += std::to_wstring(cookie_list_size) + L"  
cookie(s) found";  
                    if (!uri.empty())  
                    {  
                        result += L" on " + uri;  
                    }  
                    result += L"\n\n[";  
                    for (UINT i = 0; i < cookie_list_size; ++i)  
                    {  
                        wil::com_ptr<ICoreWebView2Cookie> cookie;  
                        CHECK_FAILURE(list->GetValueAtIndex(i,  
&cookie));  
  
                        if (cookie.get())  
                        {  
                            result += CookieToString(cookie.get());  
                            if (i != cookie_list_size - 1)  
                            {  
                                result += L",\n";  
                            }  
                        }  
                    }  
                }  
            }  
        );  
    }  
}
```

```
        }
    }
    result += L"]";
}
m_appWindow->AsyncMessageBox(std::move(result),
L"GetCookies Result");
return S_OK;
})
.Get()));
}
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2CustomSchemeRegistration

Article • 02/26/2024

```
interface ICoreWebView2CustomSchemeRegistration
: public IUnknown
```

Represents the registration of a custom scheme with the CoreWebView2Environment.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_HasAuthorityComponent</a>	Set this property to <code>true</code> if the URLs with this custom scheme will have an authority component (a host for custom schemes).
<a href="#">get_SchemeName</a>	The name of the custom scheme to register.
<a href="#">get_TreatAsSecure</a>	Whether the sites with this scheme will be treated as a <a href="#">Secure Context</a> like an HTTPS site.
<a href="#">GetAllowedOrigins</a>	List of origins that are allowed to issue requests with the custom scheme, such as XHRs and subresource requests that have an Origin header.
<a href="#">put_HasAuthorityComponent</a>	Get has authority component.
<a href="#">put_TreatAsSecure</a>	Set if the scheme will be treated as a Secure Context.
<a href="#">SetAllowedOrigins</a>	Set the array of origins that are allowed to use the scheme.

This allows the WebView2 app to be able to handle WebResourceRequested event for requests with the specified scheme and be able to navigate the WebView2 to the custom scheme. Once the environment is created, the registrations are valid and immutable throughout the lifetime of the associated WebView2's browser process and any WebView2 environments sharing the browser process must be created with identical custom scheme registrations, otherwise the environment creation will fail. Any further

attempts to register the same scheme will fail during environment creation. The URLs of registered custom schemes will be treated similar to http URLs for their origins. They will have tuple origins for URLs with host and opaque origins for URLs without host as specified in [7.5 Origin - HTML Living Standard](#)

Example: `custom-scheme-with-host://hostname/path/to/resource` has origin of `custom-scheme-with-host://hostname`. `custom-scheme-without-host:path/to/resource` has origin of `custom-scheme-without-host:path/to/resource`. For `WebResourceRequested` event, the cases of request URLs and filter URLs with custom schemes will be normalized according to generic URI syntax rules. Any non-ASCII characters will be preserved. The registered custom schemes also participate in [CORS](#) and adheres to [CSP](#). The app needs to set the appropriate access headers in its `WebResourceRequested` event handler to allow CORS requests.

C++

```
Microsoft::WRL::ComPtr<ICoreWebView2EnvironmentOptions4> options4;
if (options.As(&options4) == S_OK)
{
    const WCHAR* allowedOrigins[1] = {L"https://*.example.com"};

    auto customSchemeRegistration =
        Microsoft::WRL::Make<CoreWebView2CustomSchemeRegistration>
(L"custom-scheme");
    customSchemeRegistration->SetAllowedOrigins(1, allowedOrigins);
    auto customSchemeRegistration2 =
        Microsoft::WRL::Make<CoreWebView2CustomSchemeRegistration>
(L"wv2rocks");
    customSchemeRegistration2->put_TreatAsSecure(TRUE);
    customSchemeRegistration2->SetAllowedOrigins(1, allowedOrigins);
    customSchemeRegistration2->put_HasAuthorityComponent(TRUE);
    auto customSchemeRegistration3 =
        Microsoft::WRL::Make<CoreWebView2CustomSchemeRegistration>(
            L"custom-scheme-not-in-allowed-origins");
    ICoreWebView2CustomSchemeRegistration* registrations[3] = {
        customSchemeRegistration.Get(), customSchemeRegistration2.Get(),
        customSchemeRegistration3.Get()};
    options4->SetCustomSchemeRegistrations(
        2, static_cast<ICoreWebView2CustomSchemeRegistration**>
(registrations));
}
```

## Applies to

[ ] [Expand table](#)

Product	Introduced
WebView2 Win32	1.0.1587.40
WebView2 Win32 Prerelease	1.0.1619

## Members

### get\_HasAuthorityComponent

Set this property to `true` if the URIs with this custom scheme will have an authority component (a host for custom schemes).

```
public HRESULT get_HasAuthorityComponent(BOOL * hasAuthorityComponent)
```

Specifically, if you have a URI of the following form you should set the `HasAuthorityComponent` value as listed.

[+] Expand table

URI	Recommended HasAuthorityComponent value
<code>custom-scheme-with-authority://host/path</code>	<code>true</code>
<code>custom-scheme-without-authority:path</code>	<code>false</code>

When this property is set to `true`, the URIs with this scheme will be interpreted as having a [scheme and host](#) origin similar to an http URI. Note that the port and user information are never included in the computation of origins for custom schemes. If this property is set to `false`, URLs with this scheme will have an [opaque origin](#) similar to a data URI. This property is `false` by default.

Note: For custom schemes registered as having authority component, navigations to URIs without authority of such custom schemes will fail. However, if the content inside WebView2 references a subresource with a URI that does not have an authority component, but of a custom scheme that is registered as having authority component, the URI will be interpreted as a relative path as specified in [RFC3986](#). For example, `custom-scheme-with-authority:path` will be interpreted as `custom-scheme-with-authority://host/path`. However, this behavior cannot be guaranteed to remain in future releases so it is recommended not to rely on this behavior.

## get\_SchemeName

The name of the custom scheme to register.

```
public HRESULT get_SchemeName(LPWSTR * schemeName)
```

## get\_TreatAsSecure

Whether the sites with this scheme will be treated as a [Secure Context](#) like an HTTPS site.

```
public HRESULT get_TreatAsSecure(BOOL * treatAsSecure)
```

This flag is only effective when HasAuthorityComponent is also set to `true`. `false` by default.

## GetAllowedOrigins

List of origins that are allowed to issue requests with the custom scheme, such as XHRs and subresource requests that have an Origin header.

```
public HRESULT GetAllowedOrigins(UINT32 * allowedOriginsCount, LPWSTR ** allowedOrigins)
```

The origin of any request (requests that have the [Origin header](#)) to the custom scheme URI needs to be in this list. No-origin requests are requests that do not have an Origin header, such as link navigations, embedded images and are always allowed. Note: POST requests always contain an Origin header, therefore AllowedOrigins must be set for even for same origin POST requests. Note that cross-origin restrictions still apply. From any opaque origin (Origin header is null), no cross-origin requests are allowed. If the list is empty, no cross-origin request to this scheme is allowed. Origins are specified as a string in the format of scheme://host:port. The origins are string pattern matched with `*` (matches 0 or more characters) and `?` (matches 0 or 1 character) wildcards just like the URI matching in the [AddWebResourceRequestedFilter API](#). For example, "http://\*.example.com:80". Here's a set of examples of what is allowed and not:

[ ] Expand table

Request URI	Originating URL	AllowedOrigins	Allowed
custom-scheme:request	https://www.example.com	{"https://www.example.com"}	Yes

Request URI	Originating URL	AllowedOrigins	Allowed
custom-scheme:request	https://www.example.com	{"https://*.example.com"}	Yes
custom-scheme:request	https://www.example.com	{"https://www.example2.com"}	No
custom-scheme-with-authority://host/path	custom-scheme-with-authority://host2	{"("")"}	No
custom-scheme-with-authority://host/path	custom-scheme-with-authority2://host	{"custom-scheme-with-authority2://"}	Yes
custom-scheme-without-authority:path	custom-scheme-without-authority:path2	{"custom-scheme-without-authority:"*}"}	No
custom-scheme-without-authority:path	custom-scheme-without-authority:path2	{"*}"}	Yes

The returned strings and the array itself must be deallocated with CoTaskMemFree.

## put\_HasAuthorityComponent

Get has authority component.

```
public HRESULT put_HasAuthorityComponent(BOOL hasAuthorityComponent)
```

## put\_TreatAsSecure

Set if the scheme will be treated as a Secure Context.

```
public HRESULT put_TreatAsSecure(BOOL value)
```

## SetAllowedOrigins

Set the array of origins that are allowed to use the scheme.

```
public HRESULT SetAllowedOrigins(UINT32 allowedOriginsCount, LPCWSTR * allowedOrigins)
```

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Deferral

Article • 02/26/2024

```
interface ICoreWebView2Deferral  
: public IUnknown
```

This interface is used to complete deferrals on event args that support getting deferrals using the `GetDeferral` method.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Complete</a>	Completes the associated deferred event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### Complete

Completes the associated deferred event.

```
public HRESULT Complete()
```

Complete should only be run once for each deferral taken.

# Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2DevToolsProtocolEventArgs

Article • 02/26/2024

```
interface ICoreWebView2DevToolsProtocolEventArgs
: public IUnknown
```

Event args for the `DevToolsProtocolEventReceived` event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_ParameterObjectAsJson</a>	The parameter object of the corresponding <code>DevToolsProtocol</code> event represented as a JSON string.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### [get\\_ParameterObjectAsJson](#)

The parameter object of the corresponding `DevToolsProtocol` event represented as a JSON string.

```
public HRESULT get_ParameterObjectAsJson(LPWSTR * parameterObjectAsJson)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2DevToolsProtocolEventArgsReceivedEventArgs2

Article • 02/26/2024

```
interface ICoreWebView2DevToolsProtocolEventArgsReceivedEventArgs2
: public ICoreWebView2DevToolsProtocolEventArgsReceivedEventArgs
```

This is a continuation of the ICoreWebView2DevToolsProtocolEventArgsReceivedEventArgs interface that provides the session ID of the target where the event originates from.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_SessionId</a>	The sessionId of the target where the event originates from.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1185.39
WebView2 Win32 Prerelease	1.0.1189

## Members

### [get\\_SessionId](#)

The sessionId of the target where the event originates from.

```
public HRESULT get_SessionId(LPWSTR * sessionId)
```

Empty string is returned as sessionId if the event comes from the default session for the top page.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

C++

```
wil::com_ptr<ICoreWebView2DevToolsProtocolEventArgs2> args2;
    if (SUCCEEDED(args-
>QueryInterface(IID_PPV_ARGS(&args2))))
    {
        wil::unique_cotaskmem_string sessionId;
        CHECK_FAILURE(args2->get_SessionId(&sessionId));
        if (sessionId.get() && *sessionId.get())
        {
            title = eventName + L" (session:" +
sessionId.get() + L")";
            std::wstring targetId =
m_devToolsSessionMap[sessionId.get()];
            std::wstring targetLabel =
m_devToolsTargetLabelMap[targetId];
            details = L"From " + targetLabel + L" (session:"
+ sessionId.get() +
L")\r\n" + details;
        }
    }
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2DevToolsProtocolEventReceiver

Article • 02/26/2024

```
interface ICoreWebView2DevToolsProtocolEventReceiver
: public IUnknown
```

A Receiver is created for a particular DevTools Protocol event and allows you to subscribe and unsubscribe from that event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">add_DevToolsProtocolEventReceived</a>	Subscribe to a <code>DevToolsProtocol</code> event.
<a href="#">remove_DevToolsProtocolEventReceived</a>	Remove an event handler previously added with <code>add_DevToolsProtocolEventReceived</code> .

Obtained from the WebView object using `GetDevToolsProtocolEventReceiver`.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

## add\_DevToolsProtocolEventReceived

Subscribe to a `DevToolsProtocol` event.

```
public HRESULT  
add_DevToolsProtocolEventReceived(ICoreWebView2DevToolsProtocolEventReceive  
dEventHandler * handler, EventRegistrationToken * token)
```

The `Invoke` method of the `handler` runs whenever the corresponding `DevToolsProtocol` event runs. `Invoke` runs with an event args object containing the parameter object of the DevTools Protocol event as a JSON string.

C++

```
// Prompt the user to name a CDP event, and then subscribe to that event.  
void ScriptComponent::SubscribeToCdpEvent()  
{  
    TextInputDialog dialog(  
        m_appWindow->GetMainWindow(),  
        L"Subscribe to CDP Event",  
        L"CDP event name:",  
        L"Enter the name of the CDP event to subscribe to.\r\n"  
        L"You may also have to call the \"enable\" method of the\r\n"  
        L"event's domain to receive events (for example  
        \"Log.enable\").\r\n",  
        L"Log.entryAdded");  
    if (dialog.confirmed)  
    {  
        std::wstring eventName = dialog.input;  
        wil::com_ptr<ICoreWebView2DevToolsProtocolEventReceiver> receiver;  
        CHECK_FAILURE(  
            m_webView->GetDevToolsProtocolEventReceiver(eventName.c_str(),  
&receiver));  
  
        // If we are already subscribed to this event, unsubscribe first.  
        auto preexistingToken =  
m_devToolsProtocolEventReceivedTokenMap.find(eventName);  
        if (preexistingToken !=  
m_devToolsProtocolEventReceivedTokenMap.end())  
        {  
            CHECK_FAILURE(receiver->remove_DevToolsProtocolEventReceived(  
                preexistingToken->second));  
        }  
  
        CHECK_FAILURE(receiver->add_DevToolsProtocolEventReceived(  
            Callback<ICoreWebView2DevToolsProtocolEventReceivedEventHandler>  
            (  
                [this, eventName](  
                    ICoreWebView2* sender,  
                    ICoreWebView2DevToolsProtocolEventReceivedEventArgs*  
args) -> HRESULT
```

```

    {
        wil::unique_cotaskmem_string parameterObjectAsJson;
        CHECK_FAILURE(args-
>get_ParameterObjectAsJson(&parameterObjectAsJson));
        std::wstring title = eventName;
        std::wstring details = parameterObjectAsJson.get();

wil::com_ptr<ICoreWebView2DevToolsProtocolEventReceivedEventArgs2> args2;
        if (SUCCEEDED(args-
>QueryInterface(IID_PPV_ARGS(&args2)))
    {
        wil::unique_cotaskmem_string sessionId;
        CHECK_FAILURE(args2->get_SessionId(&sessionId));
        if (sessionId.get() && *sessionId.get())
        {
            title = eventName + L" (session:" +
sessionId.get() + L")";
            std::wstring targetId =
m_devToolsSessionMap[sessionId.get()];
            std::wstring targetLabel =
m_devToolsTargetLabelMap[targetId];
            details = L"From " + targetLabel + L" (session:"
+ sessionId.get() +
L")\r\n" + details;
        }
    }
    m_appWindow->AsyncMessageBox(details, L"CDP Event Fired:
" + title);
    return S_OK;
})
.Get(),
&m_devToolsProtocolEventReceivedTokenMap[eventName]));
}
}

```

## remove\_DevToolsProtocolEventReceived

Remove an event handler previously added with `add_DevToolsProtocolEventReceived`.

```
public HRESULT remove_DevToolsProtocolEventReceived(EventRegistrationToken
token)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2DOMContentLoadedEventArgs

Article • 02/26/2024

```
interface ICoreWebView2DOMContentLoadedEventArgs
: public IUnknown
```

Event args for the DOMContentLoaded event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_NavigationId</a>	The ID of the navigation which corresponds to other navigation ID properties on other navigation events.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.705.50
WebView2 Win32 Prerelease	1.0.721

## Members

### [get\\_NavigationId](#)

The ID of the navigation which corresponds to other navigation ID properties on other navigation events.

```
public HRESULT get_NavigationId(UINT64 * navigationId)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2DownloadOperation

Article • 02/26/2024

```
interface ICoreWebView2DownloadOperation
: public IUnknown
```

Represents a download operation.

## Summary

[ ] Expand table

Members	Descriptions
<a href="#">add_BytesReceivedChanged</a>	Add an event handler for the <code>BytesReceivedChanged</code> event.
<a href="#">add_EstimatedEndTimeChanged</a>	Add an event handler for the <code>EstimatedEndTimeChanged</code> event.
<a href="#">add_StateChanged</a>	Add an event handler for the <code>StateChanged</code> event.
<a href="#">Cancel</a>	Cancels the download.
<a href="#">get_BytesReceived</a>	The number of bytes that have been written to the download file.
<a href="#">get_CanResume</a>	Returns true if an interrupted download can be resumed.
<a href="#">get_ContentDisposition</a>	The Content-Disposition header value from the download's HTTP response.
<a href="#">get_EstimatedEndTime</a>	The estimated end time in <a href="#">ISO 8601 Date and Time Format</a> .
<a href="#">get_InterruptReason</a>	The reason why connection with file host was broken.
<a href="#">get_MimeType</a>	MIME type of the downloaded content.
<a href="#">get_ResultFilePath</a>	The absolute path to the download file, including file name.
<a href="#">get_State</a>	The state of the download.

Members	Descriptions
<a href="#">get_TotalBytesToReceive</a>	The expected size of the download in total number of bytes based on the HTTP Content-Length header.
<a href="#">get Uri</a>	The URI of the download.
<a href="#">Pause</a>	Pauses the download.
<a href="#">remove_BytesReceivedChanged</a>	Remove an event handler previously added with <code>add_BytesReceivedChanged</code> .
<a href="#">remove_EstimatedEndTimeChanged</a>	Remove an event handler previously added with <code>add_EstimatedEndTimeChanged</code> .
<a href="#">remove_StateChanged</a>	Remove an event handler previously added with <code>add_StateChanged</code> .
<a href="#">Resume</a>	Resumes a paused download.

Gives access to the download's metadata and supports a user canceling, pausing, or resuming the download.

## Applies to

[\[ \] Expand table](#)

Product	Introduced
WebView2 Win32	1.0.902.49
WebView2 Win32 Prerelease	1.0.902

## Members

### [add\\_BytesReceivedChanged](#)

Add an event handler for the `BytesReceivedChanged` event.

```
public HRESULT
add_BytesReceivedChanged(ICoreWebView2BytesReceivedChangedEventHandler *
eventHandler, EventRegistrationToken * token)
```

```
CHECK_FAILURE(download->add_BytesReceivedChanged(
    Callback<ICoreWebView2BytesReceivedChangedEventHandler>(
        [this](ICoreWebView2DownloadOperation* download, IUnknown* args)
-> HRESULT {
            // Here developer can update UI to show progress of a
download using
            // `download->get_BytesReceived` and
            // `download->get_TotalBytesToReceive`.
            return S_OK;
        }
        .Get(),
        &m_bytesReceivedChangedToken);
```

## add\_EstimatedEndTimeChanged

Add an event handler for the `EstimatedEndTimeChanged` event.

```
public HRESULT
add_EstimatedEndTimeChanged(ICoreWebView2EstimatedEndTimeChangedEventH
andler * eventHandler, EventRegistrationToken * token)
```

## add\_StateChanged

Add an event handler for the `StateChanged` event.

```
public HRESULT add_StateChanged(ICoreWebView2StateChangedEventHandler * 
eventHandler, EventRegistrationToken * token)
```

C++

```
CHECK_FAILURE(download->add_StateChanged(
    Callback<ICoreWebView2StateChangedEventHandler>(
        [this](ICoreWebView2DownloadOperation* download,
        IUnknown* args) -> HRESULT {
            COREWEBVIEW2_DOWNLOAD_STATE downloadState;
            CHECK_FAILURE(download->get_State(&downloadState));
            switch (downloadState)
            {
                case COREWEBVIEW2_DOWNLOAD_STATE_IN_PROGRESS:
                    break;
                case COREWEBVIEW2_DOWNLOAD_STATE_INTERRUPTED:
                    // Here developer can take different actions based on
`download->InterruptReason`.
                    // For example, show an error message to the end user.
                    CompleteDownload(download);
                    break;
                case COREWEBVIEW2_DOWNLOAD_STATE_COMPLETED:
```

```
        CompleteDownload(download);
        break;
    }
    return S_OK;
})
.Get(),
&m_stateChangedToken));
```

## Cancel

Cancels the download.

```
public HRESULT Cancel()
```

If canceled, the default download dialog shows that the download was canceled. Host should set the `Cancel` property from `ICoreWebView2SDownloadStartingEventArgs` if the download should be canceled without displaying the default download dialog.

## get\_BytesReceived

The number of bytes that have been written to the download file.

```
public HRESULT get_BytesReceived(INT64 * bytesReceived)
```

## get\_CanResume

Returns true if an interrupted download can be resumed.

```
public HRESULT get_CanResume(BOOL * canResume)
```

Downloads with the following interrupt reasons may automatically resume without you calling any methods: `COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_SERVER_NO_RANGE`, `COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_FILE_HASH_MISMATCH`, `COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_FILE_TOO_SHORT`. In these cases download progress may be restarted with `BytesReceived` reset to 0.

## get\_ContentDisposition

The Content-Disposition header value from the download's HTTP response.

```
public HRESULT get_ContentDisposition(LPWSTR * contentDisposition)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_EstimatedEndTime

The estimated end time in [ISO 8601 Date and Time Format](#).

```
public HRESULT get_EstimatedEndTime(LPWSTR * estimatedEndTime)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_InterruptReason

The reason why connection with file host was broken.

```
public HRESULT  
get_InterruptReason(COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON *  
interruptReason)
```

## get\_MimeType

MIME type of the downloaded content.

```
public HRESULT get_MimeType(LPWSTR * mimeType)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_ResultFilePath

The absolute path to the download file, including file name.

```
public HRESULT get_ResultFilePath(LPWSTR * resultFilePath)
```

Host can change this from `ICoreWebView2DownloadStartingEventArgs`.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_State

The state of the download.

```
public HRESULT get_State(COREWEBVIEW2_DOWNLOAD_STATE * downloadState)
```

A download can be in progress, interrupted, or completed. See `COREWEBVIEW2_DOWNLOAD_STATE` for descriptions of states.

## **get\_TotalBytesToReceive**

The expected size of the download in total number of bytes based on the HTTP Content-Length header.

```
| public HRESULT get_TotalBytesToReceive(INT64 * totalBytesToReceive)
```

Returns -1 if the size is unknown.

## **get\_Uri**

The URI of the download.

```
| public HRESULT get_Uri(LPWSTR * uri)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## **Pause**

Pauses the download.

```
| public HRESULT Pause()
```

If paused, the default download dialog shows that the download is paused. No effect if download is already paused. Pausing a download changes the state to `COREWEBVIEW2_DOWNLOAD_STATE_INTERRUPTED` with `InterruptReason` set to `COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_USER_PAUSED`.

## **remove\_BytesReceivedChanged**

Remove an event handler previously added with `add_BytesReceivedChanged`.

```
| public HRESULT remove_BytesReceivedChanged(EventRegistrationToken token)
```

## **remove\_EstimatedEndTimeChanged**

Remove an event handler previously added with `add_EstimatedEndTimeChanged`.

```
public HRESULT remove_EstimatedEndTimeChanged(EventRegistrationToken token)
```

## remove\_StateChanged

Remove an event handler previously added with `add_StateChanged`.

```
public HRESULT remove_StateChanged(EventRegistrationToken token)
```

## Resume

Resumes a paused download.

```
public HRESULT Resume()
```

May also resume a download that was interrupted for another reason, if `CanResume` returns true. Resuming a download changes the state from `COREWEBVIEW2_DOWNLOAD_STATE_INTERRUPTED` to `COREWEBVIEW2_DOWNLOAD_STATE_IN_PROGRESS`.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2DownloadStartingEventArgs

Article • 02/26/2024

```
interface ICoreWebView2DownloadStartingEventArgs
: public IUnknown
```

Event args for the `DownloadStarting` event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_Cancel</a>	The host may set this flag to cancel the download.
<a href="#">get_DownloadOperation</a>	Returns the ICoreWebView2DownloadOperation for the download that has started.
<a href="#">get_Handled</a>	The host may set this flag to <code>TRUE</code> to hide the default download dialog for this download.
<a href="#">get_ResultFilePath</a>	The path to the file.
<a href="#">GetDeferral</a>	Returns an ICoreWebView2Deferral object.
<a href="#">put_Cancel</a>	Sets the <code>Cancel</code> property.
<a href="#">put_Handled</a>	Sets the <code>Handled</code> property.
<a href="#">put_ResultFilePath</a>	Sets the <code>ResultFilePath</code> property.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.902.49
WebView2 Win32 Prerelease	1.0.902

## Members

### get\_Cancel

The host may set this flag to cancel the download.

```
public HRESULT get_Cancel(BOOL * cancel)
```

If canceled, the download save dialog is not displayed regardless of the `Handled` property.

### get\_DownloadOperation

Returns the `ICoreWebView2DownloadOperation` for the download that has started.

```
public HRESULT get_DownloadOperation(ICoreWebView2DownloadOperation ** downloadOperation)
```

### get\_Handled

The host may set this flag to `TRUE` to hide the default download dialog for this download.

```
public HRESULT get_Handled(BOOL * handled)
```

The download will progress as normal if it is not canceled, there will just be no default UI shown. By default the value is `FALSE` and the default download dialog is shown.

### get\_ResultFilePath

The path to the file.

```
public HRESULT get_ResultFilePath(LPWSTR * resultFilePath)
```

If setting the path, the host should ensure that it is an absolute path, including the file name, and that the path does not point to an existing file. If the path points to an existing file, the file will be overwritten. If the directory does not exist, it is created.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## GetDeferral

Returns an `ICoreWebView2Deferral` object.

```
public HRESULT GetDeferral(ICoreWebView2Deferral ** deferral)
```

Use this operation to complete the event at a later time.

## put\_Cancel

Sets the `Cancel` property.

```
public HRESULT put_Cancel(BOOL cancel)
```

## put\_Handled

Sets the `Handled` property.

```
public HRESULT put_Handled(BOOL handled)
```

## put\_ResultFilePath

Sets the `ResultFilePath` property.

```
public HRESULT put_ResultFilePath(LPCWSTR resultFilePath)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Environment

Article • 02/26/2024

```
interface ICoreWebView2Environment
: public IUnknown
```

Represents the WebView2 Environment.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">add_NewBrowserVersionAvailable</a>	Add an event handler for the <code>NewBrowserVersionAvailable</code> event.
<a href="#">CreateCoreWebView2Controller</a>	Asynchronously create a new WebView.
<a href="#">CreateWebResourceResponse</a>	Create a new web resource response object.
<a href="#">get_BrowserVersionString</a>	The browser version info of the current ICoreWebView2Environment, including channel name if it is not the WebView2 Runtime.
<a href="#">remove_NewBrowserVersionAvailable</a>	Remove an event handler previously added with <code>add_NewBrowserVersionAvailable</code> .

WebViews created from an environment run on the browser process specified with environment parameters and objects created from an environment should be used in the same environment. Using it in different environments are not guaranteed to be compatible and may fail.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430

Product	Introduced
WebView2 Win32 Prerelease	0.9.488

# Members

## add\_NewBrowserVersionAvailable

Add an event handler for the `NewBrowserVersionAvailable` event.

```
public HRESULT
add_NewBrowserVersionAvailable(ICoreWebView2NewBrowserVersionAvailableEven
tHandler * eventHandler, EventRegistrationToken * token)
```

`NewBrowserVersionAvailable` runs when a newer version of the WebView2 Runtime is installed and available using WebView2. To use the newer version of the browser you must create a new environment and WebView. The event only runs for new version from the same WebView2 Runtime from which the code is running. When not running with installed WebView2 Runtime, no event is run.

Because a user data folder is only able to be used by one browser process at a time, if you want to use the same user data folder in the WebView using the new version of the browser, you must close the environment and instance of WebView that are using the older version of the browser first. Or simply prompt the user to restart the app.

C++

```
// After the environment is successfully created,
// register a handler for the NewBrowserVersionAvailable event.
// This handler tells when there is a new Edge version available on the
machine.

CHECK_FAILURE(m_webViewEnvironment->add_NewBrowserVersionAvailable(
    Callback<ICoreWebView2NewBrowserVersionAvailableEventHandler>(
        [this](ICoreWebView2Environment* sender, IUnknown* args) ->
HRESULT
{
    // Don't block the event handler with a message box
    RunAsync(
        [this]
    {
        std::wstring message =
            L"We detected there is a new version for the
browser.";
        if (m_webView)
        {

```

```

        message += L"Do you want to restart the app?
\n\n";
        message +=
            L"Click No if you only want to re-create the
webviews. \n";
        message += L"Click Cancel for no action. \n";
    }
    int response = MessageBox(
        m_mainWindow, message.c_str(), L"New available
version",
        m_webView ? MB_YESNOCANCEL : MB_OK);

    if (response == IDYES)
    {
        RestartApp();
    }
    else if (response == IDNO)
    {
        ReinitializeWebViewWithNewBrowser();
    }
    else
    {
        // do nothing
    }
});

return S_OK;
}
.Get(),
nullptr));

```

## CreateCoreWebView2Controller

Asynchronously create a new WebView.

```
public HRESULT CreateCoreWebView2Controller(HWND parentWindow,
ICoreWebView2CreateCoreWebView2ControllerCompletedHandler * handler)
```

`parentWindow` is the `HWND` in which the WebView should be displayed and from which receive input. The WebView adds a child window to the provided window before this function returns. Z-order and other things impacted by sibling window order are affected accordingly. If you want to move the WebView to a different parent after it has been created, you must call `put_ParentWindow` to update tooltip positions, accessibility trees, and such.

`HWND_MESSAGE` is a valid parameter for `parentWindow` for an invisible WebView for Windows 8 and above. In this case the window will never become visible. You are not able to reparent the window after you have created the WebView. This is not supported

in Windows 7 or below. Passing this parameter in Windows 7 or below will return ERROR\_INVALID\_WINDOW\_HANDLE in the controller callback.

It is recommended that the app set Application User Model ID for the process or the app window. If none is set, during WebView creation a generated Application User Model ID is set to root window of `parentWindow`.

C++

```
// Create or recreate the WebView and its environment.
void AppWindow::InitializeWebView()
{
    // To ensure browser switches get applied correctly, we need to close
    // the existing WebView. This will result in a new browser process
    // getting created which will apply the browser switches.
    CloseWebView();
    m_dcompDevice = nullptr;
    m_wincompCompositor = nullptr;
    LPCWSTR subFolder = nullptr;

    if (m_creationModeId == IDM_CREATION_MODE_VISUAL_DCOMP ||
        m_creationModeId == IDM_CREATION_MODE_TARGET_DCOMP)
    {
        HRESULT hr = DCompositionCreateDevice2(nullptr,
        IID_PPV_ARGS(&m_dcompDevice));
        if (!SUCCEEDED(hr))
        {
            MessageBox(
                m_mainWindow,
                L"Attempting to create WebView using DComp Visual is not
supported.\r\n"
                "DComp device creation failed.\r\n"
                "Current OS may not support DComp.",
                L"Create with Windowless DComp Visual Failed", MB_OK);
            return;
        }
    }
    else if (m_creationModeId == IDM_CREATION_MODE_VISUAL_WINCOMP)
    {
        HRESULT hr = TryCreateDispatcherQueue();
        if (!SUCCEEDED(hr))
        {
            MessageBox(
                m_mainWindow,
                L"Attempting to create WebView using WinComp Visual is not
supported.\r\n"
                "WinComp compositor creation failed.\r\n"
                "Current OS may not support WinComp.",
                L"Create with Windowless WinComp Visual Failed", MB_OK);
            return;
        }
        m_wincompCompositor = winrtComp::Compositor();
    }
}
```

```

    std::wstring args;
    args.append(L"--enable-
features=ThirdPartyStoragePartitioning,PartitionedCookies");
    auto options = Microsoft::WRL::Make<CoreWebView2EnvironmentOptions>();
    options->put_AdditionalBrowserArguments(args.c_str());
    CHECK_FAILURE(
        options->put_AllowSingleSignOnUsingOSPrimaryAccount(m_AADSSOEnabled
? TRUE : FALSE));
    CHECK_FAILURE(options->put_ExclusiveUserDataFolderAccess(
        m_ExclusiveUserDataFolderAccess ? TRUE : FALSE));
    if (!m_language.empty())
        CHECK_FAILURE(options->put_Language(m_language.c_str()));
    CHECK_FAILURE(options->put_IsCustomCrashReportingEnabled(
        m_CustomCrashReportingEnabled ? TRUE : FALSE));

    Microsoft::WRL::ComPtr<ICoreWebView2EnvironmentOptions4> options4;
    if (options.As(&options4) == S_OK)
    {
        const WCHAR* allowedOrigins[1] = {L"https://*.example.com"};

        auto customSchemeRegistration =
            Microsoft::WRL::Make<CoreWebView2CustomSchemeRegistration>
(L"custom-scheme");
        customSchemeRegistration->SetAllowedOrigins(1, allowedOrigins);
        auto customSchemeRegistration2 =
            Microsoft::WRL::Make<CoreWebView2CustomSchemeRegistration>
(L"wv2rocks");
        customSchemeRegistration2->put_TreatAsSecure(TRUE);
        customSchemeRegistration2->SetAllowedOrigins(1, allowedOrigins);
        customSchemeRegistration2->put_HasAuthorityComponent(TRUE);
        auto customSchemeRegistration3 =
            Microsoft::WRL::Make<CoreWebView2CustomSchemeRegistration>(
                L"custom-scheme-not-in-allowed-origins");
        ICoreWebView2CustomSchemeRegistration* registrations[3] = {
            customSchemeRegistration.Get(), customSchemeRegistration2.Get(),
            customSchemeRegistration3.Get()};
        options4->SetCustomSchemeRegistrations(
            2, static_cast<ICoreWebView2CustomSchemeRegistration**>
(registrations));
    }

    Microsoft::WRL::ComPtr<ICoreWebView2EnvironmentOptions5> options5;
    if (options.As(&options5) == S_OK)
    {
        CHECK_FAILURE(
            options5-
>put_EnableTrackingPrevention(m_TrackingPreventionEnabled ? TRUE : FALSE));
    }

    Microsoft::WRL::ComPtr<ICoreWebView2EnvironmentOptions6> options6;
    if (options.As(&options6) == S_OK)
    {
        CHECK_FAILURE(options6->put_AreBrowserExtensionsEnabled(TRUE));
    }

```

```

HRESULT hr = CreateCoreWebView2EnvironmentWithOptions(
    subFolder, m_userDataFolder.c_str(), options.Get(),
    Callback<ICoreWebView2CreateCoreWebView2EnvironmentCompletedHandler>
)
{
    this, &AppWindow::OnCreateEnvironmentCompleted)
    .Get());
if (!SUCCEEDED(hr))
{
    switch (hr)
    {
        case HRESULT_FROM_WIN32(ERROR_FILE_NOT_FOUND):
        {
            MessageBox(
                m_mainWindow,
                L"Couldn't find Edge WebView2 Runtime. "
                "Do you have a version installed?",
                nullptr, MB_OK);
        }
        break;
        case HRESULT_FROM_WIN32(ERROR_FILE_EXISTS):
        {
            MessageBox(
                m_mainWindow,
                L"User data folder cannot be created because a file with the
same name already "
                L"exists.",
                nullptr, MB_OK);
        }
        break;
        case E_ACCESSDENIED:
        {
            MessageBox(
                m_mainWindow, L"Unable to create user data folder, Access
Denied.", nullptr,
                MB_OK);
        }
        break;
        case E_FAIL:
        {
            MessageBox(m_mainWindow, L"Edge runtime unable to start",
nullptr, MB_OK);
        }
        break;
        default:
        {
            ShowFailure(hr, L"Failed to create WebView2 environment");
        }
    }
}
}

// This is the callback passed to CreateWebViewEnvironmentWithOptions.
// Here we simply create the WebView.

HRESULT AppWindow::OnCreateEnvironmentCompleted(
    HRESULT result, ICoreWebView2Environment* environment)

```

```

{
    if (result != S_OK)
    {
        ShowFailure(result, L"Failed to create environment object.");
        return S_OK;
    }
    m_webViewEnvironment = environment;

    if (m_webviewOption.entry ==
WebViewCreateEntry::EVER_FROM_CREATE_WITH_OPTION_MENU
)
{
    return CreateControllerWithOptions();
}
auto webViewEnvironment3 =
m_webViewEnvironment.try_query<ICoreWebView2Environment3>();

if (webViewEnvironment3 && (m_dcompDevice || m_wincompCompositor))
{
    CHECK_FAILURE(webViewEnvironment3-
>CreateCoreWebView2CompositionController(
    m_mainWindow,

Callback<ICoreWebView2CreateCoreWebView2CompositionControllerCompletedHandler>(
    [this](
        HRESULT result,
        ICoreWebView2CompositionController*
compositionController) -> HRESULT
    {
        auto controller =
            wil::com_ptr<ICoreWebView2CompositionController>
(compositionController)
                .query<ICoreWebView2Controller>();
        return OnCreateCoreWebView2ControllerCompleted(result,
controller.get());
    }
    .Get()));
}
else
{
    CHECK_FAILURE(m_webViewEnvironment->CreateCoreWebView2Controller(
        m_mainWindow,
Callback<ICoreWebView2CreateCoreWebView2ControllerCompletedHandler>(
    this,
&AppWindow::OnCreateCoreWebView2ControllerCompleted)
    .Get());
}

return S_OK;
}

```

It is recommended that the app handles restart manager messages, to gracefully restart it in the case when the app is using the WebView2 Runtime from a certain installation and that installation is being uninstalled. For example, if a user installs a version of the WebView2 Runtime and opts to use another version of the WebView2 Runtime for testing the app, and then uninstalls the 1st version of the WebView2 Runtime without closing the app, the app restarts to allow un-installation to succeed.

C++

```
case WM_QUERYENDSESSION:  
{  
    // yes, we can shut down  
    // Register how we might be restarted  
    RegisterApplicationRestart(L"--restore", RESTART_NO_CRASH |  
RESTART_NO_HANG);  
    *result = TRUE;  
    return true;  
}  
break;  
case WM_ENDSESSION:  
{  
    if (wParam == TRUE)  
    {  
        // save app state and exit.  
        PostQuitMessage(0);  
        return true;  
    }  
}  
break;
```

The app should retry `CreateCoreWebView2Controller` upon failure, unless the error code is `HRESULT_FROM_WIN32(ERROR_INVALID_STATE)`. When the app retries `CreateCoreWebView2Controller` upon failure, it is recommended that the app restarts from creating a new WebView2 Environment. If a WebView2 Runtime update happens, the version associated with a WebView2 Environment may have been removed and causing the object to no longer work. Creating a new WebView2 Environment works since it uses the latest version.

WebView creation fails with `HRESULT_FROM_WIN32(ERROR_INVALID_STATE)` if a running instance using the same user data folder exists, and the Environment objects have different `EnvironmentOptions`. For example, if a WebView was created with one language, an attempt to create a WebView with a different language using the same user data folder will fail.

The creation will fail with `E_ABORT` if `parentWindow` is destroyed before the creation is finished. If this is caused by a call to `DestroyWindow`, the creation completed handler will

be invoked before `DestroyWindow` returns, so you can use this to cancel creation and clean up resources synchronously when quitting a thread.

In rare cases the creation can fail with `E_UNEXPECTED` if runtime does not have permissions to the user data folder.

## CreateWebResourceResponse

Create a new web resource response object.

```
public HRESULT CreateWebResourceResponse(IStream * content, int statusCode,
LPCWSTR reasonPhrase, LPCWSTR headers, ICoreWebView2WebResourceResponse
** response)
```

The `headers` parameter is the raw response header string delimited by newline. It is also possible to create this object with null headers string and then use the [ICoreWebView2HttpResponseHeaders](#) to construct the headers line by line. For more information about other parameters, navigate to [ICoreWebView2WebResourceResponse](#).

C++

```
if (m_blockImages)
{
    m_webView->AddWebResourceRequestedFilter(
        L"*", COREWEBVIEW2_WEB_RESOURCE_CONTEXT_IMAGE);
    CHECK_FAILURE(m_webView->add_WebResourceRequested(
        Callback<ICoreWebView2WebResourceRequestedEventHandler>(
            [this](
                ICoreWebView2* sender,
                ICoreWebView2WebResourceRequestedEventArgs* args)
        {
            COREWEBVIEW2_WEB_RESOURCE_CONTEXT resourceContext;
            CHECK_FAILURE(args-
>get_ResourceContext(&resourceContext));
            // Ensure that the type is image
            if (resourceContext !=
COREWEBVIEW2_WEB_RESOURCE_CONTEXT_IMAGE)
            {
                return E_INVALIDARG;
            }
            // Override the response with an empty one to block
            the image.
            // If put_Response is not called, the request will
            // continue as normal.
            wil::com_ptr<ICoreWebView2WebResourceResponse>
response;
            wil::com_ptr<ICoreWebView2Environment> environment;
```

```

        wil::com_ptr<ICoreWebView2_2> webview2;
        CHECK_FAILURE(m_webView-
>QueryInterface(IID_PPV_ARGS(&webview2)));
        CHECK_FAILURE(webview2-
>get_Environment(&environment));
        CHECK_FAILURE(environment-
>CreateWebResourceResponse(
            nullptr, 403 /*NoContent*/, L"Blocked",
L"Content-Type: image/jpeg",
            &response));
        CHECK_FAILURE(args->put_Response(response.get()));
        return S_OK;
    })
    .Get(),
    &m_webResourceRequestedTokenForImageBlocking));
}
else
{
    CHECK_FAILURE(m_webView->remove_WebResourceRequested(
        m_webResourceRequestedTokenForImageBlocking));
}

```

C++

```

if (m_replaceImages)
{
    m_webView->AddWebResourceRequestedFilter(
        L"*", COREWEBVIEW2_WEB_RESOURCE_CONTEXT_IMAGE);
    CHECK_FAILURE(m_webView->add_WebResourceRequested(
        Callback<ICoreWebView2WebResourceRequestedEventHandler>(
            [this](
                ICoreWebView2* sender,
                ICoreWebView2WebResourceRequestedEventArgs* args)
        {
            COREWEBVIEW2_WEB_RESOURCE_CONTEXT resourceContext;
            CHECK_FAILURE(args-
>get_ResourceContext(&resourceContext));
            // Ensure that the type is image
            if (resourceContext !=
COREWEBVIEW2_WEB_RESOURCE_CONTEXT_IMAGE)
            {
                return E_INVALIDARG;
            }
            // Override the response with an another image.
            // If put_Response is not called, the request will
            // continue as normal.
            // It's not required for this scenario, but
generally you should examine
                // relevant HTTP request headers just like an HTTP
server would do when
                // producing a response stream.
                wil::com_ptr<IStream> stream;
                CHECK_FAILURE(SHCreateStreamOnFileEx(

```

```

L"assets/EdgeWebView2-80.jpg", STGM_READ,
FILE_ATTRIBUTE_NORMAL,
    FALSE, nullptr, &stream));
wil::com_ptr<ICoreWebView2WebResourceResponse>
response;
    wil::com_ptr<ICoreWebView2Environment> environment;
    wil::com_ptr<ICoreWebView2_2> webview2;
    CHECK_FAILURE(m_webView-
>QueryInterface(IID_PPV_ARGS(&webview2)));
    CHECK_FAILURE(webview2-
>get_Environment(&environment));
    CHECK_FAILURE(environment-
>CreateWebResourceResponse(
        stream.get(), 200, L"OK", L"Content-Type:
image/jpeg", &response));
    CHECK_FAILURE(args->put_Response(response.get()));
    return S_OK;
})
.Get(),
&m_webResourceRequestedTokenForImageReplacing));
}
else
{
    CHECK_FAILURE(m_webView->remove_WebResourceRequested(
        m_webResourceRequestedTokenForImageReplacing));
}

```

## get\_BrowserVersionString

The browser version info of the current ICoreWebView2Environment, including channel name if it is not the WebView2 Runtime.

```
public HRESULT get_BrowserVersionString(LPWSTR * versionInfo)
```

It matches the format of the `GetAvailableCoreWebView2BrowserVersionString` API.  
 Channel names are `beta`, `dev`, and `canary`.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

C++

```

wil::unique_cotaskmem_string version_info;
m_webViewEnvironment->get_BrowserVersionString(&version_info);
MessageBox(
    m_mainWindow, version_info.get(), L"Browser Version Info After
WebView Creation",
    MB_OK);

```

## **remove\_NewBrowserVersionAvailable**

Remove an event handler previously added with `add_NewBrowserVersionAvailable`.

```
public HRESULT remove_NewBrowserVersionAvailable(EventRegistrationToken  
token)
```

---

## **Feedback**

Was this page helpful?



# interface ICoreWebView2Environment10

Article • 02/26/2024

```
interface ICoreWebView2Environment10
: public ICoreWebView2Environment9
```

This interface is used to create [ICoreWebView2ControllerOptions](#) object, which can be passed as a parameter in [CreateCoreWebView2ControllerWithOptions](#) and

[CreateCoreWebView2CompositionControllerWithOptions](#) function for multiple profiles support.

## Summary

[\[+\] Expand table](#)

Members	Descriptions
<a href="#">CreateCoreWebView2CompositionControllerWithOptions</a>	Create a new WebView in visual hosting mode with options.
<a href="#">CreateCoreWebView2ControllerOptions</a>	Create a new <a href="#">ICoreWebView2ControllerOptions</a> to be passed as a parameter of <a href="#">CreateCoreWebView2ControllerWithOptions</a> and <a href="#">CreateCoreWebView2CompositionControllerWithOptions</a> .
<a href="#">CreateCoreWebView2ControllerWithOptions</a>	Create a new WebView with options.

The profile will be created on disk or opened when calling

[CreateCoreWebView2ControllerWithOptions](#) or

[CreateCoreWebView2CompositionControllerWithOptions](#) no matter InPrivate mode is enabled or not, and it will be released in memory when the corresponding controller is closed but still remain on disk. If you create a WebView2Controller with {ProfileName="name", InPrivate=false} and then later create another one with {ProfileName="name", InPrivate=true}, these two controllers using the same profile would be allowed to run at the same time. As WebView2 is built on top of Edge browser, it follows Edge's behavior pattern. To create an InPrivate WebView, we gets an off-the-record profile (an InPrivate profile) from a regular profile, then create the WebView with the off-the-record profile.

C++

```
auto webViewEnvironment10 =
m_webViewEnvironment.try_query<ICoreWebView2Environment10>();
if (!webViewEnvironment10)
{
    FeatureNotAvailable();
    return S_OK;
}
```

```

wil::com_ptr<ICoreWebView2ControllerOptions> options;
// The validation of parameters occurs when setting the properties.
HRESULT hr = webViewEnvironment10->CreateCoreWebView2ControllerOptions(&options);
if (hr == E_INVALIDARG)
{
    ShowFailure(hr, L"Unable to create WebView2 due to an invalid profile
name.");
    CloseAppWindow();
    return S_OK;
}
CHECK_FAILURE(hr);

```

## Applies to

[Expand table](#)

Product	Introduced
WebView2 Win32	1.0.1210.39
WebView2 Win32 Prerelease	1.0.1222

## Members

### CreateCoreWebView2CompositionControllerWithOptions

Create a new WebView in visual hosting mode with options.

```

public HRESULT CreateCoreWebView2CompositionControllerWithOptions(HWND
parentWindow, ICoreWebView2ControllerOptions * options,
ICoreWebView2CreateCoreWebView2CompositionControllerCompletedHandler * handler)

```

### CreateCoreWebView2ControllerOptions

Create a new [ICoreWebView2ControllerOptions](#) to be passed as a parameter of [CreateCoreWebView2ControllerWithOptions](#) and [CreateCoreWebView2CompositionControllerWithOptions](#).

```

public HRESULT CreateCoreWebView2ControllerOptions(ICoreWebView2ControllerOptions ** options)

```

The 'options' is settable and in it the default value for profile name is the empty string, and the default value for IsInPrivateModeEnabled is false. Also the profile name can be reused.

### CreateCoreWebView2ControllerWithOptions

Create a new WebView with options.

```
public HRESULT CreateCoreWebView2ControllerWithOptions(HWND parentWindow,  
ICoreWebView2ControllerOptions * options,  
ICoreWebView2CreateCoreWebView2ControllerCompletedHandler * handler)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Environment11

Article • 02/26/2024

```
interface ICoreWebView2Environment11
: public ICoreWebView2Environment10
```

A continuation of the [ICoreWebView2Environment](#) interface for getting the crash dump folder path.

## Summary

[+] [Expand table](#)

Members	Descriptions
<a href="#">get_FailureReportFolderPath</a>	<code>FailureReportFolderPath</code> returns the path of the folder where minidump files are written.

## Applies to

[+] [Expand table](#)

Product	Introduced
WebView2 Win32	1.0.1518.46
WebView2 Win32 Prerelease	1.0.1549

## Members

### [get\\_FailureReportFolderPath](#)

`FailureReportFolderPath` returns the path of the folder where minidump files are written.

```
public HRESULT get_FailureReportFolderPath(LPWSTR * value)
```

Whenever a WebView2 process crashes, a crash dump file will be created in the crash dump folder. The crash dump format is minidump files. Please see [Minidump Files documentation](#) for detailed information. Normally when a single child process fails, a minidump will be generated and written to disk, then the `ProcessFailed` event is raised. But for unexpected crashes, a minidump file might not be generated at all, despite whether `ProcessFailed` event is raised. If there are multiple process failures at once, multiple minidump files could be generated. Thus `FailureReportFolderPath` could contain old minidump files that are not associated with a specific `ProcessFailed` event.

C++

```
auto environment11 =
m_webViewEnvironment.try_query<ICoreWebView2Environment11>();
CHECK FEATURE_RETURN(environment11);
wil::unique_cotaskmem_string failureReportFolder;
environment11->get_FailureReportFolderPath(&failureReportFolder);
MessageBox(m_mainWindow, failureReportFolder.get(), L"Failure Report
Folder", MB_OK);
```

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Environment12

Article • 02/26/2024

```
interface ICoreWebView2Environment12
: public ICoreWebView2Environment11
```

A continuation of the [ICoreWebView2Environment](#) interface for creating shared buffer object.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">CreateSharedBuffer</a>	Create a shared memory based buffer with the specified size in bytes.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1661.34
WebView2 Win32 Prerelease	1.0.1671

## Members

### [CreateSharedBuffer](#)

Create a shared memory based buffer with the specified size in bytes.

```
public HRESULT CreateSharedBuffer(UINT64 size, ICoreWebView2SharedBuffer **  
shared_buffer)
```

The buffer can be shared with web contents in WebView by calling `PostSharedBufferToScript` on `CoreWebView2` or `CoreWebView2Frame` object. Once shared, the same content of the buffer will be accessible from both the app process and script in WebView. Modification to the content will be visible to all parties that have access to the buffer. The shared buffer is presented to the script as ArrayBuffer. All JavaScript APIs that work for ArrayBuffer including Atomics APIs can be used on it. There is currently a limitation that only size less than 2GB is supported.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Environment13

Article • 02/26/2024

```
interface ICoreWebView2Environment13
: public ICoreWebView2Environment12
```

A continuation of the [ICoreWebView2Environment](#) interface for getting process with associated information.

## Summary

[+] [Expand table](#)

Members	Descriptions
<a href="#">GetProcessExtendedInfos</a>	Gets a snapshot collection of <code>ProcessExtendedInfo</code> s corresponding to all currently running processes associated with this <code>CoreWebView2Environment</code> . Excludes crashpad process.

## Applies to

[+] [Expand table](#)

Product	Introduced
WebView2 Win32	1.0.2210.55
WebView2 Win32 Prerelease	1.0.2357

## Members

### [GetProcessExtendedInfos](#)

Gets a snapshot collection of `ProcessExtendedInfo`s corresponding to all currently running processes associated with this `CoreWebView2Environment`. Excludes crashpad process.

```
public HRESULT  
GetProcessExtendedInfos(ICoreWebView2GetProcessExtendedInfosCompletedHandler * handler)
```

This provides the same list of `ProcessInfo`s as what's provided in `GetProcessInfos`, but additionally provides a list of associated `FrameInfo`s which are actively running (showing or hiding UI elements) in the renderer process. See `AssociatedFrameInfos` for more information.

C++

```
CHECK_FAILURE(environment13->GetProcessExtendedInfos(  
    Callback<ICoreWebView2GetProcessExtendedInfosCompletedHandler>(  
        [this](  
            HRESULT error,  
            ICoreWebView2ProcessExtendedInfoCollection*  
processCollection) -> HRESULT  
    {  
        UINT32 processCount = 0;  
        UINT32 rendererProcessCount = 0;  
        CHECK_FAILURE(processCollection-  
>get_Count(&processCount));  
        std::wstringstream otherProcessInfos;  
        std::wstringstream rendererProcessInfos;  
        for (UINT32 i = 0; i < processCount; i++)  
        {  
  
Microsoft::WRL::ComPtr<ICoreWebView2ProcessExtendedInfo>  
            processExtendedInfo;  
            CHECK_FAILURE(  
                processCollection->GetValueAtIndex(i,  
&processExtendedInfo));  
            Microsoft::WRL::ComPtr<ICoreWebView2ProcessInfo>  
processInfo;  
            CHECK_FAILURE(processExtendedInfo-  
>get_ProcessInfo(&processInfo));  
            COREWEBVIEW2_PROCESS_KIND kind;  
            CHECK_FAILURE(processInfo->get_Kind(&kind));  
            INT32 processId = 0;  
            CHECK_FAILURE(processInfo-  
>get_ProcessId(&processId));  
            if (kind == COREWEBVIEW2_PROCESS_KIND_RENDERER)  
            {  
                std::wstringstream rendererProcess;  
                wil::com_ptr<ICoreWebView2FrameInfoCollection>  
frameInfoCollection;  
                CHECK_FAILURE(processExtendedInfo-  
>get_AssociatedFrameInfos(  
                    &frameInfoCollection));  
  
wil::com_ptr<ICoreWebView2FrameInfoCollectionIterator> iterator;
```

```

        CHECK_FAILURE(frameInfoCollection-
>GetIterator(&iterator));
        BOOL hasCurrent = FALSE;
        UINT32 frameInfoCount = 0;
        while (SUCCEEDED(iterator-
>get_HasCurrent(&hasCurrent)) &&
               hasCurrent)
    {
        wil::com_ptr<ICoreWebView2FrameInfo>
frameInfo;
        CHECK_FAILURE(iterator-
>GetCurrent(&frameInfo));

        AppendFrameInfo(frameInfo, rendererProcess);

        BOOL hasNext = FALSE;
        CHECK_FAILURE(iterator->MoveNext(&hasNext));
        frameInfoCount++;
    }
    rendererProcessInfos
        << frameInfoCount
        << L" frameInfo(s) found in Renderer Process
ID:" << processId
        << L"\n"
        << rendererProcess.str() << std::endl;
    rendererProcessCount++;
}
else
{
    otherProcessInfos << L"Process Id:" << processId
        << L" | Process Kind:"
        << ProcessKindToString(kind)
<< std::endl;
}
}
std::wstringstream message;
message << processCount << L" process(es) found, from
which "
        << rendererProcessCount << L" renderer
process(es) found\n\n"
        << rendererProcessInfos.str() << L"Remaining
Process(es) Infos:\n"
        << otherProcessInfos.str();

m_appWindow->AsyncMessageBox(
    std::move(message.str()), L"Process Extended Info");
return S_OK;
}
.Get()));

```

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Environment2

Article • 02/26/2024

```
interface ICoreWebView2Environment2
: public ICoreWebView2Environment
```

A continuation of the [ICoreWebView2Environment](#) interface.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">CreateWebResourceRequest</a>	Create a new web resource request object.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.705.50
WebView2 Win32 Prerelease	1.0.721

## Members

### [CreateWebResourceRequest](#)

Create a new web resource request object.

```
public HRESULT CreateWebResourceRequest(LPCWSTR uri, LPCWSTR method,
IStream * postData, LPCWSTR headers, ICoreWebView2WebResourceRequest **  
request)
```

URI parameter must be absolute URI. The headers string is the raw request header string delimited by CRLF (optional in last header). It's also possible to create this object with null headers string and then use the [ICoreWebView2HttpRequestHeaders](#) to construct the headers line by line. For information on other parameters see [ICoreWebView2WebResourceRequest](#).

C++

```
wil::com_ptr<ICoreWebView2Environment2> webviewEnvironment2;
CHECK_FAILURE(appWindow->GetWebViewEnvironment()->QueryInterface(
    IID_PPV_ARGS(&webviewEnvironment2)));
wil::com_ptr<ICoreWebView2WebResourceRequest> webResourceRequest;
wil::com_ptr<IStream> postDataStream = SHCreateMemStream(
    reinterpret_cast<const BYTE*>(postDataBytes.get()),
sizeNeededForMultiByte);

// This is acts as a form submit to
https://www.w3schools.com/action\_page.php
CHECK_FAILURE(webviewEnvironment2->CreateWebResourceRequest(
    L"https://www.w3schools.com/action\_page.php", L"POST",
postDataStream.get(),
    L"Content-Type: application/x-www-form-urlencoded",
&webResourceRequest));
wil::com_ptr<ICoreWebView2_2> webview2;
CHECK_FAILURE(m_appWindow->GetWebView()-
>QueryInterface(IID_PPV_ARGS(&webview2)));
CHECK_FAILURE(webview2-
>NavigateWithWebResourceRequest(webResourceRequest.get()));
```

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Environment3

Article • 02/26/2024

```
interface ICoreWebView2Environment3
: public ICoreWebView2Environment2
```

A continuation of the [ICoreWebView2Environment2](#) interface.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">CreateCoreWebView2CompositionController</a>	Asynchronously create a new WebView for use with visual hosting.
<a href="#">CreateCoreWebView2PointerInfo</a>	Create an empty <a href="#">ICoreWebView2PointerInfo</a> .

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.774.44
WebView2 Win32 Prerelease	1.0.790

## Members

### [CreateCoreWebView2CompositionController](#)

Asynchronously create a new WebView for use with visual hosting.

```
public HRESULT CreateCoreWebView2CompositionController(HWND
parentWindow,
```

```
ICoreWebView2CreateCoreWebView2CompositionControllerCompletedHandler *  
handler)
```

parentWindow is the HWND in which the app will connect the visual tree of the WebView. This will be the HWND that the app will receive pointer/ mouse input meant for the WebView (and will need to use SendMouseInput/ SendPointerInput to forward). If the app moves the WebView visual tree to underneath a different window, then it needs to call put\_ParentWindow to update the new parent HWND of the visual tree.

HWND\_MESSAGE is not a valid parameter for parentWindow for visual hosting. The underlying implementation of supporting HWND\_MESSAGE would break accessibility for visual hosting. This is supported in windowed WebViews - see CreateCoreWebView2Controller.

Use put\_RootVisualTarget on the created CoreWebView2CompositionController to provide a visual to host the browser's visual tree.

It is recommended that the application set Application User Model ID for the process or the application window. If none is set, during WebView creation a generated Application User Model ID is set to root window of parentWindow.

C++

```
// Create or recreate the WebView and its environment.  
void AppWindow::InitializeWebView()  
{  
    // To ensure browser switches get applied correctly, we need to close  
    // the existing WebView. This will result in a new browser process  
    // getting created which will apply the browser switches.  
    CloseWebView();  
    m_dcompDevice = nullptr;  
    m_wincompCompositor = nullptr;  
    LPCWSTR subFolder = nullptr;  
  
    if (m_creationModeId == IDM_CREATION_MODE_VISUAL_DCOMP ||  
        m_creationModeId == IDM_CREATION_MODE_TARGET_DCOMP)  
    {  
        HRESULT hr = DCompositionCreateDevice2(nullptr,  
        IID_PPV_ARGS(&m_dcompDevice));  
        if (!SUCCEEDED(hr))  
        {  
            MessageBox(  
                m_mainWindow,  
                L"Attempting to create WebView using DComp Visual is not  
supported.\r\n"  
                "DComp device creation failed.\r\n"  
                "Current OS may not support DComp.",  
                L>Create with Windowless DComp Visual Failed", MB_OK);  
    }
```

```

        return;
    }
}

else if (m_creationModeId == IDM_CREATION_MODE_VISUAL_WINCOMP)
{
    HRESULT hr = TryCreateDispatcherQueue();
    if (!SUCCEEDED(hr))
    {
        MessageBox(
            m_mainWindow,
            L"Attempting to create WebView using WinComp Visual is not
supported.\r\n"
            "WinComp compositor creation failed.\r\n"
            "Current OS may not support WinComp.",
            L"Create with Windowless WinComp Visual Failed", MB_OK);
        return;
    }
    m_wincompCompositor = winrtComp::Compositor();
}

std::wstring args;
args.append(L"--enable-
features=ThirdPartyStoragePartitioning,PartitionedCookies");
auto options = Microsoft::WRL::Make<CoreWebView2EnvironmentOptions>();
options->put_AdditionalBrowserArguments(args.c_str());
CHECK_FAILURE(
    options->put_AllowSingleSignOnUsingOSPrimaryAccount(m_AADSSOEnabled
? TRUE : FALSE));
CHECK_FAILURE(options->put_ExclusiveUserDataFolderAccess(
    m_ExclusiveUserDataFolderAccess ? TRUE : FALSE));
if (!m_language.empty())
    CHECK_FAILURE(options->put_Language(m_language.c_str()));
CHECK_FAILURE(options->put_IsCustomCrashReportingEnabled(
    m_CustomCrashReportingEnabled ? TRUE : FALSE));

Microsoft::WRL::ComPtr<ICoreWebView2EnvironmentOptions4> options4;
if (options.As(&options4) == S_OK)
{
    const WCHAR* allowedOrigins[1] = {L"https://*.example.com"};

    auto customSchemeRegistration =
        Microsoft::WRL::Make<CoreWebView2CustomSchemeRegistration>
(L"custom-scheme");
    customSchemeRegistration->SetAllowedOrigins(1, allowedOrigins);
    auto customSchemeRegistration2 =
        Microsoft::WRL::Make<CoreWebView2CustomSchemeRegistration>
(L"wv2rocks");
    customSchemeRegistration2->put_TreatAsSecure(TRUE);
    customSchemeRegistration2->SetAllowedOrigins(1, allowedOrigins);
    customSchemeRegistration2->put_HasAuthorityComponent(TRUE);
    auto customSchemeRegistration3 =
        Microsoft::WRL::Make<CoreWebView2CustomSchemeRegistration>(
            L"custom-scheme-not-in-allowed-origins");
    ICoreWebView2CustomSchemeRegistration* registrations[3] = {
        customSchemeRegistration.Get(), customSchemeRegistration2.Get(),

```

```

        customSchemeRegistration3.Get());
options4->SetCustomSchemeRegistrations(
    2, static_cast<ICoreWebView2CustomSchemeRegistration**>
(registrations));
}

Microsoft::WRL::ComPtr<ICoreWebView2EnvironmentOptions5> options5;
if (options.As(&options5) == S_OK)
{
    CHECK_FAILURE(
        options5-
>put_EnableTrackingPrevention(m_TrackingPreventionEnabled ? TRUE : FALSE));
}

Microsoft::WRL::ComPtr<ICoreWebView2EnvironmentOptions6> options6;
if (options.As(&options6) == S_OK)
{
    CHECK_FAILURE(options6->put_AreBrowserExtensionsEnabled(TRUE));
}

HRESULT hr = CreateCoreWebView2EnvironmentWithOptions(
    subFolder, m_userDataFolder.c_str(), options.Get(),
    Callback<ICoreWebView2CreateCoreWebView2EnvironmentCompletedHandler>
(
    this, &AppWindow::OnCreateEnvironmentCompleted)
    .Get());
if (!SUCCEEDED(hr))
{
    switch (hr)
    {
        case HRESULT_FROM_WIN32(ERROR_FILE_NOT_FOUND):
        {
            MessageBox(
                m_mainWindow,
                L"Couldn't find Edge WebView2 Runtime. "
                "Do you have a version installed?",
                nullptr, MB_OK);
        }
        break;
        case HRESULT_FROM_WIN32(ERROR_FILE_EXISTS):
        {
            MessageBox(
                m_mainWindow,
                L"User data folder cannot be created because a file with the
same name already "
                L"exists.",
                nullptr, MB_OK);
        }
        break;
        case E_ACCESSDENIED:
        {
            MessageBox(
                m_mainWindow, L"Unable to create user data folder, Access
Denied.", nullptr,
                MB_OK);
        }
    }
}

```

```

        }
        break;
    case E_FAIL:
    {
        MessageBox(m_mainWindow, L"Edge runtime unable to start",
        nullptr, MB_OK);
    }
    break;
    default:
    {
        ShowFailure(hr, L"Failed to create WebView2 environment");
    }
}
}

// This is the callback passed to CreateWebViewEnvironmentWithOptions.
// Here we simply create the WebView.
HRESULT AppWindow::OnCreateEnvironmentCompleted(
    HRESULT result, ICoreWebView2Environment* environment)
{
    if (result != S_OK)
    {
        ShowFailure(result, L"Failed to create environment object.");
        return S_OK;
    }
    m_webViewEnvironment = environment;

    if (m_webviewOption.entry ==
    WebViewCreateEntry::EVER_FROM_CREATE_WITH_OPTION_MENU
    )
    {
        return CreateControllerWithOptions();
    }
    auto webViewEnvironment3 =
    m_webViewEnvironment.try_query<ICoreWebView2Environment3>();

    if (webViewEnvironment3 && (m_dcompDevice || m_wincompCompositor))
    {
        CHECK_FAILURE(webViewEnvironment3-
>CreateCoreWebView2CompositionController(
            m_mainWindow,
            Callback<ICoreWebView2CreateCoreWebView2CompositionControllerCompletedHandle
r>(
                [this](
                    HRESULT result,
                    ICoreWebView2CompositionController*
compositionController) -> HRESULT
                {
                    auto controller =
                        wil::com_ptr<ICoreWebView2CompositionController>
(compositionController)
                            .query<ICoreWebView2Controller>();
                    return OnCreateCoreWebView2ControllerCompleted(result,
controller.get());
                }
            )
        );
    }
}

```

```

        })
        .Get());
    }
    else
    {
        CHECK_FAILURE(m_webViewEnvironment->CreateCoreWebView2Controller(
            m_mainWindow,
        Callback<ICoreWebView2CreateCoreWebView2ControllerCompletedHandler>(
            this,
        &AppWindow::OnCreateCoreWebView2ControllerCompleted)
            .Get());
    }

    return S_OK;
}

```

It is recommended that the application handles restart manager messages so that it can be restarted gracefully in the case when the app is using Edge for WebView from a certain installation and that installation is being uninstalled. For example, if a user installs Edge from Dev channel and opts to use Edge from that channel for testing the app, and then uninstalls Edge from that channel without closing the app, the app will be restarted to allow uninstallation of the dev channel to succeed.

C++

```

case WM_QUERYENDSESSION:
{
    // yes, we can shut down
    // Register how we might be restarted
    RegisterApplicationRestart(L"--restore", RESTART_NO_CRASH |
RESTART_NO_HANG);
    *result = TRUE;
    return true;
}
break;
case WM_ENDSESSION:
{
    if (wParam == TRUE)
    {
        // save app state and exit.
        PostQuitMessage(0);
        return true;
    }
}
break;

```

The app should retry `CreateCoreWebView2CompositionController` upon failure, unless the error code is `HRESULT_FROM_WIN32(ERROR_INVALID_STATE)`. When the app retries `CreateCoreWebView2CompositionController` upon failure, it is recommended that the app

restarts from creating a new WebView2 Environment. If a WebView2 Runtime update happens, the version associated with a WebView2 Environment may have been removed and causing the object to no longer work. Creating a new WebView2 Environment works since it uses the latest version.

WebView creation fails with `HRESULT_FROM_WIN32(ERROR_INVALID_STATE)` if a running instance using the same user data folder exists, and the Environment objects have different `EnvironmentOptions`. For example, if a WebView was created with one language, an attempt to create a WebView with a different language using the same user data folder will fail.

The creation will fail with `E_ABORT` if `parentWindow` is destroyed before the creation is finished. If this is caused by a call to `DestroyWindow`, the creation completed handler will be invoked before `DestroyWindow` returns, so you can use this to cancel creation and clean up resources synchronously when quitting a thread.

In rare cases the creation can fail with `E_UNEXPECTED` if runtime does not have permissions to the user data folder.

`CreateCoreWebView2CompositionController` is supported in the following versions of Windows:

- Windows 11
- Windows 10
- Windows Server 2019
- Windows Server 2016

## CreateCoreWebView2PointerInfo

Create an empty `ICoreWebView2PointerInfo`.

```
public HRESULT CreateCoreWebView2PointerInfo(ICoreWebView2PointerInfo **  
pointerInfo)
```

The returned `ICoreWebView2PointerInfo` needs to be populated with all of the relevant info before calling `SendPointerInput`.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Environment4

Article • 02/26/2024

```
interface ICoreWebView2Environment4
: public ICoreWebView2Environment3
```

A continuation of the [ICoreWebView2Environment3](#) interface.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">GetAutomationProviderForWindow</a>	Returns the Automation Provider for the WebView that matches the provided window.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.774.44
WebView2 Win32 Prerelease	1.0.824

## Members

### [GetAutomationProviderForWindow](#)

Returns the Automation Provider for the WebView that matches the provided window.

```
public HRESULT GetAutomationProviderForWindow(HWND hwnd, IUnknown **  
provider)
```

Host apps are expected to implement `IRawElementProviderOverride`. When `GetOverrideProviderForHwnd` is called, the app can pass the HWND to `GetAutomationProviderForWindow` to find the matching WebView Automation Provider.

---

## Feedback

Was this page helpful?



Yes



No

# interface ICoreWebView2Environment5

Article • 02/26/2024

```
interface ICoreWebView2Environment5
: public ICoreWebView2Environment4
```

A continuation of the [ICoreWebView2Environment4](#) interface that supports the `BrowserProcessExited` event.

## Summary

[+] [Expand table](#)

Members	Descriptions
<a href="#">add_BrowserProcessExited</a>	Add an event handler for the <code>BrowserProcessExited</code> event.
<a href="#">remove_BrowserProcessExited</a>	Remove an event handler previously added with <code>add_BrowserProcessExited</code> .

## Applies to

[+] [Expand table](#)

Product	Introduced
WebView2 Win32	1.0.992.28
WebView2 Win32 Prerelease	1.0.1010

## Members

### [add\\_BrowserProcessExited](#)

Add an event handler for the `BrowserProcessExited` event.

```
public HRESULT
add_BrowserProcessExited(ICoreWebView2BrowserProcessExitedEventHandler *
```

```
eventHandler, EventRegistrationToken * token)
```

The `BrowserProcessExited` event is raised when the collection of WebView2 Runtime processes for the browser process of this environment terminate due to browser process failure or normal shutdown (for example, when all associated WebViews are closed), after all resources have been released (including the user data folder). To learn about what these processes are, go to [Process model](#).

A handler added with this method is called until removed with `remove_BrowserProcessExited`, even if a new browser process is bound to this environment after earlier `BrowserProcessExited` events are raised.

Multiple app processes can share a browser process by creating their webviews from a `ICoreWebView2Environment` with the same user data folder. When the entire collection of `WebView2Runtime` processes for the browser process exit, all associated `ICoreWebView2Environment` objects receive the `BrowserProcessExited` event. Multiple processes sharing the same browser process need to coordinate their use of the shared user data folder to avoid race conditions and unnecessary waits. For example, one process should not clear the user data folder at the same time that another process recovers from a crash by recreating its `WebView` controls; one process should not block waiting for the event if other app processes are using the same browser process (the browser process will not exit until those other processes have closed their webviews too).

Note this is an event from the `ICoreWebView2Environment3` interface, not the `ICoreWebView2` one. The difference between `BrowserProcessExited` and `ICoreWebView2`'s `ProcessFailed` is that `BrowserProcessExited` is raised for any **browser process** exit (expected or unexpected, after all associated processes have exited too), while `ProcessFailed` is raised for **unexpected** process exits of any kind (browser, render, GPU, and all other types), or for main frame **render process** unresponsiveness. To learn more about the `WebView2` Process Model, go to [Process model](#).

In the case the browser process crashes, both `BrowserProcessExited` and `ProcessFailed` events are raised, but the order is not guaranteed. These events are intended for different scenarios. It is up to the app to coordinate the handlers so they do not try to perform reliability recovery while also trying to move to a new `WebView2` Runtime version or remove the user data folder.

C++

```
// Close the WebView and deinitialize related state. This doesn't close the  
// app window.
```

```

bool AppWindow::CloseWebView(bool cleanupUserDataFolder)
{
    if (auto file = GetComponent<FileComponent>())
    {
        if (file->IsPrintToPdfInProgress())
        {
            int selection = MessageBox(
                m_mainWindow, L"Print to PDF is in progress. Continue
closing?", 
                L"Print to PDF", MB_YESNO);
            if (selection == IDNO)
            {
                return false;
            }
        }
    }
    // 1. Delete components.
    DeleteAllComponents();

    // 2. If cleanup needed and BrowserProcessExited event interface
available,
    // register to cleanup upon browser exit.
    wil::com_ptr<ICoreWebView2Environment5> environment5;
    if (m_webViewEnvironment)
    {
        environment5 =
m_webViewEnvironment.try_query<ICoreWebView2Environment5>();
    }
    if (cleanupUserDataFolder && environment5)
    {
        // Before closing the WebView, register a handler with code to run
once the
        // browser process and associated processes are terminated.
        CHECK_FAILURE(environment5->add_BrowserProcessExited(
            Callback<ICoreWebView2BrowserProcessExitedEventHandler>(
                [environment5, this](
                    ICoreWebView2Environment* sender,
                    ICoreWebView2BrowserProcessExitedEventArgs* args)
            {
                COREWEBVIEW2_BROWSER_PROCESS_EXIT_KIND kind;
                UINT32 pid;
                CHECK_FAILURE(args->get_BrowserProcessExitKind(&kind));
                CHECK_FAILURE(args->get_BrowserProcessId(&pid));

                // If a new WebView is created from this
CoreWebView2Environment after
                    // the browser has exited but before our handler gets to
run, a new
data folder
                    // browser process will be created and lock the user
in these
                    // again. Do not attempt to cleanup the user data folder
against the
                    // cases. We check the PID of the exited browser process
                        // PID of the browser process to which our last

```

```

CoreWebView2 attached.
    if (pid == m_newestBrowserPid)
    {
        // Watch for graceful browser process exit. Let
ProcessFailed event
        // handler take care of failed browser process
termination.
        if (kind ==
COREWEBVIEW2_BROWSER_PROCESS_EXIT_KIND_NORMAL)
        {
            CHECK_FAILURE(environment5-
>remove_BrowserProcessExited(
                m_browserExitedEventToken));
            // Release the environment only after the
handler is invoked.
            // Otherwise, there will be no environment to
raise the event when
            // the collection of WebView2 Runtime processes
exit.
            m_webViewEnvironment = nullptr;
            RunAsync([this]() { CleanupUserDataFolder(); });
        }
        else
        {
            // The exiting process is not the last in use. Do
not attempt cleanup
            // as we might still have a webview open over the
user data folder.
            // Do not block from event handler.
            AsyncMessageBox(
                L"A new browser process prevented cleanup of the
user data folder.",
                L"Cleanup User Data Folder");
        }
        return S_OK;
    })
    .Get(),
    &m_browserExitedEventToken));
}

// 3. Close the webview.
if (m_controller)
{
    m_controller->Close();
    m_controller = nullptr;
    m_webView = nullptr;
    m_webView3 = nullptr;
}

// 4. If BrowserProcessExited event interface is not available, release
// environment and proceed to cleanup immediately. If the interface is
// available, release environment only if not waiting for the event.
if (!environment5)

```

```
{  
    m_webViewEnvironment = nullptr;  
    if (cleanupUserDataFolder)  
    {  
        CleanupUserDataFolder();  
    }  
}  
else if (!cleanupUserDataFolder)  
{  
    // Release the environment object here only if no cleanup is needed.  
    // If cleanup is needed, the environment object release is deferred  
    // until the browser process exits, otherwise the handler for the  
    // BrowserProcessExited event will not be called.  
    m_webViewEnvironment = nullptr;  
}  
  
// reset profile name  
m_profileName = L"";  
m_documentTitle = L"";  
return true;  
}
```

## remove\_BrowserProcessExited

Remove an event handler previously added with `add_BrowserProcessExited`.

```
public HRESULT remove_BrowserProcessExited(EventRegistrationToken token)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Environment6

Article • 02/26/2024

```
interface ICoreWebView2Environment6
    : public ICoreWebView2Environment5
```

This interface is an extension of the [ICoreWebView2Environment](#) that supports creating print settings for printing to PDF.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">CreatePrintSettings</a>	Creates the <a href="#">ICoreWebView2PrintSettings</a> used by the <code>PrintToPdf</code> method.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1020.30
WebView2 Win32 Prerelease	1.0.1056

## Members

### CreatePrintSettings

Creates the [ICoreWebView2PrintSettings](#) used by the `PrintToPdf` method.

```
public HRESULT CreatePrintSettings(ICoreWebView2PrintSettings ** printSettings)
```

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Environment7

Article • 02/26/2024

```
interface ICoreWebView2Environment7
: public ICoreWebView2Environment6
```

This interface is an extension of the [ICoreWebView2Environment](#).

## Summary

[+] [Expand table](#)

Members	Descriptions
<a href="#">get_UserDataFolder</a>	Returns the user data folder that all CoreWebView2's created from this environment are using.

An object implementing the ICoreWebView2Environment7 interface will also implement [ICoreWebView2Environment](#).

## Applies to

[+] [Expand table](#)

Product	Introduced
WebView2 Win32	1.0.1054.31
WebView2 Win32 Prerelease	1.0.1056

## Members

### get\_UserDataFolder

Returns the user data folder that all CoreWebView2's created from this environment are using.

```
public HRESULT get_UserDataFolder(LPWSTR * value)
```

This could be either the value passed in by the developer when creating the environment object or the calculated one for default handling. It will always be an absolute path.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

C++

```
auto environment7 =
m_webViewEnvironment.try_query<ICoreWebView2Environment7>();
CHECK_FEATURE_RETURN(environment7);
wil::unique_cotaskmem_string userDataFolder;
environment7->get_UserDataFolder(&userDataFolder);
MessageBox(m_mainWindow, userDataFolder.get(), L"User Data Folder",
MB_OK);
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Environment8

Article • 02/26/2024

```
interface ICoreWebView2Environment8
    : public ICoreWebView2Environment7
```

A continuation of the [ICoreWebView2Environment7](#) interface that supports the `ProcessInfosChanged` event.

## Summary

[ ] Expand table

Members	Descriptions
<a href="#">add_ProcessInfosChanged</a>	Adds an event handler for the <code>ProcessInfosChanged</code> event.
<a href="#">GetProcessInfos</a>	Returns the <a href="#">ICoreWebView2ProcessInfoCollection</a> Provide a list of all process using same user data folder except for crashpad process.
<a href="#">remove_ProcessInfosChanged</a>	Remove an event handler previously added with <code>add_ProcessInfosChanged</code> .

## Applies to

[ ] Expand table

Product	Introduced
WebView2 Win32	1.0.1108.44
WebView2 Win32 Prerelease	1.0.1133

## Members

### [add\\_ProcessInfosChanged](#)

Adds an event handler for the `ProcessInfosChanged` event.

```
public HRESULT  
add_ProcessInfosChanged(ICoreWebView2ProcessInfosChangedEventHandler *  
eventHandler, EventRegistrationToken * token)
```

C++

```
    CHECK_FAILURE(environment8->add_ProcessInfosChanged(  
        Callback<ICoreWebView2ProcessInfosChangedEventHandler>(  
            [this](ICoreWebView2Environment* sender, IUnknown* args) ->  
HRESULT {  
            wil::com_ptr<ICoreWebView2Environment8>  
webviewEnvironment;  
            sender-  
>QueryInterface(IID_PPV_ARGS(&webviewEnvironment));  
            CHECK_FAILURE(  
                webviewEnvironment-  
>GetProcessInfos(&m_processCollection));  
            return S_OK;  
        })  
        .Get(),  
        &m_processInfosChangedToken));
```

C++

```
void ProcessComponent::PerformanceInfo()  
{  
    std::wstring result;  
    UINT processListCount;  
    CHECK_FAILURE(m_processCollection->get_Count(&processListCount));  
  
    if (processListCount == 0)  
    {  
        result += L"No process found.";  
    }  
    else  
    {  
        result += std::to_wstring(processListCount) + L" process(s) found";  
        result += L"\n\n";  
        for (UINT i = 0; i < processListCount; ++i)  
        {  
            wil::com_ptr<ICoreWebView2ProcessInfo> processInfo;  
            CHECK_FAILURE(m_processCollection->GetValueAtIndex(i,  
&processInfo));  
  
            INT32 processId = 0;  
            COREWEBVIEW2_PROCESS_KIND kind;  
            CHECK_FAILURE(processInfo->get_ProcessId(&processId));  
            CHECK_FAILURE(processInfo->get_Kind(&kind));
```

```

        WCHAR id[4096] = L"";
        StringCchPrintf(id, ARRAYSIZE(id), L"Process ID: %u",
processId);

        HANDLE processHandle =
            OpenProcess(PROCESS_QUERY_LIMITED_INFORMATION, FALSE,
processId);
        PROCESS_MEMORY_COUNTERS_EX pmc;
        GetProcessMemoryInfo(
            processHandle, reinterpret_cast<PROCESS_MEMORY_COUNTERS*>
(&pmc), sizeof(pmc));
        SIZE_T virtualMemUsed = pmc.PrivateUsage / 1024;
        WCHAR memory[4096] = L"";
        StringCchPrintf(memory, ARRAYSIZE(memory), L"Memory: %u",
virtualMemUsed);
        CloseHandle(processHandle);

        result = result + id + L" | Process Kind: " +
ProcessKindToString(kind) + L" | " +
                memory + L" KB\n";
    }
}
MessageBox(nullptr, result.c_str(), L"Memory Usage", MB_OK);
}

```

## GetProcessInfos

Returns the [ICoreWebView2ProcessInfoCollection](#) Provide a list of all process using same user data folder except for crashpad process.

```
public HRESULT GetProcessInfos(ICoreWebView2ProcessInfoCollection ** value)
```

## remove\_ProcessInfosChanged

Remove an event handler previously added with [add\\_ProcessInfosChanged](#).

```
public HRESULT remove_ProcessInfosChanged(EventRegistrationToken token)
```

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Environment9

Article • 02/26/2024

```
interface ICoreWebView2Environment9
    : public ICoreWebView2Environment8
```

A continuation of the [ICoreWebView2Environment](#) interface for creating CoreWebView2 ContextMenuItem objects.

## Summary

[+] [Expand table](#)

Members	Descriptions
<a href="#">CreateContextMenuItem</a>	Create a custom <code>ContextMenuItem</code> object to insert into the WebView context menu.

## Applies to

[+] [Expand table](#)

Product	Introduced
WebView2 Win32	1.0.1185.39
WebView2 Win32 Prerelease	1.0.1189

## Members

### [CreateContextMenuItem](#)

Create a custom `ContextMenuItem` object to insert into the WebView context menu.

```
public HRESULT CreateContextMenuItem(LPCWSTR label, IStream * iconStream,
COREWEBVIEW2_CONTEXT_MENU_ITEM_KIND kind,
ICoreWebView2ContextMenuItem ** item)
```

CoreWebView2 will rewind the icon stream before decoding. There is a limit of 1000 active custom context menu items at a given time. Attempting to create more before deleting existing ones will fail with ERROR\_NOT\_ENOUGH\_QUOTA. It is recommended to reuse ContextMenuItems across ContextMenuRequested events for performance. The returned ContextMenuItem object's `IsEnabled` property will default to `TRUE` and `.IsChecked` property will default to `FALSE`. A `CommandId` will be assigned to the ContextMenuItem object that's unique across active custom context menu items, but command ID values of deleted ContextMenuItems can be reassigned.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2EnvironmentOptions

Article • 02/26/2024

```
interface ICoreWebView2EnvironmentOptions
: public IUnknown
```

Options used to create WebView2 Environment.

## Summary

[ ] Expand table

Members	Descriptions
<a href="#">get_AdditionalBrowserArguments</a>	Changes the behavior of the WebView.
<a href="#">get_AllowSingleSignOnUsingOSPrimaryAccount</a>	The <code>AllowSingleSignOnUsingOSPrimaryAccount</code> property is used to enable single sign on with Azure Active Directory (AAD) and personal Microsoft Account (MSA) resources inside WebView.
<a href="#">get_Language</a>	The default display language for WebView.
<a href="#">get_TargetCompatibleBrowserVersion</a>	Specifies the version of the WebView2 Runtime binaries required to be compatible with your app.
<a href="#">put_AdditionalBrowserArguments</a>	Sets the <code>AdditionalBrowserArguments</code> property.
<a href="#">put_AllowSingleSignOnUsingOSPrimaryAccount</a>	Sets the <code>AllowSingleSignOnUsingOSPrimaryAccount</code> property.
<a href="#">put_Language</a>	Sets the <code>Language</code> property.
<a href="#">put_TargetCompatibleBrowserVersion</a>	Sets the <code>TargetCompatibleBrowserVersion</code> property.

A default implementation is provided in `WebView2EnvironmentOptions.h`.

C++

```
    std::wstring args;
    args.append(L"--enable-
features=ThirdPartyStoragePartitioning,PartitionedCookies");
    auto options = Microsoft::WRL::Make<CoreWebView2EnvironmentOptions>();
    options->put_AdditionalBrowserArguments(args.c_str());
    CHECK_FAILURE(
        options->put_AllowSingleSignOnUsingOSPrimaryAccount(m_AADSSOEnabled
? TRUE : FALSE));
    CHECK_FAILURE(options->put_ExclusiveUserDataFolderAccess(
        m_ExclusiveUserDataFolderAccess ? TRUE : FALSE));
    if (!m_language.empty())
        CHECK_FAILURE(options->put_Language(m_language.c_str()));
    CHECK_FAILURE(options->put_IsCustomCrashReportingEnabled(
        m_CustomCrashReportingEnabled ? TRUE : FALSE));

    Microsoft::WRL::ComPtr<ICoreWebView2EnvironmentOptions4> options4;
    if (options.As(&options4) == S_OK)
    {
        const WCHAR* allowedOrigins[1] = {L"https://*.example.com"};

        auto customSchemeRegistration =
            Microsoft::WRL::Make<CoreWebView2CustomSchemeRegistration>
(L"custom-scheme");
        customSchemeRegistration->SetAllowedOrigins(1, allowedOrigins);
        auto customSchemeRegistration2 =
            Microsoft::WRL::Make<CoreWebView2CustomSchemeRegistration>
(L"wv2rocks");
        customSchemeRegistration2->put_TreatAsSecure(TRUE);
        customSchemeRegistration2->SetAllowedOrigins(1, allowedOrigins);
        customSchemeRegistration2->put_HasAuthorityComponent(TRUE);
        auto customSchemeRegistration3 =
            Microsoft::WRL::Make<CoreWebView2CustomSchemeRegistration>(
                L"custom-scheme-not-in-allowed-origins");
        ICoreWebView2CustomSchemeRegistration* registrations[3] = {
            customSchemeRegistration.Get(), customSchemeRegistration2.Get(),
            customSchemeRegistration3.Get()};
        options4->SetCustomSchemeRegistrations(
            2, static_cast<ICoreWebView2CustomSchemeRegistration**>
(registrations));
    }

    Microsoft::WRL::ComPtr<ICoreWebView2EnvironmentOptions5> options5;
    if (options.As(&options5) == S_OK)
    {
        CHECK_FAILURE(
            options5-
>put_EnableTrackingPrevention(m_TrackingPreventionEnabled ? TRUE : FALSE));
    }

    Microsoft::WRL::ComPtr<ICoreWebView2EnvironmentOptions6> options6;
    if (options.As(&options6) == S_OK)
```

```

{
    CHECK_FAILURE(options6->put_AreBrowserExtensionsEnabled(TRUE));
}

HRESULT hr = CreateCoreWebView2EnvironmentWithOptions(
    subFolder, m_userDataFolder.c_str(), options.Get(),
    Callback<ICoreWebView2CreateCoreWebView2EnvironmentCompletedHandler>
(
    this, &AppWindow::OnCreateEnvironmentCompleted)
    .Get());

```

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.488
WebView2 Win32 Prerelease	0.9.488

## Members

### get\_AdditionalBrowserArguments

Changes the behavior of the WebView.

```
public HRESULT get_AdditionalBrowserArguments(LPWSTR * value)
```

The arguments are passed to the browser process as part of the command. For more information about using command-line switches with Chromium browser processes, navigate to [Run Chromium with Flags](#). The value appended to a switch is appended to the browser process, for example, in `--edge-webview-switches=xxx` the value is `xxx`. If you specify a switch that is important to WebView functionality, it is ignored, for example, `--user-data-dir`. Specific features are disabled internally and blocked from being enabled. If a switch is specified multiple times, only the last instance is used.

#### ! Note

A merge of the different values of the same switch is not attempted, except for disabled and enabled features. The features specified by `--enable-features` and `--disable-features` are merged with simple logic.

- The features is the union of the specified features and built-in features. If a feature is disabled, it is removed from the enabled features list.

If you specify command-line switches and use the `additionalBrowserArguments` parameter, the `--edge-webview-switches` value takes precedence and is processed last. If a switch fails to parse, the switch is ignored. The default state for the operation is to run the browser process with no extra flags.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_AllowSingleSignOnUsingOSPrimaryAccount

The `AllowSingleSignOnUsingOSPrimaryAccount` property is used to enable single sign on with Azure Active Directory (AAD) and personal Microsoft Account (MSA) resources inside WebView.

```
public HRESULT get_AllowSingleSignOnUsingOSPrimaryAccount(BOOL * allow)
```

All AAD accounts, connected to Windows and shared for all apps, are supported. For MSA, SSO is only enabled for the account associated for Windows account login, if any. Default is disabled. Universal Windows Platform apps must also declare `enterpriseCloudSSO` [Restricted capabilities](#) for the single sign on (SSO) to work.

## get\_Language

The default display language for WebView.

```
public HRESULT get_Language(LPWSTR * value)
```

It applies to browser UI such as context menu and dialogs. It also applies to the `accept-languages` HTTP header that WebView sends to websites. The intended locale value is in the format of BCP 47 Language Tags. More information can be found from [IETF BCP47](#).

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_TargetCompatibleBrowserVersion

Specifies the version of the WebView2 Runtime binaries required to be compatible with your app.

```
public HRESULT get_TargetCompatibleBrowserVersion(LPWSTR * value)
```

This defaults to the WebView2 Runtime version that corresponds with the version of the SDK the app is using. The format of this value is the same as the format of the `BrowserVersionString` property and other `BrowserVersion` values. Only the version part of the `BrowserVersion` value is respected. The channel suffix, if it exists, is ignored. The version of the WebView2 Runtime binaries actually used may be different from the specified `TargetCompatibleBrowserVersion`. The binaries are only guaranteed to be compatible. Verify the actual version on the `BrowserVersionString` property on the `ICoreWebView2Environment`.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## put\_AdditionalBrowserArguments

Sets the `AdditionalBrowserArguments` property.

```
public HRESULT put_AdditionalBrowserArguments(LPCWSTR value)
```

Please note that calling this API twice will replace the previous value rather than appending to it. If there are multiple switches, there should be a space in between them. The one exception is if multiple features are being enabled/disabled for a single switch, in which case the features should be comma-separated. Ex. "--disable-features=feature1,feature2 --some-other-switch --do-something"

## put\_AllowSingleSignOnUsingOSPrimaryAccount

Sets the `AllowSingleSignOnUsingOSPrimaryAccount` property.

```
public HRESULT put_AllowSingleSignOnUsingOSPrimaryAccount(BOOL allow)
```

## put\_Language

Sets the `Language` property.

```
public HRESULT put_Language(LPCWSTR value)
```

## put\_TargetCompatibleBrowserVersion

Sets the `TargetCompatibleBrowserVersion` property.

```
public HRESULT put_TargetCompatibleBrowserVersion(LPCWSTR value)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2EnvironmentOptions2

Article • 02/26/2024

```
interface ICoreWebView2EnvironmentOptions2
: public IUnknown
```

Additional options used to create WebView2 Environment.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_ExclusiveUserDataFolderAccess</a>	Whether other processes can create WebView2 from WebView2Environment created with the same user data folder and therefore sharing the same WebView browser process instance.
<a href="#">put_ExclusiveUserDataFolderAccess</a>	Sets the <code>ExclusiveUserDataFolderAccess</code> property.

A default implementation is provided in `WebView2EnvironmentOptions.h`.

C++

```
std::wstring args;
args.append(L"--enable-
features=ThirdPartyStoragePartitioning,PartitionedCookies");
auto options = Microsoft::WRL::Make<CoreWebView2EnvironmentOptions>();
options->put_AdditionalBrowserArguments(args.c_str());
CHECK_FAILURE(
    options->put_AllowSingleSignOnUsingOSPrimaryAccount(m_AADSSOEnabled
? TRUE : FALSE));
CHECK_FAILURE(options->put_ExclusiveUserDataFolderAccess(
    m_ExclusiveUserDataFolderAccess ? TRUE : FALSE));
if (!m_language.empty())
    CHECK_FAILURE(options->put_Language(m_language.c_str()));
CHECK_FAILURE(options->put_IsCustomCrashReportingEnabled(
    m_CustomCrashReportingEnabled ? TRUE : FALSE));

Microsoft::WRL::ComPtr<ICoreWebView2EnvironmentOptions4> options4;
```

```

    if (options.As(&options4) == S_OK)
    {
        const WCHAR* allowedOrigins[1] = {L"https://*.example.com"};

        auto customSchemeRegistration =
            Microsoft::WRL::Make<CoreWebView2CustomSchemeRegistration>
(L"custom-scheme");
        customSchemeRegistration->SetAllowedOrigins(1, allowedOrigins);
        auto customSchemeRegistration2 =
            Microsoft::WRL::Make<CoreWebView2CustomSchemeRegistration>
(L"wv2rocks");
        customSchemeRegistration2->put_TreatAsSecure(TRUE);
        customSchemeRegistration2->SetAllowedOrigins(1, allowedOrigins);
        customSchemeRegistration2->put_HasAuthorityComponent(TRUE);
        auto customSchemeRegistration3 =
            Microsoft::WRL::Make<CoreWebView2CustomSchemeRegistration>(
                L"custom-scheme-not-in-allowed-origins");
        ICoreWebView2CustomSchemeRegistration* registrations[3] = {
            customSchemeRegistration.Get(), customSchemeRegistration2.Get(),
            customSchemeRegistration3.Get()};
        options4->SetCustomSchemeRegistrations(
            2, static_cast<ICoreWebView2CustomSchemeRegistration**>
(registrations));
    }

    Microsoft::WRL::ComPtr<ICoreWebView2EnvironmentOptions5> options5;
    if (options.As(&options5) == S_OK)
    {
        CHECK_FAILURE(
            options5-
>put_EnableTrackingPrevention(m_TrackingPreventionEnabled ? TRUE : FALSE));
    }

    Microsoft::WRL::ComPtr<ICoreWebView2EnvironmentOptions6> options6;
    if (options.As(&options6) == S_OK)
    {
        CHECK_FAILURE(options6->put_AreBrowserExtensionsEnabled(TRUE));
    }

    HRESULT hr = CreateCoreWebView2EnvironmentWithOptions(
        subFolder, m_userDataFolder.c_str(), options.Get(),
        Callback<ICoreWebView2CreateCoreWebView2EnvironmentCompletedHandler>
(
        this, &AppWindow::OnCreateEnvironmentCompleted)
        .Get());

```

## Applies to

[ ] Expand table

Product	Introduced
WebView2 Win32	1.0.1185.39
WebView2 Win32 Prerelease	1.0.1189

## Members

### get\_ExclusiveUserDataFolderAccess

Whether other processes can create WebView2 from WebView2Environment created with the same user data folder and therefore sharing the same WebView browser process instance.

```
public HRESULT get_ExclusiveUserDataFolderAccess(BOOL * value)
```

Default is FALSE.

### put\_ExclusiveUserDataFolderAccess

Sets the `ExclusiveUserDataFolderAccess` property.

```
public HRESULT put_ExclusiveUserDataFolderAccess(BOOL value)
```

The `ExclusiveUserDataFolderAccess` property specifies that the WebView environment obtains exclusive access to the user data folder. If the user data folder is already being used by another WebView environment with a different value for `ExclusiveUserDataFolderAccess` property, the creation of a WebView2Controller using the environment object will fail with `HRESULT_FROM_WIN32(ERROR_INVALID_STATE)`. When set as TRUE, no other WebView can be created from other processes using WebView2Environment objects with the same UserDataFolder. This prevents other processes from creating WebViews which share the same browser process instance, since sharing is performed among WebViews that have the same UserDataFolder. When another process tries to create a WebView2Controller from an WebView2Environment object created with the same user data folder, it will fail with `HRESULT_FROM_WIN32(ERROR_INVALID_STATE)`.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2EnvironmentOptions3

Article • 02/26/2024

```
interface ICoreWebView2EnvironmentOptions3
: public IUnknown
```

Additional options used to create WebView2 Environment to manage crash reporting.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_IsCustomCrashReportingEnabled</a>	When <code>IsCustomCrashReportingEnabled</code> is set to <code>TRUE</code> , Windows won't send crash data to Microsoft endpoint.
<a href="#">put_IsCustomCrashReportingEnabled</a>	Sets the <code>IsCustomCrashReportingEnabled</code> property.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1518.46
WebView2 Win32 Prerelease	1.0.1549

## Members

### [get\\_IsCustomCrashReportingEnabled](#)

When `IsCustomCrashReportingEnabled` is set to `TRUE`, Windows won't send crash data to Microsoft endpoint.

```
public HRESULT get_IsCustomCrashReportingEnabled(BOOL * value)
```

`IsCustomCrashReportingEnabled` is default to be `FALSE`, in this case, WebView will respect OS consent.

## **put\_IsCustomCrashReportingEnabled**

Sets the `IsCustomCrashReportingEnabled` property.

```
public HRESULT put_IsCustomCrashReportingEnabled(BOOL value)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2EnvironmentOptions4

Article • 02/26/2024

```
interface ICoreWebView2EnvironmentOptions4
: public IUnknown
```

Additional options used to create WebView2 Environment that manages custom scheme registration.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">GetCustomSchemeRegistrations</a>	Array of custom scheme registrations.
<a href="#">SetCustomSchemeRegistrations</a>	Set the array of custom scheme registrations to be used.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1587.40
WebView2 Win32 Prerelease	1.0.1619

## Members

### [GetCustomSchemeRegistrations](#)

Array of custom scheme registrations.

```
public HRESULT GetCustomSchemeRegistrations(UINT32 * count,
ICoreWebView2CustomSchemeRegistration *** schemeRegistrations)
```

The returned `ICoreWebView2CustomSchemeRegistration` pointers must be released, and the array itself must be deallocated with `CoTaskMemFree`.

## SetCustomSchemeRegistrations

Set the array of custom scheme registrations to be used.

```
public HRESULT SetCustomSchemeRegistrations(UINT32 count,
ICoreWebView2CustomSchemeRegistration ** schemeRegistrations)
```

C++

```
Microsoft::WRL::ComPtr<ICoreWebView2EnvironmentOptions4> options4;
if (options.As(&options4) == S_OK)
{
    const WCHAR* allowedOrigins[1] = {L"https://*.example.com"};

    auto customSchemeRegistration =
        Microsoft::WRL::Make<CoreWebView2CustomSchemeRegistration>
(L"custom-scheme");
    customSchemeRegistration->SetAllowedOrigins(1, allowedOrigins);
    auto customSchemeRegistration2 =
        Microsoft::WRL::Make<CoreWebView2CustomSchemeRegistration>
(L"wv2rocks");
    customSchemeRegistration2->put_TreatAsSecure(TRUE);
    customSchemeRegistration2->SetAllowedOrigins(1, allowedOrigins);
    customSchemeRegistration2->put_HasAuthorityComponent(TRUE);
    auto customSchemeRegistration3 =
        Microsoft::WRL::Make<CoreWebView2CustomSchemeRegistration>(
            L"custom-scheme-not-in-allowed-origins");
    ICoreWebView2CustomSchemeRegistration* registrations[3] = {
        customSchemeRegistration.Get(), customSchemeRegistration2.Get(),
        customSchemeRegistration3.Get()};
    options4->SetCustomSchemeRegistrations(
        2, static_cast<ICoreWebView2CustomSchemeRegistration**>
(registrations));
}
```

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2EnvironmentOptions5

Article • 02/26/2024

```
interface ICoreWebView2EnvironmentOptions5
: public IUnknown
```

Additional options used to create WebView2 Environment to manage tracking prevention.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_EnableTrackingPrevention</a>	The <code>EnableTrackingPrevention</code> property is used to enable/disable tracking prevention feature in WebView2.
<a href="#">put_EnableTrackingPrevention</a>	Sets the <code>EnableTrackingPrevention</code> property.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1661.34
WebView2 Win32 Prerelease	1.0.1619

## Members

### `get_EnableTrackingPrevention`

The `EnableTrackingPrevention` property is used to enable/disable tracking prevention feature in WebView2.

```
public HRESULT get_EnableTrackingPrevention(BOOL * value)
```

This property enable/disable tracking prevention for all the WebView2's created in the same environment. By default this feature is enabled to block potentially harmful trackers and trackers from sites that aren't visited before and set to `COREWEBVIEW2_TRACKING_PREVENTION_LEVEL_BALANCED` or whatever value was last changed/persisted on the profile.

You can set this property to false to disable the tracking prevention feature if the app only renders content in the WebView2 that is known to be safe. Disabling this feature when creating environment also improves runtime performance by skipping related code.

You shouldn't disable this property if WebView2 is being used as a "full browser" with arbitrary navigation and should protect end user privacy.

There is `ICoreWebView2Profile3::PreferredTrackingPreventionLevel` property to control levels of tracking prevention of the WebView2's associated with a same profile. However, you can also disable tracking prevention later using `ICoreWebView2Profile3::PreferredTrackingPreventionLevel` property and `COREWEBVIEW2_TRACKING_PREVENTION_LEVEL_NONE` value but that doesn't improves runtime performance.

See `ICoreWebView2Profile3::PreferredTrackingPreventionLevel` for more details.

Tracking prevention protects users from online tracking by restricting the ability of trackers to access browser-based storage as well as the network. See [Tracking prevention](#).

## put\_EnableTrackingPrevention

Sets the `EnableTrackingPrevention` property.

```
public HRESULT put_EnableTrackingPrevention(BOOL value)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2EnvironmentOptions6

Article • 02/26/2024

```
interface ICoreWebView2EnvironmentOptions6
: public IUnknown
```

Additional options used to create WebView2 Environment to manage browser extensions.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_AreBrowserExtensionsEnabled</a>	When <code>AreBrowserExtensionsEnabled</code> is set to <code>TRUE</code> , new extensions can be added to user profile and used.
<a href="#">put_AreBrowserExtensionsEnabled</a>	Sets the <code>AreBrowserExtensionsEnabled</code> property.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.2210.55
WebView2 Win32 Prerelease	1.0.2194

## Members

### `get_AreBrowserExtensionsEnabled`

When `AreBrowserExtensionsEnabled` is set to `TRUE`, new extensions can be added to user profile and used.

```
public HRESULT get_AreBrowserExtensionsEnabled(BOOL * value)
```

`AreBrowserExtensionsEnabled` is default to be `FALSE`, in this case, new extensions can't be installed, and already installed extension won't be available to use in user profile. If connecting to an already running environment with a different value for `AreBrowserExtensionsEnabled` property, it will fail with `HRESULT_FROM_WIN32(ERROR_INVALID_STATE)`. See [ICoreWebView2BrowserExtension](#) for Extensions API details.

## put\_AreBrowserExtensionsEnabled

Sets the `AreBrowserExtensionsEnabled` property.

```
public HRESULT put_AreBrowserExtensionsEnabled(BOOL value)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2ExecuteScriptResult

Article • 02/26/2024

```
interface ICoreWebView2ExecuteScriptResult
: public IUnknown
```

This is the result for ExecuteScriptWithResult.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_Exception</a>	If Succeeded is false, you can use this property to get the unhandled exception thrown by script execution Note that due to the compatibility of the WinRT/.NET interface, S_OK will be returned even if the acquisition fails.
<a href="#">get_ResultAsJson</a>	A function that has no explicit return value returns undefined.
<a href="#">get_Succeeded</a>	This property is true if ExecuteScriptWithResult successfully executed script with no unhandled exceptions and the result is available in the ResultAsJson property or via the TryGetResultAsString method.
<a href="#">TryGetResultAsString</a>	If Succeeded is true and the result of script execution is a string, this method provides the value of the string result, and we will get the FALSE var value when the js result is not string type.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.2277.86
WebView2 Win32 Prerelease	1.0.2357

# Members

## get\_Exception

If Succeeded is false, you can use this property to get the unhandled exception thrown by script execution Note that due to the compatibility of the WinRT/.NET interface, S\_OK will be returned even if the acquisition fails.

```
public HRESULT get_Exception(ICoreWebView2ScriptException ** exception)
```

We can determine whether the acquisition is successful by judging whether the `exception` is `nullptr`.

## get\_ResultAsJson

A function that has no explicit return value returns undefined.

```
public HRESULT get_ResultAsJson(LPWSTR * jsonResult)
```

If the script that was run throws an unhandled exception, then the result is also "null". This method is applied asynchronously. If the method is run before `ContentLoading`, the script will not be executed and the string "null" will be returned. The return value description is as follows

1. S\_OK: Execution succeeds.
2. E\_POINTER: When the `jsonResult` is `nullptr`.

## get\_Succeeded

This property is true if ExecuteScriptWithResult successfully executed script with no unhandled exceptions and the result is available in the ResultAsJson property or via the TryGetResultAsString method.

```
public HRESULT get_Succeeded(BOOL * value)
```

If it is false then the script execution had an unhandled exception which you can get via the Exception property.

## TryGetResultAsString

If Succeeded is true and the result of script execution is a string, this method provides the value of the string result, and we will get the FALSE var value when the js result is not string type.

```
public HRESULT TryGetResultAsString(LPWSTR * stringResult, BOOL * value)
```

The return value description is as follows

1. S\_OK: Execution succeeds.
  2. E\_POINTER: When the `stringResult` or `value` is nullptr. NOTE: If the `value` returns FALSE , the `stringResult` will be set to a empty string.
- 

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2File

Article • 02/26/2024

```
interface ICoreWebView2File
: public IUnknown
```

Representation of a DOM [File](#) object passed via WebMessage.

## Summary

[\[+\] Expand table](#)

Members	Descriptions
<a href="#">get_Path</a>	Get the absolute file path.

You can use this object to obtain the path of a File dropped on WebView2.

C++

```
CHECK_FAILURE(m_webView->add_WebMessageReceived(
    Callback<ICoreWebView2WebMessageReceivedEventHandler>(
        [this](ICoreWebView2* sender,
    ICoreWebView2WebMessageReceivedEventArgs* args)
    {
        wil::com_ptr<ICoreWebView2WebMessageReceivedEventArgs2>
args2 =
        wil::com_ptr<ICoreWebView2WebMessageReceivedEventArgs>
(args)
        .query<ICoreWebView2WebMessageReceivedEventArgs2>();
        wil::com_ptr<ICoreWebView2ObjectCollectionView>
objectsCollection;
        args2->get_AdditionalObjects(&objectsCollection);
        unsigned int length;
        objectsCollection->get_Count(&length);

        // Array of file paths to be sent back to the webview as
```

JSON

```
        std::wstring pathObjects = L"[";
        for (unsigned int i = 0; i < length; i++)
        {
            wil::com_ptr<IUnknown> object;
            objectsCollection->GetValueAtIndex(i, &object);

            wil::com_ptr<ICoreWebView2File> file =
```

```

object.query<ICoreWebView2File>();
    if (file)
    {
        // Add the file to message to be sent back to
        webview

        wil::unique_cotaskmem_string path;
        file->get_Path(&path);
        std::wstring pathObject =
            L"{"L"path":L"" + std::wstring(path.get()) +
            L"\n}";

        // Escape backslashes
        std::wstring pathObjectEscaped;
        for (const auto& c : pathObject)
        {
            if (c == L'\\')
            {
                pathObjectEscaped += L"\\\\";
            }
            else
            {
                pathObjectEscaped += c;
            }
        }
        pathObjects += pathObjectEscaped;

        if (i < length - 1)
        {
            pathObjects += L",";
        }
    }
    pathObjects += L"]";

    // Post the message back to the webview so path is
    accessible to content
    m_webView->PostWebMessageAsJson(pathObjects.c_str());
}

return S_OK;
}
.Get(),
&m_webMessageReceivedToken));

```

## Applies to

[\[\] Expand table](#)

Product	Introduced
WebView2 Win32	1.0.1774.30
WebView2 Win32 Prerelease	1.0.1777

# Members

## get\_Path

Get the absolute file path.

```
public HRESULT get_Path(LPWSTR * path)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Frame

Article • 02/26/2024

```
interface ICoreWebView2Frame
: public IUnknown
```

ICoreWebView2Frame provides direct access to the iframes information.

## Summary

[ ] [Expand table](#)

Members	Descriptions
<a href="#">add_Destroyed</a>	The Destroyed event is raised when the iframe corresponding to this CoreWebView2Frame object is removed or the document containing that iframe is destroyed.
<a href="#">add_NameChanged</a>	Raised when the iframe changes its window.name property.
<a href="#">AddHostObjectToScriptWithOrigins</a>	Add the provided host object to script running in the iframe with the specified name for the list of the specified origins.
<a href="#">get_Name</a>	The value of iframe's window.name property.
<a href="#">IsDestroyed</a>	Check whether a frame is destroyed.
<a href="#">remove_Destroyed</a>	Remove an event handler previously added with add_Destroyed.
<a href="#">remove_NameChanged</a>	Remove an event handler previously added with add_NameChanged.
<a href="#">RemoveHostObjectFromScript</a>	Remove the host object specified by the name so that it is no longer accessible from JavaScript code in the iframe.

You can get an ICoreWebView2Frame by handling the ICoreWebView2\_4::add\_FrameCreated event.

## Applies to

Product	Introduced
WebView2 Win32	1.0.902.49
WebView2 Win32 Prerelease	1.0.902

## Members

### **add\_Destroyed**

The Destroyed event is raised when the iframe corresponding to this CoreWebView2Frame object is removed or the document containing that iframe is destroyed.

```
public HRESULT add_Destroyed(ICoreWebView2FrameDestroyedEventHandler *  
eventHandler, EventRegistrationToken * token)
```

### **add\_NameChanged**

Raised when the iframe changes its window.name property.

```
public HRESULT  
add_NameChanged(ICoreWebView2FrameNameChangedEventHandler *  
eventHandler, EventRegistrationToken * token)
```

### **AddHostObjectToScriptWithOrigins**

Add the provided host object to script running in the iframe with the specified name for the list of the specified origins.

```
public HRESULT AddHostObjectToScriptWithOrigins(LPCWSTR name, VARIANT *  
object, UINT32 originsCount, LPCWSTR * origins)
```

The host object will be accessible for this iframe only if the iframe's origin during access matches one of the origins which are passed. The provided origins will be normalized before comparing to the origin of the document. So the scheme name is made lower case, the host will be punycode decoded as appropriate, default port values will be removed, and so on. This means the origin's host may be punycode encoded or not and will match regardless. If list contains malformed origin the call will fail. The method can

be called multiple times in a row without calling RemoveHostObjectFromScript for the same object name. It will replace the previous object with the new object and new list of origins. List of origins will be treated as following:

1. empty list - call will succeed and object will be added for the iframe but it will not be exposed to any origin;
2. list with origins - during access to host object from iframe the origin will be checked that it belongs to this list;
3. list with "\*" element - host object will be available for iframe for all origins. We suggest not to use this feature without understanding security implications of giving access to host object from iframes with unknown origins.
4. list with "file://" element - host object will be available for iframes loaded via file protocol. Calling this method fails if it is called after the iframe is destroyed.

C++

```
wil::unique_variant remoteObjectAsVariant;
// It will throw if m_hostObject fails the QI, but because
it is our object
// it should always succeed.
m_hostObject.query_to<IDispatch>
(&remoteObjectAsVariant.pdispVal);
remoteObjectAsVariant.vt = VT_DISPATCH;

// Create list of origins which will be checked.
// iframe will have access to host object only if its origin
belongs
// to this list.
LPCWSTR origin = L"https://appassets.example/";

CHECK_FAILURE(webviewFrame-
>AddHostObjectToScriptWithOrigins(
    L"sample", &remoteObjectAsVariant, 1, &origin));
```

For more information about host objects navigate to  
[ICoreWebView2::AddHostObjectToScript](#).

## get\_Name

The value of iframe's window.name property.

```
public HRESULT get_Name(LPWSTR * name)
```

The default value equals to iframe html tag declaring it. You can access this property even if the iframe is destroyed.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## IsDestroyed

Check whether a frame is destroyed.

```
public HRESULT IsDestroyed(BOOL * destroyed)
```

Returns true during the Destroyed event.

## remove\_Destroyed

Remove an event handler previously added with add\_Destroyed.

```
public HRESULT remove_Destroyed(EventRegistrationToken token)
```

## remove\_NameChanged

Remove an event handler previously added with add\_NameChanged.

```
public HRESULT remove_NameChanged(EventRegistrationToken token)
```

## RemoveHostObjectFromScript

Remove the host object specified by the name so that it is no longer accessible from JavaScript code in the iframe.

```
public HRESULT RemoveHostObjectFromScript(LPCWSTR name)
```

While new access attempts are denied, if the object is already obtained by JavaScript code in the iframe, the JavaScript code continues to have access to that object. Calling this method for a name that is already removed or was never added fails. If the iframe is destroyed this method will return fail also.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Frame2

Article • 02/26/2024

```
interface ICoreWebView2Frame2
    : public ICoreWebView2Frame
```

A continuation of the ICoreWebView2Frame interface with navigation events, executing script and posting web messages.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">add_ContentLoading</a>	Add an event handler for the <code>ContentLoading</code> event.
<a href="#">add_DOMContentLoaded</a>	Add an event handler for the <code>DOMContentLoaded</code> event.
<a href="#">add_NavigationCompleted</a>	Add an event handler for the <code>NavigationCompleted</code> event.
<a href="#">add_NavigationStarting</a>	Add an event handler for the <code>NavigationStarting</code> event.
<a href="#">add_WebMessageReceived</a>	Add an event handler for the <code>WebMessageReceived</code> event.
<a href="#">ExecuteScript</a>	Run JavaScript code from the <code>javascript</code> parameter in the current frame.
<a href="#">PostWebMessageAsJson</a>	Posts the specified <code>webMessage</code> to the frame.
<a href="#">PostWebMessageAsString</a>	Posts a message that is a simple string rather than a JSON string representation of a JavaScript object.
<a href="#">remove_ContentLoading</a>	Remove an event handler previously added with <code>add_ContentLoading</code> .
<a href="#">remove_DOMContentLoaded</a>	Remove an event handler previously added with <code>add_DOMContentLoaded</code> .
<a href="#">remove_NavigationCompleted</a>	Remove an event handler previously added with <code>add_NavigationCompleted</code> .
<a href="#">remove_NavigationStarting</a>	Remove an event handler previously added with <code>add_NavigationStarting</code> .

Members	Descriptions
<a href="#">remove_WebMessageReceived</a>	Remove an event handler previously added with <code>add_WebMessageReceived</code> .

## Applies to

[Expand table](#)

Product	Introduced
WebView2 Win32	1.0.1108.44
WebView2 Win32 Prerelease	1.0.1133

## Members

### [add\\_ContentLoading](#)

Add an event handler for the `ContentLoading` event.

```
public HRESULT  
add_ContentLoading(ICoreWebView2FrameContentLoadingEventHandler *  
eventHandler, EventRegistrationToken * token)
```

`ContentLoading` triggers before any content is loaded, including scripts added with `AddScriptToExecuteOnDocumentCreated`. `ContentLoading` does not trigger if a same page navigation occurs (such as through `fragment` navigations or `history.pushState` navigations). This operation follows the `NavigationStarting` and precedes `NavigationCompleted` events.

### [add\\_DOMContentLoaded](#)

Add an event handler for the `DOMContentLoaded` event.

```
public HRESULT  
add_DOMContentLoaded(ICoreWebView2FrameDOMContentLoadedEventHandler *  
eventHandler, EventRegistrationToken * token)
```

`DOMContentLoaded` is raised when the iframe html document has been parsed. This aligns with the document's `DOMContentLoaded` event in html.

## **add\_NavigationCompleted**

Add an event handler for the `NavigationCompleted` event.

```
public HRESULT  
add_NavigationCompleted(ICoreWebView2FrameNavigationCompletedEventHandler  
r * eventHandler, EventRegistrationToken * token)
```

`NavigationCompleted` runs when the `CoreWebView2Frame` has completely loaded (concurrently when `body.onload` runs) or loading stopped with error.

## **add\_NavigationStarting**

Add an event handler for the `NavigationStarting` event.

```
public HRESULT  
add_NavigationStarting(ICoreWebView2FrameNavigationStartingEventHandler *  
eventHandler, EventRegistrationToken * token)
```

A frame navigation will raise a `NavigationStarting` event and a `CoreWebView2.FrameNavigationStarting` event. All of the `FrameNavigationStarting` event handlers for the current frame will be run before the `NavigationStarting` event handlers. All of the event handlers share a common `NavigationStartingEventArgs` object. Whichever event handler is last to change the `NavigationStartingEventArgs.Cancel` property will decide if the frame navigation will be cancelled. Redirects raise this event as well, and the navigation id is the same as the original one.

Navigations will be blocked until all `NavigationStarting` and `CoreWebView2.FrameNavigationStarting` event handlers return.

## **add\_WebMessageReceived**

Add an event handler for the `WebMessageReceived` event.

```
public HRESULT  
add_WebMessageReceived(ICoreWebView2FrameWebMessageReceivedEventHandler  
er * handler, EventRegistrationToken * token)
```

`WebMessageReceived` runs when the `ICoreWebView2Settings::IsWebMessageEnabled` setting is set and the frame document runs `window.chrome.webview.postMessage`. The `postMessage` function is `void postMessage(object)` where object is any object supported by JSON conversion.

#### HTML

```
        window.chrome.webview.addEventListener('message', arg => {
            if ("SetColor" in arg.data) {
                document.getElementById("colorable").style.color =
arg.data.SetColor;
            }
            if ("WindowBounds" in arg.data) {
                document.getElementById("window-bounds").value =
arg.data.WindowBounds;
            }
        });

        function SetTitleText() {
            let titleText = document.getElementById("title-text");
            window.chrome.webview.postMessage(`SetTitleText
${titleText.value}`);
        }
        function GetWindowBounds() {
            window.chrome.webview.postMessage("GetWindowBounds");
        }
    }
```

When the frame calls `postMessage`, the object parameter is converted to a JSON string and is posted asynchronously to the host process. This will result in the handlers `Invoke` method being called with the JSON string as its parameter.

#### C++

```
// Setup the web message received event handler before
navigating to
// ensure we don't miss any messages.
CHECK_FAILURE(webviewFrame2->add_WebMessageReceived(
    Microsoft::WRL::Callback<ICoreWebView2FrameWebMessageReceivedEventHandler>(
        [this](
            ICoreWebView2Frame* sender,
            ICoreWebView2WebMessageReceivedEventArgs* args) {
                wil::unique_cotaskmem_string uri;
                CHECK_FAILURE(args->get_Source(&uri));

                // Always validate that the origin of the message is
what you expect.
                if (uri.get() != m_sampleUri)
                {
                    // Ignore messages from untrusted sources.
                }
            }
        )
    )
);
```

```

        return S_OK;
    }
    wil::unique_cotaskmem_string messageRaw;
    HRESULT hr = args-
>TryGetWebMessageAsString(&messageRaw);
    if (hr == E_INVALIDARG)
    {
        // Was not a string message. Ignore.
        return S_OK;
    }
    // Any other problems are fatal.
    CHECK_FAILURE(hr);
    std::wstring message = messageRaw.get();

    if (message.compare(0, 13, L"SetTitleText ") == 0)
    {
        m_appWindow-
>SetDocumentTitle(message.substr(13).c_str());
    }
    else if (message.compare(L"GetWindowBounds") == 0)
    {
        RECT bounds = m_appWindow->GetWindowBounds();
        std::wstring reply = L"
{\\"WindowBounds\":\"Left:\" +
std::to_wstring(bounds.left) + L"\\"\\nTop:\" +
                                         std::to_wstring(bounds.top)
+ L"\\"\\nRight:\" +
std::to_wstring(bounds.right) + L"\\"\\nBottom:\" +
std::to_wstring(bounds.bottom) + L"\\"}";
        wil::com_ptr<ICoreWebView2Frame2> webviewFrame2;
        if (sender-
>QueryInterface(IID_PPV_ARGS(&webviewFrame2)) == S_OK)
        {
            CHECK_FAILURE(
                webviewFrame2-
>PostWebMessageAsJson(reply.c_str()));
        }
    }
    else
    {
        // Ignore unrecognized messages, but log for
        further investigation
        // since it suggests a mismatch between the web
        content and the host.
        OutputDebugString(
            std::wstring(L"Unexpected message from
frame:" + message).c_str());
    }
    return S_OK;
}
.Get(),
NULL));

```

## ExecuteScript

Run JavaScript code from the javascript parameter in the current frame.

```
public HRESULT ExecuteScript(LPCWSTR javaScript,  
ICoreWebView2ExecuteScriptCompletedHandler * handler)
```

The result of evaluating the provided JavaScript is passed to the completion handler. The result value is a JSON encoded string. If the result is undefined, contains a reference cycle, or otherwise is not able to be encoded into JSON, then the result is considered to be null, which is encoded in JSON as the string "null".

### ⓘ Note

A function that has no explicit return value returns undefined. If the script that was run throws an unhandled exception, then the result is also "null". This method is applied asynchronously. If the method is run before `ContentLoading`, the script will not be executed and the string "null" will be returned. This operation executes the script even if `ICoreWebView2Settings::IsScriptEnabled` is set to `FALSE`.

C++

```
wil::com_ptr<ICoreWebView2_4> webview2_4 =  
m_webView.try_query<ICoreWebView2_4>();  
if (webview2_4)  
{  
    CHECK_FAILURE(webview2_4->add_FrameCreated(  
        Callback<ICoreWebView2FrameCreatedEventHandler>(  
            [](ICoreWebView2* sender,  
                ICoreWebView2FrameCreatedEventArgs* args) -> HRESULT {  
                    wil::com_ptr<ICoreWebView2Frame> webviewFrame;  
                    CHECK_FAILURE(args->get_Frame(&webviewFrame));  
                    wil::com_ptr<ICoreWebView2Frame2> frame2 =  
                        webviewFrame.try_query<ICoreWebView2Frame2>();  
                    if (frame2)  
                    {  
                        frame2->add_DOMContentLoaded(  
                            Callback<ICoreWebView2FrameDOMContentLoadedEventHandler>(  
                                [](ICoreWebView2Frame* frame,  
                                    ICoreWebView2DOMContentLoadedEventArgs*  
                                    args) -> HRESULT {  
                                        wil::com_ptr<ICoreWebView2Frame2>  
                                        frame2;
```

```

frame-
>QueryInterface(IID_PPV_ARGS(&frame2));
frame2->ExecuteScript(
    LR"~(
let content =
content.style.color = 'blue';
content.textContent = "This text was
document.body.appendChild(content);
)~",
Callback<ICoreWebView2ExecuteScriptCompletedHandler>(
    [](HRESULT error, PCWSTR result)
-> HRESULT {
    error and result here if needed
    // Handle ExecuteScript
    callback parametr otherwise.
    // or pass nullptr as
    return S_OK;
})
    .Get());
return S_OK;
})
    .Get(),
    NULL);
}
return S_OK;
})
    .Get(),
&m_frameCreatedToken));
}

```

## PostWebMessageAsJson

Posts the specified webMessage to the frame.

```
public HRESULT PostWebMessageAsJson(LPCWSTR webMessageAsJson)
```

The frame receives the message by subscribing to the `message` event of the `window.chrome.webview` of the frame document.

C++

```
window.chrome.webview.addEventListener('message', handler)
window.chrome.webview.removeEventListener('message', handler)
```

The event args is an instances of `MessageEvent`. The `ICoreWebView2Settings::IsWebMessageEnabled` setting must be `TRUE` or the message will not be sent. The `data` property of the event args is the `webMessage` string parameter parsed as a JSON string into a JavaScript object. The `source` property of the event args is a reference to the `window.chrome.webview` object. For information about sending messages from the HTML document in the WebView to the host, navigate to [add\\_WebMessageReceived](#). The message is delivered asynchronously. If a navigation occurs before the message is posted to the page, the message is discarded.

## PostWebMessageAsString

Posts a message that is a simple string rather than a JSON string representation of a JavaScript object.

```
public HRESULT PostWebMessageAsString(LPCWSTR webMessageAsString)
```

This behaves in exactly the same manner as `PostWebMessageAsJson`, but the `data` property of the event args of the `window.chrome.webview` message is a string with the same value as `webMessageAsString`. Use this instead of `PostWebMessageAsJson` if you want to communicate using simple strings rather than JSON objects.

## remove\_ContentLoading

Remove an event handler previously added with `add_ContentLoading`.

```
public HRESULT remove_ContentLoading(EventRegistrationToken token)
```

## remove\_DOMContentLoaded

Remove an event handler previously added with `add_DOMContentLoaded`.

```
public HRESULT remove_DOMContentLoaded(EventRegistrationToken token)
```

## remove\_NavigationCompleted

Remove an event handler previously added with `add_NavigationCompleted`.

```
public HRESULT remove_NavigationCompleted(EventRegistrationToken token)
```

## remove\_NavigationStarting

Remove an event handler previously added with `add_NavigationStarting`.

```
public HRESULT remove_NavigationStarting(EventRegistrationToken token)
```

## remove\_WebMessageReceived

Remove an event handler previously added with `add_WebMessageReceived`.

```
public HRESULT remove_WebMessageReceived(EventRegistrationToken token)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Frame3

Article • 02/26/2024

```
interface ICoreWebView2Frame3
: public ICoreWebView2Frame2
```

This is an extension of the ICoreWebView2Frame interface that supports PermissionRequested.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">add_PermissionRequested</a>	Add an event handler for the <code>PermissionRequested</code> event.
<a href="#">remove_PermissionRequested</a>	Remove an event handler previously added with <code>add_PermissionRequested</code>

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1185.39
WebView2 Win32 Prerelease	1.0.1158

## Members

### [add\\_PermissionRequested](#)

Add an event handler for the `PermissionRequested` event.

```
public HRESULT
```

```
add_PermissionRequested(ICoreWebView2FramePermissionRequestedEventHandler
```

```
* handler, EventRegistrationToken * token)
```

`PermissionRequested` is raised when content in an iframe any of its descendant iframes requests permission to privileged resources.

This relates to the `PermissionRequested` event on the `CoreWebView2`. Both these events will be raised in the case of an iframe requesting permission. The `CoreWebView2Frame`'s event handlers will be invoked before the event handlers on the `CoreWebView2`. If the `Handled` property of the `PermissionRequestedEventArgs` is set to TRUE within the `CoreWebView2Frame` event handler, then the event will not be raised on the `CoreWebView2`, and it's event handlers will not be invoked.

In the case of nested iframes, the 'PermissionRequested' event will be raised from the top level iframe.

If a deferral is not taken on the event args, the subsequent scripts are blocked until the event handler returns. If a deferral is taken, the scripts are blocked until the deferral is completed.

C++

```
m_webView4 = m_webView.try_query<ICoreWebView2_4>();
if (m_webView4)
{
    CHECK_FAILURE(m_webView4->add_FrameCreated(
        Callback<ICoreWebView2FrameCreatedEventHandler>(
            [this](ICoreWebView2* sender,
ICoreWebView2FrameCreatedEventArgs* args) -> HRESULT {
                wil::com_ptr<ICoreWebView2Frame> webviewFrame;
                CHECK_FAILURE(args->get_Frame(&webviewFrame));

                m_frame3 = webviewFrame.try_query<ICoreWebView2Frame3>()
                    ;
                    if (m_frame3)
                    {
                        CHECK_FAILURE(m_frame3->add_PermissionRequested(
                            Callback<ICoreWebView2FramePermissionRequestedEventHandler>(
                                this,
                                &ScenarioIFrameDevicePermission::OnPermissionRequested
                                    ).Get(),
                                &m_PermissionRequestedToken));
                    }
                    else {
                        m_appWindow->RunAsync([]{ FeatureNotAvailable(); });
                    }
                    m_webView4->remove_FrameCreated(m_FrameCreatedToken);
                    return S_OK;
                }).Get(),
            );
```

```
    &m_FrameCreatedToken));
}
else
{
    FeatureNotAvailable();
}
```

C++

```
HRESULT ScenarioIFrameDevicePermission::OnPermissionRequested(
    ICoreWebView2Frame* sender, ICoreWebView2PermissionRequestedEventArgs2*
args)
{
    // If we set Handled to true, then we will not fire the
    PermissionRequested
    // event off of the CoreWebView2.
    args->put_Handled(true);

    // Obtain a deferral for the event so that the CoreWebView2
    // doesn't examine the properties we set on the event args until
    // after we call the Complete method asynchronously later.
    wil::com_ptr<ICoreWebView2Deferral> deferral;
    CHECK_FAILURE(args->GetDeferral(&deferral));

    // Do the rest asynchronously, to avoid calling MessageBox in an event
    handler.
    m_appWindow->RunAsync([this, deferral, args]
    {
        COREWEBVIEW2_PERMISSION_KIND kind =
        COREWEBVIEW2_PERMISSION_KIND_UNKNOWN_PERMISSION;
        BOOL userInitiated = FALSE;
        wil::unique_cotaskmem_string uri;
        CHECK_FAILURE(args->get_PermissionKind(&kind));
        CHECK_FAILURE(args->get_IsUserInitiated(&userInitiated));
        CHECK_FAILURE(args->get_Uri(&uri));

        COREWEBVIEW2_PERMISSION_STATE state;

        auto cached_key = std::make_tuple(std::wstring(uri.get()), kind,
userInitiated);
        auto cached_permission = m_cached_permissions.find(cached_key);
        if (cached_permission != m_cached_permissions.end())
        {
            state = (cached_permission->second
                ? COREWEBVIEW2_PERMISSION_STATE_ALLOW
                : COREWEBVIEW2_PERMISSION_STATE_DENY);
        }
        else
        {
            std::wstring message = L"An iframe has requested device
permission for ";
            message += PermissionKindToString(kind);
            message += L" to the website at ";
        }
    });
}
```

```

        message += uri.get();
        message += L"?\\n\\n";
        message += L"Do you want to grant permission?\\n";
        message += (userInitiated
            ? L"This request came from a user gesture."
            : L"This request did not come from a user gesture.");

    int response = MessageBox(
        nullptr, message.c_str(), L"Permission Request",
        MB_YESNOCANCEL | MB_ICONWARNING);
    switch (response) {
        case IDYES:
            m_cached_permissions[cached_key] = true;
            state = COREWEBVIEW2_PERMISSION_STATE_ALLOW;
            break;
        case IDNO:
            m_cached_permissions[cached_key] = false;
            state = COREWEBVIEW2_PERMISSION_STATE_DENY;
            break;
        default:
            state = COREWEBVIEW2_PERMISSION_STATE_DEFAULT;
            break;
    }
}

CHECK_FAILURE(args->put_State(state));
CHECK_FAILURE(deferral->Complete());
});

return S_OK;
}

```

## remove\_PermissionRequested

Remove an event handler previously added with `add_PermissionRequested`

---

`public HRESULT remove_PermissionRequested(EventRegistrationToken token)`

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Frame4

Article • 02/26/2024

```
interface ICoreWebView2Frame4
: public ICoreWebView2Frame3
```

This is an extension of the [ICoreWebView2Frame](#) interface that supports shared buffer based on file mapping.

## Summary

[+] [Expand table](#)

Members	Descriptions
<a href="#">PostSharedBufferToScript</a>	Share a shared buffer object with script of the iframe in the WebView.

## Applies to

[+] [Expand table](#)

Product	Introduced
WebView2 Win32	1.0.1661.34
WebView2 Win32 Prerelease	1.0.1671

## Members

### [PostSharedBufferToScript](#)

Share a shared buffer object with script of the iframe in the WebView.

```
public HRESULT PostSharedBufferToScript(ICoreWebView2SharedBuffer *
sharedBuffer, COREWEBVIEW2_SHARED_BUFFER_ACCESS access, LPCWSTR
additionalDataAsJson)
```

The script will receive a `sharedbufferreceived` event from `chrome.webview`. The event arg for that event will have the following methods and properties: `getBuffer()`: return an `ArrayBuffer` object with the backing content from the shared buffer. `additionalData`: an object as the result of parsing `additionalDataAsJson` as JSON string. This property will be `undefined` if `additionalDataAsJson` is `nullptr` or empty string. `source`: with a value set as `chrome.webview` object. If a string is provided as `additionalDataAsJson` but it is not a valid JSON string, the API will fail with `E_INVALIDARG`. If `access` is `COREWEBVIEW2_SHARED_BUFFER_ACCESS_READ_ONLY`, the script will only have read access to the buffer. If the script tries to modify the content in a read only buffer, it will cause an access violation in WebView renderer process and crash the renderer process. If the shared buffer is already closed, the API will fail with `RO_E_CLOSED`.

The script code should call `chrome.webview.releaseBuffer` with the shared buffer as the parameter to release underlying resources as soon as it does not need access to the shared buffer any more.

The application can post the same shared buffer object to multiple web pages or iframes, or post to the same web page or iframe multiple times. Each `PostSharedBufferToScript` will create a separate `ArrayBuffer` object with its own view of the memory and is separately released. The underlying shared memory will be released when all the views are released.

For example, if we want to send data to script for one time read only consumption.

C++

```
wil::com_ptr<ICoreWebView2Environment12> environment;
CHECK_FAILURE(
    m_appWindow->GetWebViewEnvironment()-
>QueryInterface(IID_PPV_ARGS(&environment)));

wil::com_ptr<ICoreWebView2SharedBuffer> sharedBuffer;
CHECK_FAILURE(environment->CreateSharedBuffer(bufferSize,
&sharedBuffer));
// Set data into the shared memory via IStream.
wil::com_ptr<IStream> stream;
CHECK_FAILURE(sharedBuffer->OpenStream(&stream));
CHECK_FAILURE(stream->Write(data, sizeof(data), nullptr));
PCWSTR additionalDataAsJson = L"\"myBufferType\":\"bufferType1\"";
if (fromFrame)
{
    m_webviewFrame4->PostSharedBufferToScript(
        sharedBuffer.get(),
        COREWEBVIEW2_SHARED_BUFFER_ACCESS_READ_ONLY,
        additionalDataAsJson);
}
else
```

```
{  
    m_webView17->PostSharedBufferToScript(  
        sharedBuffer.get(),  
        COREWEBVIEW2_SHARED_BUFFER_ACCESS_READ_ONLY,  
        additionalDataAsJson);  
}  
// Explicitly close the one time shared buffer to ensure that the  
resource is released.  
sharedBuffer->Close();
```

In the HTML document,

HTML

```
window.chrome.webview.addEventListener("sharedbufferreceived", e  
=> {  
    SharedBufferReceived(e);});
```

HTML

```
let readOnlySharedBuffer;  
function ShowReadOnlySharedBuffer() {  
    if (readOnlySharedBuffer) {  
        DisplaySharedBufferData(readOnlySharedBuffer);  
    } else {  
        // Post a web message to ask host to share the one time read  
only buffer.  
        chrome.webview.postMessage("RequestOneTimeShareBuffer");  
    }  
}  
  
function DisplaySharedBufferData(buffer) {  
    document.getElementById("shared-buffer-data").value =  
        new TextDecoder().decode(new Uint8Array(buffer));  
}  
  
function SharedBufferReceived(e) {  
    if (e.additionalData && e.additionalData.myBufferType ==  
"bufferType1") {  
        readOnlySharedBuffer = e.getBuffer();  
    } else {  
        sharedBuffer = e.getBuffer();  
    }  
    DisplaySharedBufferData(e.getBuffer());  
}  
  
function ReleaseBuffer(buffer) {  
    window.chrome.webview.releaseBuffer(buffer);  
}
```

Sharing a buffer to script has security risk. You should only share buffer with trusted site. If a buffer is shared to a untrusted site, possible sensitive information could be leaked. If a buffer is shared as modifiable by the script and the script modifies it in an unexpected way, it could result in corrupted data that might even crash the application.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Frame5

Article • 02/26/2024

```
interface ICoreWebView2Frame5
    : public ICoreWebView2Frame4
```

This is an extension of the [ICoreWebView2Frame](#) interface that provides the `FrameId` property.

## Summary

[+] [Expand table](#)

Members	Descriptions
<a href="#">get_FrameId</a>	The unique identifier of the current frame.

## Applies to

[+] [Expand table](#)

Product	Introduced
WebView2 Win32	1.0.2210.55
WebView2 Win32 Prerelease	1.0.2357

## Members

### [get\\_FrameId](#)

The unique identifier of the current frame.

```
public HRESULT get_FrameId(UINT32 * id)
```

It's the same kind of ID as with the `FrameId` in `CoreWebView2` and via `CoreWebView2FrameInfo`.

---

# Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2FrameCreatedEventArgs

Article • 02/26/2024

```
interface ICoreWebView2FrameCreatedEventArgs
: public IUnknown
```

Event args for the `FrameCreated` events.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_Frame</a>	The frame which was created.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.902.49
WebView2 Win32 Prerelease	1.0.902

## Members

### `get_Frame`

The frame which was created.

```
public HRESULT get_Frame(ICoreWebView2Frame ** frame)
```

# Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2FrameInfo

Article • 02/26/2024

```
interface ICoreWebView2FrameInfo
: public IUnknown
```

Provides a set of properties for a frame in the [ICoreWebView2](#).

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_Name</a>	The value of iframe's window.name property.
<a href="#">get_Source</a>	The URI of the document in the frame.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.818.41
WebView2 Win32 Prerelease	1.0.824

## Members

### [get\\_Name](#)

The value of iframe's window.name property.

```
public HRESULT get\_Name(LPWSTR * name)
```

The default value equals to iframe html tag declaring it, as in `<iframe name="frame-name" ...>`. The returned string is empty when the frame has no name attribute and no assigned value for window.name.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_Source

The URI of the document in the frame.

```
public HRESULT get_Source(LPWSTR * source)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2FrameInfo2

Article • 02/26/2024

```
interface ICoreWebView2FrameInfo2
: public ICoreWebView2FrameInfo
```

A continuation of the ICoreWebView2FrameInfo interface that provides `ParentFrameInfo`, `FrameId` and `FrameKind` properties.

## Summary

[ ] Expand table

Members	Descriptions
<a href="#">get_FrameId</a>	The unique identifier of the frame associated with the current <code>FrameInfo</code> .
<a href="#">get_FrameKind</a>	The frame kind of the frame.
<a href="#">get_ParentFrameInfo</a>	This parent frame's <code>FrameInfo</code> .

## Applies to

[ ] Expand table

Product	Introduced
WebView2 Win32	1.0.2210.55
WebView2 Win32 Prerelease	1.0.2357

## Members

### get\_FrameId

The unique identifier of the frame associated with the current `FrameInfo`.

```
public HRESULT get_FrameId(UINT32 * id)
```

It's the same kind of ID as with the `FrameId` in `CoreWebView2` and via `CoreWebView2Frame`. `FrameId` will only be populated (non-zero) when obtained calling `CoreWebView2ProcessExtendedInfo.AssociatedFrameInfos`. `CoreWebView2FrameInfo` objects obtained via `CoreWebView2.ProcessFailed` will always have an invalid frame Id 0. Note that this `FrameId` could be out of date as it's a snapshot. If there's WebView2 created or destroyed or `FrameCreated/FrameDestroyed` events after the asynchronous call `CoreWebView2Environment.GetProcessExtendedInfos` starts, you may want to call asynchronous method again to get the updated `FrameInfos`.

## get\_FrameKind

The frame kind of the frame.

```
public HRESULT get_FrameKind(COREWEBVIEW2_FRAME_KIND * kind)
```

`FrameKind` will only be populated when obtained calling `CoreWebView2ProcessExtendedInfo.AssociatedFrameInfos`. `CoreWebView2FrameInfo` objects obtained via `CoreWebView2.ProcessFailed` will always have the default value `COREWEBVIEW2_FRAME_KIND_UNKNOWN`. Note that this `FrameKind` could be out of date as it's a snapshot.

## get\_ParentFrameInfo

This parent frame's `FrameInfo`.

```
public HRESULT get_ParentFrameInfo(ICoreWebView2FrameInfo ** frameInfo)
```

`ParentFrameInfo` will only be populated when obtained via calling `CoreWebView2ProcessExtendedInfo.AssociatedFrameInfos`. `CoreWebView2FrameInfo` objects obtained via `CoreWebView2.ProcessFailed` will always have a `null ParentFrameInfo`. This property is also `null` for the main frame in the WebView2 which has no parent frame. Note that this `ParentFrameInfo` could be out of date as it's a snapshot.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2FrameInfoCollection

Article • 02/26/2024

```
interface ICoreWebView2FrameInfoCollection
: public IUnknown
```

Collection of `FrameInfo`s (name and source).

## Summary

[+] Expand table

Members	Descriptions
<a href="#">GetIterator</a>	Gets an iterator over the collection of <code>FrameInfo</code> s.

Used to list the affected frames' info when a frame-only render process failure occurs in the `ICoreWebView2`.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.818.41
WebView2 Win32 Prerelease	1.0.824

## Members

### GetIterator

Gets an iterator over the collection of `FrameInfo`s.

```
public HRESULT GetIterator(ICoreWebView2FrameInfoCollectionIterator ** iterator)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2FrameInfoCollectionIterator

Article • 02/26/2024

```
interface ICoreWebView2FrameInfoCollectionIterator
: public IUnknown
```

Iterator for a collection of [FrameInfos](#).

## Summary

[+] [Expand table](#)

Members	Descriptions
<a href="#">get_HasCurrent</a>	<code>TRUE</code> when the iterator has not run out of <a href="#">FrameInfos</a> .
<a href="#">GetCurrent</a>	Get the current ICoreWebView2FrameInfo of the iterator.
<a href="#">MoveNext</a>	Move the iterator to the next <a href="#">FrameInfo</a> in the collection.

For more info, see [ICoreWebView2ProcessFailedEventArgs2](#) and [ICoreWebView2FrameInfoCollection](#).

## Applies to

[+] [Expand table](#)

Product	Introduced
WebView2 Win32	1.0.818.41
WebView2 Win32 Prerelease	1.0.824

## Members

## get\_HasCurrent

`TRUE` when the iterator has not run out of `FrameInfo`s.

```
| public HRESULT get_HasCurrent(BOOL * hasCurrent)
```

If the collection over which the iterator is iterating is empty or if the iterator has gone past the end of the collection, then this is `FALSE`.

## GetCurrent

Get the current `ICoreWebView2FrameInfo` of the iterator.

```
| public HRESULT GetCurrent(ICoreWebView2FrameInfo ** frameInfo)
```

Returns `HRESULT_FROM_WIN32(ERROR_INVALID_INDEX)` if HasCurrent is `FALSE`.

## MoveNext

Move the iterator to the next `FrameInfo` in the collection.

```
| public HRESULT MoveNext(BOOL * hasNext)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2HttpHeadersCollectionIterator

Article • 02/26/2024

```
interface ICoreWebView2HttpHeadersCollectionIterator
: public IUnknown
```

Iterator for a collection of HTTP headers.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_HasCurrentHeader</a>	TRUE when the iterator has not run out of headers.
<a href="#">GetCurrentHeader</a>	Get the name and value of the current HTTP header of the iterator.
<a href="#">MoveNext</a>	Move the iterator to the next HTTP header in the collection.

For more information, navigate to [ICoreWebView2HttpRequestHeaders](#) and [ICoreWebView2HttpResponseHeaders](#).

C++

```
std::wstring RequestHeadersToJsonString(ICoreWebView2HttpRequestHeaders*
requestHeaders)
{
    wil::com_ptr<ICoreWebView2HttpHeadersCollectionIterator> iterator;
    CHECK_FAILURE(requestHeaders->GetIterator(&iterator));
    BOOL hasCurrent = FALSE;
    std::wstring result = L"[";
    while (SUCCEEDED(iterator->get_HasCurrentHeader(&hasCurrent)) &&
    hasCurrent)
    {
        wil::unique_cotaskmem_string name;
        wil::unique_cotaskmem_string value;
        CHECK_FAILURE(iterator->GetCurrentHeader(&name, &value));
        result += L"{" + name + L": " + value + L", ";
    }
    result += L"]";
}
```

```

        result += L"{"name": " + EncodeQuote(name.get())
        + L", \"value\": " + EncodeQuote(value.get()) + L"}";

    BOOL hasNext = FALSE;
    CHECK_FAILURE(iterator->MoveNext(&hasNext));
    if (hasNext)
    {
        result += L", ";
    }
}

return result + L"]";
}

```

## Applies to

[] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### get\_HasCurrentHeader

`TRUE` when the iterator has not run out of headers.

```
public HRESULT get_HasCurrentHeader(BOOL * hasCurrent)
```

If the collection over which the iterator is iterating is empty or if the iterator has gone past the end of the collection then this is `FALSE`.

### GetCurrentHeader

Get the name and value of the current HTTP header of the iterator.

```
public HRESULT GetCurrentHeader(LPWSTR * name, LPWSTR * value)
```

If the previous `MoveNext` operation set the `hasNext` parameter to `FALSE`, this method fails.

The caller must free the returned strings with `CoTaskMemFree`. See [API Conventions](#).

## MoveNext

Move the iterator to the next HTTP header in the collection.

```
public HRESULT MoveNext(BOOL * hasNext)
```

### ⓘ Note

If no more HTTP headers exist, the `hasNext` parameter is set to `FALSE`. After this occurs the `GetCurrentHeader` method fails.

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2HttpRequestHeaders

Article • 02/26/2024

```
interface ICoreWebView2HttpRequestHeaders
: public IUnknown
```

HTTP request headers.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Contains</a>	Verifies that the headers contain an entry that matches the header name.
<a href="#">GetHeader</a>	Gets the header value matching the name.
<a href="#">GetHeaders</a>	Gets the header value matching the name using an iterator.
<a href="#">GetIterator</a>	Gets an iterator over the collection of request headers.
<a href="#">RemoveHeader</a>	Removes header that matches the name.
<a href="#">SetHeader</a>	Adds or updates header that matches the name.

Used to inspect the HTTP request on `WebResourceRequested` event and `NavigationStarting` event.

### (!) Note

It is possible to modify the HTTP request from a `WebResourceRequested` event, but not from a `NavigationStarting` event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### Contains

Verifies that the headers contain an entry that matches the header name.

```
public HRESULT Contains(LPCWSTR name, BOOL * contains)
```

### GetHeader

Gets the header value matching the name.

```
public HRESULT GetHeader(LPCWSTR name, LPWSTR * value)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

### GetHeaders

Gets the header value matching the name using an iterator.

```
public HRESULT GetHeaders(LPCWSTR name,  
ICoreWebView2HttpHeadersCollectionIterator ** iterator)
```

### GetIterator

Gets an iterator over the collection of request headers.

```
public HRESULT GetIterator(ICoreWebView2HttpHeadersCollectionIterator **  
iterator)
```

### RemoveHeader

Removes header that matches the name.

```
public HRESULT RemoveHeader(LPCWSTR name)
```

## SetHeader

Adds or updates header that matches the name.

```
public HRESULT SetHeader(LPCWSTR name, LPCWSTR value)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2HttpResponseHeaders

Article • 02/26/2024

```
interface ICoreWebView2HttpResponseHeaders
: public IUnknown
```

HTTP response headers.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">AppendHeader</a>	Appends header line with name and value.
<a href="#">Contains</a>	Verifies that the headers contain entries that match the header name.
<a href="#">GetHeader</a>	Gets the first header value in the collection matching the name.
<a href="#">GetHeaders</a>	Gets the header values matching the name.
<a href="#">GetIterator</a>	Gets an iterator over the collection of entire response headers.

Used to construct a `WebResourceResponse` for the `WebResourceRequested` event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

## AppendHeader

Appends header line with name and value.

```
public HRESULT AppendHeader(LPCWSTR name, LPCWSTR value)
```

## Contains

Verifies that the headers contain entries that match the header name.

```
public HRESULT Contains(LPCWSTR name, BOOL * contains)
```

## GetHeader

Gets the first header value in the collection matching the name.

```
public HRESULT GetHeader(LPCWSTR name, LPWSTR * value)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## GetHeaders

Gets the header values matching the name.

```
public HRESULT GetHeaders(LPCWSTR name,  
ICoreWebView2HttpHeadersCollectionIterator ** iterator)
```

## GetIterator

Gets an iterator over the collection of entire response headers.

```
public HRESULT GetIterator(ICoreWebView2HttpHeadersCollectionIterator **  
iterator)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2LaunchingExternalUriSchemeEventArgs

Article • 02/26/2024

```
interface ICoreWebView2LaunchingExternalUriSchemeEventArgs
: public IUnknown
```

Event args for `LaunchingExternalUriScheme` event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_Cancel</a>	The event handler may set this property to <code>TRUE</code> to cancel the external URI scheme launch.
<a href="#">get_InitiatingOrigin</a>	The origin initiating the external URI scheme launch.
<a href="#">get_IsUserInitiated</a>	<code>TRUE</code> when the external URI scheme request was initiated through a user gesture.
<a href="#">get Uri</a>	The URI with the external URI scheme to be launched.
<a href="#">GetDeferral</a>	Returns an <code>ICoreWebView2Deferral</code> object.
<a href="#">put_Cancel</a>	Sets the <code>Cancel</code> property.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1823.32
WebView2 Win32 Prerelease	1.0.1905

# Members

## get\_Cancel

The event handler may set this property to `TRUE` to cancel the external URI scheme launch.

```
public HRESULT get_Cancel(BOOL * value)
```

If set to `TRUE`, the external URI scheme will not be launched, and the default dialog is not displayed. This property can be used to replace the normal handling of launching an external URI scheme. The initial value of the `Cancel` property is `FALSE`.

## get\_InitiatingOrigin

The origin initiating the external URI scheme launch.

```
public HRESULT get_InitiatingOrigin(LPWSTR * value)
```

The origin will be an empty string if the request is initiated by calling `CoreWebView2.Navigate` on the external URI scheme. If a script initiates the navigation, the `InitiatingOrigin` will be the top-level document's `Source`, for example, if `window.location` is set to "calculator://", the `InitiatingOrigin` will be set to calculator://. If the request is initiated from a child frame, the `InitiatingOrigin` will be the source of that child frame.

## get\_IsUserInitiated

`TRUE` when the external URI scheme request was initiated through a user gesture.

```
public HRESULT get_IsUserInitiated(BOOL * value)
```

### ⓘ Note

Being initiated through a user gesture does not mean that user intended to access the associated resource.

## get\_Uri

The URI with the external URI scheme to be launched.

```
public HRESULT get Uri(LPWSTR * value)
```

## GetDeferral

Returns an ICoreWebView2Deferral object.

```
public HRESULT GetDeferral(ICoreWebView2Deferral ** value)
```

Use this operation to complete the event at a later time.

## put\_Cancel

Sets the `Cancel` property.

```
public HRESULT put_Cancel(BOOL value)
```

---

## Feedback

Was this page helpful?



# interface

## ICoreWebView2MoveFocusRequestedEventArgs

Article • 02/26/2024

```
interface ICoreWebView2MoveFocusRequestedEventArgs
: public IUnknown
```

Event args for the `MoveFocusRequested` event.

## Summary

Expand table

Members	Descriptions
<code>get_Handled</code>	Indicates whether the event has been handled by the app.
<code>get_Reason</code>	The reason for WebView to run the <code>MoveFocusRequested</code> event.
<code>put_Handled</code>	Sets the <code>Handled</code> property.

## Applies to

Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### `get_Handled`

Indicates whether the event has been handled by the app.

```
public HRESULT get_Handled(BOOL * value)
```

If the app has moved the focus to another desired location, it should set the `Handled` property to `TRUE`. When the `Handled` property is `FALSE` after the event handler returns, default action is taken. The default action is to try to find the next tab stop child window in the app and try to move focus to that window. If no other window exists to move focus, focus is cycled within the web content of the `WebView`.

## get\_Reason

The reason for `WebView` to run the `MoveFocusRequested` event.

```
public HRESULT get_Reason(COREWEBVIEW2_MOVE_FOCUS_REASON * reason)
```

## put\_Handled

Sets the `Handled` property.

```
public HRESULT put_Handled(BOOL value)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2NavigationCompletedEventArgs

Article • 02/26/2024

```
interface ICoreWebView2NavigationCompletedEventArgs
: public IUnknown
```

Event args for the `NavigationCompleted` event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_IsSuccess</a>	<code>TRUE</code> when the navigation is successful.
<a href="#">get_NavigationId</a>	The ID of the navigation.
<a href="#">get_WebErrorStatus</a>	The error code if the navigation failed.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### get\_IsSuccess

`TRUE` when the navigation is successful.

```
public HRESULT get_IsSuccess(BOOL * isSuccess)
```

`FALSE` for a navigation that ended up in an error page (failures due to no network, DNS lookup failure, HTTP server responds with 4xx), but may also be `FALSE` for additional scenarios such as `window.stop()` run on navigated page. Note that WebView2 will report the navigation as 'unsuccessful' if the load for the navigation did not reach the expected completion for any reason. Such reasons include potentially catastrophic issues such network and certificate issues, but can also be the result of intended actions such as the app canceling a navigation or navigating away before the original navigation completed. Applications should not just rely on this flag, but also consider the reported WebErrorStatus to determine whether the failure is indeed catastrophic in their context. WebErrorStatuses that may indicate a non-catastrophic failure include:

- `COREWEBVIEW2_WEB_ERROR_STATUS_OPERATION_CANCELED`
- `COREWEBVIEW2_WEB_ERROR_STATUS_VALID_AUTHENTICATION_CREDENTIALS_REQUIRED`
- `COREWEBVIEW2_WEB_ERROR_STATUS_VALID_PROXY_AUTHENTICATION_REQUIRED`

## get\_NavigationId

The ID of the navigation.

```
public HRESULT get_NavigationId(UINT64 * navigationId)
```

## get\_WebErrorStatus

The error code if the navigation failed.

```
public HRESULT get_WebErrorStatus(COREWEBVIEW2_WEB_ERROR_STATUS * webErrorStatus)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2NavigationCompletedEventArgs2

Article • 02/26/2024

```
interface ICoreWebView2NavigationCompletedEventArgs2
    : public ICoreWebView2NavigationCompletedEventArgs
```

This is an interface for the StatusCode property of ICoreWebView2NavigationCompletedEventArgs.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_HttpStatusCode</a>	The HTTP status code of the navigation if it involved an HTTP request.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1245.22
WebView2 Win32 Prerelease	1.0.1248

## Members

### get\_HttpStatusCode

The HTTP status code of the navigation if it involved an HTTP request.

```
public HRESULT get_HttpStatusCode(int * http_status_code)
```

For instance, this will usually be 200 if the request was successful, 404 if a page was not found, etc. See <https://developer.mozilla.org/docs/Web/HTTP>Status> for a list of common status codes.

The `HttpStatusCode` property will be 0 in the following cases:

- The navigation did not involve an HTTP request. For instance, if it was a navigation to a file:// URL, or if it was a same-document navigation.
- The navigation failed before a response was received. For instance, if the hostname was not found, or if there was a network error.

In those cases, you can get more information from the `IsSuccess` and `WebErrorStatus` properties.

If the navigation receives a successful HTTP response, but the navigated page calls `window.stop()` before it finishes loading, then `HttpStatusCode` may contain a success code like 200, but `IsSuccess` will be FALSE and `WebErrorStatus` will be `COREWEBVIEW2_WEB_ERROR_STATUS_CONNECTION_ABORTED`.

Since WebView2 handles HTTP continuations and redirects automatically, it is unlikely for `HttpStatusCode` to ever be in the 1xx or 3xx ranges.

---

## Feedback

Was this page helpful?



# interface ICoreWebView2NavigationStartingEventArgs

Article • 02/26/2024

```
interface ICoreWebView2NavigationStartingEventArgs
: public IUnknown
```

Event args for the `NavigationStarting` event.

## Summary

[+] Expand table

Members	Descriptions
<code>get_Cancel</code>	The host may set this flag to cancel the navigation.
<code>get_IsRedirected</code>	<code>TRUE</code> when the navigation is redirected.
<code>get_IsUserInitiated</code>	<code>TRUE</code> when the navigation was initiated through a user gesture as opposed to programmatic navigation by page script.
<code>get_NavigationId</code>	The ID of the navigation.
<code>get_RequestHeaders</code>	The HTTP request headers for the navigation.
<code>get_Uri</code>	The uri of the requested navigation.
<code>put_Cancel</code>	Sets the <code>Cancel</code> property.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

# Members

## get\_Cancel

The host may set this flag to cancel the navigation.

```
public HRESULT get_Cancel(BOOL * cancel)
```

If set, the navigation is not longer present and the content of the current page is intact. For performance reasons, `GET` HTTP requests may happen, while the host is responding. You may set cookies and use part of a request for the navigation. Navigations to about schemes are cancellable, unless `msWebView2CancellableAboutNavigations` feature flag is disabled. Cancellation of frame navigation to `srcdoc` is not supported and will be ignored. A cancelled navigation will fire a `NavigationCompleted` event with a `WebErrorStatus` of `COREWEBVIEW2_WEB_ERROR_STATUS_OPERATION_CANCELED`.

## get\_IsRedirected

`TRUE` when the navigation is redirected.

```
public HRESULT get_IsRedirected(BOOL * isRedirected)
```

## get\_IsUserInitiated

`TRUE` when the navigation was initiated through a user gesture as opposed to programmatic navigation by page script.

```
public HRESULT get_IsUserInitiated(BOOL * isUserInitiated)
```

Navigations initiated via WebView2 APIs are considered as user initiated.

## get\_NavigationId

The ID of the navigation.

```
public HRESULT get_NavigationId(UINT64 * navigationId)
```

## get\_RequestHeaders

The HTTP request headers for the navigation.

```
public HRESULT get_RequestHeaders(ICoreWebView2HttpRequestHeaders **  
requestHeaders)
```

### ⓘ Note

You are not able to modify the HTTP request headers in a `NavigationStarting` event.

## get Uri

The uri of the requested navigation.

```
public HRESULT get Uri(LPWSTR * uri)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## put Cancel

Sets the `Cancel` property.

```
public HRESULT put Cancel(BOOL cancel)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2NavigationStartingEventArgs2

Article • 02/26/2024

```
interface ICoreWebView2NavigationStartingEventArgs2
: public ICoreWebView2NavigationStartingEventArgs
```

The AdditionalAllowedFrameAncestors API that enable developers to provide additional allowed frame ancestors.

## Summary

Expand table

Members	Descriptions
<a href="#">get_AdditionalAllowedFrameAncestors</a>	Get additional allowed frame ancestors set by the host app.
<a href="#">put_AdditionalAllowedFrameAncestors</a>	The app may set this property to allow a frame to be embedded by additional ancestors besides what is allowed by http header <a href="#">X-Frame-Options</a> and <a href="#">Content-Security-Policy frame-ancestors directive</a> .

## Applies to

Expand table

Product	Introduced
WebView2 Win32	1.0.1108.44
WebView2 Win32 Prerelease	1.0.1133

## Members

## get\_AdditionalAllowedFrameAncestors

Get additional allowed frame ancestors set by the host app.

```
public HRESULT get_AdditionalAllowedFrameAncestors(LPWSTR * value)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## put\_AdditionalAllowedFrameAncestors

The app may set this property to allow a frame to be embedded by additional ancestors besides what is allowed by http header [X-Frame-Options](#) and [Content-Security-Policy frame-ancestors directive](#).

```
public HRESULT put_AdditionalAllowedFrameAncestors(LPCWSTR value)
```

If set, a frame ancestor is allowed if it is allowed by the additional allowed frame ancestors or original http header from the site. Whether an ancestor is allowed by the additional allowed frame ancestors is done the same way as if the site provided it as the source list of the Content-Security-Policy frame-ancestors directive. For example, if

`https://example.com` and `https://www.example.com` are the origins of the top page and intermediate iframes that embed a nested site-embedding iframe, and you fully trust those origins, you should set this property to `https://example.com`

`https://www.example.com`. This property gives the app the ability to use iframe to embed sites that otherwise could not be embedded in an iframe in trusted app pages. This could potentially subject the embedded sites to [Clickjacking](#) attack from the code running in the embedding web page. Therefore, you should only set this property with origins of fully trusted embedding page and any intermediate iframes. Whenever possible, you should use the list of specific origins of the top and intermediate frames instead of wildcard characters for this property. This API is to provide limited support for app scenarios that used to be supported by `<webview>` element in other solutions like JavaScript UWP apps and Electron. You should limit the usage of this property to trusted pages, and specific navigation target url, by checking the `source` of the WebView2, and `Uri` of the event args.

This property is ignored for top level document navigation.

C++

```
const std::wstring myTrustedSite = L"https://appassets.example";
const std::wstring siteToEmbed = L"https://www.microsoft.com";
```

```

// The trusted page is using <iframe name="my_site_embedding_frame">
// element to embed other sites.
const std::wstring siteEmbeddingFrameName = L"my_site_embedding_frame";

bool AreSitesSame(PCWSTR url1, PCWSTR url2)
{
    wil::com_ptr<IUri> uri1;
    CHECK_FAILURE(CreateUri(url1, Uri_CREATE_CANONICALIZE, 0, &uri1));
    DWORD scheme1 = -1;
    DWORD port1 = 0;
    wil::unique_bstr host1;
    CHECK_FAILURE(uri1->GetScheme(&scheme1));
    CHECK_FAILURE(uri1->GetHost(&host1));
    CHECK_FAILURE(uri1->GetPort(&port1));
    wil::com_ptr<IUri> uri2;
    CHECK_FAILURE(CreateUri(url2, Uri_CREATE_CANONICALIZE, 0, &uri2));
    DWORD scheme2 = -1;
    DWORD port2 = 0;
    wil::unique_bstr host2;
    CHECK_FAILURE(uri2->GetScheme(&scheme2));
    CHECK_FAILURE(uri2->GetHost(&host2));
    CHECK_FAILURE(uri2->GetPort(&port2));
    return (scheme1 == scheme2) && (port1 == port2) && (wcscmp(host1.get(),
host2.get()) == 0);
}

// App specific logic to decide whether the page is fully trusted.
bool IsAppContentUri(PCWSTR pageUrl)
{
    return AreSitesSame(pageUrl, myTrustedSite.c_str());
}

// App specific logic to decide whether a site is the one it wants to embed.
bool IsTargetSite(PCWSTR siteUrl)
{
    return AreSitesSame(siteUrl, siteToEmbed.c_str());
}

```

C++

```

        // Set up the event listeners to handle site embedding scenario. The
code will take effect
        // when the site embedding page is navigated to and the embedding
iframe navigates to the
        // site that we want to embed.

        // This part is trying to scope the API usage to the specific
scenario where we are
        // embedding a site. The result is recorded in
m_siteEmbeddingIFrameCount.
        CHECK_FAILURE(webview2_4->add_FrameCreated(
            Callback<ICoreWebView2FrameCreatedEventHandler>(
                [this](

```

```

        ICoreWebView2* sender,
ICoreWebView2FrameCreatedEventArgs* args) -> HRESULT
{
    wil::com_ptr<ICoreWebView2Frame> webviewFrame;
CHECK_FAILURE(args->get_Frame(&webviewFrame));
wil::unique_cotaskmem_string page_url;
CHECK_FAILURE(m_webView->get_Source(&page_url));
// IsAppContentUri verifies that page_url belongs to
the app.

    if (IsAppContentUri(page_url.get()))
    {
        // We are on trusted pages. Now check whether it is
the iframe we plan
        // to embed arbitrary sites.
        wil::unique_cotaskmem_string frame_name;
CHECK_FAILURE(webviewFrame->get_Name(&frame_name));
if (siteEmbeddingFrameName == frame_name.get())
{
    ++m_siteEmbeddingIFrameCount;
CHECK_FAILURE(webviewFrame->add_Destroyed(
    Microsoft::WRL::Callback<
        ICoreWebView2FrameDestroyedEventHandler>
(
    [this](
        ICoreWebView2Frame* sender,
IUnknown* args) -> HRESULT
{
    --m_siteEmbeddingIFrameCount;
    return S_OK;
}
    .Get(),
    nullptr));
}
}
return S_OK;
})
.Get(),
nullptr));
}

// Using FrameNavigationStarting event instead of NavigationStarting
event of
// CoreWebViewFrame to cover all possible nested iframes inside the
embedded site as
// CoreWebViewFrame object currently only support first level
iframes in the top page.
CHECK_FAILURE(m_webView->add_FrameNavigationStarting(
Microsoft::WRL::Callback<ICoreWebView2NavigationStartingEventHandler>(
    [this](ICoreWebView2* sender,
ICoreWebView2NavigationStartingEventArgs* args)
-> HRESULT
{
    if (m_siteEmbeddingIFrameCount > 0)
{
        wil::unique_cotaskmem_string navigationTargetUri;

```

```
        CHECK_FAILURE(args->get_Uri(&navigationTargetUri));
        if (IsTargetSite(navigationTargetUri.get()))
        {

wil::com_ptr<ICoreWebView2NavigationStartingEventArgs2>
            navigationStartArgs;
            if (SUCCEEDED(args-
>QueryInterface(IID_PPV_ARGS(&navigationStartArgs))))
            {
                navigationStartArgs-
>put_AdditionalAllowedFrameAncestors(
                    myTrustedSite.c_str());
            }
        }
    }
    return S_OK;
})
.Get(),
nullptr));
```

---

## Feedback

Was this page helpful?



# interface

## ICoreWebView2NavigationStartingEventArgs3

Article • 02/26/2024

```
interface ICoreWebView2NavigationStartingEventArgs3
    : public ICoreWebView2NavigationStartingEventArgs2
```

The NavigationKind API that enables developers to get more information about navigation type.

## Summary

[ ] Expand table

Members	Descriptions
<a href="#">get_NavigationKind</a>	Get the navigation kind of this navigation.

## Applies to

[ ] Expand table

Product	Introduced
WebView2 Win32	1.0.1901.177
WebView2 Win32 Prerelease	1.0.1905

## Members

### [get\\_NavigationKind](#)

Get the navigation kind of this navigation.

```
public HRESULT get_NavigationKind(COREWEBVIEW2_NAVIGATION_KIND *  
navigation_kind)
```

---

## Feedback

Was this page helpful?



# interface

## ICoreWebView2NewWindowRequestedEventArgs

Article • 02/26/2024

```
interface ICoreWebView2NewWindowRequestedEventArgs
: public IUnknown
```

Event args for the `NewWindowRequested` event.

## Summary

[ ] Expand table

Members	Descriptions
<code>get_Handled</code>	Gets whether the <code>NewWindowRequested</code> event is handled by host.
<code>get_IsUserInitiated</code>	<code>TRUE</code> when the new window request was initiated through a user gesture.
<code>get_NewWindow</code>	Gets the new window.
<code>get_Uri</code>	The target uri of the new window requested.
<code>get_WindowFeatures</code>	Window features specified by the <code>window.open</code> .
<code>GetDeferral</code>	Obtain an <code>ICoreWebView2Deferral</code> object and put the event into a deferred state.
<code>put_Handled</code>	Sets whether the <code>NewWindowRequested</code> event is handled by host.
<code>put_NewWindow</code>	Sets a <code>CoreWebView2</code> as a result of the <code>NewWindowRequested</code> event.

The event is run when content inside webview requested to open a new window (through `window.open()` and so on).

## Applies to

[ ] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### get\_Handled

Gets whether the `NewWindowRequested` event is handled by host.

```
public HRESULT get_Handled(BOOL * handled)
```

### get\_IsUserInitiated

`TRUE` when the new window request was initiated through a user gesture.

```
public HRESULT get_IsUserInitiated(BOOL * isUserInitiated)
```

Examples of user initiated requests are:

- Selecting an anchor tag with target
- Programmatic window open from a script that directly run as a result of user interaction such as via onclick handlers.

Non-user initiated requests are programmatic window opens from a script that are not directly triggered by user interaction, such as those that run while loading a new page or via timers. The Microsoft Edge popup blocker is disabled for WebView so the app is able to use this flag to block non-user initiated popups.

### get\_NewWindow

Gets the new window.

```
public HRESULT get_NewWindow(ICoreWebView2 ** newWindow)
```

### get\_Uri

The target uri of the new window requested.

```
public HRESULT get Uri(LPWSTR * uri)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_WindowFeatures

Window features specified by the `window.open`.

```
public HRESULT get_WindowFeatures(ICoreWebView2WindowFeatures ** value)
```

The features should be considered for positioning and sizing of new webview windows.

## GetDeferral

Obtain an `ICoreWebView2Deferral` object and put the event into a deferred state.

```
public HRESULT GetDeferral(ICoreWebView2Deferral ** deferral)
```

Use the `ICoreWebView2Deferral` object to complete the window open request at a later time. While this event is deferred the opener window returns a `WindowProxy` to an un-navigated window, which navigates when the deferral is complete.

## put\_Handled

Sets whether the `NewWindowRequested` event is handled by host.

```
public HRESULT put_Handled(BOOL handled)
```

If this is `FALSE` and no `NewWindow` is set, the WebView opens a popup window and it returns as opened `WindowProxy`. If set to `TRUE` and no `NewWindow` is set for `window.open`, the opened `WindowProxy` is for an testing window object and no window loads. The default value is `FALSE`.

## put\_NewWindow

Sets a `CoreWebView2` as a result of the `NewWindowRequested` event.

```
public HRESULT put_NewWindow(ICoreWebView2 * newWindow)
```

Provides a WebView as the target for a `window.open()` from inside the requesting WebView. If this is set, the top-level window of this WebView is returned as the opened [WindowProxy](#) to the opener script. If this is not set, then `Handled` is checked to determine behavior for NewWindowRequested event. CoreWebView2 provided in the `NewWindow` property must be on the same Environment as the opener WebView and should not have been navigated previously. Don't use methods that cause navigation or interact with the DOM on this CoreWebView2 until the target content has loaded. Setting event handlers, changing Settings properties, or other methods are fine to call. Changes to settings should be made before `put_NewWindow` is called to ensure that those settings take effect for the newly setup WebView. Once the NewWindow is set the underlying web contents of this CoreWebView2 will be replaced and navigated as appropriate for the new window. After setting new window it cannot be changed and error will be return otherwise.

The methods which should affect the new web contents like `AddScriptToExecuteOnDocumentCreated` has to be called and completed before setting `NewWindow`. Other methods which should affect the new web contents like `add_WebResourceRequested` have to be called after setting `NewWindow`. It is best not to use `RemoveScriptToExecuteOnDocumentCreated` before setting `NewWindow`, otherwise it may not work for later added scripts.

The new WebView must have the same profile as the opener WebView.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2NewWindowRequestedEventArgs2

Article • 02/26/2024

```
interface ICoreWebView2NewWindowRequestedEventArgs2
: public ICoreWebView2NewWindowRequestedEventArgs
```

This is a continuation of the ICoreWebView2NewWindowRequestedEventArgs interface.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_Name</a>	Gets the name of the new window.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.992.28
WebView2 Win32 Prerelease	1.0.1010

## Members

### get\_Name

Gets the name of the new window.

```
public HRESULT get_Name(LPWSTR * value)
```

This window can be created via `window.open(url, windowName)`, where the `windowName` parameter corresponds to `Name` property. If no `windowName` is passed to `window.open`, then the `Name` property will be set to an empty string. Additionally, if window is opened through other means, such as `<a target="windowName">...</a>` or `<iframe name="windowName">...</iframe>`, then the `Name` property will be set accordingly. In the case of `target=_blank`, the `Name` property will be an empty string. Opening a window via `ctrl+clicking` a link would result in the `Name` property being set to an empty string.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2NewWindowRequestedEventArgs3

Article • 02/26/2024

```
interface ICoreWebView2NewWindowRequestedEventArgs3
: public ICoreWebView2NewWindowRequestedEventArgs2
```

This is a continuation of the ICoreWebView2NewWindowRequestedEventArgs interface.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_OriginalSourceFrameInfo</a>	The frame info of the frame where the new window request originated.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.2151.40
WebView2 Win32 Prerelease	1.0.2106

## Members

### [get\\_OriginalSourceFrameInfo](#)

The frame info of the frame where the new window request originated.

```
public HRESULT get_OriginalSourceFrameInfo(ICoreWebView2FrameInfo **  
frameInfo)
```

The `OriginalSourceFrameInfo` is a snapshot of frame information at the time when the new window was requested. See `ICoreWebView2FrameInfo` for details on frame properties.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2ObjectCollectionView

Article • 02/26/2024

```
interface ICoreWebView2ObjectCollectionView
: public IUnknown
```

Read-only collection of generic objects.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_Count</a>	Gets the number of items in the collection.
<a href="#">GetValueAtIndex</a>	Gets the object at the specified index.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1774.30
WebView2 Win32 Prerelease	1.0.1777

## Members

### [get\\_Count](#)

Gets the number of items in the collection.

```
public HRESULT get_Count(UINT32 * value)
```

## GetValueAtIndex

Gets the object at the specified index.

```
public HRESULT GetValueAtIndex(UINT32 index, IUnknown ** value)
```

Cast the object to the native type to access its specific properties.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2PermissionRequestedEventArgs

Article • 02/26/2024

```
interface ICoreWebView2PermissionRequestedEventArgs
: public IUnknown
```

Event args for the `PermissionRequested` event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_IsUserInitiated</a>	<code>TRUE</code> when the permission request was initiated through a user gesture.
<a href="#">get_PermissionKind</a>	The type of the permission that is requested.
<a href="#">get_State</a>	The status of a permission request, (for example is the request is granted).
<a href="#">get_Uri</a>	The origin of the web content that requests the permission.
<a href="#">GetDeferral</a>	Gets an <code>ICoreWebView2Deferral</code> object.
<a href="#">put_State</a>	Sets the <code>State</code> property.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

# Members

## get\_IsUserInitiated

`TRUE` when the permission request was initiated through a user gesture.

```
public HRESULT get_IsUserInitiated(BOOL * isUserInitiated)
```

### ⓘ Note

Being initiated through a user gesture does not mean that user intended to access the associated resource.

## get\_PermissionKind

The type of the permission that is requested.

```
public HRESULT get_PermissionKind(COREWEBVIEW2_PERMISSION_KIND * permissionKind)
```

## get\_State

The status of a permission request, (for example is the request is granted).

```
public HRESULT get_State(COREWEBVIEW2_PERMISSION_STATE * state)
```

The default value is `COREWEBVIEW2_PERMISSION_STATE_DEFAULT`.

## get Uri

The origin of the web content that requests the permission.

```
public HRESULT get Uri(LPWSTR * uri)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## GetDeferral

Gets an `ICoreWebView2Deferral` object.

```
public HRESULT GetDeferral(ICoreWebView2Deferral ** deferral)
```

Use the deferral object to make the permission decision at a later time. The deferral only applies to the current request, and does not prevent the `PermissionRequested` event from getting raised for new requests. However, for some permission kinds the WebView will avoid creating a new request if there is a pending request of the same kind.

## put\_State

Sets the `state` property.

```
public HRESULT put_State(COREWEBVIEW2_PERMISSION_STATE state)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2PermissionRequestedEventArgs2

Article • 02/26/2024

```
interface ICoreWebView2PermissionRequestedEventArgs2
: public ICoreWebView2PermissionRequestedEventArgs
```

This is a continuation of the ICoreWebView2PermissionRequestedEventArgs interface.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_Handled</a>	By default, both the <code>PermissionRequested</code> event handlers on the <code>CoreWebView2Frame</code> and the <code>CoreWebView2</code> will be invoked, with the <code>CoreWebView2Frame</code> event handlers invoked first.
<a href="#">put_Handled</a>	Sets the <code>Handled</code> property.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1185.39
WebView2 Win32 Prerelease	1.0.1158

## Members

### get\_Handled

By default, both the `PermissionRequested` event handlers on the `CoreWebView2Frame` and the `CoreWebView2` will be invoked, with the `CoreWebView2Frame` event handlers invoked first.

```
| public HRESULT get_Handled(BOOL * handled)
```

The host may set this flag to `TRUE` within the `CoreWebView2Frame` event handlers to prevent the remaining `CoreWebView2` event handlers from being invoked.

If a deferral is taken on the event args, then you must synchronously set `Handled` to `TRUE` prior to taking your deferral to prevent the `CoreWebView2`'s event handlers from being invoked.

## **put\_Handled**

Sets the `Handled` property.

```
| public HRESULT put_Handled(BOOL handled)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2PermissionRequestedEventArgs3

Article • 02/26/2024

```
interface ICoreWebView2PermissionRequestedEventArgs3
: public ICoreWebView2PermissionRequestedEventArgs2
```

This is a continuation of the ICoreWebView2PermissionRequestedEventArgs2 interface.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_SavesInProfile</a>	The permission state set from the <code>PermissionRequested</code> event is saved in the profile by default; it persists across sessions and becomes the new default behavior for future <code>PermissionRequested</code> events.
<a href="#">put_SavesInProfile</a>	Sets the <code>SavesInProfile</code> property.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1661.34
WebView2 Win32 Prerelease	1.0.1671

## Members

### [get\\_SavesInProfile](#)

The permission state set from the `PermissionRequested` event is saved in the profile by default; it persists across sessions and becomes the new default behavior for future `PermissionRequested` events.

```
| public HRESULT get_SavesInProfile(BOOL * value)
```

Browser heuristics can affect whether the event continues to be raised when the state is saved in the profile. Set the `SavesInProfile` property to `FALSE` to not persist the state beyond the current request, and to continue to receive `PermissionRequested` events for this origin and permission kind.

## put\_SavesInProfile

Sets the `SavesInProfile` property.

```
| public HRESULT put_SavesInProfile(BOOL value)
```

---

## Feedback

Was this page helpful?



# interface ICoreWebView2PermissionSetting

Article • 02/26/2024

```
interface ICoreWebView2PermissionSetting
: public IUnknown
```

Provides a set of properties for a permission setting.

## Summary

[ ] Expand table

Members	Descriptions
<a href="#">get_PermissionKind</a>	The kind of the permission setting.
<a href="#">get_PermissionOrigin</a>	The origin of the permission setting.
<a href="#">get_PermissionState</a>	The state of the permission setting.

## Applies to

[ ] Expand table

Product	Introduced
WebView2 Win32	1.0.1661.34
WebView2 Win32 Prerelease	1.0.1671

## Members

### get\_PermissionKind

The kind of the permission setting.

```
public HRESULT get_PermissionKind(COREWEBVIEW2_PERMISSION_KIND * value)
```

See `COREWEBVIEW2_PERMISSION_KIND` for more details.

## get\_PermissionOrigin

The origin of the permission setting.

```
public HRESULT get_PermissionOrigin(LPWSTR * value)
```

## get\_PermissionState

The state of the permission setting.

```
public HRESULT get_PermissionState(COREWEBVIEW2_PERMISSION_STATE * value)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2PermissionSettingCollectionView

Article • 02/26/2024

```
interface ICoreWebView2PermissionSettingCollectionView
: public IUnknown
```

Read-only collection of `PermissionSetting`s (origin, kind, and state).

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_Count</a>	The number of ICoreWebView2PermissionSettings in the collection.
<a href="#">GetValueAtIndex</a>	Gets the ICoreWebView2PermissionSetting at the specified index.

Used to list the nondefault permission settings on the profile that are persisted across sessions.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1661.34
WebView2 Win32 Prerelease	1.0.1671

## Members

### `get_Count`

The number of ICoreWebView2PermissionSettings in the collection.

```
public HRESULT get_Count(UINT32 * value)
```

## GetValueAtIndex

Gets the ICoreWebView2PermissionSetting at the specified index.

```
public HRESULT GetValueAtIndex(UINT32 index, ICoreWebView2PermissionSetting  
** permissionSetting)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2PointerInfo

Article • 02/26/2024

```
interface ICoreWebView2PointerInfo
: public IUnknown
```

This mostly represents a combined win32  
POINTER\_INFO/POINTER\_TOUCH\_INFO/POINTER\_PEN\_INFO object.

## Summary

[ ] Expand table

Members	Descriptions
<a href="#">get_ButtonChangeKind</a>	Get the ButtonChangeKind of the pointer event.
<a href="#">get_DisplayRect</a>	Get the DisplayRect of the sourceDevice property of the POINTER_INFO struct as defined in the Windows SDK (winuser.h).
<a href="#">get_FrameId</a>	Get the FrameID of the pointer event.
<a href="#">get_HimetricLocation</a>	Get the HimetricLocation of the pointer event.
<a href="#">get_HimetricLocationRaw</a>	Get the HimetricLocationRaw of the pointer event.
<a href="#">get_HistoryCount</a>	Get the HistoryCount of the pointer event.
<a href="#">get_InputData</a>	Get the InputData of the pointer event.
<a href="#">get_KeyStates</a>	Get the KeyStates of the pointer event.
<a href="#">get_PenFlags</a>	Get the PenFlags of the pointer event.
<a href="#">get_PenMask</a>	Get the PenMask of the pointer event.
<a href="#">get_PenPressure</a>	Get the PenPressure of the pointer event.
<a href="#">get_PenRotation</a>	Get the PenRotation of the pointer event.
<a href="#">get_PenTiltX</a>	Get the PenTiltX of the pointer event.
<a href="#">get_PenTiltY</a>	Get the PenTiltY of the pointer event.
<a href="#">get_PerformanceCount</a>	Get the PerformanceCount of the pointer event.

Members	Descriptions
<a href="#">get_PixelLocation</a>	Get the PixelLocation of the pointer event.
<a href="#">get_PixelLocationRaw</a>	Get the PixelLocationRaw of the pointer event.
<a href="#">get_PointerDeviceRect</a>	Get the PointerDeviceRect of the sourceDevice property of the <code>POINTER_INFO</code> struct as defined in the Windows SDK ( <code>winuser.h</code> ).
<a href="#">get_PointerFlags</a>	Get the PointerFlags of the pointer event.
<a href="#">get_PointerId</a>	Get the PointerId of the pointer event.
<a href="#">get_PointerKind</a>	Get the PointerKind of the pointer event.
<a href="#">get_Time</a>	Get the Time of the pointer event.
<a href="#">get_TouchContact</a>	Get the TouchContact of the pointer event.
<a href="#">get_TouchContactRaw</a>	Get the TouchContactRaw of the pointer event.
<a href="#">get_TouchFlags</a>	Get the TouchFlags of the pointer event.
<a href="#">get_TouchMask</a>	Get the TouchMask of the pointer event.
<a href="#">get_TouchOrientation</a>	Get the TouchOrientation of the pointer event.
<a href="#">get_TouchPressure</a>	Get the TouchPressure of the pointer event.
<a href="#">put_ButtonChangeKind</a>	Set the ButtonChangeKind of the pointer event.
<a href="#">put_DisplayRect</a>	Set the DisplayRect of the sourceDevice property of the <code>POINTER_INFO</code> struct as defined in the Windows SDK ( <code>winuser.h</code> ).
<a href="#">put_FrameId</a>	Set the FrameID of the pointer event.
<a href="#">put_HimetricLocation</a>	Set the HimetricLocation of the pointer event.
<a href="#">put_HimetricLocationRaw</a>	Set the HimetricLocationRaw of the pointer event.
<a href="#">put_HistoryCount</a>	Set the HistoryCount of the pointer event.
<a href="#">put_InputData</a>	Set the InputData of the pointer event.
<a href="#">put_KeyStates</a>	Set the KeyStates of the pointer event.
<a href="#">put_PenFlags</a>	Set the PenFlags of the pointer event.
<a href="#">put_PenMask</a>	Set the PenMask of the pointer event.
<a href="#">put_PenPressure</a>	Set the PenPressure of the pointer event.
<a href="#">put_PenRotation</a>	Set the PenRotation of the pointer event.

Members	Descriptions
<a href="#">put_PenTiltX</a>	Set the PenTiltX of the pointer event.
<a href="#">put_PenTiltY</a>	Set the PenTiltY of the pointer event.
<a href="#">put_PerformanceCount</a>	Set the PerformanceCount of the pointer event.
<a href="#">put_PixelLocation</a>	Set the PixelLocation of the pointer event.
<a href="#">put_PixelLocationRaw</a>	Set the PixelLocationRaw of the pointer event.
<a href="#">put_PointerDeviceRect</a>	Set the PointerDeviceRect of the sourceDevice property of the <code>POINTER_INFO</code> struct as defined in the Windows SDK ( <code>winuser.h</code> ).
<a href="#">put_PointerFlags</a>	Set the PointerFlags of the pointer event.
<a href="#">put_PointerId</a>	Set the PointerId of the pointer event.
<a href="#">put_PointerKind</a>	Set the PointerKind of the pointer event.
<a href="#">put_Time</a>	Set the Time of the pointer event.
<a href="#">put_TouchContact</a>	Set the TouchContact of the pointer event.
<a href="#">put_TouchContactRaw</a>	Set the TouchContactRaw of the pointer event.
<a href="#">put_TouchFlags</a>	Set the TouchFlags of the pointer event.
<a href="#">put_TouchMask</a>	Set the TouchMask of the pointer event.
<a href="#">put_TouchOrientation</a>	Set the TouchOrientation of the pointer event.
<a href="#">put_TouchPressure</a>	Set the TouchPressure of the pointer event.

It takes fields from all three and excludes some win32 specific data types like `HWND` and `HANDLE`. Note, `sourceDevice` is taken out but we expect the `PointerDeviceRect` and `DisplayRect` to cover the existing use cases of `sourceDevice`. Another big difference is that any of the point or rect locations are expected to be in WebView physical coordinates. That is, coordinates relative to the WebView and no DPI scaling applied.

## Applies to

 Expand table

Product	Introduced
WebView2 Win32	1.0.774.44

Product	Introduced
WebView2 Win32 Prerelease	1.0.790

## Members

### get\_ButtonChangeKind

Get the ButtonChangeKind of the pointer event.

```
public HRESULT get_ButtonChangeKind(INT32 * buttonChangeKind)
```

This corresponds to the ButtonChangeKind property of the POINTER\_INFO struct. The values are defined by the POINTER\_BUTTON\_CHANGE\_KIND enum in the Windows SDK (winuser.h).

### get\_DisplayRect

Get the DisplayRect of the sourceDevice property of the POINTER\_INFO struct as defined in the Windows SDK (winuser.h).

```
public HRESULT get_DisplayRect(RECT * displayRect)
```

### get\_Frameld

Get the Frameld of the pointer event.

```
public HRESULT get_Frameld(UINT32 * frameld)
```

This corresponds to the frameld property of the POINTER\_INFO struct as defined in the Windows SDK (winuser.h).

### get\_HimetricLocation

Get the HimetricLocation of the pointer event.

```
public HRESULT get_HimetricLocation(POINT * himetricLocation)
```

This corresponds to the ptHimetricLocation property of the POINTER\_INFO struct as defined in the Windows SDK (winuser.h).

## **get\_HimetricLocationRaw**

Get the HimetricLocationRaw of the pointer event.

```
public HRESULT get_HimetricLocationRaw(POINT * himetricLocationRaw)
```

This corresponds to the ptHimetricLocationRaw property of the POINTER\_INFO struct as defined in the Windows SDK (winuser.h).

## **get\_HistoryCount**

Get the HistoryCount of the pointer event.

```
public HRESULT get_HistoryCount(UINT32 * historyCount)
```

This corresponds to the historyCount property of the POINTER\_INFO struct as defined in the Windows SDK (winuser.h).

## **get\_InputData**

Get the InputData of the pointer event.

```
public HRESULT get_InputData(INT32 * inputData)
```

This corresponds to the InputData property of the POINTER\_INFO struct as defined in the Windows SDK (winuser.h).

## **get\_KeyStates**

Get the KeyStates of the pointer event.

```
public HRESULT get_KeyStates(DWORD * keyStates)
```

This corresponds to the dwKeyStates property of the POINTER\_INFO struct as defined in the Windows SDK (winuser.h).

## **get\_PenFlags**

Get the PenFlags of the pointer event.

```
public HRESULT get_PenFlags(UINT32 * penFlags)
```

This corresponds to the penFlags property of the POINTER\_PEN\_INFO struct. The values are defined by the PEN\_FLAGS constants in the Windows SDK (winuser.h).

## get\_PenMask

Get the PenMask of the pointer event.

```
public HRESULT get_PenMask(UINT32 * penMask)
```

This corresponds to the penMask property of the POINTER\_PEN\_INFO struct. The values are defined by the PEN\_MASK constants in the Windows SDK (winuser.h).

## get\_PenPressure

Get the PenPressure of the pointer event.

```
public HRESULT get_PenPressure(UINT32 * penPressure)
```

This corresponds to the pressure property of the POINTER\_PEN\_INFO struct as defined in the Windows SDK (winuser.h).

## get\_PenRotation

Get the PenRotation of the pointer event.

```
public HRESULT get_PenRotation(UINT32 * penRotation)
```

This corresponds to the rotation property of the POINTER\_PEN\_INFO struct as defined in the Windows SDK (winuser.h).

## get\_PenTiltX

Get the PenTiltX of the pointer event.

```
public HRESULT get_PenTiltX(INT32 * penTiltX)
```

This corresponds to the tiltX property of the POINTER\_PEN\_INFO struct as defined in the Windows SDK (winuser.h).

## get\_PenTiltY

Get the PenTiltY of the pointer event.

```
public HRESULT get_PenTiltY(INT32 * penTiltY)
```

This corresponds to the tiltY property of the POINTER\_PEN\_INFO struct as defined in the Windows SDK (winuser.h).

## get\_PerformanceCount

Get the PerformanceCount of the pointer event.

```
public HRESULT get_PerformanceCount(UINT64 * performanceCount)
```

This corresponds to the PerformanceCount property of the POINTER\_INFO struct as defined in the Windows SDK (winuser.h).

## get\_PixelLocation

Get the PixelLocation of the pointer event.

```
public HRESULT get_PixelLocation(POINT * pixelLocation)
```

This corresponds to the ptPixelLocation property of the POINTER\_INFO struct as defined in the Windows SDK (winuser.h).

## get\_PixelLocationRaw

Get the PixelLocationRaw of the pointer event.

```
public HRESULT get_PixelLocationRaw(POINT * pixelLocationRaw)
```

This corresponds to the ptPixelLocationRaw property of the POINTER\_INFO struct as defined in the Windows SDK (winuser.h).

## get\_PointerDeviceRect

Get the PointerDeviceRect of the sourceDevice property of the POINTER\_INFO struct as defined in the Windows SDK (winuser.h).

```
public HRESULT get_PointerDeviceRect(RECT * pointerDeviceRect)
```

## **get\_PointerFlags**

Get the PointerFlags of the pointer event.

```
public HRESULT get_PointerFlags(UINT32 * pointerFlags)
```

This corresponds to the pointerFlags property of the POINTER\_INFO struct. The values are defined by the POINTER\_FLAGS constants in the Windows SDK (winuser.h).

## **get\_PointerId**

Get the PointerId of the pointer event.

```
public HRESULT get_PointerId(UINT32 * pointerId)
```

This corresponds to the pointerId property of the POINTER\_INFO struct as defined in the Windows SDK (winuser.h).

## **get\_PointerKind**

Get the PointerKind of the pointer event.

```
public HRESULT get_PointerKind(DWORD * pointerKind)
```

This corresponds to the pointerKind property of the POINTER\_INFO struct. The values are defined by the POINTER\_INPUT\_KIND enum in the Windows SDK (winuser.h). Supports PT\_PEN and PT\_TOUCH.

## **get\_Time**

Get the Time of the pointer event.

```
public HRESULT get_Time(DWORD * time)
```

This corresponds to the dwTime property of the POINTER\_INFO struct as defined in the Windows SDK (winuser.h).

## **get\_TouchContact**

Get the TouchContact of the pointer event.

```
public HRESULT get_TouchContact(RECT * touchContact)
```

This corresponds to the rcContact property of the POINTER\_TOUCH\_INFO struct as defined in the Windows SDK (winuser.h).

## get\_TouchContactRaw

Get the TouchContactRaw of the pointer event.

```
public HRESULT get_TouchContactRaw(RECT * touchContactRaw)
```

This corresponds to the rcContactRaw property of the POINTER\_TOUCH\_INFO struct as defined in the Windows SDK (winuser.h).

## get\_TouchFlags

Get the TouchFlags of the pointer event.

```
public HRESULT get_TouchFlags(UINT32 * touchFlags)
```

This corresponds to the touchFlags property of the POINTER\_TOUCH\_INFO struct. The values are defined by the TOUCH\_FLAGS constants in the Windows SDK (winuser.h).

## get\_TouchMask

Get the TouchMask of the pointer event.

```
public HRESULT get_TouchMask(UINT32 * touchMask)
```

This corresponds to the touchMask property of the POINTER\_TOUCH\_INFO struct. The values are defined by the TOUCH\_MASK constants in the Windows SDK (winuser.h).

## get\_TouchOrientation

Get the TouchOrientation of the pointer event.

```
public HRESULT get_TouchOrientation(UINT32 * touchOrientation)
```

This corresponds to the orientation property of the POINTER\_TOUCH\_INFO struct as defined in the Windows SDK (winuser.h).

## **get\_TouchPressure**

Get the TouchPressure of the pointer event.

```
public HRESULT get_TouchPressure(UINT32 * touchPressure)
```

This corresponds to the pressure property of the `POINTER_TOUCH_INFO` struct as defined in the Windows SDK (`winuser.h`).

## **put\_ButtonChangeKind**

Set the ButtonChangeKind of the pointer event.

```
public HRESULT put_ButtonChangeKind(INT32 buttonChangeKind)
```

This corresponds to the `ButtonChangeKind` property of the `POINTER_INFO` struct. The values are defined by the `POINTER_BUTTON_CHANGE_KIND` enum in the Windows SDK (`winuser.h`).

## **put\_DisplayRect**

Set the DisplayRect of the sourceDevice property of the `POINTER_INFO` struct as defined in the Windows SDK (`winuser.h`).

```
public HRESULT put_DisplayRect(RECT displayRect)
```

## **put\_Frameld**

Set the FrameID of the pointer event.

```
public HRESULT put_Frameld(UINT32 frameld)
```

This corresponds to the `frameld` property of the `POINTER_INFO` struct as defined in the Windows SDK (`winuser.h`).

## **put\_HimetricLocation**

Set the HimetricLocation of the pointer event.

```
public HRESULT put_HimetricLocation(POINT himetricLocation)
```

This corresponds to the ptHimetricLocation property of the POINTER\_INFO struct as defined in the Windows SDK (winuser.h).

## **put\_HimetricLocationRaw**

Set the HimetricLocationRaw of the pointer event.

```
| public HRESULT put_HimetricLocationRaw(POINT himetricLocationRaw)
```

This corresponds to the ptHimetricLocationRaw property of the POINTER\_INFO struct as defined in the Windows SDK (winuser.h).

## **put\_HistoryCount**

Set the HistoryCount of the pointer event.

```
| public HRESULT put_HistoryCount(UINT32 historyCount)
```

This corresponds to the historyCount property of the POINTER\_INFO struct as defined in the Windows SDK (winuser.h).

## **put\_InputData**

Set the InputData of the pointer event.

```
| public HRESULT put_InputData(INT32 inputData)
```

This corresponds to the InputData property of the POINTER\_INFO struct as defined in the Windows SDK (winuser.h).

## **put\_KeyStates**

Set the KeyStates of the pointer event.

```
| public HRESULT put_KeyStates(DWORD keyStates)
```

This corresponds to the dwKeyStates property of the POINTER\_INFO struct as defined in the Windows SDK (winuser.h).

## **put\_PenFlags**

Set the PenFlags of the pointer event.

```
public HRESULT put_PenFlags(UINT32 penFLags)
```

This corresponds to the penFlags property of the POINTER\_PEN\_INFO struct. The values are defined by the PEN\_FLAGS constants in the Windows SDK (winuser.h).

## **put\_PenMask**

Set the PenMask of the pointer event.

```
public HRESULT put_PenMask(UINT32 penMask)
```

This corresponds to the penMask property of the POINTER\_PEN\_INFO struct. The values are defined by the PEN\_MASK constants in the Windows SDK (winuser.h).

## **put\_PenPressure**

Set the PenPressure of the pointer event.

```
public HRESULT put_PenPressure(UINT32 penPressure)
```

This corresponds to the pressure property of the POINTER\_PEN\_INFO struct as defined in the Windows SDK (winuser.h).

## **put\_PenRotation**

Set the PenRotation of the pointer event.

```
public HRESULT put_PenRotation(UINT32 penRotation)
```

This corresponds to the rotation property of the POINTER\_PEN\_INFO struct as defined in the Windows SDK (winuser.h).

## **put\_PenTiltX**

Set the PenTiltX of the pointer event.

```
public HRESULT put_PenTiltX(INT32 penTiltX)
```

This corresponds to the `tiltX` property of the `POINTER_PEN_INFO` struct as defined in the Windows SDK (`winuser.h`).

## **put\_PenTiltY**

Set the `PenTiltY` of the pointer event.

```
| public HRESULT put_PenTiltY(INT32 penTiltY)
```

This corresponds to the `tiltY` property of the `POINTER_PEN_INFO` struct as defined in the Windows SDK (`winuser.h`).

## **put\_PerformanceCount**

Set the `PerformanceCount` of the pointer event.

```
| public HRESULT put_PerformanceCount(UINT64 performanceCount)
```

This corresponds to the `PerformanceCount` property of the `POINTER_INFO` struct as defined in the Windows SDK (`winuser.h`).

## **put\_PixelLocation**

Set the `PixelLocation` of the pointer event.

```
| public HRESULT put_PixelLocation(POINT pixelLocation)
```

This corresponds to the `ptPixelLocation` property of the `POINTER_INFO` struct as defined in the Windows SDK (`winuser.h`).

## **put\_PixelLocationRaw**

Set the `PixelLocationRaw` of the pointer event.

```
| public HRESULT put_PixelLocationRaw(POINT pixelLocationRaw)
```

This corresponds to the `ptPixelLocationRaw` property of the `POINTER_INFO` struct as defined in the Windows SDK (`winuser.h`).

## **put\_PointerDeviceRect**

Set the PointerDeviceRect of the sourceDevice property of the POINTER\_INFO struct as defined in the Windows SDK (winuser.h).

```
public HRESULT put_PointerDeviceRect(RECT pointerDeviceRect)
```

## put\_PointerFlags

Set the PointerFlags of the pointer event.

```
public HRESULT put_PointerFlags(UINT32 pointerFlags)
```

This corresponds to the pointerFlags property of the POINTER\_INFO struct. The values are defined by the POINTER\_FLAGS constants in the Windows SDK (winuser.h).

## put\_PointerId

Set the PointerId of the pointer event.

```
public HRESULT put_PointerId(UINT32 pointerId)
```

This corresponds to the pointerId property of the POINTER\_INFO struct as defined in the Windows SDK (winuser.h).

## put\_PointerKind

Set the PointerKind of the pointer event.

```
public HRESULT put_PointerKind(DWORD pointerKind)
```

This corresponds to the pointerKind property of the POINTER\_INFO struct. The values are defined by the POINTER\_INPUT\_KIND enum in the Windows SDK (winuser.h). Supports PT\_PEN and PT\_TOUCH.

## put\_Time

Set the Time of the pointer event.

```
public HRESULT put_Time(DWORD time)
```

This corresponds to the dwTime property of the POINTER\_INFO struct as defined in the Windows SDK (winuser.h).

## **put\_TouchContact**

Set the TouchContact of the pointer event.

```
| public HRESULT put_TouchContact(RECT touchContact)
```

This corresponds to the rcContact property of the POINTER\_TOUCH\_INFO struct as defined in the Windows SDK (winuser.h).

## **put\_TouchContactRaw**

Set the TouchContactRaw of the pointer event.

```
| public HRESULT put_TouchContactRaw(RECT touchContactRaw)
```

This corresponds to the rcContactRaw property of the POINTER\_TOUCH\_INFO struct as defined in the Windows SDK (winuser.h).

## **put\_TouchFlags**

Set the TouchFlags of the pointer event.

```
| public HRESULT put_TouchFlags(UINT32 touchFlags)
```

This corresponds to the touchFlags property of the POINTER\_TOUCH\_INFO struct. The values are defined by the TOUCH\_FLAGS constants in the Windows SDK (winuser.h).

## **put\_TouchMask**

Set the TouchMask of the pointer event.

```
| public HRESULT put_TouchMask(UINT32 touchMask)
```

This corresponds to the touchMask property of the POINTER\_TOUCH\_INFO struct. The values are defined by the TOUCH\_MASK constants in the Windows SDK (winuser.h).

## **put\_TouchOrientation**

Set the TouchOrientation of the pointer event.

```
public HRESULT put_TouchOrientation(UINT32 touchOrientation)
```

This corresponds to the orientation property of the `POINTER_TOUCH_INFO` struct as defined in the Windows SDK (`winuser.h`).

## **put\_TouchPressure**

Set the TouchPressure of the pointer event.

```
public HRESULT put_TouchPressure(UINT32 touchPressure)
```

This corresponds to the pressure property of the `POINTER_TOUCH_INFO` struct as defined in the Windows SDK (`winuser.h`).

---

## **Feedback**

Was this page helpful?

 Yes

 No

# interface ICoreWebView2PrintSettings

Article • 02/26/2024

```
interface ICoreWebView2PrintSettings
: public IUnknown
```

Settings used by the `PrintToPdf` method.

## Summary

[+] Expand table

Members	Descriptions
<code>get_FooterUri</code>	The URI in the footer if <code>ShouldPrintHeaderAndFooter</code> is <code>TRUE</code> .
<code>get_HeaderTitle</code>	The title in the header if <code>ShouldPrintHeaderAndFooter</code> is <code>TRUE</code> .
<code>get_MarginBottom</code>	The bottom margin in inches. The default is 1 cm, or ~0.4 inches.
<code>get_MarginLeft</code>	The left margin in inches. The default is 1 cm, or ~0.4 inches.
<code>get_MarginRight</code>	The right margin in inches. The default is 1 cm, or ~0.4 inches.
<code>get_MarginTop</code>	The top margin in inches. The default is 1 cm, or ~0.4 inches.
<code>get_Orientation</code>	The orientation can be portrait or landscape.
<code>get_PageHeight</code>	The page height in inches. The default height is 11 inches.
<code>get_PageWidth</code>	The page width in inches. The default width is 8.5 inches.
<code>get_ScaleFactor</code>	The scale factor is a value between 0.1 and 2.0. The default is 1.0.
<code>get_ShouldPrintBackgrounds</code>	<code>TRUE</code> if background colors and images should be printed.
<code>get_ShouldPrintHeaderAndFooter</code>	<code>TRUE</code> if header and footer should be printed.
<code>get_ShouldPrintSelectionOnly</code>	<code>TRUE</code> if only the current end user's selection of HTML in the document should be printed.
<code>put_FooterUri</code>	Set the <code>FooterUri</code> property.

Members	Descriptions
<a href="#">put_HeaderTitle</a>	Set the <code>HeaderTitle</code> property.
<a href="#">put_MarginBottom</a>	Sets the <code>MarginBottom</code> property.
<a href="#">put_MarginLeft</a>	Sets the <code>MarginLeft</code> property.
<a href="#">put_MarginRight</a>	Set the <code>MarginRight</code> property. A margin cannot be less than zero.
<a href="#">put_MarginTop</a>	Sets the <code>MarginTop</code> property.
<a href="#">put_Orientation</a>	Sets the <code>Orientation</code> property.
<a href="#">put_PageHeight</a>	Sets the <code>PageHeight</code> property.
<a href="#">put_PageWidth</a>	Sets the <code>PageWidth</code> property.
<a href="#">put_ScaleFactor</a>	Sets the <code>ScaleFactor</code> property.
<a href="#">put_ShouldPrintBackgrounds</a>	Set the <code>ShouldPrintBackgrounds</code> property.
<a href="#">put_ShouldPrintHeaderAndFooter</a>	Set the <code>ShouldPrintHeaderAndFooter</code> property.
<a href="#">put_ShouldPrintSelectionOnly</a>	Set the <code>ShouldPrintSelectionOnly</code> property.

## Applies to

[Expand table](#)

Product	Introduced
WebView2 Win32	1.0.1020.30
WebView2 Win32 Prerelease	1.0.1056

## Members

### get\_FooterUri

The URI in the footer if `ShouldPrintHeaderAndFooter` is `TRUE`.

```
public HRESULT get_FooterUri(LPWSTR * footerUri)
```

The default value is the current URI.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_HeaderTitle

The title in the header if `ShouldPrintHeaderAndFooter` is `TRUE`.

```
public HRESULT get_HeaderTitle(LPWSTR * headerTitle)
```

The default value is the title of the current document.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_MarginBottom

The bottom margin in inches. The default is 1 cm, or ~0.4 inches.

```
public HRESULT get_MarginBottom(double * marginBottom)
```

## get\_MarginLeft

The left margin in inches. The default is 1 cm, or ~0.4 inches.

```
public HRESULT get_MarginLeft(double * marginLeft)
```

## get\_MarginRight

The right margin in inches. The default is 1 cm, or ~0.4 inches.

```
public HRESULT get_MarginRight(double * marginRight)
```

## get\_MarginTop

The top margin in inches. The default is 1 cm, or ~0.4 inches.

```
public HRESULT get_MarginTop(double * marginTop)
```

## get\_Orientation

The orientation can be portrait or landscape.

```
public HRESULT get_Orientation(COREWEBVIEW2_PRINT_ORIENTATION *  
orientation)
```

The default orientation is portrait. See `COREWEBVIEW2_PRINT_ORIENTATION`.

## get\_PageHeight

The page height in inches. The default height is 11 inches.

```
public HRESULT get_PageHeight(double * pageHeight)
```

## get\_PageWidth

The page width in inches. The default width is 8.5 inches.

```
public HRESULT get_PageWidth(double * pageWidth)
```

## get\_ScaleFactor

The scale factor is a value between 0.1 and 2.0. The default is 1.0.

```
public HRESULT get_ScaleFactor(double * scaleFactor)
```

## get\_ShouldPrintBackgrounds

`TRUE` if background colors and images should be printed.

```
public HRESULT get_ShouldPrintBackgrounds(BOOL * shouldPrintBackgrounds)
```

The default value is `FALSE`.

## get\_ShouldPrintHeaderAndFooter

`TRUE` if header and footer should be printed.

```
public HRESULT get_ShouldPrintHeaderAndFooter(BOOL *  
shouldPrintHeaderAndFooter)
```

The default value is `FALSE`. The header consists of the date and time of printing, and the title of the page. The footer consists of the URI and page number. The height of the

header and footer is 0.5 cm, or ~0.2 inches.

## get\_ShouldPrintSelectionOnly

`TRUE` if only the current end user's selection of HTML in the document should be printed.

```
| public HRESULT get_ShouldPrintSelectionOnly(BOOL * shouldPrintSelectionOnly)
```

The default value is `FALSE`.

## put\_FooterUri

Set the `FooterUri` property.

```
| public HRESULT put_FooterUri(LPCWSTR footerUri)
```

If an empty string or null value is provided, no URI is shown in the footer.

## put\_HeaderTitle

Set the `HeaderTitle` property.

```
| public HRESULT put_HeaderTitle(LPCWSTR headerTitle)
```

If an empty string or null value is provided, no title is shown in the header.

## put\_MarginBottom

Sets the `MarginBottom` property.

```
| public HRESULT put_MarginBottom(double marginBottom)
```

A margin cannot be less than zero. Returns `E_INVALIDARG` if an invalid value is provided, and the current value is not changed.

## put\_MarginLeft

Sets the `MarginLeft` property.

```
public HRESULT put_MarginLeft(double marginLeft)
```

A margin cannot be less than zero. Returns `E_INVALIDARG` if an invalid value is provided, and the current value is not changed.

## put\_MarginRight

Set the `MarginRight` property. A margin cannot be less than zero.

```
public HRESULT put_MarginRight(double marginRight)
```

Returns `E_INVALIDARG` if an invalid value is provided, and the current value is not changed.

## put\_MarginTop

Sets the `MarginTop` property.

```
public HRESULT put_MarginTop(double marginTop)
```

A margin cannot be less than zero. Returns `E_INVALIDARG` if an invalid value is provided, and the current value is not changed.

## put\_Orientation

Sets the `Orientation` property.

```
public HRESULT put_Orientation(COREWEBVIEW2_PRINT_ORIENTATION orientation)
```

## put\_PageHeight

Sets the `PageHeight` property.

```
public HRESULT put_PageHeight(double pageHeight)
```

Returns `E_INVALIDARG` if the page height is less than or equal to zero, and the current value is not changed.

## put\_PageWidth

Sets the `PageWidth` property.

```
public HRESULT put_PageWidth(double pageWidth)
```

Returns `E_INVALIDARG` if the page width is less than or equal to zero, and the current value is not changed.

## put\_ScaleFactor

Sets the `ScaleFactor` property.

```
public HRESULT put_ScaleFactor(double scaleFactor)
```

Returns `E_INVALIDARG` if an invalid value is provided, and the current value is not changed.

## put\_ShouldPrintBackgrounds

Set the `ShouldPrintBackgrounds` property.

```
public HRESULT put_ShouldPrintBackgrounds(BOOL shouldPrintBackgrounds)
```

## put\_ShouldPrintHeaderAndFooter

Set the `ShouldPrintHeaderAndFooter` property.

```
public HRESULT put_ShouldPrintHeaderAndFooter(BOOL  
shouldPrintHeaderAndFooter)
```

## put\_ShouldPrintSelectionOnly

Set the `ShouldPrintSelectionOnly` property.

```
public HRESULT put_ShouldPrintSelectionOnly(BOOL shouldPrintSelectionOnly)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2PrintSettings2

Article • 02/26/2024

```
interface ICoreWebView2PrintSettings2
    : public ICoreWebView2PrintSettings
```

Settings used by the `Print` method.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_Collation</a>	Printer collation.
<a href="#">get_ColorMode</a>	Printer color mode.
<a href="#">get_Copies</a>	Number of copies to print.
<a href="#">get_Duplex</a>	Printer duplex settings.
<a href="#">get_MediaSize</a>	Printer media size.
<a href="#">get_PageRanges</a>	Page range to print.
<a href="#">get_PagesPerSide</a>	Prints multiple pages of a document on a single piece of paper.
<a href="#">get_PrinterName</a>	The name of the printer to use.
<a href="#">put_Collation</a>	Set the <code>Collation</code> property.
<a href="#">put_ColorMode</a>	Set the <code>ColorMode</code> property.
<a href="#">put_Copies</a>	Set the <code>Copies</code> property.
<a href="#">put_Duplex</a>	Set the <code>Duplex</code> property.
<a href="#">put_MediaSize</a>	Set the <code>MediaSize</code> property.
<a href="#">put_PageRanges</a>	Set the <code>PageRanges</code> property.
<a href="#">put_PagesPerSide</a>	Set the <code>PagesPerSide</code> property.
<a href="#">put_PrinterName</a>	Set the <code>PrinterName</code> property.

# Applies to

 Expand table

Product	Introduced
WebView2 Win32	1.0.1518.46
WebView2 Win32 Prerelease	1.0.1549

## Members

### get\_Collation

Printer collation.

```
public HRESULT get_Collation(COREWEBVIEW2_PRINT_COLLATION * value)
```

See `COREWEBVIEW2_PRINT_COLLATION` for descriptions of collation. The default value is `COREWEBVIEW2_PRINT_COLLATION_DEFAULT`.

Printing uses default value of printer's collation if an invalid value is provided for the specific printer.

This value is ignored in `PrintToPdfStream` method.

### get\_ColorMode

Printer color mode.

```
public HRESULT get_ColorMode(COREWEBVIEW2_PRINT_COLOR_MODE * value)
```

See `COREWEBVIEW2_PRINT_COLOR_MODE` for descriptions of color modes. The default value is `COREWEBVIEW2_PRINT_COLOR_MODE_DEFAULT`.

Printing uses default value of printer supported color if an invalid value is provided for the specific printer.

### get\_Copies

Number of copies to print.

```
public HRESULT get_Copies(INT32 * value)
```

Minimum value is 1 and the maximum copies count is 999. The default value is 1.

This value is ignored in PrintToPdfStream method.

## get\_Duplex

Printer duplex settings.

```
public HRESULT get_Duplex(COREWEBVIEW2_PRINT_DUPLEX * value)
```

See COREWEBVIEW2\_PRINT\_DUPLEX for descriptions of duplex. The default value is COREWEBVIEW2\_PRINT\_DUPLEX\_DEFAULT.

Printing uses default value of printer's duplex if an invalid value is provided for the specific printer.

This value is ignored in PrintToPdfStream method.

## get\_MediaSize

Printer media size.

```
public HRESULT get_MediaSize(COREWEBVIEW2_PRINT_MEDIA_SIZE * value)
```

See COREWEBVIEW2\_PRINT\_MEDIA\_SIZE for descriptions of media size. The default value is COREWEBVIEW2\_PRINT\_MEDIA\_SIZE\_DEFAULT.

If media size is COREWEBVIEW2\_PRINT\_MEDIA\_SIZE\_CUSTOM, you should set the PageWidth and PageHeight.

Printing uses default value of printer supported media size if an invalid value is provided for the specific printer.

This value is ignored in PrintToPdfStream method.

## get\_PageRanges

Page range to print.

```
public HRESULT get_PageRanges(LPWSTR * value)
```

Defaults to empty string, which means print all pages. If the Page range is empty string or null, then it applies the default.

The PageRanges property is a list of page ranges specifying one or more pages that should be printed separated by commas. Any whitespace between page ranges is ignored. A valid page range is either a single integer identifying the page to print, or a range in the form `[start page]-[last page]` where `start page` and `last page` are integers identifying the first and last inclusive pages respectively to print. Every page identifier is an integer greater than 0 unless wildcards are used (see below examples). The first page is 1.

In a page range of the form `[start page]-[last page]` the start page number must be larger than 0 and less than or equal to the document's total page count. If the `start page` is not present, then 1 is used as the `start page`. The `last page` must be larger than the `start page`. If the `last page` is not present, then the document total page count is used as the `last page`.

Repeating a page does not print it multiple times. To print multiple times, use the `Copies` property.

The pages are always printed in ascending order, even if specified in non-ascending order.

If page range is not valid or if a page is greater than document total page count, `ICoreWebView2PrintCompletedHandler` or `ICoreWebView2PrintToPdfStreamCompletedHandler` handler will return `E_INVALIDARG`.

The following examples assume a document with 20 total pages.

Expand table

Example	Result	Notes
"2"	Page 2	
"1-4, 9, 3-6, 10, 11"	Pages 1-6, 9- 11	
"1-4, -6"	Pages 1-6	The "-6" is interpreted as "1-6".
"2-"	Pages 2-20	The "2-" is interpreted as "pages 2 to the end of the document".
"4-2, 11, -6"	Invalid	"4-2" is an invalid range.

Example	Result	Notes
"-"	Pages 1-20	The "-" is interpreted as "page 1 to the end of the document".
"1-4dsf, 11"	Invalid	
"2-2"	Page 2	

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#)

## get\_PagesPerSide

Prints multiple pages of a document on a single piece of paper.

```
public HRESULT get_PagesPerSide(INT32 * value)
```

Choose from 1, 2, 4, 6, 9 or 16. The default value is 1.

## get\_PrinterName

The name of the printer to use.

```
public HRESULT get_PrinterName(LPWSTR * value)
```

Defaults to empty string. If the printer name is empty string or null, then it prints to the default printer on the user OS.

This value is ignored in `PrintToPdfStream` method.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#)

## put\_Collation

Set the `Collation` property.

```
public HRESULT put_Collation(COREWEBVIEW2_PRINT_COLLATION value)
```

## put\_ColorMode

Set the `ColorMode` property.

```
public HRESULT put_ColorMode(COREWEBVIEW2_PRINT_COLOR_MODE value)
```

## **put\_Copies**

Set the `Copies` property.

```
public HRESULT put_Copies(INT32 value)
```

Returns `E_INVALIDARG` if an invalid value is provided and the current value is not changed.

## **put\_Duplex**

Set the `Duplex` property.

```
public HRESULT put_Duplex(COREWEBVIEW2_PRINT_DUPLEX value)
```

## **put\_MediaSize**

Set the `MediaSize` property.

```
public HRESULT put_MediaSize(COREWEBVIEW2_PRINT_MEDIA_SIZE value)
```

## **put\_PageRanges**

Set the `PageRanges` property.

```
public HRESULT put_PageRanges(LPCWSTR value)
```

## **put\_PagesPerSide**

Set the `PagesPerSide` property.

```
public HRESULT put_PagesPerSide(INT32 value)
```

Returns `E_INVALIDARG` if an invalid value is provided, and the current value is not changed.

Below examples shows print output for PagesPerSide and Duplex.

 Expand table

PagesPerSide	Total pages	Two-sided printing	Result
1	1	-	1 page on the front side.
2	1	Yes	1 page on the front side.
2	4	-	2 pages on the first paper and 2 pages on the next paper.
2	4	Yes	2 pages on the front side and 2 pages on back side.
4	4	Yes	4 pages on the front side.
4	8	Yes	4 pages on the front side and 4 pages on the back side.

## put\_PrinterName

Set the `PrinterName` property.

```
public HRESULT put_PrinterName(LPCWSTR value)
```

If provided printer name doesn't match with the name of any installed printers on the user OS, ICoreWebView2PrintCompletedHandler handler will return `errorCode` as `S_OK` and `printStatus` as `COREWEBVIEW2_PRINT_STATUS_PRINTER_UNAVAILABLE`.

Use [Enum Printers](#) to enumerate available printers.

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2ProcessExtendedInfo

Article • 02/26/2024

```
interface ICoreWebView2ProcessExtendedInfo
: public IUnknown
```

Provides process with associated extended information in the ICoreWebView2Environment.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_AssociatedFrameInfos</a>	The collection of associated <code>FrameInfo</code> s which are actively running (showing or hiding UI elements) in this renderer process.
<a href="#">get_ProcessInfo</a>	The process info of the current process.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.2210.55
WebView2 Win32 Prerelease	1.0.2357

## Members

### [get\\_AssociatedFrameInfos](#)

The collection of associated `FrameInfo`s which are actively running (showing or hiding UI elements) in this renderer process.

```
public HRESULT get_AssociatedFrameInfos(ICoreWebView2FrameInfoCollection **  
frames)
```

`AssociatedFrameInfos` will only be populated when this `CoreWebView2ProcessExtendedInfo` corresponds to a renderer process. Non-renderer processes will always have an empty `AssociatedFrameInfos`. The `AssociatedFrameInfos` may also be empty for renderer processes that have no active frames.

C++

```
        std::wstringstream rendererProcess;  
        wil::com_ptr<ICoreWebView2FrameInfoCollection>  
frameInfoCollection;  
        CHECK_FAILURE(processExtendedInfo-  
>get_AssociatedFrameInfos(  
                            &frameInfoCollection));  
  
        wil::com_ptr<ICoreWebView2FrameInfoCollectionIterator> iterator;  
        CHECK_FAILURE(frameInfoCollection-  
>GetIterator(&iterator));  
        BOOL hasCurrent = FALSE;  
        UINT32 frameInfoCount = 0;  
        while (SUCCEEDED(iterator-  
>get_HasCurrent(&hasCurrent)) &&  
               hasCurrent)  
        {  
            wil::com_ptr<ICoreWebView2FrameInfo>  
frameInfo;  
            CHECK_FAILURE(iterator-  
>GetCurrent(&frameInfo));  
  
            AppendFrameInfo(frameInfo, rendererProcess);  
  
            BOOL hasNext = FALSE;  
            CHECK_FAILURE(iterator->MoveNext(&hasNext));  
            frameInfoCount++;  
        }  
        rendererProcessInfos  
        << frameInfoCount  
        << L" frameInfo(s) found in Renderer Process  
ID:" << processId  
        << L"\n"  
        << rendererProcess.str() << std::endl;
```

## get\_ProcessInfo

The process info of the current process.

```
public HRESULT get_ProcessInfo(ICoreWebView2ProcessInfo ** processInfo)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2ProcessExtendedInfoCollection

Article • 02/26/2024

```
interface ICoreWebView2ProcessExtendedInfoCollection
: public IUnknown
```

A list containing processInfo and associated extended information.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_Count</a>	The number of process contained in the ICoreWebView2ProcessExtendedInfoCollection.
<a href="#">GetValueAtIndex</a>	Gets the <a href="#">ICoreWebView2ProcessExtendedInfo</a> located in the ICoreWebView2ProcessExtendedInfoCollection at the given index.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.2210.55
WebView2 Win32 Prerelease	1.0.2357

## Members

### [get\\_Count](#)

The number of process contained in the ICoreWebView2ProcessExtendedInfoCollection.

```
public HRESULT get_Count(UINT * count)
```

## GetValueAtIndex

Gets the [ICoreWebView2ProcessExtendedInfo](#) located in the [ICoreWebView2ProcessExtendedInfoCollection](#) at the given index.

```
public HRESULT GetValueAtIndex(UINT32 index,  
    ICoreWebView2ProcessExtendedInfo ** processInfo)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2ProcessFailedEventArgs

Article • 02/26/2024

```
interface ICoreWebView2ProcessFailedEventArgs
: public IUnknown
```

Event args for the `ProcessFailed` event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_ProcessFailedKind</a>	The kind of process failure that has occurred.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### [get\\_ProcessFailedKind](#)

The kind of process failure that has occurred.

```
public HRESULT get\_ProcessFailedKind(COREWEBVIEW2_PROCESS_FAILED_KIND *
processFailedKind)
```

This is a combination of process kind (for example, browser, renderer, gpu) and failure (exit, unresponsiveness). Renderer processes are further divided in *main frame* renderer (`RenderProcessExited`, `RenderProcessUnresponsive`) and *subframe* renderer (`FrameRenderProcessExited`). To learn about the conditions under which each failure kind occurs, see `COREWEBVIEW2_PROCESS_FAILED_KIND`.

---

## Feedback

Was this page helpful?



# interface ICoreWebView2ProcessFailedEventArgs2

Article • 02/26/2024

```
interface ICoreWebView2ProcessFailedEventArgs2
: public ICoreWebView2ProcessFailedEventArgs
```

A continuation of the [ICoreWebView2ProcessFailedEventArgs](#) interface.

## Summary

 Expand table

Members	Descriptions
<a href="#">get_ExitCode</a>	The exit code of the failing process, for telemetry purposes.
<a href="#">get_FrameInfosForFailedProcess</a>	The collection of <code>FrameInfo</code> s for frames in the <a href="#">ICoreWebView2</a> that were being rendered by the failed process.
<a href="#">get_ProcessDescription</a>	Description of the process assigned by the WebView2 Runtime.
<a href="#">get_Reason</a>	The reason for the process failure.

## Applies to

 Expand table

Product	Introduced
WebView2 Win32	1.0.818.41
WebView2 Win32 Prerelease	1.0.824

## Members

### get\_ExitCode

The exit code of the failing process, for telemetry purposes.

```
public HRESULT get_ExitCode(int * exitCode)
```

The exit code is always `STILL_ACTIVE` (259) when `ProcessFailedKind` is `COREWEBVIEW2_PROCESS_FAILED_KIND_RENDER_PROCESS_UNRESPONSIVE`.

### get\_FrameInfosForFailedProcess

The collection of `FrameInfo`s for frames in the [ICoreWebView2](#) that were being rendered by the failed process.

```
public HRESULT get_FrameInfosForFailedProcess(ICoreWebView2FrameInfoCollection ** frames)
```

The content in these frames is replaced with an error page. This is only available when `ProcessFailedKind` is `COREWEBVIEW2_PROCESS_FAILED_KIND_FRAME_RENDER_PROCESS_EXITED`; `frames` is `null` for all other process failure kinds, including the case in which the failed process was the renderer for the main frame and subframes within it, for which the failure kind is `COREWEBVIEW2_PROCESS_FAILED_KIND_RENDER_PROCESS_EXITED`.

## get\_ProcessDescription

Description of the process assigned by the WebView2 Runtime.

```
public HRESULT get_ProcessDescription(LPWSTR * processDescription)
```

This is a technical English term appropriate for logging or development purposes, and not localized for the end user. It applies to utility processes (for example, "Audio Service", "Video Capture") and plugin processes (for example, "Flash"). The returned `processDescription` is empty if the WebView2 Runtime did not assign a description to the process.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_Reason

The reason for the process failure.

```
public HRESULT get_Reason(COREWEBVIEW2_PROCESS_FAILED_REASON * reason)
```

Some of the reasons are only applicable to specific values of

`ICoreWebView2ProcessFailedEventArgs::ProcessFailedKind`, and the following `ProcessFailedKind` values always return the indicated reason value:

[+] Expand table

ProcessFailedKind	Reason
<code>COREWEBVIEW2_PROCESS_FAILED_KIND_BROWSER_PROCESS_EXITED</code>	<code>COREWEBVIEW2_PROCESS_FAILED_REASON_UNEXPECTED</code>
<code>COREWEBVIEW2_PROCESS_FAILED_KIND_RENDER_PROCESS_UNRESPONSIVE</code>	<code>COREWEBVIEW2_PROCESS_FAILED_REASON_UNRESPONSIVE</code>

For other `ProcessFailedKind` values, the reason may be any of the reason values. To learn about what these values mean, see `COREWEBVIEW2_PROCESS_FAILED_REASON`.

---

## Feedback

Was this page helpful?

Yes  No

# interface ICoreWebView2ProcessInfo

Article • 02/26/2024

```
interface ICoreWebView2ProcessInfo
: public IUnknown
```

Provides a set of properties for a process in the ICoreWebView2Environment.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_Kind</a>	The kind of the process.
<a href="#">get_ProcessId</a>	The process id of the process.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1108.44
WebView2 Win32 Prerelease	1.0.1133

## Members

### [get\\_Kind](#)

The kind of the process.

```
public HRESULT get_Kind(COREWEBVIEW2_PROCESS_KIND * kind)
```

### [get\\_ProcessId](#)

The process id of the process.

```
public HRESULT get_ProcessId(INT32 * value)
```

---

## Feedback

Was this page helpful?



# interface ICoreWebView2ProcessInfoCollection

Article • 02/26/2024

```
interface ICoreWebView2ProcessInfoCollection
: public IUnknown
```

A list containing process id and corresponding process type.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_Count</a>	The number of process contained in the ICoreWebView2ProcessInfoCollection.
<a href="#">GetValueAtIndex</a>	Gets the ICoreWebView2ProcessInfo located in the ICoreWebView2ProcessInfoCollection at the given index.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1108.44
WebView2 Win32 Prerelease	1.0.1133

## Members

### [get\\_Count](#)

The number of process contained in the ICoreWebView2ProcessInfoCollection.

```
public HRESULT get_Count(UINT * count)
```

## GetValueAtIndex

Gets the ICoreWebView2ProcessInfo located in the ICoreWebView2ProcessInfoCollection at the given index.

```
public HRESULT GetValueAtIndex(UINT32 index, ICoreWebView2ProcessInfo **  
processInfo)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Profile

Article • 02/26/2024

```
interface ICoreWebView2Profile
: public IUnknown
```

Provides a set of properties to configure a Profile object.

## Summary

[\[+\] Expand table](#)

Members	Descriptions
<a href="#">get_DefaultDownloadFolderPath</a>	Gets the <code>DefaultDownloadFolderPath</code> property.
<a href="#">get_IsInPrivateModeEnabled</a>	InPrivate mode is enabled or not.
<a href="#">get_PreferredColorScheme</a>	The PreferredColorScheme property sets the overall color scheme of the WebView2s associated with this profile.
<a href="#">get_ProfileName</a>	Name of the profile.
<a href="#">get_ProfilePath</a>	Full path of the profile directory.
<a href="#">put_DefaultDownloadFolderPath</a>	Sets the <code>DefaultDownloadFolderPath</code> property.
<a href="#">put_PreferredColorScheme</a>	Sets the <code>PreferredColorScheme</code> property.

C++

```
    CHECK_FAILURE(webViewEnvironment10-
>CreateCoreWebView2CompositionControllerWithOptions(
    m_mainWindow, options.get(),

Callback<ICoreWebView2CreateCoreWebView2CompositionControllerCompletedHandle
r>(
    [this]{
        HRESULT result,
        ICoreWebView2CompositionController*
compositionController) -> HRESULT
    {
        auto controller =
            wil::com_ptr<ICoreWebView2CompositionController>
(compositionController)
```

```
        .query<ICoreWebView2Controller>());
    return OnCreateCoreWebView2ControllerCompleted(result,
controller.get());
    })
    .Get());
```

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1210.39
WebView2 Win32 Prerelease	1.0.1222

## Members

### get\_DefaultDownloadFolderPath

Gets the `DefaultDownloadFolderPath` property.

```
public HRESULT get_DefaultDownloadFolderPath(LPWSTR * value)
```

The default value is the system default download folder path for the user.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

### get\_IsInPrivateModeEnabled

InPrivate mode is enabled or not.

```
public HRESULT get_IsInPrivateModeEnabled(BOOL * value)
```

### get\_PreferredColorScheme

The PreferredColorScheme property sets the overall color scheme of the WebView2s associated with this profile.

```
public HRESULT
get_PreferredColorScheme(COREWEBVIEW2_PREFERRED_COLOR_SCHEME * value)
```

This sets the color scheme for WebView2 UI like dialogs, prompts, and context menus by setting the media feature `prefers-color-scheme` for websites to respond to.

The default value for this is `COREWEBVIEW2_PREFERRED_COLOR_AUTO`, which will follow whatever theme the OS is currently set to.

C++

```
void
ViewComponent::SetPreferredColorScheme(COREWEBVIEW2_PREFERRED_COLOR_SCHEME
value)
{
    wil::com_ptr<ICoreWebView2_13> webView2_13;
    webView2_13 = m_webView.try_query<ICoreWebView2_13>();

    if (webView2_13)
    {
        wil::com_ptr<ICoreWebView2Profile> profile;
        CHECK_FAILURE(webView2_13->get_Profile(&profile));
        profile->put_PreferredColorScheme(value);
    }
}
```

Returns the value of the `PreferredColorScheme` property.

## get\_ProfileName

Name of the profile.

```
public HRESULT get_ProfileName(LPWSTR * value)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_ProfilePath

Full path of the profile directory.

```
public HRESULT get_ProfilePath(LPWSTR * value)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## put\_DefaultDownloadFolderPath

Sets the `DefaultDownloadFolderPath` property.

```
public HRESULT put_DefaultDownloadFolderPath(LPCWSTR value)
```

The default download folder path is persisted in the user data folder across sessions. The value should be an absolute path to a folder that the user and application can write to. Returns `E_INVALIDARG` if the value is invalid, and the default download folder path is not changed. Otherwise the path is changed immediately. If the directory does not yet exist, it is created at the time of the next download. If the host application does not have permission to create the directory, then the user is prompted to provide a new path through the Save As dialog. The user can override the default download folder path for a given download by choosing a different path in the Save As dialog.

## put\_PREFERRED\_COLOR\_SCHEME

Sets the `PreferredColorScheme` property.

```
public HRESULT  
put_PREFERRED_COLOR_SCHEME(COREWEBVIEW2_PREFERRED_COLOR_SCHEME value)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Profile2

Article • 02/26/2024

```
interface ICoreWebView2Profile2
    : public ICoreWebView2Profile
```

Profile2 interface.

## Summary

[+] [Expand table](#)

Members	Descriptions
<a href="#">ClearBrowsingData</a>	Clear browsing data based on a data type.
<a href="#">ClearBrowsingDataAll</a>	ClearBrowsingDataAll behaves like ClearBrowsingData except that it clears the entirety of the data associated with the profile it is called on.
<a href="#">ClearBrowsingDataInTimeRange</a>	ClearBrowsingDataInTimeRange behaves like ClearBrowsingData except that it takes in two additional parameters for the start and end time for which it should clear the data between.

## Applies to

[+] [Expand table](#)

Product	Introduced
WebView2 Win32	1.0.1245.22
WebView2 Win32 Prerelease	1.0.1248

## Members

### [ClearBrowsingData](#)

Clear browsing data based on a data type.

```
public HRESULT ClearBrowsingData(COREWEBVIEW2_BROWSING_DATA_KINDS  
dataKinds, ICoreWebView2ClearBrowsingDataCompletedHandler * handler)
```

This method takes two parameters, the first being a mask of one or more `COREWEBVIEW2_BROWSING_DATA_KINDS`. OR operation(s) can be applied to multiple `COREWEBVIEW2_BROWSING_DATA_KINDS` to create a mask representing those data types. The browsing data kinds that are supported are listed below. These data kinds follow a hierarchical structure in which nested bullet points are included in their parent bullet point's data kind. Ex: All DOM storage is encompassed in all site data which is encompassed in all profile data.

- All Profile
- All Site Data
- All DOM Storage: File Systems, Indexed DB, Local Storage, Web SQL, Cache Storage
- Cookies
- Disk Cache
- Download History
- General Autocomplete
- Password Autosave
- Browsing History
- Settings The completed handler will be invoked when the browsing data has been cleared and will indicate if the specified data was properly cleared. In the case in which the operation is interrupted and the corresponding data is not fully cleared the handler will return `E_ABORT` and otherwise will return `S_OK`. Because this is an asynchronous operation, code that is dependent on the cleared data must be placed in the callback of this operation. If the WebView object is closed before the clear browsing data operation has completed, the handler will be released, but not invoked. In this case the clear browsing data operation may or may not be completed. `ClearBrowsingData` clears the `dataKinds` regardless of timestamp.

## ClearBrowsingDataAll

`ClearBrowsingDataAll` behaves like `ClearBrowsingData` except that it clears the entirety of the data associated with the profile it is called on.

```
public HRESULT  
ClearBrowsingDataAll(ICoreWebView2ClearBrowsingDataCompletedHandler *  
handler)
```

It clears the data regardless of timestamp.

C++

```
bool AppWindow::ClearBrowsingData(COREWEBVIEW2_BROWSING_DATA_KINDS  
dataKinds)  
{  
    auto webView2_13 = m_webView.try_query<ICoreWebView2_13>();  
    CHECK_FEATURE_RETURN(webView2_13);  
    wil::com_ptr<ICoreWebView2Profile> webView2Profile;  
    CHECK_FAILURE(webView2_13->get_Profile(&webView2Profile));  
    CHECK_FEATURE_RETURN(webView2Profile);  
    auto webView2Profile2 = webView2Profile.try_query<ICoreWebView2Profile2>()  
    ;  
    CHECK_FEATURE_RETURN(webView2Profile2);  
    // Clear the browsing data from the last hour.  
    double endTime = (double)std::time(nullptr);  
    double startTime = endTime - 3600.0;  
    CHECK_FAILURE(webView2Profile2->ClearBrowsingDataInTimeRange(  
        dataKinds, startTime, endTime,  
        Callback<ICoreWebView2ClearBrowsingDataCompletedHandler>(  
            [this](HRESULT error) -> HRESULT  
            {  
                AsyncMessageBox(L"Completed", L"Clear Browsing Data");  
                return S_OK;  
            })  
            .Get());  
    return true;  
}
```

## ClearBrowsingDataInTimeRange

`ClearBrowsingDataInTimeRange` behaves like `ClearBrowsingData` except that it takes in two additional parameters for the start and end time for which it should clear the data between.

```
public HRESULT  
ClearBrowsingDataInTimeRange(COREWEBVIEW2_BROWSING_DATA_KINDS  
dataKinds, double startTime, double endTime,  
ICoreWebView2ClearBrowsingDataCompletedHandler * handler)
```

The `startTime` and `endTime` parameters correspond to the number of seconds since the UNIX epoch. `startTime` is inclusive while `endTime` is exclusive, therefore the data will be cleared between [startTime, endTime).

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Profile3

Article • 02/26/2024

```
interface ICoreWebView2Profile3
: public ICoreWebView2Profile2
```

This is an extension of the [ICoreWebView2Profile](#) interface to control levels of tracking prevention.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_PREFERREDTRACKINGPREVENTIONLEVEL</a>	The <code>PreferredTrackingPreventionLevel</code> property allows you to control levels of tracking prevention for WebView2 which are associated with a profile.
<a href="#">put_PREFERREDTRACKINGPREVENTIONLEVEL</a>	Set the <code>PreferredTrackingPreventionLevel</code> property.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1661.34
WebView2 Win32 Prerelease	1.0.1619

## Members

### [get\\_PREFERREDTRACKINGPREVENTIONLEVEL](#)

The `PreferredTrackingPreventionLevel` property allows you to control levels of tracking prevention for WebView2 which are associated with a profile.

```
public HRESULT  
get_PREFERREDTRACKINGPREVENTIONLEVEL(COREWEBVIEW2_TRACKING_PREVENTION_LEVEL * value)
```

This level would apply to the context of the profile. That is, all WebView2s sharing the same profile will be affected and also the value is persisted in the user data folder.

See `COREWEBVIEW2_TRACKING_PREVENTION_LEVEL` for descriptions of levels.

If tracking prevention feature is enabled when creating the WebView2 environment, you can also disable tracking prevention later using this property and `COREWEBVIEW2_TRACKING_PREVENTION_LEVEL_NONE` value but that doesn't improve runtime performance.

There is `ICoreWebView2EnvironmentOptions5::EnableTrackingPrevention` property to enable/disable tracking prevention feature for all the WebView2's created in the same environment. If enabled, `PreferredTrackingPreventionLevel` is set to `COREWEBVIEW2_TRACKING_PREVENTION_LEVEL_BALANCED` by default for all the WebView2's and profiles created in the same environment or is set to the level whatever value was last changed/persisted to the profile. If disabled `PreferredTrackingPreventionLevel` is not respected by WebView2. If `PreferredTrackingPreventionLevel` is set when the feature is disabled, the property value gets changed and persisted but it will take effect only if `ICoreWebView2EnvironmentOptions5::EnableTrackingPrevention` is true.

See `ICoreWebView2EnvironmentOptions5::EnableTrackingPrevention` for more details.

C++

```
void  
SettingsComponent::SetTrackingPreventionLevel(COREWEBVIEW2_TRACKING_PREVENTION_LEVEL value)  
{  
    wil::com_ptr<ICoreWebView2_13> webView2_13;  
    webView2_13 = m_webView.try_query<ICoreWebView2_13>();  
  
    if (webView2_13)  
    {  
        wil::com_ptr<ICoreWebView2Profile> profile;  
        CHECK_FAILURE(webView2_13->get_Profile(&profile));  
  
        auto profile3 = profile.try_query<ICoreWebView2Profile3>();  
        if (profile3)  
        {  
            CHECK_FAILURE(profile3->put_PREFERREDTRACKINGPREVENTIONLEVEL(value));  
            MessageBox(
```

```
    nullptr, L"Tracking prevention level is set successfully",
    L"Tracking Prevention Level", MB_OK);
}
}
```

## put\_PreferredTrackingPreventionLevel

Set the `PreferredTrackingPreventionLevel` property.

```
public HRESULT
put_PreferredTrackingPreventionLevel(COREWEBVIEW2_TRACKING_PREVENTION_LEVEL value)
```

If `ICoreWebView2EnvironmentOptions5::EnableTrackingPrevention` is false, this property will be changed and persisted to the profile but the WebView2 ignores the level silently.

---

## Feedback

Was this page helpful?



# interface ICoreWebView2Profile4

Article • 02/26/2024

```
interface ICoreWebView2Profile4
: public ICoreWebView2Profile3
```

This is the [ICoreWebView2Profile](#) interface for the permission management APIs.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">GetNonDefaultPermissionSettings</a>	Invokes the handler with a collection of all nondefault permission settings.
<a href="#">SetPermissionState</a>	Sets permission state for the given permission kind and origin asynchronously.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1661.34
WebView2 Win32 Prerelease	1.0.1671

## Members

### [GetNonDefaultPermissionSettings](#)

Invokes the handler with a collection of all nondefault permission settings.

```
public HRESULT
```

```
GetNonDefaultPermissionSettings(ICoreWebView2GetNonDefaultPermissionSettings
```

```
CompletedHandler * completedHandler)
```

Use this method to get the permission state set in the current and previous sessions.

C++

```
CHECK_FAILURE(webView2_13->add_DOMContentLoaded(
    Callback<ICoreWebView2DOMContentLoadedEventHandler>(
        [this](
            ICoreWebView2* sender,
            ICoreWebView2DOMContentLoadedEventArgs* args) -> HRESULT
    {
        wil::unique_cotaskmem_string source;
        CHECK_FAILURE(sender->get_Source(&source));
        // Permission management APIs are only used on the app's
        // permission management page.
        if (source.get() != m_sampleUri)
        {
            return S_OK;
        }
        // Get all the nondefault permissions and post them to the
        // app's permission management page. The permission
management
        // page can present a list of custom permissions set for
this
        // profile and let the end user modify them.
        CHECK_FAILURE(m_webViewProfile4-
>GetNonDefaultPermissionSettings()

Callback<ICoreWebView2GetNonDefaultPermissionSettingsCompletedHandler>(
    [this, sender](
        HRESULT code,
        ICoreWebView2PermissionSettingCollectionView*
collectionView)
        -> HRESULT
    {
        UINT32 count;
        collectionView->get_Count(&count);
        for (UINT32 i = 0; i < count; i++)
        {
            wil::com_ptr<ICoreWebView2PermissionSetting>
setting;
            CHECK_FAILURE(collectionView-
>GetValueAtIndex(i, &setting));
            COREWEBVIEW2_PERMISSION_KIND kind;
            CHECK_FAILURE(setting-
>get_PermissionKind(&kind));
            PermissionKindToString(kind);
            COREWEBVIEW2_PERMISSION_STATE state;
            CHECK_FAILURE(setting-
>get_PermissionState(&state));
            wil::unique_cotaskmem_string origin;
```

```

        CHECK_FAILURE(setting-
>get_PermissionOrigin(&origin));
        std::wstring state_string =
PermissionStateToString(state);
        std::wstring reply = L"
{\\"PermissionSetting\\": \\""
kind_string + L", " +
origin.get() +
L"\\"}";
        CHECK_FAILURE(sender-
>PostWebMessageAsJson(reply.c_str()));
    }
    return S_OK;
}
.Get());
return S_OK;
)
.Get(),
&m_DOMContentLoadedToken));

```

## SetPermissionState

Sets permission state for the given permission kind and origin asynchronously.

```

public HRESULT SetPermissionState(COREWEBVIEW2_PERMISSION_KIND
permissionKind, LPCWSTR origin, COREWEBVIEW2_PERMISSION_STATE state,
ICoreWebView2SetPermissionStateCompletedHandler * completedHandler)

```

The change persists across sessions until it is changed by another call to `SetPermissionState`, or by setting the `State` property in `PermissionRequestedEventArgs`. Setting the state to `COREWEBVIEW2_PERMISSION_STATE_DEFAULT` will erase any state saved in the profile and restore the default behavior. The origin should have a valid scheme and host (e.g. "https://www.example.com"), otherwise the method fails with `E_INVALIDARG`. Additional URI parts like path and fragment are ignored. For example, "https://www.example.com/app1/index.html/" is treated the same as "https://www.example.com". See the [MDN origin definition](#) for more details.

C++

```

// The app's permission management page can provide a way for the end user
to
// change permissions. For example, a button on the page can trigger a
dialog,
// where the user specifies the desired permission kind, origin, and state.
void ScenarioPermissionManagement::ShowSetPermissionDialog()
{

```

```
    PermissionDialog dialog(m_appWindow->GetMainWindow(), permissionKinds,
permissionStates);
    if (dialog.confirmed && m_webViewProfile4)
    {
        // Example: m_webViewProfile4->SetPermissionState(
        //     COREWEBVIEW2_PERMISSION_KIND_GEOLOCATION,
        //     L"https://example.com",
        //     COREWEBVIEW2_PERMISSION_STATE_DENY
        //     SetPermissionStateCallback);
        CHECK_FAILURE(m_webViewProfile4->SetPermissionState(
            dialog.kind, dialog.origin.c_str(), dialog.state,
            Callback<ICoreWebView2SetPermissionStateCompletedHandler>(
                [this](HRESULT error) -> HRESULT
                {
                    m_webView->Reload();
                    return S_OK;
                }
                .Get())));
    }
}
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Profile5

Article • 02/26/2024

```
interface ICoreWebView2Profile5
: public ICoreWebView2Profile4
```

This is the [ICoreWebView2Profile](#) interface for cookie manager.

## Summary

[ ] [Expand table](#)

Members	Descriptions
<a href="#">get_CookieManager</a>	Get the cookie manager for the profile.

## Applies to

[ ] [Expand table](#)

Product	Introduced
WebView2 Win32	1.0.1774.30
WebView2 Win32 Prerelease	1.0.1777

## Members

### [get\\_CookieManager](#)

Get the cookie manager for the profile.

```
public HRESULT get_CookieManager(ICoreWebView2CookieManager **  
cookieManager)
```

All CoreWebView2s associated with this profile share the same cookie values. Changes to cookies in this cookie manager apply to all CoreWebView2s associated with this

profile. See [ICoreWebView2CookieManager](#).

C++

```
auto webView2_13 = m_webView.try_query<ICoreWebView2_13>();
CHECK_FEATURE_RETURN_EMPTY(webView2_13);
wil::com_ptr<ICoreWebView2Profile> webView2Profile;
CHECK_FAILURE(webView2_13->get_Profile(&webView2Profile));
auto webView2Profile5 =
webView2Profile.try_query<ICoreWebView2Profile5>();
CHECK_FEATURE_RETURN_EMPTY(webView2Profile5);
CHECK_FAILURE(webView2Profile5-
>get_CookieManager(&m_cookieManager));
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Profile6

Article • 02/26/2024

```
interface ICoreWebView2Profile6
: public ICoreWebView2Profile5
```

Interfaces in profile for managing password-autosave and general-autofill.

## Summary

[ ] [Expand table](#)

Members	Descriptions
<a href="#">get_IsGeneralAutofillEnabled</a>	IsGeneralAutofillEnabled controls whether autofill for information like names, street and email addresses, phone numbers, and arbitrary input is enabled.
<a href="#">get_IsPasswordAutosaveEnabled</a>	IsPasswordAutosaveEnabled controls whether autosave for password information is enabled.
<a href="#">put_IsGeneralAutofillEnabled</a>	Set the IsGeneralAutofillEnabled property.
<a href="#">put_IsPasswordAutosaveEnabled</a>	Set the IsPasswordAutosaveEnabled property.

## Applies to

[ ] [Expand table](#)

Product	Introduced
WebView2 Win32	1.0.1823.32
WebView2 Win32 Prerelease	1.0.1829

## Members

### [get\\_IsGeneralAutofillEnabled](#)

`IsGeneralAutofillEnabled` controls whether autofill for information like names, street and email addresses, phone numbers, and arbitrary input is enabled.

```
public HRESULT get_IsGeneralAutofillEnabled(BOOL * value)
```

This excludes password and credit card information. When `IsGeneralAutofillEnabled` is false, no suggestions appear, and no new information is saved. When `IsGeneralAutofillEnabled` is true, information is saved, suggestions appear and clicking on one will populate the form fields. It will take effect immediately after setting. The default value is `TRUE`. This property has the same value as

`CoreWebView2Settings.IsGeneralAutofillEnabled`, and changing one will change the other. All `CoreWebView2`s with the same `CoreWebView2Profile` will share the same value for this property, so for the `CoreWebView2`s with the same profile, their `CoreWebView2Settings.IsGeneralAutofillEnabled` and `CoreWebView2Profile.IsGeneralAutofillEnabled` will always have the same value.

C++

```
// Get the profile object.
auto webView2_13 = m_webView.try_query<ICoreWebView2_13>();
CHECK_FEATURE_RETURN(webView2_13);
wil::com_ptr<ICoreWebView2Profile> webView2Profile;
CHECK_FAILURE(webView2_13->get_Profile(&webView2Profile));
CHECK_FEATURE_RETURN(webView2Profile);
auto webView2Profile6 =
webView2Profile.try_query<ICoreWebView2Profile6>();
CHECK_FEATURE_RETURN(webView2Profile6);

BOOL enabled;
CHECK_FAILURE(webView2Profile6-
>get_IsGeneralAutofillEnabled(&enabled));
// Set general-autofill property to the opposite value to
current value.
if (enabled)
{
    CHECK_FAILURE(webView2Profile6-
>put_IsGeneralAutofillEnabled(FALSE));
    MessageBox(
        nullptr,
        L"General autofill will be disabled immediately in all
WebView2 with the "
        L"same profile.",
        L"Profile settings change", MB_OK);
}
else
{
    CHECK_FAILURE(webView2Profile6-
>put_IsGeneralAutofillEnabled(TRUE));
```

```
        MessageBox(
            nullptr,
            L"General autofill will be enabled immediately in all
WebView2 with the "
            L"same profile.",
            L"Profile settings change", MB_OK);
    }
```

## get\_IsPasswordAutosaveEnabled

IsPasswordAutosaveEnabled controls whether autosave for password information is enabled.

```
public HRESULT get_IsPasswordAutosaveEnabled(BOOL * value)
```

The IsPasswordAutosaveEnabled property behaves independently of the IsGeneralAutofillEnabled property. When IsPasswordAutosaveEnabled is false, no new password data is saved and no Save/Update Password prompts are displayed. However, if there was password data already saved before disabling this setting, then that password information is auto-populated, suggestions are shown and clicking on one will populate the fields. When IsPasswordAutosaveEnabled is true, password information is auto-populated, suggestions are shown and clicking on one will populate the fields, new data is saved, and a Save/Update Password prompt is displayed. It will take effect immediately after setting. The default value is FALSE. This property has the same value as CoreWebView2Settings.IsPasswordAutosaveEnabled, and changing one will change the other. All CoreWebView2s with the same CoreWebView2Profile will share the same value for this property, so for the CoreWebView2s with the same profile, their

CoreWebView2Settings.IsPasswordAutosaveEnabled and

CoreWebView2Profile.IsPasswordAutosaveEnabled will always have the same value.

C++

```
// Get the profile object.
auto webView2_13 = m_webView.try_query<ICoreWebView2_13>();
CHECK_FEATURE_RETURN(webView2_13);
wil::com_ptr<ICoreWebView2Profile> webView2Profile;
CHECK_FAILURE(webView2_13->get_Profile(&webView2Profile));
CHECK_FEATURE_RETURN(webView2Profile);
auto webView2Profile6 =
webView2Profile.try_query<ICoreWebView2Profile6>();
CHECK_FEATURE_RETURN(webView2Profile6);

BOOL enabled;
CHECK_FAILURE(webView2Profile6-
>get_IsPasswordAutosaveEnabled(&enabled));
```

```
// Set password-autosave property to the opposite value to
current value.
    if (enabled)
    {
        CHECK_FAILURE(webView2Profile6-
>put_IsPasswordAutosaveEnabled(FALSE));
        MessageBox(
            nullptr,
            L>Password autosave will be disabled immediately in all
        "
            L"WebView2 with the same profile.",
            L"Profile settings change", MB_OK);
    }
    else
    {
        CHECK_FAILURE(webView2Profile6-
>put_IsPasswordAutosaveEnabled(TRUE));
        MessageBox(
            nullptr,
            L>Password autosave will be enabled immediately in all "
            L"WebView2 with the same profile.",
            L"Profile settings change", MB_OK);
    }
}
```

## put\_IsGeneralAutofillEnabled

Set the IsGeneralAutofillEnabled property.

```
public HRESULT put_IsGeneralAutofillEnabled(BOOL value)
```

## put\_IsPasswordAutosaveEnabled

Set the IsPasswordAutosaveEnabled property.

```
public HRESULT put_IsPasswordAutosaveEnabled(BOOL value)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Profile7

Article • 02/26/2024

```
interface ICoreWebView2Profile7
: public ICoreWebView2Profile6
```

Interfaces in profile for managing browser extensions.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">AddBrowserExtension</a>	Adds the <a href="#">browser extension</a> using the extension path for unpacked extensions from the local device.
<a href="#">GetBrowserExtensions</a>	Gets a snapshot of the set of extensions installed at the time <a href="#">GetBrowserExtensions</a> is called.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.2210.55
WebView2 Win32 Prerelease	1.0.2194

## Members

### AddBrowserExtension

Adds the [browser extension](#) using the extension path for unpacked extensions from the local device.

```
public HRESULT AddBrowserExtension(LPCWSTR extensionFolderPath,  
ICoreWebView2ProfileAddBrowserExtensionCompletedHandler * handler)
```

Extension is running right after installation. The extension folder path is the topmost folder of an unpacked browser extension and contains the browser extension manifest file. If the `extensionFolderPath` is an invalid path or doesn't contain the extension `manifest.json` file, this function will return `ERROR_FILE_NOT_FOUND` to callers. Installed extension will default `.IsEnabled` to true. When `AreBrowserExtensionsEnabled` is `FALSE`, `AddBrowserExtension` will fail and return `HRESULT_ERROR_NOT_SUPPORTED`. During installation, the content of the extension is not copied to the user data folder. Once the extension is installed, changing the content of the extension will cause the extension to be removed from the installed profile. When an extension is added the extension is persisted in the corresponding profile. The extension will still be installed the next time you use this profile. When an extension is installed from a folder path, adding the same extension from the same folder path means reinstalling this extension. When two extensions with the same Id are installed, only the later installed extension will be kept.

Extensions that are designed to include any UI interactions (e.g. icon, badge, pop up, etc.) can be loaded and used but will have missing UI entry points due to the lack of browser UI elements to host these entry points in WebView2.

The following summarizes the possible error values that can be returned from `AddBrowserExtension` and a description of why these errors occur.

[+] Expand table

Error value	Description
<code>HRESULT_FROM_WIN32(ERROR_NOT_SUPPORTED)</code>	Extensions are disabled.
<code>HRESULT_FROM_WIN32(ERROR_FILE_NOT_FOUND)</code>	Cannot find <code>manifest.json</code> file or it is not a valid extension manifest.
<code>E_ACCESSDENIED</code>	Cannot load extension with file or directory name starting with "_", reserved for use by the system.
<code>E_FAIL</code>	Extension failed to install with other unknown reasons.

## GetBrowserExtensions

Gets a snapshot of the set of extensions installed at the time `GetBrowserExtensions` is called.

```
public HRESULT  
GetBrowserExtensions(ICoreWebView2ProfileGetBrowserExtensionsCompletedHandler * handler)
```

If an extension is installed or uninstalled after `GetBrowserExtensions` completes, the list returned by `GetBrowserExtensions` remains the same. When `AreBrowserExtensionsEnabled` is `FALSE`, `GetBrowserExtensions` won't return any extensions on current user profile.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Profile8

Article • 02/26/2024

```
interface ICoreWebView2Profile8
: public ICoreWebView2Profile7
```

This is the profile interface that manages profile deletion.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">add_Deleted</a>	Add an event handler for the <code>Deleted</code> event.
<a href="#">Delete</a>	After the API is called, the profile will be marked for deletion.
<a href="#">remove_Deleted</a>	Removes an event handler previously added with <code>add_Deleted</code> .

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.2210.55
WebView2 Win32 Prerelease	1.0.2194

## Members

### [add\\_Deleted](#)

Add an event handler for the `Deleted` event.

```
public HRESULT add_Deleted(ICoreWebView2ProfileDeletedEventHandler *
eventHandler, EventRegistrationToken * token)
```

The `Deleted` event is raised when the profile is marked for deletion. When this event is raised, the `CoreWebView2Profile` and its corresponding `CoreWebView2s` have been closed, and cannot be used anymore.

C++

```
auto webView2_13 = m_webView.try_query<ICoreWebView2_13>();
CHECK_FEATURE_RETURN_EMPTY(webView2_13);
wil::com_ptr<ICoreWebView2Profile> webView2Profile;
CHECK_FAILURE(webView2_13->get_Profile(&webView2Profile));
CHECK_FEATURE_RETURN_EMPTY(webView2Profile);
auto webView2Profile8 = webView2Profile.try_query<ICoreWebView2Profile8>()
();
CHECK_FEATURE_RETURN_EMPTY(webView2Profile8);
CHECK_FAILURE(webView2Profile8->add_Deleted(
    Microsoft::WRL::Callback<ICoreWebView2ProfileDeletedEventHandler>(
        [this](ICoreWebView2Profile* sender, IUnknown* args)
    {
        RunAsync(
            [this]()
        {
            std::wstring message = L"The profile has been marked
for deletion. Any "
                                L"associated webview2 objects
will be closed.";
            MessageBox(m_mainWindow, message.c_str(), L"webview2
closed", MB_OK);
            CloseAppWindow();
        });
        return S_OK;
    })
    .Get(),
    nullptr));
```

## Delete

After the API is called, the profile will be marked for deletion.

```
public HRESULT Delete()
```

The local profile's directory will be deleted at browser process exit. If it fails to delete, because something else is holding the files open, WebView2 will try to delete the profile at all future browser process starts until successful. The corresponding `CoreWebView2s` will be closed and the `CoreWebView2Profile.Deleted` event will be raised. See `CoreWebView2Profile.Deleted` for more information. If you try to create a new profile with the same name as an existing profile that has been marked as deleted but hasn't

yet been deleted, profile creation will fail with  
HRESULT\_FROM\_WIN32(ERROR\_DELETE\_PENDING).

C++

```
CHECK_FEATURE_RETURN(m_webView2_13);
wil::com_ptr<ICoreWebView2Profile> webView2ProfileBase;
m_webView2_13->get_Profile(&webView2ProfileBase);
CHECK_FEATURE_RETURN(webView2ProfileBase);
auto webView2Profile =
webView2ProfileBase.try_query<ICoreWebView2Profile8>();
CHECK_FEATURE_RETURN(webView2Profile);
webView2Profile->Delete();
```

## remove\_Deleted

Removes an event handler previously added with `add_Deleted`.

```
public HRESULT remove_Deleted(EventRegistrationToken token)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2ScriptDialogOpeningEventArgs

Article • 02/26/2024

```
interface ICoreWebView2ScriptDialogOpeningEventArgs
: public IUnknown
```

Event args for the `ScriptDialogOpening` event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Accept</a>	The host may run this to respond with OK to <code>confirm</code> , <code>prompt</code> , and <code>beforeunload</code> dialogs.
<a href="#">get_DefaultText</a>	The second parameter passed to the JavaScript prompt dialog.
<a href="#">get_Kind</a>	The kind of JavaScript dialog box.
<a href="#">get_Message</a>	The message of the dialog box.
<a href="#">get_ResultText</a>	The return value from the JavaScript prompt function if <code>Accept</code> is run.
<a href="#">get_Uri</a>	The URI of the page that requested the dialog box.
<a href="#">GetDeferral</a>	Returns an <code>ICoreWebView2Deferral</code> object.
<a href="#">put_ResultText</a>	Sets the <code>ResultText</code> property.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430

Product	Introduced
WebView2 Win32 Prerelease	0.9.488

## Members

### Accept

The host may run this to respond with **OK** to `confirm`, `prompt`, and `beforeunload` dialogs.

```
public HRESULT Accept()
```

Do not run this method to indicate cancel. From JavaScript, this means that the `confirm` and `beforeunload` function returns `TRUE` if `Accept` is run. And for the prompt function it returns the value of `ResultText` if `Accept` is run and otherwise returns `FALSE`.

### get\_DefaultText

The second parameter passed to the JavaScript prompt dialog.

```
public HRESULT get_DefaultText(LPWSTR * defaultText)
```

The result of the prompt JavaScript function uses this value as the default value.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

### get\_Kind

The kind of JavaScript dialog box.

```
public HRESULT get_Kind(COREWEBVIEW2_SCRIPT_DIALOG_KIND * kind)
```

`alert`, `confirm`, `prompt`, or `beforeunload`.

### get\_Message

The message of the dialog box.

```
public HRESULT get_Message(LPWSTR * message)
```

From JavaScript this is the first parameter passed to `alert`, `confirm`, and `prompt` and is empty for `beforeunload`.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_ResultText

The return value from the JavaScript prompt function if `Accept` is run.

```
public HRESULT get_ResultText(LPWSTR * resultText)
```

This value is ignored for dialog kinds other than prompt. If `Accept` is not run, this value is ignored and `FALSE` is returned from prompt.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_Uri

The URI of the page that requested the dialog box.

```
public HRESULT get_Uri(LPWSTR * uri)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## GetDeferral

Returns an `ICoreWebView2Deferral` object.

```
public HRESULT GetDeferral(ICoreWebView2Deferral ** deferral)
```

Use this operation to complete the event at a later time.

## put\_ResultText

Sets the `ResultText` property.

```
public HRESULT put_ResultText(LPCWSTR resultText)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2ScriptException

Article • 02/26/2024

```
interface ICoreWebView2ScriptException
: public IUnknown
```

This interface represents a JavaScript exception.

## Summary

[ ] Expand table

Members	Descriptions
<a href="#">get_ColumnNumber</a>	The column number of the source where the exception occurred.
<a href="#">get_LineNumber</a>	The line number of the source where the exception occurred.
<a href="#">get_Message</a>	The Message is the exception's message and potentially stack.
<a href="#">get_Name</a>	The Name is the exception's class name.
<a href="#">get_ToJson</a>	This will return all details of the exception as a JSON string.

If the CoreWebView2.ExecuteScriptWithResult result has Succeeded as false, you can use the result's Exception property to get the script exception.

## Applies to

[ ] Expand table

Product	Introduced
WebView2 Win32	1.0.2277.86
WebView2 Win32 Prerelease	1.0.2357

## Members

## **get\_ColumnNumber**

The column number of the source where the exception occurred.

```
| public HRESULT get_ColumnNumber(UINT32 * value)
```

In the JSON it is `exceptionDetail.columnNumber`. Note that this position starts at 0.

## **get\_LineNumber**

The line number of the source where the exception occurred.

```
| public HRESULT get_LineNumber(UINT32 * value)
```

In the JSON it is `exceptionDetail.lineNumber`. Note that this position starts at 0.

## **get\_Message**

The Message is the exception's message and potentially stack.

```
| public HRESULT get_Message(LPWSTR * value)
```

In the JSON it is `exceptionDetail.exception.description`. This is the empty string if the exception doesn't have a description. This can happen if the script throws a non-Error object such as `throw "abc";`

## **get\_Name**

The Name is the exception's class name.

```
| public HRESULT get_Name(LPWSTR * value)
```

In the JSON it is `exceptionDetail.exception.className`. This is the empty string if the exception doesn't have a class name. This can happen if the script throws a non-Error object such as `throw "abc";`

## **get\_ToJson**

This will return all details of the exception as a JSON string.

```
public HRESULT get_ToJson(LPWSTR * value)
```

In the case that script has thrown a non-Error object such as `throw "abc";` or any other non-Error object, you can get object specific properties.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2ServerCertificateErrorDetectedEventArgs

Article • 02/26/2024

```
interface ICoreWebView2ServerCertificateErrorDetectedEventArgs
: public IUnknown
```

Event args for the `ServerCertificateErrorDetected` event.

## Summary

Expand table

Members	Descriptions
<code>get_Action</code>	The action of the server certificate error detection.
<code>get_ErrorStatus</code>	The TLS error code for the invalid certificate.
<code>get_RequestUri</code>	URI associated with the request for the invalid certificate.
<code>get_ServerCertificate</code>	Returns the server certificate.
<code>GetDeferral</code>	Returns an <code>ICoreWebView2Deferral</code> object.
<code>put_Action</code>	Sets the <code>Action</code> property.

## Applies to

Expand table

Product	Introduced
WebView2 Win32	1.0.1245.22
WebView2 Win32 Prerelease	1.0.1248

# Members

## get\_Action

The action of the server certificate error detection.

```
public HRESULT get_Action(COREWEBVIEW2_SERVER_CERTIFICATE_ERROR_ACTION * value)
```

The default value is `COREWEBVIEW2_SERVER_CERTIFICATE_ERROR_ACTION_DEFAULT`.

## get\_ErrorStatus

The TLS error code for the invalid certificate.

```
public HRESULT get_ErrorStatus(COREWEBVIEW2_WEB_ERROR_STATUS * value)
```

## get\_RequestUri

URI associated with the request for the invalid certificate.

```
public HRESULT get_RequestUri(LPWSTR * value)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_ServerCertificate

Returns the server certificate.

```
public HRESULT get_ServerCertificate(ICoreWebView2Certificate ** value)
```

## GetDeferral

Returns an `ICoreWebView2Deferral` object.

```
public HRESULT GetDeferral(ICoreWebView2Deferral ** deferral)
```

Use this operation to complete the event at a later time.

## put\_Action

Sets the `Action` property.

```
public HRESULT put_Action(COREWEBVIEW2_SERVER_CERTIFICATE_ERROR_ACTION  
value)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Settings

Article • 02/26/2024

```
interface ICoreWebView2Settings  
: public IUnknown
```

Defines properties that enable, disable, or modify WebView features.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_AreDefaultContextMenuEnabled</a>	The <code>AreDefaultContextMenuEnabled</code> property is used to prevent default context menus from being shown to user in WebView.
<a href="#">get_AreDefaultScriptDialogsEnabled</a>	<code>AreDefaultScriptDialogsEnabled</code> is used when loading a new HTML document.
<a href="#">get_AreDevToolsEnabled</a>	<code>AreDevToolsEnabled</code> controls whether the user is able to use the context menu or keyboard shortcuts to open the DevTools window.
<a href="#">get_AreHostObjectsAllowed</a>	The <code>AreHostObjectsAllowed</code> property is used to control whether host objects are accessible from the page in WebView.
<a href="#">get_IsBuiltInErrorPageEnabled</a>	The <code>IsBuiltInErrorPageEnabled</code> property is used to disable built in error page for navigation failure and render process failure.
<a href="#">get_IsScriptEnabled</a>	Controls if running JavaScript is enabled in all future navigations in the WebView.
<a href="#">get_IsStatusBarEnabled</a>	<code>IsStatusBarEnabled</code> controls whether the status bar is displayed.
<a href="#">get_IsWebMessageEnabled</a>	The <code>IsWebMessageEnabled</code> property is used when loading a new HTML document.
<a href="#">get_IsZoomControlEnabled</a>	The <code>IsZoomControlEnabled</code> property is used to prevent the user from impacting the zoom of the WebView.

Members	Descriptions
<code>put_AreDefaultContextMenusEnabled</code>	Sets the <code>AreDefaultContextMenusEnabled</code> property.
<code>put_AreDefaultScriptDialogsEnabled</code>	Sets the <code>AreDefaultScriptDialogsEnabled</code> property.
<code>put_AreDevToolsEnabled</code>	Sets the <code>AreDevToolsEnabled</code> property.
<code>put_AreHostObjectsAllowed</code>	Sets the <code>AreHostObjectsAllowed</code> property.
<code>put_IsBuiltInErrorPageEnabled</code>	Sets the <code>IsBuiltInErrorPageEnabled</code> property.
<code>put_IsScriptEnabled</code>	Sets the <code>IsScriptEnabled</code> property.
<code>put_IsStatusBarEnabled</code>	Sets the <code>IsStatusBarEnabled</code> property.
<code>put_IsWebMessageEnabled</code>	Sets the <code>IsWebMessageEnabled</code> property.
<code>put_IsZoomControlEnabled</code>	Sets the <code>IsZoomControlEnabled</code> property.

Changes to `IsGeneralAutofillEnabled` and `IsPasswordAutosaveEnabled` apply immediately, while other setting changes made after `NavigationStarting` event do not apply until the next top-level navigation.

## Applies to

[ ] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### `get_AreDefaultContextMenusEnabled`

The `AreDefaultContextMenusEnabled` property is used to prevent default context menus from being shown to user in WebView.

```
public HRESULT get_AreDefaultContextMenusEnabled(BOOL * enabled)
```

The default value is `TRUE`.

C++

```
    BOOL allowContextMenus;
    CHECK_FAILURE(m_settings-
>get_AreDefaultContextMenuEnabled(&allowContextMenus));
    if (allowContextMenus)
    {
        CHECK_FAILURE(m_settings-
>put_AreDefaultContextMenuEnabled(FALSE));
        MessageBox(
            nullptr, L"Context menus will be disabled after the next
navigation.",
            L"Settings change", MB_OK);
    }
    else
    {
        CHECK_FAILURE(m_settings-
>put_AreDefaultContextMenuEnabled(TRUE));
        MessageBox(
            nullptr, L"Context menus will be enabled after the next
navigation.",
            L"Settings change", MB_OK);
    }
```

## get\_AreDefaultScriptDialogsEnabled

`AreDefaultScriptDialogsEnabled` is used when loading a new HTML document.

```
public HRESULT get_AreDefaultScriptDialogsEnabled(BOOL *
areDefaultScriptDialogsEnabled)
```

If set to `FALSE`, WebView2 does not render the default JavaScript dialog box (Specifically those displayed by the JavaScript alert, confirm, prompt functions and `beforeunload` event). Instead, if an event handler is set using `add_ScriptDialogOpening`, WebView sends an event that contains all of the information for the dialog and allow the host app to show a custom UI. The default value is `TRUE`.

## get\_AreDevToolsEnabled

`AreDevToolsEnabled` controls whether the user is able to use the context menu or keyboard shortcuts to open the DevTools window.

```
public HRESULT get_AreDevToolsEnabled(BOOL * areDevToolsEnabled)
```

The default value is `TRUE`.

## get\_AreHostObjectsAllowed

The `AreHostObjectsAllowed` property is used to control whether host objects are accessible from the page in WebView.

```
public HRESULT get_AreHostObjectsAllowed(BOOL * allowed)
```

The default value is `TRUE`.

C++

```
    BOOL allowHostObjects;
    CHECK_FAILURE(m_settings-
>get_AreHostObjectsAllowed(&allowHostObjects));
    if (allowHostObjects)
    {
        CHECK_FAILURE(m_settings->put_AreHostObjectsAllowed(FALSE));
        MessageBox(
            nullptr,
            L"Access to host objects will be denied after the next
navigation.",
            L"Settings change", MB_OK);
    }
    else
    {
        CHECK_FAILURE(m_settings->put_AreHostObjectsAllowed(TRUE));
        MessageBox(
            nullptr,
            L"Access to host objects will be allowed after the next
navigation.",
            L"Settings change", MB_OK);
    }
```

## get\_IsBuiltInErrorPageEnabled

The `IsBuiltInErrorPageEnabled` property is used to disable built in error page for navigation failure and render process failure.

```
public HRESULT get_IsBuiltInErrorPageEnabled(BOOL * enabled)
```

When disabled, a blank page is displayed when the related error happens. The default value is `TRUE`.

C++

```
    BOOL enabled;
    CHECK_FAILURE(m_settings-
```

```
>get_IsBuiltInErrorPageEnabled(&enabled));
    if (enabled)
    {
        CHECK_FAILURE(m_settings-
>put_IsBuiltInErrorPageEnabled(FALSE));
        MessageBox(
            nullptr, L"Built-in error page will be disabled for
future navigation.",
            L"Settings change", MB_OK);
    }
    else
    {
        CHECK_FAILURE(m_settings-
>put_IsBuiltInErrorPageEnabled(TRUE));
        MessageBox(
            nullptr, L"Built-in error page will be enabled for
future navigation.",
            L"Settings change", MB_OK);
    }
}
```

## get\_IsScriptEnabled

Controls if running JavaScript is enabled in all future navigations in the WebView.

```
public HRESULT get_IsScriptEnabled(BOOL * isScriptEnabled)
```

This only affects scripts in the document. Scripts injected with `ExecuteScript` runs even if script is disabled. The default value is `TRUE`.

C++

```
// Changes to settings will apply at the next navigation,
which includes the
// navigation after a NavigationStarting event. We can use
this to change
// settings according to what site we're visiting.
if (ShouldBlockScriptForUri(uri.get()))
{
    m_settings->put_IsScriptEnabled(FALSE);
}
else
{
    m_settings->put_IsScriptEnabled(m_isScriptEnabled);
}
```

## get\_IsStatusBarEnabled

`IsStatusBarEnabled` controls whether the status bar is displayed.

```
public HRESULT get_IsStatusBarEnabled(BOOL * isStatusBarEnabled)
```

The status bar is usually displayed in the lower left of the WebView and shows things such as the URI of a link when the user hovers over it and other information. The default value is `TRUE`. The status bar UI can be altered by web content and should not be considered secure.

## get\_IsWebMessageEnabled

The `IsWebMessageEnabled` property is used when loading a new HTML document.

```
public HRESULT get_IsWebMessageEnabled(BOOL * isWebMessageEnabled)
```

If set to `TRUE`, communication from the host to the top-level HTML document of the WebView is allowed using `PostWebMessageAsJson`, `PostWebMessageAsString`, and message event of `window.chrome.webview`. For more information, navigate to `PostWebMessageAsJson`. Communication from the top-level HTML document of the WebView to the host is allowed using the `postMessage` function of `window.chrome.webview` and `add_WebMessageReceived` method. For more information, navigate to `add_WebMessageReceived`. If set to false, then communication is disallowed. `PostWebMessageAsJson` and `PostWebMessageAsString` fails with `E_ACCESSDENIED` and `window.chrome.webview.postMessage` fails by throwing an instance of an `Error` object. The default value is `TRUE`.

C++

```
ComPtr<ICoreWebView2Settings> settings;
CHECK_FAILURE(m_webView->get_Settings(&settings));

CHECK_FAILURE(settings->put_IsWebMessageEnabled(TRUE));
```

## get\_IsZoomControlEnabled

The `IsZoomControlEnabled` property is used to prevent the user from impacting the zoom of the WebView.

```
public HRESULT get_IsZoomControlEnabled(BOOL * enabled)
```

When disabled, the user is not able to zoom using Ctrl++, Ctrl--, or Ctrl+mouse wheel, but the zoom is set using `ZoomFactor` API. The default value is `TRUE`.

C++

```
BOOL zoomControlEnabled;
CHECK_FAILURE(m_settings-
>get_IsZoomControlEnabled(&zoomControlEnabled));
if (zoomControlEnabled)
{
    CHECK_FAILURE(m_settings->put_IsZoomControlEnabled(FALSE));
    MessageBox(
        nullptr, L"Zoom control will be disabled after the next
navigation.",
        L"Settings change", MB_OK);
}
else
{
    CHECK_FAILURE(m_settings->put_IsZoomControlEnabled(TRUE));
    MessageBox(
        nullptr, L"Zoom control will be enabled after the next
navigation.",
        L"Settings change", MB_OK);
}
```

## **put\_AreDefaultContextMenusEnabled**

Sets the `AreDefaultContextMenusEnabled` property.

```
public HRESULT put_AreDefaultContextMenusEnabled(BOOL enabled)
```

## **put\_AreDefaultScriptDialogsEnabled**

Sets the `AreDefaultScriptDialogsEnabled` property.

```
public HRESULT put_AreDefaultScriptDialogsEnabled(BOOL
areDefaultScriptDialogsEnabled)
```

## **put\_AreDevToolsEnabled**

Sets the `AreDevToolsEnabled` property.

```
public HRESULT put_AreDevToolsEnabled(BOOL areDevToolsEnabled)
```

## **put\_AreHostObjectsAllowed**

Sets the `AreHostObjectsAllowed` property.

```
public HRESULT put_AreHostObjectsAllowed(BOOL allowed)
```

## **put\_IsBuiltInErrorPageEnabled**

Sets the `IsBuiltInErrorPageEnabled` property.

```
public HRESULT put_IsBuiltInErrorPageEnabled(BOOL enabled)
```

## **put\_IsScriptEnabled**

Sets the `IsScriptEnabled` property.

```
public HRESULT put_IsScriptEnabled(BOOL isScriptEnabled)
```

## **put\_IsStatusBarEnabled**

Sets the `IsStatusBarEnabled` property.

```
public HRESULT put_IsStatusBarEnabled(BOOL isStatusBarEnabled)
```

## **put\_IsWebMessageEnabled**

Sets the `IsWebMessageEnabled` property.

```
public HRESULT put_IsWebMessageEnabled(BOOL isWebMessageEnabled)
```

## **put\_IsZoomControlEnabled**

Sets the `IsZoomControlEnabled` property.

```
public HRESULT put_IsZoomControlEnabled(BOOL enabled)
```

---

## **Feedback**

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Settings2

Article • 02/26/2024

```
interface ICoreWebView2Settings2
: public ICoreWebView2Settings
```

A continuation of the [ICoreWebView2Settings](#) interface that manages the user agent.

## Summary

[+] [Expand table](#)

Members	Descriptions
<a href="#">get_UserAgent</a>	Returns the User Agent.
<a href="#">put_UserAgent</a>	Sets the <code>UserAgent</code> property.

## Applies to

[+] [Expand table](#)

Product	Introduced
WebView2 Win32	1.0.864.35
WebView2 Win32 Prerelease	1.0.824

## Members

### get\_UserAgent

Returns the User Agent.

```
public HRESULT get_UserAgent(LPWSTR * userAgent)
```

The default value is the default User Agent of the Microsoft Edge browser.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

C++

```
if (m_settings2)
{
    static const PCWSTR url_compare_example =
L"fourthcoffee.com";
    wil::unique_bstr domain = GetDomainOfUri(uri.get());
    const wchar_t* domains = domain.get();

    if (wcscmp(url_compare_example, domains) == 0)
    {
        SetUserAgent(L"example_navigation_ua");
    }
}
```

## put\_UserAgent

Sets the `UserAgent` property.

```
public HRESULT put_UserAgent(LPCWSTR userAgent)
```

This property may be overridden if the User-Agent header is set in a request. If the parameter is empty the User Agent will not be updated and the current User Agent will remain. Returns `HRESULT_FROM_WIN32(ERROR_INVALID_STATE)` if the owning WebView is closed.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Settings3

Article • 02/26/2024

```
interface ICoreWebView2Settings3
: public ICoreWebView2Settings2
```

A continuation of the [ICoreWebView2Settings](#) interface that manages whether browser accelerator keys are enabled.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_AreBrowserAcceleratorKeysEnabled</a>	When this setting is set to FALSE, it disables all accelerator keys that access features specific to a web browser, including but not limited to:
<a href="#">put_AreBrowserAcceleratorKeysEnabled</a>	Sets the <code>AreBrowserAcceleratorKeysEnabled</code> property.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.864.35
WebView2 Win32 Prerelease	1.0.865

## Members

### [get\\_AreBrowserAcceleratorKeysEnabled](#)

When this setting is set to FALSE, it disables all accelerator keys that access features specific to a web browser, including but not limited to:

```
public HRESULT get_AreBrowserAcceleratorKeysEnabled(BOOL *  
areBrowserAcceleratorKeysEnabled)
```

- Ctrl-F and F3 for Find on Page
- Ctrl-P for Print
- Ctrl-R and F5 for Reload
- Ctrl-Plus and Ctrl-Minus for zooming
- Ctrl-Shift-C and F12 for DevTools
- Special keys for browser functions, such as Back, Forward, and Search

It does not disable accelerator keys related to movement and text editing, such as:

- Home, End, Page Up, and Page Down
- Ctrl-X, Ctrl-C, Ctrl-V
- Ctrl-A for Select All
- Ctrl-Z for Undo

Those accelerator keys will always be enabled unless they are handled in the `AcceleratorKeyPressed` event.

This setting has no effect on the `AcceleratorKeyPressed` event. The event will be fired for all accelerator keys, whether they are enabled or not.

The default value for `AreBrowserAcceleratorKeysEnabled` is TRUE.

C++

```
CHECK_FEATURE_RETURN(m_settings3);  
  
    BOOL enabled;  
    CHECK_FAILURE(m_settings3-  
>get_AreBrowserAcceleratorKeysEnabled(&enabled));  
    if (enabled)  
    {  
        CHECK_FAILURE(m_settings3-  
>put_AreBrowserAcceleratorKeysEnabled(FALSE));  
        MessageBox(  
            nullptr,  
            L"Browser-specific accelerator keys will be disabled  
after the next "  
            L"navigation.",
```

```
        L"Settings change", MB_OK);
    }
    else
    {
        CHECK_FAILURE(m_settings3-
>put_AreBrowserAcceleratorKeysEnabled(TRUE));
        MessageBox(
            nullptr,
            L"Browser-specific accelerator keys will be enabled
after the next "
            L"navigation.",
            L"Settings change", MB_OK);
    }
}
```

## put\_AreBrowserAcceleratorKeysEnabled

Sets the `AreBrowserAcceleratorKeysEnabled` property.

```
public HRESULT put_AreBrowserAcceleratorKeysEnabled(BOOL
areBrowserAcceleratorKeysEnabled)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Settings4

Article • 02/26/2024

```
interface ICoreWebView2Settings4
: public ICoreWebView2Settings3
```

A continuation of the [ICoreWebView2Settings](#) interface to manage autofill.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_IsGeneralAutofillEnabled</a>	IsGeneralAutofillEnabled controls whether autofill for information like names, street and email addresses, phone numbers, and arbitrary input is enabled.
<a href="#">get_IsPasswordAutosaveEnabled</a>	IsPasswordAutosaveEnabled controls whether autosave for password information is enabled.
<a href="#">put_IsGeneralAutofillEnabled</a>	Set the IsGeneralAutofillEnabled property.
<a href="#">put_IsPasswordAutosaveEnabled</a>	Set the IsPasswordAutosaveEnabled property.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.902.49
WebView2 Win32 Prerelease	1.0.902

## Members

### [get\\_IsGeneralAutofillEnabled](#)

`IsGeneralAutofillEnabled` controls whether autofill for information like names, street and email addresses, phone numbers, and arbitrary input is enabled.

```
public HRESULT get_IsGeneralAutofillEnabled(BOOL * value)
```

This excludes password and credit card information. When `IsGeneralAutofillEnabled` is false, no suggestions appear, and no new information is saved. When `IsGeneralAutofillEnabled` is true, information is saved, suggestions appear and clicking on one will populate the form fields. It will take effect immediately after setting. The default value is `TRUE`. This property has the same value as

`CoreWebView2Profile.IsGeneralAutofillEnabled`, and changing one will change the other. All `CoreWebView2`s with the same `CoreWebView2Profile` will share the same value for this property, so for the `CoreWebView2`s with the same profile, their `CoreWebView2Settings.IsGeneralAutofillEnabled` and `CoreWebView2Profile.IsGeneralAutofillEnabled` will always have the same value.

C++

```
CHECK_FEATURE_RETURN(m_settings4);

BOOL enabled;
CHECK_FAILURE(m_settings4-
>get_IsGeneralAutofillEnabled(&enabled));
if (enabled)
{
    CHECK_FAILURE(m_settings4-
>put_IsGeneralAutofillEnabled(FALSE));
    MessageBox(
        nullptr, L"General autofill will be disabled
immediately.",
        L"Settings change", MB_OK);
}
else
{
    CHECK_FAILURE(m_settings4-
>put_IsGeneralAutofillEnabled(TRUE));
    MessageBox(
        nullptr, L"General autofill will be enabled
immediately.",
        L"Settings change", MB_OK);
}
```

## get\_IsPasswordAutosaveEnabled

`IsPasswordAutosaveEnabled` controls whether autosave for password information is enabled.

```
public HRESULT get_IsPasswordAutosaveEnabled(BOOL * value)
```

The IsPasswordAutosaveEnabled property behaves independently of the IsGeneralAutofillEnabled property. When IsPasswordAutosaveEnabled is false, no new password data is saved and no Save/Update Password prompts are displayed. However, if there was password data already saved before disabling this setting, then that password information is auto-populated, suggestions are shown and clicking on one will populate the fields. When IsPasswordAutosaveEnabled is true, password information is auto-populated, suggestions are shown and clicking on one will populate the fields, new data is saved, and a Save/Update Password prompt is displayed. It will take effect immediately after setting. The default value is FALSE. This property has the same value as CoreWebView2Profile.IsPasswordAutosaveEnabled, and changing one will change the other. All CoreWebView2s with the same CoreWebView2Profile will share the same value for this property, so for the CoreWebView2s with the same profile, their CoreWebView2Settings.IsPasswordAutosaveEnabled and CoreWebView2Profile.IsPasswordAutosaveEnabled will always have the same value.

C++

```
CHECK_FEATURE_RETURN(m_settings4);

BOOL enabled;
CHECK_FAILURE(m_settings4-
>get_IsPasswordAutosaveEnabled(&enabled));
if (enabled)
{
    CHECK_FAILURE(m_settings4-
>put_IsPasswordAutosaveEnabled(FALSE));
    MessageBox(
        nullptr, L>Password autosave will be disabled
immediately.,
        L"Settings change", MB_OK);
}
else
{
    CHECK_FAILURE(m_settings4-
>put_IsPasswordAutosaveEnabled(TRUE));
    MessageBox(
        nullptr, L>Password autosave will be enabled
immediately.,
        L"Settings change", MB_OK);
}
```

## put\_IsGeneralAutofillEnabled

Set the IsGeneralAutofillEnabled property.

```
public HRESULT put_IsGeneralAutofillEnabled(BOOL value)
```

## **put\_IsPasswordAutosaveEnabled**

Set the IsPasswordAutosaveEnabled property.

```
public HRESULT put_IsPasswordAutosaveEnabled(BOOL value)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Settings5

Article • 02/26/2024

```
interface ICoreWebView2Settings5
: public ICoreWebView2Settings4
```

A continuation of the [ICoreWebView2Settings](#) interface to manage pinch zoom.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_IsPinchZoomEnabled</a>	Pinch-zoom, referred to as "Page Scale" zoom, is performed as a post-rendering step, it changes the page scale factor property and scales the surface the web page is rendered onto when user performs a pinch zooming action.
<a href="#">put_IsPinchZoomEnabled</a>	Set the <code>IsPinchZoomEnabled</code> property.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.902.49
WebView2 Win32 Prerelease	1.0.902

## Members

### **get\_IsPinchZoomEnabled**

Pinch-zoom, referred to as "Page Scale" zoom, is performed as a post-rendering step, it changes the page scale factor property and scales the surface the web page is rendered onto when user performs a pinch zooming action.

```
public HRESULT get_IsPinchZoomEnabled(BOOL * enabled)
```

It does not change the layout but rather changes the viewport and clips the web content, the content outside of the viewport isn't visible onscreen and users can't reach this content using mouse.

The `IsPinchZoomEnabled` property enables or disables the ability of the end user to use a pinching motion on touch input enabled devices to scale the web content in the WebView2. It defaults to `TRUE`. When set to `FALSE`, the end user cannot pinch zoom after the next navigation. Disabling/Enabling `IsPinchZoomEnabled` only affects the end user's ability to use pinch motions and does not change the page scale factor. This API only affects the Page Scale zoom and has no effect on the existing browser zoom properties (`IsZoomControlEnabled` and `ZoomFactor`) or other end user mechanisms for zooming.

C++

```
CHECK FEATURE RETURN(m_settings5);

BOOL pinchZoomEnabled;
CHECK FAILURE(m_settings5-
>get_IsPinchZoomEnabled(&pinchZoomEnabled));
if (pinchZoomEnabled)
{
    CHECK FAILURE(m_settings5->put_IsPinchZoomEnabled(FALSE));
    MessageBox(
        nullptr, L"Pinch Zoom is disabled after the next
navigation.",
        L"Settings change", MB_OK);
}
else
{
    CHECK FAILURE(m_settings5->put_IsPinchZoomEnabled(TRUE));
    MessageBox(
        nullptr, L"Pinch Zoom is enabled after the next
navigation.",
        L"Settings change", MB_OK);
}
```

## put\_IsPinchZoomEnabled

Set the `IsPinchZoomEnabled` property.

```
public HRESULT put_IsPinchZoomEnabled(BOOL enabled)
```

# Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Settings6

Article • 02/26/2024

```
interface ICoreWebView2Settings6
: public ICoreWebView2Settings5
```

A continuation of the [ICoreWebView2Settings](#) interface to manage swipe navigation.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_IsSwipeNavigationEnabled</a>	The <code>IsSwipeNavigationEnabled</code> property enables or disables the ability of the end user to use swiping gesture on touch input enabled devices to navigate in WebView2.
<a href="#">put_IsSwipeNavigationEnabled</a>	Set the <code>IsSwipeNavigationEnabled</code> property.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.992.28
WebView2 Win32 Prerelease	1.0.1010

## Members

### [get\\_IsSwipeNavigationEnabled](#)

The `IsSwipeNavigationEnabled` property enables or disables the ability of the end user to use swiping gesture on touch input enabled devices to navigate in WebView2.

```
public HRESULT get_IsSwipeNavigationEnabled(BOOL * enabled)
```

It defaults to `TRUE`.

When this property is `TRUE`, then all configured navigation gestures are enabled:

1. Swiping left and right to navigate forward and backward is always configured.
2. Swiping down to refresh is off by default and not exposed via our API currently, it requires the "--pull-to-refresh" option to be included in the additional browser arguments to be configured. (See `put_AdditionalBrowserArguments`.)

When set to `FALSE`, the end user cannot swipe to navigate or pull to refresh. This API only affects the overscrolling navigation functionality and has no effect on the scrolling interaction used to explore the web content shown in `WebView2`.

Disabling/Enabling `IsSwipeNavigationEnabled` takes effect after the next navigation.

C++

```
    CHECK_FEATURE_RETURN(m_settings6);
    BOOL swipeNavigationEnabled;
    CHECK_FAILURE(m_settings6-
>get_IsSwipeNavigationEnabled(&swipeNavigationEnabled));
    if (swipeNavigationEnabled)
    {
        CHECK_FAILURE(m_settings6-
>put_IsSwipeNavigationEnabled(FALSE));
        MessageBox(
            nullptr, L"Swipe to navigate is disabled after the next
navigation.",
            L"Settings change", MB_OK);
    }
    else
    {
        CHECK_FAILURE(m_settings6-
>put_IsSwipeNavigationEnabled(TRUE));
        MessageBox(
            nullptr, L"Swipe to navigate is enabled after the next
navigation.",
            L"Settings change", MB_OK);
    }
```

## `put_IsSwipeNavigationEnabled`

Set the `IsSwipeNavigationEnabled` property.

```
public HRESULT put_IsSwipeNavigationEnabled(BOOL enabled)
```

# Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Settings7

Article • 02/26/2024

```
interface ICoreWebView2Settings7
: public ICoreWebView2Settings6
```

A continuation of the [ICoreWebView2Settings](#) interface to hide Pdf toolbar items.

## Summary

[+] [Expand table](#)

Members	Descriptions
<a href="#">get_HiddenPdfToolbarItems</a>	<code>HiddenPdfToolbarItems</code> is used to customize the PDF toolbar items.
<a href="#">put_HiddenPdfToolbarItems</a>	Set the <code>HiddenPdfToolbarItems</code> property.

## Applies to

[+] [Expand table](#)

Product	Introduced
WebView2 Win32	1.0.1185.39
WebView2 Win32 Prerelease	1.0.1189

## Members

### get\_HiddenPdfToolbarItems

`HiddenPdfToolbarItems` is used to customize the PDF toolbar items.

```
public HRESULT get_HiddenPdfToolbarItems(COREWEBVIEW2_PDF_TOOLBAR_ITEMS
* hidden_pdf_toolbar_items)
```

By default, it is COREWEBVIEW2\_PDF\_TOOLBAR\_ITEMS\_NONE and so it displays all of the items. Changes to this property apply to all CoreWebView2s in the same environment and using the same profile. Changes to this setting apply only after the next navigation.

C++

```
CHECK_FEATURE_RETURN(m_settings7);

COREWEBVIEW2_PDF_TOOLBAR_ITEMS hiddenPdfToolbarItems;
CHECK_FAILURE(m_settings7-
>get_HiddenPdfToolbarItems(&hiddenPdfToolbarItems));
if (hiddenPdfToolbarItems ==

COREWEBVIEW2_PDF_TOOLBAR_ITEMS::COREWEBVIEW2_PDF_TOOLBAR_ITEMS_NONE)
{
    CHECK_FAILURE(
        m_settings7->put_HiddenPdfToolbarItems(
COREWEBVIEW2_PDF_TOOLBAR_ITEMS::COREWEBVIEW2_PDF_TOOLBAR_ITEMS_PRINT |

COREWEBVIEW2_PDF_TOOLBAR_ITEMS::COREWEBVIEW2_PDF_TOOLBAR_ITEMS_SAVE) |

COREWEBVIEW2_PDF_TOOLBAR_ITEMS::COREWEBVIEW2_PDF_TOOLBAR_ITEMS_BOOKMARKS |

COREWEBVIEW2_PDF_TOOLBAR_ITEMS::COREWEBVIEW2_PDF_TOOLBAR_ITEMS_FIT_PAGE |

COREWEBVIEW2_PDF_TOOLBAR_ITEMS::COREWEBVIEW2_PDF_TOOLBAR_ITEMS_PAGE_LAYOUT |

COREWEBVIEW2_PDF_TOOLBAR_ITEMS::COREWEBVIEW2_PDF_TOOLBAR_ITEMS_ROTATE |

COREWEBVIEW2_PDF_TOOLBAR_ITEMS::COREWEBVIEW2_PDF_TOOLBAR_ITEMS_SEARCH |

COREWEBVIEW2_PDF_TOOLBAR_ITEMS::COREWEBVIEW2_PDF_TOOLBAR_ITEMS_ZOOM_IN |

COREWEBVIEW2_PDF_TOOLBAR_ITEMS::COREWEBVIEW2_PDF_TOOLBAR_ITEMS_ZOOM_OUT |

COREWEBVIEW2_PDF_TOOLBAR_ITEMS::COREWEBVIEW2_PDF_TOOLBAR_ITEMS_SAVE_AS |

COREWEBVIEW2_PDF_TOOLBAR_ITEMS::
    COREWEBVIEW2_PDF_TOOLBAR_ITEMS_PAGE_SELECTOR |

COREWEBVIEW2_PDF_TOOLBAR_ITEMS::COREWEBVIEW2_PDF_TOOLBAR_ITEMS_FULLSCREEN |

COREWEBVIEW2_PDF_TOOLBAR_ITEMS::
    COREWEBVIEW2_PDF_TOOLBAR_ITEMS_MORE_SETTINGS);
MessageBox(
    nullptr,
    L"PDF toolbar print and save buttons are hidden after
the next navigation.",
    L"Settings change", MB_OK);
}
else
{
    CHECK_FAILURE(m_settings7->put_HiddenPdfToolbarItems(
```

```
COREWEBVIEW2_PDF_TOOLBAR_ITEMS::COREWEBVIEW2_PDF_TOOLBAR_ITEMS_NONE));  
    MessageBox(  
        nullptr,  
        L"PDF toolbar print and save buttons are shown after the  
next navigation.",  
        L"Settings change", MB_OK);  
}
```

## put\_HiddenPdfToolbarItems

Set the `HiddenPdfToolbarItems` property.

```
public HRESULT put_HiddenPdfToolbarItems(COREWEBVIEW2_PDF_TOOLBAR_ITEMS  
hidden_pdf_toolbar_items)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2Settings8

Article • 02/26/2024

```
interface ICoreWebView2Settings8
: public ICoreWebView2Settings7
```

A continuation of the [ICoreWebView2Settings](#) interface to manage smartscreen.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_IsReputationCheckingRequired</a>	SmartScreen helps webviews identify reported phishing and malware websites and also helps users make informed decisions about downloads.
<a href="#">put_IsReputationCheckingRequired</a>	Sets whether this webview2 instance needs SmartScreen protection for its content.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1722.45
WebView2 Win32 Prerelease	1.0.1724

## Members

### **get\_IsReputationCheckingRequired**

SmartScreen helps webviews identify reported phishing and malware websites and also helps users make informed decisions about downloads.

```
public HRESULT get_IsReputationCheckingRequired(BOOL * value)
```

`IsReputationCheckingRequired` is used to control whether SmartScreen enabled or not. SmartScreen is enabled or disabled for all CoreWebView2s using the same user data folder. If CoreWebView2Setting.IsReputationCheckingRequired is true for any CoreWebView2 using the same user data folder, then SmartScreen is enabled. If CoreWebView2Setting.IsReputationCheckingRequired is false for all CoreWebView2 using the same user data folder, then SmartScreen is disabled. When it is changed, the change will be applied to all WebViews using the same user data folder on the next navigation or download. The default value for `IsReputationCheckingRequired` is true. If the newly created CoreWebView2 does not set SmartScreen to false, when navigating(Such as Navigate(), LoadDataUrl(), ExecuteScript(), etc.), the default value will be applied to all CoreWebView2 using the same user data folder. SmartScreen of WebView2 apps can be controlled by Windows system setting "SmartScreen for Microsoft Edge", specially, for WebView2 in Windows Store apps, SmartScreen is controlled by another Windows system setting "SmartScreen for Microsoft Store apps". When the Windows setting is enabled, the SmartScreen operates under the control of the `IsReputationCheckingRequired`. When the Windows setting is disabled, the SmartScreen will be disabled regardless of the `IsReputationCheckingRequired` value set in WebView2 apps. In other words, under this circumstance the value of `IsReputationCheckingRequired` will be saved but overridden by system setting. Upon re-enabling the Windows setting, the CoreWebView2 will reference the `IsReputationCheckingRequired` to determine the SmartScreen status.

C++

```
CHECK FEATURE RETURN(m_settings8);

BOOL is_smart_screen_enabled;
CHECK FAILURE(
    m_settings8-
>get_IsReputationCheckingRequired(&is_smart_screen_enabled));
    if (is_smart_screen_enabled)
    {
        CHECK FAILURE(m_settings8-
>put_IsReputationCheckingRequired(false));
        MessageBox(
            nullptr, L"SmartScreen is disabled after the next
navigation.",
            L"Settings change", MB_OK);
    }
    else
    {
        CHECK FAILURE(m_settings8-
>put_IsReputationCheckingRequired(true));
```

```
        MessageBox(
            nullptr, L"SmartScreen is enabled after the next
navigation.",
            L"Settings change", MB_OK);
    }
```

## put\_IsReputationCheckingRequired

Sets whether this webview2 instance needs SmartScreen protection for its content.

```
public HRESULT put_IsReputationCheckingRequired(BOOL value)
```

Set the `IsReputationCheckingRequired` property.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2SharedBuffer

Article • 02/26/2024

```
interface ICoreWebView2SharedBuffer
: public IUnknown
```

The shared buffer object that is created by [CreateSharedBuffer](#).

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Close</a>	Release the backing shared memory.
<a href="#">get_Buffer</a>	The memory address of the shared buffer.
<a href="#">get_FileMappingHandle</a>	Returns a handle to the file mapping object that backs this shared buffer.
<a href="#">get_Size</a>	The size of the shared buffer in bytes.
<a href="#">OpenStream</a>	Get an <a href="#">IStream</a> object that can be used to access the shared buffer.

The object is presented to script as [ArrayBuffer](#) when posted to script with [PostSharedBufferToScript](#).

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1661.34
WebView2 Win32 Prerelease	1.0.1671

## Members

## Close

Release the backing shared memory.

```
public HRESULT Close()
```

The application should call this API when no access to the buffer is needed any more, to ensure that the underlying resources are released timely even if the shared buffer object itself is not released due to some leaked reference. After the shared buffer is closed, the buffer address and file mapping handle previously obtained becomes invalid and cannot be used anymore. Accessing properties of the object will fail with `RO_E_CLOSED`.

Operations like Read or Write on the `IStream` objects returned from `OpenStream` will fail with `RO_E_CLOSED`. `PostSharedBufferToScript` will also fail with `RO_E_CLOSED`.

The script code should call `chrome.webview.releaseBuffer` with the shared buffer as the parameter to release underlying resources as soon as it does not need access the shared buffer any more. When script tries to access the buffer after calling

`chrome.webview.releaseBuffer`, JavaScript `TypeError` exception will be raised complaining about accessing a detached `ArrayBuffer`, the same exception when trying to access a transferred `ArrayBuffer`.

Closing the buffer object on native side doesn't impact access from Script and releasing the buffer from script doesn't impact access to the buffer from native side. The underlying shared memory will be released by the OS when both native and script side release the buffer.

## get\_Buffer

The memory address of the shared buffer.

```
public HRESULT get_Buffer(BYTE ** value)
```

## get\_FileMappingHandle

Returns a handle to the file mapping object that backs this shared buffer.

```
public HRESULT get_FileMappingHandle(HANDLE * value)
```

The returned handle is owned by the shared buffer object. You should not call `CloseHandle` on it. Normal app should use `Buffer` or `OpenStream` to get memory address or `IStream` object to access the buffer. For advanced scenarios, you could use

file mapping APIs to obtain other views or duplicate this handle to another application process and create a view from the duplicated handle in that process to access the buffer from that separate process.

## get\_Size

The size of the shared buffer in bytes.

```
public HRESULT get_Size(UINT64 * value)
```

## OpenStream

Get an IStream object that can be used to access the shared buffer.

```
public HRESULT OpenStream(IStream ** value)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2SourceChangedEventArgs

Article • 02/26/2024

```
interface ICoreWebView2SourceChangedEventArgs
: public IUnknown
```

Event args for the `SourceChanged` event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_IsNewDocument</a>	<code>TRUE</code> if the page being navigated to is a new document.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### `get_IsNewDocument`

`TRUE` if the page being navigated to is a new document.

```
public HRESULT get_IsNewDocument(BOOL * isNewDocument)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2StringCollection

Article • 02/26/2024

```
interface ICoreWebView2StringCollection
: public IUnknown
```

A collection of strings.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_Count</a>	The number of strings contained in ICoreWebView2StringCollection.
<a href="#">GetValueAtIndex</a>	Gets the value at a given index.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.961.33
WebView2 Win32 Prerelease	1.0.955

## Members

### [get\\_Count](#)

The number of strings contained in ICoreWebView2StringCollection.

```
public HRESULT get_Count(UINT * value)
```

## GetValueAtIndex

Gets the value at a given index.

```
public HRESULT GetValueAtIndex(UINT index, LPWSTR * value)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2WebMessageReceivedEventArgs

Article • 02/26/2024

```
interface ICoreWebView2WebMessageReceivedEventArgs
: public IUnknown
```

Event args for the `WebMessageReceived` event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_Source</a>	The URI of the document that sent this web message.
<a href="#">get_WebMessageAsJson</a>	The message posted from the WebView content to the host converted to a JSON string.
<a href="#">TryGetWebMessageAsString</a>	If the message posted from the WebView content to the host is a string type, this method returns the value of that string.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### `get_Source`

The URI of the document that sent this web message.

```
public HRESULT get_Source(LPWSTR * source)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_WebMessageAsJson

The message posted from the WebView content to the host converted to a JSON string.

```
public HRESULT get_WebMessageAsJson(LPWSTR * webMessageAsJson)
```

Run this operation to communicate using JavaScript objects.

For example, the following `postMessage` runs result in the following `WebMessageAsJson` values.

JSON

```
postMessage({ 'a': 'b'})      L"\"a\": \"b\""  
postMessage(1.2)            L"1.2"  
postMessage('example')     L"\"example\""
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## TryGetWebMessageAsString

If the message posted from the WebView content to the host is a string type, this method returns the value of that string.

```
public HRESULT TryGetWebMessageAsString(LPWSTR * webMessageAsString)
```

If the message posted is some other kind of JavaScript type this method fails with the following error.

text

```
E_INVALIDARG
```

Run this operation to communicate using simple strings.

For example, the following `postMessage` runs result in the following `WebMessageAsString` values.

JSON

<code>postMessage({ 'a': 'b'})</code>	<code>E_INVALIDARG</code>
<code>postMessage(1.2)</code>	<code>E_INVALIDARG</code>
<code>postMessage('example')</code>	<code>L"example"</code>

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2WebMessageReceivedEventArgs2

Article • 02/26/2024

```
interface ICoreWebView2WebMessageReceivedEventArgs2
    : public ICoreWebView2WebMessageReceivedEventArgs
```

Extension of WebMessageReceivedEventArgs to provide access to additional WebMessage objects.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_AdditionalObjects</a>	Additional received WebMessage objects.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1774.30
WebView2 Win32 Prerelease	1.0.1777

## Members

### get\_AdditionalObjects

Additional received WebMessage objects.

```
public HRESULT get_AdditionalObjects(ICoreWebView2ObjectCollectionView **  
value)
```

To pass `additionalObjects` via WebMessage to the app, use the

`chrome.webview.postMessageWithAdditionalObjects` content API. Any DOM object type that can be natively representable that has been passed in to `additionalObjects` parameter will be accessible here. Currently a WebMessage object can be the `ICoreWebView2File` type. Entries in the collection can be `nullptr` if `null` or `undefined` was passed.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2WebResourceRequest

Article • 02/26/2024

```
interface ICoreWebView2WebResourceRequest
: public IUnknown
```

An HTTP request used with the `WebResourceRequested` event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_Content</a>	The HTTP request message body as stream.
<a href="#">get_Headers</a>	The mutable HTTP request headers.
<a href="#">get_Method</a>	The HTTP request method.
<a href="#">get_Uri</a>	The request URI.
<a href="#">put_Content</a>	Sets the <code>Content</code> property.
<a href="#">put_Method</a>	Sets the <code>Method</code> property.
<a href="#">put_Uri</a>	Sets the <code>Uri</code> property.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

# Members

## get\_Content

The HTTP request message body as stream.

```
| public HRESULT get_Content(IStream ** content)
```

POST data should be here. If a stream is set, which overrides the message body, the stream must have all the content data available by the time the `WebResourceRequested` event deferral of this response is completed. Stream should be agile or be created from a background STA to prevent performance impact to the UI thread. `Null` means no content data. `IStream` semantics apply (return `S_OK` to `Read` runs until all data is exhausted).

## get\_Headers

The mutable HTTP request headers.

```
| public HRESULT get_Headers(ICoreWebView2HttpRequestHeaders ** headers)
```

## get\_Method

The HTTP request method.

```
| public HRESULT get_Method(LPWSTR * method)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_Uri

The request URI.

```
| public HRESULT get_Uri(LPWSTR * uri)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## put\_Content

Sets the `Content` property.

```
public HRESULT put_Content(IStream * content)
```

## put\_Method

Sets the `Method` property.

```
public HRESULT put_Method(LPCWSTR method)
```

## put\_Uri

Sets the `Uri` property.

```
public HRESULT put_Uri(LPCWSTR uri)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2WebResourceRequestedEventArgs

Article • 02/26/2024

```
interface ICoreWebView2WebResourceRequestedEventArgs
: public IUnknown
```

Event args for the `WebResourceRequested` event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_Request</a>	The Web resource request.
<a href="#">get_ResourceManager</a>	The web resource request context.
<a href="#">get_Response</a>	A placeholder for the web resource response object.
<a href="#">GetDeferral</a>	Obtain an ICoreWebView2Deferral object and put the event into a deferred state.
<a href="#">put_Response</a>	Sets the <code>Response</code> property.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

## get\_Request

The Web resource request.

```
public HRESULT get_Request(ICoreWebView2WebResourceRequest ** request)
```

The request object may be missing some headers that are added by network stack at a later time.

## get\_ResourceContext

The web resource request context.

```
public HRESULT get_ResourceContext(COREWEBVIEW2_WEB_RESOURCE_CONTEXT * context)
```

## get\_Response

A placeholder for the web resource response object.

```
public HRESULT get_Response(ICoreWebView2WebResourceResponse ** response)
```

If this object is set, the web resource request is completed with the specified response.

## GetDeferral

Obtain an ICoreWebView2Deferral object and put the event into a deferred state.

```
public HRESULT GetDeferral(ICoreWebView2Deferral ** deferral)
```

Use the ICoreWebView2Deferral object to complete the request at a later time.

## put\_Response

Sets the `Response` property.

```
public HRESULT put_Response(ICoreWebView2WebResourceResponse * response)
```

Create an empty web resource response object with `CreateWebResourceResponse` and then modify it to construct the response.

---

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2WebResourceRequestedEventArgs2

Article • 02/26/2024

```
interface ICoreWebView2WebResourceRequestedEventArgs2
: public ICoreWebView2WebResourceRequestedEventArgs
```

Event args for the `WebResourceRequested` event.

## Summary

Expand table

Members	Descriptions
<a href="#">get_RequestedSourceKind</a>	The web resource requested source.

## Applies to

Expand table

Product	Introduced
WebView2 Win32	N/A
WebView2 Win32 Prerelease	1.0.2357

## Members

### [get\\_RequestedSourceKind](#)

The web resource requested source.

```
public HRESULT
```

```
get_RequestedSourceKind(COREWEBVIEW2_WEB_RESOURCE_REQUEST_SOURCE_KI
```

NDS \* requestedSourceKind)

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2WebResourceResponse

Article • 02/26/2024

```
interface ICoreWebView2WebResourceResponse
: public IUnknown
```

An HTTP response used with the `WebResourceRequested` event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_Content</a>	HTTP response content as stream.
<a href="#">get_Headers</a>	Overridden HTTP response headers.
<a href="#">get_ReasonPhrase</a>	The HTTP response reason phrase.
<a href="#">get_StatusCode</a>	The HTTP response status code.
<a href="#">put_Content</a>	Sets the <code>Content</code> property.
<a href="#">put_ReasonPhrase</a>	Sets the <code>ReasonPhrase</code> property.
<a href="#">put_StatusCode</a>	Sets the <code>StatusCode</code> property.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

# Members

## get\_Content

HTTP response content as stream.

```
public HRESULT get_Content(IStream ** content)
```

Stream must have all the content data available by the time the `WebResourceRequested` event deferral of this response is completed. Stream should be agile or be created from a background thread to prevent performance impact to the UI thread. `Null` means no content data. `IStream` semantics apply (return `S_OK` to `Read` runs until all data is exhausted). When providing the response data, you should consider relevant HTTP request headers just like an HTTP server would do. For example, if the request was for a video resource in a HTML video element, the request may contain the [Range](#) header to request only a part of the video that is streaming. In this case, your response stream should be only the portion of the video specified by the range HTTP request headers and you should set the appropriate [Content-Range](#) header in the response.

## get\_Headers

Overridden HTTP response headers.

```
public HRESULT get_Headers(ICoreWebView2HttpResponseHeaders ** headers)
```

## get\_ReasonPhrase

The HTTP response reason phrase.

```
public HRESULT get_ReasonPhrase(LPWSTR * reasonPhrase)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_StatusCode

The HTTP response status code.

```
public HRESULT get_StatusCode(int * statusCode)
```

## put\_Content

Sets the `Content` property.

```
public HRESULT put_Content(IStream * content)
```

## **put\_RaisonPhrase**

Sets the `RaisonPhrase` property.

```
public HRESULT put_RaisonPhrase(LPCWSTR raisonPhrase)
```

## **put\_StatusCode**

Sets the `StatusCode` property.

```
public HRESULT put_StatusCode(int statusCode)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2WebResourceResponseReceivedEventArgs

Article • 02/26/2024

```
interface ICoreWebView2WebResourceResponseReceivedEventArgs
: public IUnknown
```

Event args for the WebResourceResponseReceived event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_Request</a>	The request object for the web resource, as committed.
<a href="#">get_Response</a>	View of the response object received for the web resource.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.705.50
WebView2 Win32 Prerelease	1.0.721

## Members

### get\_Request

The request object for the web resource, as committed.

```
public HRESULT get_Request(ICoreWebView2WebResourceRequest ** request)
```

This includes headers added by the network stack that were not included during the associated WebResourceRequested event, such as Authentication headers. Modifications to this object have no effect on how the request is processed as it has already been sent.

## get\_Response

View of the response object received for the web resource.

```
public HRESULT get_Response(ICoreWebView2WebResourceResponseView **  
response)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2WebResourceResponseView

Article • 02/26/2024

```
interface ICoreWebView2WebResourceResponseView
: public IUnknown
```

View of the HTTP representation for a web resource response.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">get_Headers</a>	The HTTP response headers as received.
<a href="#">get_ReasonPhrase</a>	The HTTP response reason phrase.
<a href="#">get_StatusCode</a>	The HTTP response status code.
<a href="#">GetContent</a>	Get the response content asynchronously.

The properties of this object are not mutable. This response view is used with the `WebResourceResponseReceived` event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.705.50
WebView2 Win32 Prerelease	1.0.721

## Members

## get\_Headers

The HTTP response headers as received.

```
public HRESULT get_Headers(ICoreWebView2HttpResponseHeaders ** headers)
```

## get\_ReasonPhrase

The HTTP response reason phrase.

```
public HRESULT get_ReasonPhrase(LPWSTR * reasonPhrase)
```

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

## get\_StatusCode

The HTTP response status code.

```
public HRESULT get_StatusCode(int * statusCode)
```

## GetContent

Get the response content asynchronously.

```
public HRESULT  
GetContent(ICoreWebView2WebResourceResponseViewGetContentCompletedHandler * handler)
```

The handler will receive the response content stream.

This method returns null if content size is more than 123MB or for navigations that become downloads or if response is downloadable content type (e.g., application/octet-stream). See `add_DownloadStarting` event to handle the response.

If this method is being called again before a first call has completed, the handler will be invoked at the same time the handlers from prior calls are invoked. If this method is being called after a first call has completed, the handler will be invoked immediately.

C++

```
webResourceResponse->GetContent(  
    Callback<
```

```
ICoreWebView2WebResourceResponseViewGetContentCompletedHandler>
    [this, webResourceRequest,
     webResourceResponse](HRESULT result, IStream*
content) {
    std::wstring message =
        L"{ \"kind\": \"event\", \"name\": "
        L"\\"WebResourceResponseReceived\\",
        L"\"args\": \""
        L"\"request\": " +
    RequestToJsonString(webResourceRequest.get()) +
        L", "
        L"\"response\": " +
    ResponseToJsonString(webResourceResponse.get(), content) +
        L"}";
    message +=

    WebViewPropertiesToJsonString(m_webviewEventSource.get());
    message += L"}";
    PostEventMessage(message);
    return S_OK;
})
.Get());
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2WindowFeatures

Article • 02/26/2024

```
interface ICoreWebView2WindowFeatures
: public IUnknown
```

The window features for a WebView popup window.

## Summary

[ ] Expand table

Members	Descriptions
<a href="#">get_HasPosition</a>	Specifies left and top values.
<a href="#">get_HasSize</a>	Specifies height and width values.
<a href="#">get_Height</a>	Specifies the height of the window.
<a href="#">get_Left</a>	Specifies the left position of the window.
<a href="#">get_ShouldDisplayMenuBar</a>	Indicates that the menu bar is displayed.
<a href="#">get_ShouldDisplayScrollBars</a>	Indicates that the scroll bars are displayed.
<a href="#">get_ShouldDisplayStatus</a>	Indicates that the status bar is displayed.
<a href="#">get_ShouldDisplayToolbar</a>	Indicates that the browser toolbar is displayed.
<a href="#">get_Top</a>	Specifies the top position of the window.
<a href="#">get_Width</a>	Specifies the width of the window.

The fields match the `windowFeatures` passed to `window.open` as specified in [Window features](#) on MDN.

There is no requirement for you to respect the values. If your app does not have corresponding UI features (for example, no toolbar) or if all instances of WebView are opened in tabs and do not have distinct size or positions, then your app does not respect the values. You may want to respect values, but perhaps only some apply to the

UI of your app. Accordingly, you may respect all, some, or none of the properties as appropriate for your app. For all numeric properties, if the value that is passed to `window.open` is outside the range of an unsigned 32bit int, the resulting value is the absolute value of the maximum for unsigned 32bit integer. If you are not able to parse the value as an integer, it is considered `0`. If the value is a floating point value, it is rounded down to an integer.

In runtime versions 98 or later, the values of `ShouldDisplayMenuBar`, `ShouldDisplayStatus`, `ShouldDisplayToolbar`, and `ShouldDisplayScrollBars` will not directly depend on the equivalent fields in the `windowFeatures` string. Instead, they will all be false if the window is expected to be a popup, and true if it is not.

## Applies to

[Expand table](#)

Product	Introduced
WebView2 Win32	0.9.622.11
WebView2 Win32 Prerelease	0.9.628

## Members

### get\_HasPosition

Specifies left and top values.

```
public HRESULT get_HasPosition(BOOL * value)
```

### get\_HasSize

Specifies height and width values.

```
public HRESULT get_HasSize(BOOL * value)
```

### get\_Height

Specifies the height of the window.

```
public HRESULT get_Height(UINT32 * value)
```

Minimum value is 100. If `HasSize` is set to FALSE, this field is ignored.

## get\_Left

Specifies the left position of the window.

```
public HRESULT get_Left(UINT32 * value)
```

If `HasPosition` is set to FALSE, this field is ignored.

## get\_ShouldDisplayMenuBar

Indicates that the menu bar is displayed.

```
public HRESULT get_ShouldDisplayMenuBar(BOOL * value)
```

## get\_ShouldDisplayScrollBars

Indicates that the scroll bars are displayed.

```
public HRESULT get_ShouldDisplayScrollBars(BOOL * value)
```

## get\_ShouldDisplayStatus

Indicates that the status bar is displayed.

```
public HRESULT get_ShouldDisplayStatus(BOOL * value)
```

## get\_ShouldDisplayToolbar

Indicates that the browser toolbar is displayed.

```
public HRESULT get_ShouldDisplayToolbar(BOOL * value)
```

## get\_Top

Specifies the top position of the window.

```
public HRESULT get_Top(UINT32 * value)
```

If `HasPosition` is set to FALSE, this field is ignored.

## get\_Width

Specifies the width of the window.

```
public HRESULT get_Width(UINT32 * value)
```

Minimum value is 100. If HasSize is set to FALSE, this field is ignored.

---

## Feedback

Was this page helpful?

 Yes

 No

# Globals

Article • 02/26/2024

## Summary

[Expand table](#)

Members	Descriptions
<a href="#">COREWEBVIEW2_BOUNDS_MODE</a>	Mode for how the Bounds property is interpreted in relation to the RasterizationScale property.
<a href="#">COREWEBVIEW2_BROWSER_PROCESS_EXIT_KIND</a>	Specifies the browser process exit type used in the ICoreWebView2BrowserProcessExitedEventArgs interface.
<a href="#">COREWEBVIEW2_BROWSING_DATA_KINDS</a>	Specifies the datatype for the ICoreWebView2Profile2::ClearBrowsingData method.
<a href="#">COREWEBVIEW2_CAPTURE_PREVIEW_IMAGE_FORMAT</a>	Specifies the image format for the ICoreWebView2::CapturePreview method.
<a href="#">COREWEBVIEW2_CLIENT_CERTIFICATE_KIND</a>	Specifies the client certificate kind.
<a href="#">COREWEBVIEW2_CONTEXT_MENU_ITEM_KIND</a>	Specifies the menu item kind for the ICoreWebView2ContextMenuItem::get_Kind method.
<a href="#">COREWEBVIEW2_CONTEXT_MENU_TARGET_KIND</a>	Indicates the kind of context for which the context menu was created for the ICoreWebView2ContextMenuTarget::get_Kind method.
<a href="#">COREWEBVIEW2_COOKIE_SAME_SITE_KIND</a>	Kind of cookie SameSite status used in the ICoreWebView2Cookie interface.
<a href="#">COREWEBVIEW2_DEFAULT_DOWNLOAD_DIALOG_CORNER_ALIGNMENT</a>	The default download dialog can be aligned to any of the WebView corners by setting the <code>DefaultDownloadDialogCornerAlignment</code> property.
<a href="#">COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON</a>	Reason why a download was interrupted.
<a href="#">COREWEBVIEW2_DOWNLOAD_STATE</a>	State of the download operation.
<a href="#">COREWEBVIEW2_FAVICON_IMAGE_FORMAT</a>	Specifies the image format to use for favicon.
<a href="#">COREWEBVIEW2_FRAME_KIND</a>	Indicates the frame type used in the ICoreWebView2FrameInfo interface.
<a href="#">COREWEBVIEW2_HOST_RESOURCE_ACCESS_KIND</a>	Kind of cross origin resource access allowed for host resources during download.
<a href="#">COREWEBVIEW2_KEY_EVENT_KIND</a>	Specifies the key event type that triggered an <code>AcceleratorKeyPressed</code> event.
<a href="#">COREWEBVIEW2_MEMORY_USAGE_TARGET_LEVEL</a>	Specifies memory usage target level of WebView.
<a href="#">COREWEBVIEW2_MOUSE_EVENT_KIND</a>	Mouse event type used by <code>SendMouseInput</code> to convey the type of mouse event being sent to WebView.

Members	Descriptions
<a href="#">COREWEBVIEW2_MOUSE_EVENT_VIRTUAL_KEYS</a>	Mouse event virtual keys associated with a COREWEBVIEW2_MOUSE_EVENT_KIND for SendMouseInput.
<a href="#">COREWEBVIEW2_MOVE_FOCUS_REASON</a>	Specifies the reason for moving focus.
<a href="#">COREWEBVIEW2_NAVIGATION_KIND</a>	Specifies the navigation kind of each navigation.
<a href="#">COREWEBVIEW2_PDF_TOOLBAR_ITEMS</a>	Specifies the PDF toolbar item types used for the <code>ICoreWebView2Settings::put_HiddenPdfToolbarItems</code> method.
<a href="#">COREWEBVIEW2_PERMISSION_KIND</a>	Indicates the type of a permission request.
<a href="#">COREWEBVIEW2_PERMISSION_STATE</a>	Specifies the response to a permission request.
<a href="#">COREWEBVIEW2_POINTER_EVENT_KIND</a>	Pointer event type used by SendPointerInput to convey the type of pointer event being sent to WebView.
<a href="#">COREWEBVIEW2_PREFERRED_COLOR_SCHEME</a>	An enum to represent the options for WebView2 color scheme: auto, light, or dark.
<a href="#">COREWEBVIEW2_PRINT_COLLATION</a>	Specifies the collation for a print.
<a href="#">COREWEBVIEW2_PRINT_COLOR_MODE</a>	Specifies the color mode for a print.
<a href="#">COREWEBVIEW2_PRINT_DIALOG_KIND</a>	Specifies the print dialog kind.
<a href="#">COREWEBVIEW2_PRINT_DUPLEX</a>	Specifies the duplex option for a print.
<a href="#">COREWEBVIEW2_PRINT_MEDIA_SIZE</a>	Specifies the media size for a print.
<a href="#">COREWEBVIEW2_PRINT_ORIENTATION</a>	The orientation for printing, used by the <code>Orientation</code> property on <code>ICoreWebView2PrintSettings</code> .
<a href="#">COREWEBVIEW2_PRINT_STATUS</a>	Indicates the status for printing.
<a href="#">COREWEBVIEW2_PROCESS_FAILED_KIND</a>	Specifies the process failure type used in the <code>ICoreWebView2ProcessFailedEventArgs</code> interface.
<a href="#">COREWEBVIEW2_PROCESS_FAILED_REASON</a>	Specifies the process failure reason used in the <code>ICoreWebView2ProcessFailedEventArgs</code> interface.
<a href="#">COREWEBVIEW2_PROCESS_KIND</a>	Indicates the process type used in the <code>ICoreWebView2ProcessInfo</code> interface.
<a href="#">COREWEBVIEW2_SCRIPT_DIALOG_KIND</a>	Specifies the JavaScript dialog type used in the <code>ICoreWebView2ScriptDialogOpeningEventHandler</code> interface.
<a href="#">COREWEBVIEW2_SERVER_CERTIFICATE_ERROR_ACTION</a>	Specifies the action type when server certificate error is detected to be used in the <code>ICoreWebView2ServerCertificateErrorDetectedEventArgs</code> interface.
<a href="#">COREWEBVIEW2_SHARED_BUFFER_ACCESS</a>	Specifies the desired access from script to <code>CoreWebView2SharedBuffer</code> .
<a href="#">COREWEBVIEW2_TRACKING_PREVENTION_LEVEL</a>	Tracking prevention levels.
<a href="#">COREWEBVIEW2_WEB_ERROR_STATUS</a>	Indicates the error status values for web navigations.

Members	Descriptions
<a href="#">COREWEBVIEW2_WEB_RESOURCE_CONTEXT</a>	Specifies the web resource request contexts.
<a href="#">COREWEBVIEW2_WEB_RESOURCE_REQUEST_SOURCE_KINDS</a>	Specifies the source of <code>WebResourceRequested</code> event.
<a href="#">COREWEBVIEW2_COLOR</a>	A value representing RGBA color (Red, Green, Blue, Alpha) for WebView2.
<a href="#">CompareBrowserVersions</a>	This method is for anyone want to compare version correctly to determine which version is newer, older or same.
<a href="#">CreateCoreWebView2Environment</a>	Creates an evergreen WebView2 Environment using the installed WebView2 Runtime version.
<a href="#">CreateCoreWebView2EnvironmentWithOptions</a>	DLL export to create a WebView2 environment with a custom version of WebView2 Runtime, user data folder, and with or without additional options.
<a href="#">CreateWebViewEnvironmentWithOptionsInternal</a>	This is a DLL export out of <code>EmbeddedBrowserWebView.dll</code> which can be found in the installation folder of the WebView2 Runtime you wish to use.
<a href="#">GetAvailableCoreWebView2BrowserVersionString</a>	Get the browser version info including channel name if it is not the WebView2 Runtime.

## Members

### COREWEBVIEW2\_BOUNDS\_MODE

enum [COREWEBVIEW2\\_BOUNDS\\_MODE](#)

[Expand table](#)

Values	Descriptions
<code>COREWEBVIEW2_BOUNDS_MODE_USE_RAW_PIXELS</code>	Bounds property represents raw pixels. Physical size of Webview is not impacted by RasterizationScale.
<code>COREWEBVIEW2_BOUNDS_MODE_USE_RASTERIZATION_SCALE</code>	Bounds property represents logical pixels and the RasterizationScale property is used to get the physical size of the WebView.

Mode for how the Bounds property is interpreted in relation to the RasterizationScale property.

### COREWEBVIEW2\_BROWSER\_PROCESS\_EXIT\_KIND

enum [COREWEBVIEW2\\_BROWSER\\_PROCESS\\_EXIT\\_KIND](#)

[Expand table](#)

Values	Descriptions
<code>COREWEBVIEW2_BROWSER_PROCESS_EXIT_KIND_NORMAL</code>	Indicates that the browser process ended normally.

Values	Descriptions
COREWEBVIEW2_BROWSER_PROCESS_EXIT_KIND_FAILED	Indicates that the browser process ended unexpectedly.

Specifies the browser process exit type used in the ICoreWebView2BrowserProcessExitedEventArgs interface.

## COREWEBVIEW2\_BROWSING\_DATA\_KINDS

enum [COREWEBVIEW2\\_BROWSING\\_DATA\\_KINDS](#)

[Expand table](#)

Values	Descriptions
COREWEBVIEW2_BROWSING_DATA_KINDS_FILE_SYSTEMS	Specifies file systems data.
COREWEBVIEW2_BROWSING_DATA_KINDS_INDEXED_DB	Specifies data stored by the IndexedDB DOM feature.
COREWEBVIEW2_BROWSING_DATA_KINDS_LOCAL_STORAGE	Specifies data stored by the localStorage DOM API.
COREWEBVIEW2_BROWSING_DATA_KINDS_WEB_SQL	Specifies data stored by the Web SQL database DOM API.
COREWEBVIEW2_BROWSING_DATA_KINDS_CACHE_STORAGE	Specifies data stored by the CacheStorage DOM API.
COREWEBVIEW2_BROWSING_DATA_KINDS_ALL_DOM_STORAGE	Specifies DOM storage data, now and future.
COREWEBVIEW2_BROWSING_DATA_KINDS_COOKIES	Specifies HTTP cookies data.
COREWEBVIEW2_BROWSING_DATA_KINDS_ALL_SITE	Specifies all site data, now and future.
COREWEBVIEW2_BROWSING_DATA_KINDS_DISK_CACHE	Specifies disk cache.
COREWEBVIEW2_BROWSING_DATA_KINDS_DOWNLOAD_HISTORY	Specifies download history data.
COREWEBVIEW2_BROWSING_DATA_KINDS_GENERAL_AUTOFILL	Specifies general autofill form data.
COREWEBVIEW2_BROWSING_DATA_KINDS_PASSWORD_AUTOSAVE	Specifies password autosave data.
COREWEBVIEW2_BROWSING_DATA_KINDS_BROWSING_HISTORY	Specifies browsing history data.
COREWEBVIEW2_BROWSING_DATA_KINDS_SETTINGS	Specifies settings data.
COREWEBVIEW2_BROWSING_DATA_KINDS_ALL_PROFILE	Specifies profile data that should be wiped to make it look like a new profile.
COREWEBVIEW2_BROWSING_DATA_KINDS_SERVICE_WORKERS	Specifies service workers registered for an origin, and clear will result in termination and deregistration of them.

Specifies the datatype for the ICoreWebView2Profile2::ClearBrowsingData method.

## COREWEBVIEW2\_CAPTURE\_PREVIEW\_IMAGE\_FORMAT

enum [COREWEBVIEW2\\_CAPTURE\\_PREVIEW\\_IMAGE\\_FORMAT](#)

[Expand table](#)

Values	Descriptions
COREWEBVIEW2_CAPTURE_PREVIEW_IMAGE_FORMAT_PNG	Indicates that the PNG image format is used.
COREWEBVIEW2_CAPTURE_PREVIEW_IMAGE_FORMAT_JPEG	Indicates the JPEG image format is used.

Specifies the image format for the `ICoreWebView2::CapturePreview` method.

## COREWEBVIEW2\_CLIENT\_CERTIFICATE\_KIND

enum [COREWEBVIEW2\\_CLIENT\\_CERTIFICATE\\_KIND](#)

[Expand table](#)

Values	Descriptions
COREWEBVIEW2_CLIENT_CERTIFICATE_KIND_SMART_CARD	Specifies smart card certificate.
COREWEBVIEW2_CLIENT_CERTIFICATE_KIND_PIN	Specifies PIN certificate.
COREWEBVIEW2_CLIENT_CERTIFICATE_KIND_OTHER	Specifies other certificate.

Specifies the client certificate kind.

## COREWEBVIEW2\_CONTEXT\_MENU\_ITEM\_KIND

enum [COREWEBVIEW2\\_CONTEXT\\_MENU\\_ITEM\\_KIND](#)

[Expand table](#)

Values	Descriptions
COREWEBVIEW2_CONTEXT_MENU_ITEM_KIND_COMMAND	Specifies a command menu item kind.
COREWEBVIEW2_CONTEXT_MENU_ITEM_KIND_CHECK_BOX	Specifies a check box menu item kind.
COREWEBVIEW2_CONTEXT_MENU_ITEM_KIND_RADIO	Specifies a radio button menu item kind.
COREWEBVIEW2_CONTEXT_MENU_ITEM_KIND_SEPARATOR	Specifies a separator menu item kind.
COREWEBVIEW2_CONTEXT_MENU_ITEM_KIND_SUBMENU	Specifies a submenu menu item kind.

Specifies the menu item kind for the `ICoreWebView2ContextMenuItem::get_Kind` method.

## COREWEBVIEW2\_CONTEXT\_MENU\_TARGET\_KIND

enum [COREWEBVIEW2\\_CONTEXT\\_MENU\\_TARGET\\_KIND](#)

[Expand table](#)

Values	Descriptions
COREWEBVIEW2_CONTEXT_MENU_TARGET_KIND_PAGE	Indicates that the context menu was created for the page without any additional content.

Values	Descriptions
COREWEBVIEW2_CONTEXT_MENU_TARGET_KIND_IMAGE	Indicates that the context menu was created for an image element.
COREWEBVIEW2_CONTEXT_MENU_TARGET_KIND_SELECTED_TEXT	Indicates that the context menu was created for selected text.
COREWEBVIEW2_CONTEXT_MENU_TARGET_KIND_AUDIO	Indicates that the context menu was created for an audio element.
COREWEBVIEW2_CONTEXT_MENU_TARGET_KIND_VIDEO	Indicates that the context menu was created for a video element.

Indicates the kind of context for which the context menu was created for the `ICoreWebView2ContextMenuTarget::get_Kind` method.

This enum will always represent the active element that caused the context menu request. If there is a selection with multiple images, audio and text, for example, the element that the end user right clicks on within this selection will be the option represented by this enum.

## COREWEBVIEW2\_COOKIE\_SAME\_SITE\_KIND

enum [COREWEBVIEW2\\_COOKIE\\_SAME\\_SITE\\_KIND](#)

[Expand table](#)

Values	Descriptions
COREWEBVIEW2_COOKIE_SAME_SITE_KIND_NONE	None SameSite type. No restrictions on cross-site requests.
COREWEBVIEW2_COOKIE_SAME_SITE_KIND_LAX	Lax SameSite type. The cookie will be sent with "same-site" requests, and with "cross-site" top level navigation.
COREWEBVIEW2_COOKIE_SAME_SITE_KIND_STRICT	Strict SameSite type. The cookie will only be sent along with "same-site" requests.

Kind of cookie SameSite status used in the `ICoreWebView2Cookie` interface.

These fields match those as specified in <https://developer.mozilla.org/docs/Web/HTTP/Cookies#>. Learn more about SameSite cookies here: <https://tools.ietf.org/html/draft-west-first-party-cookies-07>

## COREWEBVIEW2\_DEFAULT\_DOWNLOAD\_DIALOG\_CORNER\_ALIGNMENT

enum [COREWEBVIEW2\\_DEFAULT\\_DOWNLOAD\\_DIALOG\\_CORNER\\_ALIGNMENT](#)

[Expand table](#)

Values	Descriptions
COREWEBVIEW2_DEFAULT_DOWNLOAD_DIALOG_CORNER_ALIGNMENT_TOP_LEFT	Top-left corner of the WebView.
COREWEBVIEW2_DEFAULT_DOWNLOAD_DIALOG_CORNER_ALIGNMENT_TOP_RIGHT	Top-right corner of the WebView.

Values	Descriptions
COREWEBVIEW2_DEFAULT_DOWNLOAD_DIALOG_CORNER_ALIGNMENT_BOTTOM_LEFT	Bottom-left corner of the WebView.
COREWEBVIEW2_DEFAULT_DOWNLOAD_DIALOG_CORNER_ALIGNMENT_BOTTOM_RIGHT	Bottom-right corner of the WebView.

The default download dialog can be aligned to any of the WebView corners by setting the `DefaultDownloadDialogCornerAlignment` property.

The default position is top-right corner.

## COREWEBVIEW2\_DOWNLOAD\_INTERRUPT\_REASON

enum `COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON`

 Expand table

Values	Descriptions
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_NONE	
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_FILE_FAILED	Generic file error.
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_FILE_ACCESS_DENIED	Access denied due to security restrictions.
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_FILE_NO_SPACE	Disk full.
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_FILE_NAME_TOO_LONG	Result file path with file name is too long.
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_FILE_TOO_LARGE	File is too large for file system.
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_FILE_MALICIOUS	Microsoft Defender Smartscreen detected a virus in the file.
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_FILE_TRANSIENT_ERROR	File was in use, too many files opened, or out of memory.
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_FILE_BLOCKED_BY_POLICY	File blocked by local policy.
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_FILE_SECURITY_CHECK_FAILED	Security check failed unexpectedly.
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_FILE_TOO_SHORT	Seeking past the end of a file in opening a file, as part of resuming an interrupted download.
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_FILE_HASH_MISMATCH	Partial file did not match the expected hash and was deleted.

Values	Descriptions
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_NETWORK_FAILED	Generic network error. User can retry the download manually.
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_NETWORK_TIMEOUT	Network operation timed out.
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_NETWORK_DISCONNECTED	Network connection lost. User can retry the download manually.
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_NETWORK_SERVER_DOWN	Server has gone down. User can retry the download manually.
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_NETWORK_INVALID_REQUEST	Network request invalid because original or redirected URI is invalid, has an unsupported scheme, or is disallowed by network policy.
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_SERVER_FAILED	Generic server error. User can retry the download manually.
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_SERVER_NO_RANGE	Server does not support range requests.
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_SERVER_BAD_CONTENT	Server does not have the requested data.
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_SERVER_UNAUTHORIZED	Server did not authorize access to resource.
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_SERVER_CERTIFICATE_PROBLEM	Server certificate problem.
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_SERVER_FORBIDDEN	Server access forbidden.
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_SERVER_UNEXPECTED_RESPONSE	Unexpected server response.
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_SERVER_CONTENT_LENGTH_MISMATCH	Server sent fewer bytes than the Content-Length header.
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_SERVER_CROSS_ORIGIN_REDIRECT	Unexpected cross-origin redirect.
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_USER_CANCELED	User canceled the download.
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_USER_SHUTDOWN	User shut down the WebView.
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_USER_PAUSED	User paused the download.
COREWEBVIEW2_DOWNLOAD_INTERRUPT_REASON_DOWNLOAD_PROCESS_CRASHED	WebView crashed.

Reason why a download was interrupted.

## COREWEBVIEW2\_DOWNLOAD\_STATE

enum [COREWEBVIEW2\\_DOWNLOAD\\_STATE](#)

[Expand table](#)

Values	Descriptions
COREWEBVIEW2_DOWNLOAD_STATE_IN_PROGRESS	The download is in progress.
COREWEBVIEW2_DOWNLOAD_STATE_INTERRUPTED	The connection with the file host was broken.
COREWEBVIEW2_DOWNLOAD_STATE_COMPLETED	The download completed successfully.

State of the download operation.

## COREWEBVIEW2\_FAVICON\_IMAGE\_FORMAT

enum [COREWEBVIEW2\\_FAVICON\\_IMAGE\\_FORMAT](#)

[Expand table](#)

Values	Descriptions
COREWEBVIEW2_FAVICON_IMAGE_FORMAT_PNG	Indicates that the PNG image format is used.
COREWEBVIEW2_FAVICON_IMAGE_FORMAT_JPEG	Indicates the JPEG image format is used.

Specifies the image format to use for favicon.

## COREWEBVIEW2\_FRAME\_KIND

enum [COREWEBVIEW2\\_FRAME\\_KIND](#)

[Expand table](#)

Values	Descriptions
COREWEBVIEW2_FRAME_KIND_UNKNOWN	Indicates that the frame is an unknown type frame.
COREWEBVIEW2_FRAME_KIND_MAIN_FRAME	Indicates that the frame is a primary main frame(webview).
COREWEBVIEW2_FRAME_KIND_IFRAME	Indicates that the frame is an iframe.
COREWEBVIEW2_FRAME_KIND_EMBED	Indicates that the frame is an embed element.
COREWEBVIEW2_FRAME_KIND_OBJECT	Indicates that the frame is an object element.

Indicates the frame type used in the ICoreWebView2FrameInfo interface.

## COREWEBVIEW2\_HOST\_RESOURCE\_ACCESS\_KIND

enum [COREWEBVIEW2\\_HOST\\_RESOURCE\\_ACCESS\\_KIND](#)

[Expand table](#)

Values	Descriptions
COREWEBVIEW2_HOST_RESOURCE_ACCESS_KIND_DENY	All cross origin resource access is denied, including normal sub resource access as src of a script or image element.
COREWEBVIEW2_HOST_RESOURCE_ACCESS_KIND_ALLOW	All cross origin resource access is allowed, including accesses that are subject to Cross-Origin Resource Sharing(CORS) check.
COREWEBVIEW2_HOST_RESOURCE_ACCESS_KIND_DENY_CORS	Cross origin resource access is allowed for normal sub resource access like as src of a script or image element, while any access that subjects to CORS check will be denied.

Kind of cross origin resource access allowed for host resources during download.

Note that other normal access checks like same origin DOM access check and [Content Security Policy](#) still apply. The following table illustrates the host resource cross origin access according to access context and `COREWEBVIEW2_HOST_RESOURCE_ACCESS_KIND`.

[Expand table](#)

Cross Origin Access Context	DENY	ALLOW	DENY_CORS
From DOM like src of img, script or iframe element	Deny	Allow	Allow
From Script like Fetch or XMLHttpRequest	Deny	Allow	Deny

## COREWEBVIEW2\_KEY\_EVENT\_KIND

enum [COREWEBVIEW2\\_KEY\\_EVENT\\_KIND](#)

[Expand table](#)

Values	Descriptions
COREWEBVIEW2_KEY_EVENT_KIND_KEY_DOWN	Specifies that the key event type corresponds to window message <code>WM_KEYDOWN</code> .
COREWEBVIEW2_KEY_EVENT_KIND_KEY_UP	Specifies that the key event type corresponds to window message <code>WM_KEYUP</code> .
COREWEBVIEW2_KEY_EVENT_KIND_SYSTEM_KEY_DOWN	Specifies that the key event type corresponds to window message <code>WM_SYSKEYDOWN</code> .
COREWEBVIEW2_KEY_EVENT_KIND_SYSTEM_KEY_UP	Specifies that the key event type corresponds to window message <code>WM_SYSKEYUP</code> .

Specifies the key event type that triggered an `AcceleratorKeyPressed` event.

## COREWEBVIEW2\_MEMORY\_USAGE\_TARGET\_LEVEL

enum [COREWEBVIEW2\\_MEMORY\\_USAGE\\_TARGET\\_LEVEL](#)

[Expand table](#)

Values	Descriptions
COREWEBVIEW2_MEMORY_USAGE_TARGET_LEVEL_NORMAL	Specifies normal memory usage target level.
COREWEBVIEW2_MEMORY_USAGE_TARGET_LEVEL_LOW	Specifies low memory usage target level.

Specifies memory usage target level of WebView.

## COREWEBVIEW2\_MOUSE\_EVENT\_KIND

enum [COREWEBVIEW2\\_MOUSE\\_EVENT\\_KIND](#)

[Expand table](#)

Values	Descriptions
COREWEBVIEW2_MOUSE_EVENT_KIND_HORIZONTAL_WHEEL	Mouse horizontal wheel scroll event, WM_MOUSEWHEEL.
COREWEBVIEW2_MOUSE_EVENT_KIND_LEFT_BUTTON_DOUBLE_CLICK	Left button double click mouse event, WM_LBUTTONDOWNDBLCLK.
COREWEBVIEW2_MOUSE_EVENT_KIND_LEFT_BUTTON_DOWN	Left button down mouse event, WM_LBUTTONDOWN.
COREWEBVIEW2_MOUSE_EVENT_KIND_LEFT_BUTTON_UP	Left button up mouse event, WM_LBUTTONUP.
COREWEBVIEW2_MOUSE_EVENT_KIND_LEAVE	Mouse leave event, WM_MOUSELEAVE.
COREWEBVIEW2_MOUSE_EVENT_KIND_MIDDLE_BUTTON_DOUBLE_CLICK	Middle button double click mouse event, WM_MBUTTONDOWNDBLCLK.
COREWEBVIEW2_MOUSE_EVENT_KIND_MIDDLE_BUTTON_DOWN	Middle button down mouse event, WM_MBUTTONDOWN.
COREWEBVIEW2_MOUSE_EVENT_KIND_MIDDLE_BUTTON_UP	Middle button up mouse event, WM_MBUTTONUP.
COREWEBVIEW2_MOUSE_EVENT_KIND_MOVE	Mouse move event, WM_MOUSEMOVE.
COREWEBVIEW2_MOUSE_EVENT_KIND_RIGHT_BUTTON_DOUBLE_CLICK	Right button double click mouse event, WM_RBUTTONDOWNDBLCLK.
COREWEBVIEW2_MOUSE_EVENT_KIND_RIGHT_BUTTON_DOWN	Right button down mouse event, WM_RBUTTONDOWN.
COREWEBVIEW2_MOUSE_EVENT_KIND_RIGHT_BUTTON_UP	Right button up mouse event, WM_RBUTTONUP.
COREWEBVIEW2_MOUSE_EVENT_KIND_WHEEL	Mouse wheel scroll event, WM_MOUSEWHEEL.
COREWEBVIEW2_MOUSE_EVENT_KIND_X_BUTTON_DOUBLE_CLICK	First or second X button double click mouse event, WM_XBUTTONDOWNDBLCLK.
COREWEBVIEW2_MOUSE_EVENT_KIND_X_BUTTON_DOWN	First or second X button down mouse event, WM_XBUTTONDOWN.

Values	Descriptions
COREWEBVIEW2_MOUSE_EVENT_KIND_X_BUTTON_UP	First or second X button up mouse event, WM_XBUTTONUP.
COREWEBVIEW2_MOUSE_EVENT_KIND_NON_CLIENT_RIGHT_BUTTON_DOWN	Mouse Right Button Down event over a nonclient area, WM_NCRBUTTONDOWN.
COREWEBVIEW2_MOUSE_EVENT_KIND_NON_CLIENT_RIGHT_BUTTON_UP	Mouse Right Button up event over a nonclient area, WM_NCRBUTTONUP.

Mouse event type used by SendMouseInput to convey the type of mouse event being sent to WebView.

The values of this enum align with the matching WM\_\* window messages.

## COREWEBVIEW2\_MOUSE\_EVENT\_VIRTUAL\_KEYS

enum [COREWEBVIEW2\\_MOUSE\\_EVENT\\_VIRTUAL\\_KEYS](#)

[Expand table](#)

Values	Descriptions
COREWEBVIEW2_MOUSE_EVENT_VIRTUAL_KEYS_NONE	No additional keys pressed.
COREWEBVIEW2_MOUSE_EVENT_VIRTUAL_KEYS_LEFT_BUTTON	Left mouse button is down, MK_LBUTTON.
COREWEBVIEW2_MOUSE_EVENT_VIRTUAL_KEYS_RIGHT_BUTTON	Right mouse button is down, MK_RBUTTON.
COREWEBVIEW2_MOUSE_EVENT_VIRTUAL_KEYS_SHIFT	SHIFT key is down, MK_SHIFT.
COREWEBVIEW2_MOUSE_EVENT_VIRTUAL_KEYS_CONTROL	CTRL key is down, MK_CONTROL.
COREWEBVIEW2_MOUSE_EVENT_VIRTUAL_KEYS_MIDDLE_BUTTON	Middle mouse button is down, MK_MBUTTON.
COREWEBVIEW2_MOUSE_EVENT_VIRTUAL_KEYS_X_BUTTON1	First X button is down, MK_XBUTTON1.
COREWEBVIEW2_MOUSE_EVENT_VIRTUAL_KEYS_X_BUTTON2	Second X button is down, MK_XBUTTON2.

Mouse event virtual keys associated with a COREWEBVIEW2\_MOUSE\_EVENT\_KIND for SendMouseInput.

These values can be combined into a bit flag if more than one virtual key is pressed for the event. The values of this enum align with the matching MK\_\* mouse keys.

## COREWEBVIEW2\_MOVE\_FOCUS\_REASON

enum [COREWEBVIEW2\\_MOVE\\_FOCUS\\_REASON](#)

[Expand table](#)

Values	Descriptions
COREWEBVIEW2_MOVE_FOCUS_REASON_PROGRAMMATIC	Specifies that the code is setting focus into WebView.
COREWEBVIEW2_MOVE_FOCUS_REASON_NEXT	Specifies that the focus is moving due to Tab traversal forward.

Values	Descriptions
COREWEBVIEW2_MOVE_FOCUS_REASON_PREVIOUS	Specifies that the focus is moving due to Tab traversal backward.

Specifies the reason for moving focus.

## COREWEBVIEW2\_NAVIGATION\_KIND

enum [COREWEBVIEW2\\_NAVIGATION\\_KIND](#)

[Expand table](#)

Values	Descriptions
COREWEBVIEW2_NAVIGATION_KIND_RELOAD	A navigation caused by <code>CoreWebView2.Reload()</code> , <code>location.reload()</code> , the end user using F5 or other UX, or other reload mechanisms to reload the current document without modifying the navigation history.
COREWEBVIEW2_NAVIGATION_KIND_BACK_OR_FORWARD	A navigation back or forward to a different entry in the session navigation history, like via <code>CoreWebView2.Back()</code> , <code>location.back()</code> , the end user pressing Alt+Left or other UX, or other mechanisms to navigate back or forward in the current session navigation history.
COREWEBVIEW2_NAVIGATION_KIND_NEW_DOCUMENT	A navigation to another document, which can be caused by <code>CoreWebView2.Navigate()</code> , <code>window.location.href = ...</code> , or other WebView2 or DOM APIs that navigate to a new URI.

Specifies the navigation kind of each navigation.

## COREWEBVIEW2\_PDF\_TOOLBAR\_ITEMS

enum [COREWEBVIEW2\\_PDF\\_TOOLBAR\\_ITEMS](#)

[Expand table](#)

Values	Descriptions
COREWEBVIEW2_PDF_TOOLBAR_ITEMS_NONE	No item.
COREWEBVIEW2_PDF_TOOLBAR_ITEMS_SAVE	The save button.
COREWEBVIEW2_PDF_TOOLBAR_ITEMS_PRINT	The print button.
COREWEBVIEW2_PDF_TOOLBAR_ITEMS_SAVE_AS	The save as button.
COREWEBVIEW2_PDF_TOOLBAR_ITEMS_ZOOM_IN	The zoom in button.
COREWEBVIEW2_PDF_TOOLBAR_ITEMS_ZOOM_OUT	The zoom out button.
COREWEBVIEW2_PDF_TOOLBAR_ITEMS_ROTATE	The rotate button.
COREWEBVIEW2_PDF_TOOLBAR_ITEMS_FIT_PAGE	The fit page button.
COREWEBVIEW2_PDF_TOOLBAR_ITEMS_PAGE_LAYOUT	The page layout button.

Values	Descriptions
COREWEBVIEW2_PDF_TOOLBAR_ITEMS_BOOKMARKS	The bookmarks button.
COREWEBVIEW2_PDF_TOOLBAR_ITEMS_PAGE_SELECTOR	The page select button.
COREWEBVIEW2_PDF_TOOLBAR_ITEMS_SEARCH	The search button.
COREWEBVIEW2_PDF_TOOLBAR_ITEMS_FULL_SCREEN	The full screen button.
COREWEBVIEW2_PDF_TOOLBAR_ITEMS_MORE_SETTINGS	The more settings button.

Specifies the PDF toolbar item types used for the `ICoreWebView2Settings::put_HiddenPdfToolbarItems` method.

## COREWEBVIEW2\_PERMISSION\_KIND

enum `COREWEBVIEW2_PERMISSION_KIND`

 Expand table

Values	Descriptions
COREWEBVIEW2_PERMISSION_KIND_UNKNOWN_PERMISSION	Indicates an unknown permission.
COREWEBVIEW2_PERMISSION_KIND_MICROPHONE	Indicates permission to capture audio.
COREWEBVIEW2_PERMISSION_KIND_CAMERA	Indicates permission to capture video.
COREWEBVIEW2_PERMISSION_KIND_GEOLOCATION	Indicates permission to access geolocation.
COREWEBVIEW2_PERMISSION_KIND_NOTIFICATIONS	Indicates permission to send web notifications.
COREWEBVIEW2_PERMISSION_KIND_OTHER_SENSORS	Indicates permission to access generic sensor.
COREWEBVIEW2_PERMISSION_KIND_CLIPBOARD_READ	Indicates permission to read the system clipboard without a user gesture.
COREWEBVIEW2_PERMISSION_KIND_MULTIPLE_AUTOMATIC_DOWNLOADS	Indicates permission to automatically download multiple files.
COREWEBVIEW2_PERMISSION_KIND_FILE_READ_WRITE	Indicates permission to read and write to files or folders on the device.
COREWEBVIEW2_PERMISSION_KIND_AUTOPLAY	Indicates permission to play audio and video automatically on sites.
COREWEBVIEW2_PERMISSION_KIND_LOCAL_FONTS	Indicates permission to use fonts on the device.
COREWEBVIEW2_PERMISSION_KIND_MIDI_SYSTEM_EXCLUSIVE_MESSAGES	Indicates permission to send and receive system exclusive messages to/from MIDI (Musical Instrument Digital Interface) devices.
COREWEBVIEW2_PERMISSION_KIND_WINDOW_MANAGEMENT	Indicates permission to open and place windows on the screen.

Indicates the type of a permission request.

## COREWEBVIEW2\_PERMISSION\_STATE

enum [COREWEBVIEW2\\_PERMISSION\\_STATE](#)

 [Expand table](#)

Values	Descriptions
COREWEBVIEW2_PERMISSION_STATE_DEFAULT	Specifies that the default browser behavior is used, which normally prompt users for decision.
COREWEBVIEW2_PERMISSION_STATE_ALLOW	Specifies that the permission request is granted.
COREWEBVIEW2_PERMISSION_STATE_DENY	Specifies that the permission request is denied.

Specifies the response to a permission request.

## COREWEBVIEW2\_POINTER\_EVENT\_KIND

enum [COREWEBVIEW2\\_POINTER\\_EVENT\\_KIND](#)

 [Expand table](#)

Values	Descriptions
COREWEBVIEW2_POINTER_EVENT_KIND_ACTIVATE	Corresponds to WM_POINTERACTIVATE.
COREWEBVIEW2_POINTER_EVENT_KIND_DOWN	Corresponds to WM_POINTERDOWN.
COREWEBVIEW2_POINTER_EVENT_KIND_ENTER	Corresponds to WM_POINTERENTER.
COREWEBVIEW2_POINTER_EVENT_KIND_LEAVE	Corresponds to WM_POINTERLEAVE.
COREWEBVIEW2_POINTER_EVENT_KIND_UP	Corresponds to WM_POINTERUP.
COREWEBVIEW2_POINTER_EVENT_KIND_UPDATE	Corresponds to WM_POINTERUPDATE.

Pointer event type used by `SendPointerInput` to convey the type of pointer event being sent to WebView.

The values of this enum align with the matching WM\_POINTER\* window messages.

## COREWEBVIEW2\_PREFERRED\_COLOR\_SCHEME

enum [COREWEBVIEW2\\_PREFERRED\\_COLOR\\_SCHEME](#)

 [Expand table](#)

Values	Descriptions
COREWEBVIEW2_PREFERRED_COLOR_SCHEME_AUTO	Auto color scheme.
COREWEBVIEW2_PREFERRED_COLOR_SCHEME_LIGHT	Light color scheme.

Values	Descriptions
COREWEBVIEW2_PREFERRED_COLOR_SCHEME_DARK	Dark color scheme.

An enum to represent the options for WebView2 color scheme: auto, light, or dark.

## COREWEBVIEW2\_PRINT\_COLLATION

| enum [COREWEBVIEW2\\_PRINT\\_COLLATION](#)

 [Expand table](#)

Values	Descriptions
COREWEBVIEW2_PRINT_COLLATION_DEFAULT	The default collation for a printer.
COREWEBVIEW2_PRINT_COLLATION_COLLATED	Indicate that the collation has been selected for the printed output.
COREWEBVIEW2_PRINT_COLLATION_UNCOLLATED	Indicate that the collation has not been selected for the printed output.

Specifies the collation for a print.

## COREWEBVIEW2\_PRINT\_COLOR\_MODE

| enum [COREWEBVIEW2\\_PRINT\\_COLOR\\_MODE](#)

 [Expand table](#)

Values	Descriptions
COREWEBVIEW2_PRINT_COLOR_MODE_DEFAULT	The default color mode for a printer.
COREWEBVIEW2_PRINT_COLOR_MODE_COLOR	Indicate that the printed output will be in color.
COREWEBVIEW2_PRINT_COLOR_MODE_GRAYSCALE	Indicate that the printed output will be in shades of gray.

Specifies the color mode for a print.

## COREWEBVIEW2\_PRINT\_DIALOG\_KIND

| enum [COREWEBVIEW2\\_PRINT\\_DIALOG\\_KIND](#)

 [Expand table](#)

Values	Descriptions
COREWEBVIEW2_PRINT_DIALOG_KIND_BROWSER	Opens the browser print preview dialog.
COREWEBVIEW2_PRINT_DIALOG_KIND_SYSTEM	Opens the system print dialog.

Specifies the print dialog kind.

## COREWEBVIEW2\_PRINT\_DUPLEX

enum [COREWEBVIEW2\\_PRINT\\_DUPLEX](#)

[Expand table](#)

Values	Descriptions
COREWEBVIEW2_PRINT_DUPLEX_DEFAULT	The default duplex for a printer.
COREWEBVIEW2_PRINT_DUPLEX_ONE_SIDED	Print on only one side of the sheet.
COREWEBVIEW2_PRINT_DUPLEX_TWO_SIDED_LONG_EDGE	Print on both sides of the sheet, flipped along the long edge.
COREWEBVIEW2_PRINT_DUPLEX_TWO_SIDED_SHORT_EDGE	Print on both sides of the sheet, flipped along the short edge.

Specifies the duplex option for a print.

## COREWEBVIEW2\_PRINT\_MEDIA\_SIZE

enum [COREWEBVIEW2\\_PRINT\\_MEDIA\\_SIZE](#)

[Expand table](#)

Values	Descriptions
COREWEBVIEW2_PRINT_MEDIA_SIZE_DEFAULT	The default media size for a printer.
COREWEBVIEW2_PRINT_MEDIA_SIZE_CUSTOM	Indicate custom media size that is specific to the printer.

Specifies the media size for a print.

## COREWEBVIEW2\_PRINT\_ORIENTATION

enum [COREWEBVIEW2\\_PRINT\\_ORIENTATION](#)

[Expand table](#)

Values	Descriptions
COREWEBVIEW2_PRINT_ORIENTATION_PORTRAIT	Print the page(s) in portrait orientation.
COREWEBVIEW2_PRINT_ORIENTATION_LANDSCAPE	Print the page(s) in landscape orientation.

The orientation for printing, used by the `Orientation` property on `ICoreWebView2PrintSettings`.

## COREWEBVIEW2\_PRINT\_STATUS

enum [COREWEBVIEW2\\_PRINT\\_STATUS](#)

[Expand table](#)

Values	Descriptions
COREWEBVIEW2_PRINT_STATUS_SUCCEEDED	Indicates that the print operation is succeeded.
COREWEBVIEW2_PRINT_STATUS_PRINTER_UNAVAILABLE	Indicates that the printer is not available.
COREWEBVIEW2_PRINT_STATUS_OTHER_ERROR	Indicates that the print operation is failed.

Indicates the status for printing.

## COREWEBVIEW2\_PROCESS\_FAILED\_KIND

enum [COREWEBVIEW2\\_PROCESS\\_FAILED\\_KIND](#)

[\[+\] Expand table](#)

Values	Descriptions
COREWEBVIEW2_PROCESS_FAILED_KIND_BROWSER_PROCESS_EXITED	Indicates that the browser process ended unexpectedly.
COREWEBVIEW2_PROCESS_FAILED_KIND_RENDER_PROCESS_EXITED	Indicates that the main frame's render process ended unexpectedly.
COREWEBVIEW2_PROCESS_FAILED_KIND_RENDER_PROCESS_UNRESPONSIVE	Indicates that the main frame's render process is unresponsive.
COREWEBVIEW2_PROCESS_FAILED_KIND_FRAME_RENDER_PROCESS_EXITED	Indicates that a frame-only render process ended unexpectedly.
COREWEBVIEW2_PROCESS_FAILED_KIND.Utility_PROCESS_EXITED	Indicates that a utility process ended unexpectedly.
COREWEBVIEW2_PROCESS_FAILED_KIND_SANDBOX_HELPER_PROCESS_EXITED	Indicates that a sandbox helper process ended unexpectedly.
COREWEBVIEW2_PROCESS_FAILED_KIND_GPU_PROCESS_EXITED	Indicates that the GPU process ended unexpectedly.
COREWEBVIEW2_PROCESS_FAILED_KIND_PPAPI_PLUGIN_PROCESS_EXITED	Indicates that a PPAPI plugin process ended unexpectedly.
COREWEBVIEW2_PROCESS_FAILED_KIND_PPAPI_BROKER_PROCESS_EXITED	Indicates that a PPAPI plugin broker process ended unexpectedly.
COREWEBVIEW2_PROCESS_FAILED_KIND_UNKNOWN_PROCESS_EXITED	Indicates that a process of unspecified kind ended unexpectedly.

Specifies the process failure type used in the `ICoreWebView2ProcessFailedEventArgs` interface.

The values in this enum make reference to the process kinds in the Chromium architecture. For more information about what these processes are and what they do, see [Browser Architecture - Inside look at modern web browser](#).

## COREWEBVIEW2\_PROCESS\_FAILED\_REASON

enum [COREWEBVIEW2\\_PROCESS\\_FAILED\\_REASON](#)

[Expand table](#)

Values	Descriptions
COREWEBVIEW2_PROCESS_FAILED_REASON_UNEXPECTED	An unexpected process failure occurred.
COREWEBVIEW2_PROCESS_FAILED_REASON_UNRESPONSIVE	The process became unresponsive.
COREWEBVIEW2_PROCESS_FAILED_REASON_TERMINATED	The process was terminated. For example, from Task Manager.
COREWEBVIEW2_PROCESS_FAILED_REASON_CRASHED	The process crashed.
COREWEBVIEW2_PROCESS_FAILED_REASON_LAUNCH_FAILED	The process failed to launch.
COREWEBVIEW2_PROCESS_FAILED_REASON_OUT_OF_MEMORY	The process terminated due to running out of memory.
COREWEBVIEW2_PROCESS_FAILED_REASON_PROFILE_DELETED	The process exited because its corresponding profile was deleted.

Specifies the process failure reason used in the `ICoreWebView2ProcessFailedEventArgs` interface.

For process failures where a process has exited, it indicates the type of issue that produced the process exit.

## COREWEBVIEW2\_PROCESS\_KIND

enum [COREWEBVIEW2\\_PROCESS\\_KIND](#)

[Expand table](#)

Values	Descriptions
COREWEBVIEW2_PROCESS_KIND_BROWSER	Indicates the browser process kind.
COREWEBVIEW2_PROCESS_KIND_RENDERER	Indicates the render process kind.
COREWEBVIEW2_PROCESS_KIND.Utility	Indicates the utility process kind.
COREWEBVIEW2_PROCESS_KIND_SANDBOX_HELPER	Indicates the sandbox helper process kind.
COREWEBVIEW2_PROCESS_KIND_GPU	Indicates the GPU process kind.
COREWEBVIEW2_PROCESS_KIND_PPAPI_PLUGIN	Indicates the PPAPI plugin process kind.
COREWEBVIEW2_PROCESS_KIND_PPAPI_BROKER	Indicates the PPAPI plugin broker process kind.

Indicates the process type used in the `ICoreWebView2ProcessInfo` interface.

## COREWEBVIEW2\_SCRIPT\_DIALOG\_KIND

enum [COREWEBVIEW2\\_SCRIPT\\_DIALOG\\_KIND](#)

[Expand table](#)

Values	Descriptions
COREWEBVIEW2_SCRIPT_DIALOG_KIND_ALERT	Indicates that the dialog uses the <code>window.alert</code> JavaScript

Values	Descriptions
COREWEBVIEW2_SCRIPT_DIALOG_KIND_CONFIRM	Indicates that the dialog uses the <code>window.confirm</code> JavaScript function.
COREWEBVIEW2_SCRIPT_DIALOG_KIND_PROMPT	Indicates that the dialog uses the <code>window.prompt</code> JavaScript function.
COREWEBVIEW2_SCRIPT_DIALOG_KIND_BEFOREUNLOAD	Indicates that the dialog uses the <code>beforeunload</code> JavaScript event.

Specifies the JavaScript dialog type used in the `ICoreWebView2ScriptDialogOpeningEventHandler` interface.

## COREWEBVIEW2\_SERVER\_CERTIFICATE\_ERROR\_ACTION

enum `COREWEBVIEW2_SERVER_CERTIFICATE_ERROR_ACTION`

 [Expand table](#)

Values	Descriptions
COREWEBVIEW2_SERVER_CERTIFICATE_ERROR_ACTION_ALWAYS_ALLOW	Indicates to ignore the warning and continue the request with the TLS certificate.
COREWEBVIEW2_SERVER_CERTIFICATE_ERROR_ACTION_CANCEL	Indicates to reject the certificate and cancel the request.
COREWEBVIEW2_SERVER_CERTIFICATE_ERROR_ACTION_DEFAULT	Indicates to display the default TLS interstitial error page to user for page navigations.

Specifies the action type when server certificate error is detected to be used in the `ICoreWebView2ServerErrorDetectedEventArgs` interface.

## COREWEBVIEW2\_SHARED\_BUFFER\_ACCESS

enum `COREWEBVIEW2_SHARED_BUFFER_ACCESS`

 [Expand table](#)

Values	Descriptions
COREWEBVIEW2_SHARED_BUFFER_ACCESS_READ_ONLY	Script from web page only has read access to the shared buffer.
COREWEBVIEW2_SHARED_BUFFER_ACCESS_READ_WRITE	Script from web page has read and write access to the shared buffer.

Specifies the desired access from script to `coreWebView2SharedBuffer`.

## COREWEBVIEW2\_TRACKING\_PREVENTION\_LEVEL

enum `COREWEBVIEW2_TRACKING_PREVENTION_LEVEL`

[Expand table](#)

Values	Descriptions
COREWEBVIEW2_TRACKING_PREVENTION_LEVEL_NONE	Tracking prevention is turned off.
COREWEBVIEW2_TRACKING_PREVENTION_LEVEL_BASIC	The least restrictive level of tracking prevention.
COREWEBVIEW2_TRACKING_PREVENTION_LEVEL_BALANCED	The default level of tracking prevention.
COREWEBVIEW2_TRACKING_PREVENTION_LEVEL_STRICT	The most restrictive level of tracking prevention.

Tracking prevention levels.

## COREWEBVIEW2\_WEB\_ERROR\_STATUS

enum [COREWEBVIEW2\\_WEB\\_ERROR\\_STATUS](#)

[Expand table](#)

Values	Descriptions
COREWEBVIEW2_WEB_ERROR_STATUS_UNKNOWN	Indicates that an unknown error occurred.
COREWEBVIEW2_WEB_ERROR_STATUS_CERTIFICATE_COMMON_NAME_IS_INCORRECT	Indicates that the SSL certificate common name does not match the web address.
COREWEBVIEW2_WEB_ERROR_STATUS_CERTIFICATE_EXPIRED	Indicates that the SSL certificate has expired.
COREWEBVIEW2_WEB_ERROR_STATUS_CLIENT_CERTIFICATE_CONTAINS_ERRORS	Indicates that the SSL client certificate contains errors.
COREWEBVIEW2_WEB_ERROR_STATUS_CERTIFICATE_REVOKED	Indicates that the SSL certificate has been revoked.
COREWEBVIEW2_WEB_ERROR_STATUS_CERTIFICATE_IS_INVALID	Indicates that the SSL certificate is not valid.
COREWEBVIEW2_WEB_ERROR_STATUS_SERVER_UNREACHABLE	Indicates that the host is unreachable.
COREWEBVIEW2_WEB_ERROR_STATUS_TIMEOUT	Indicates that the connection has timed out.
COREWEBVIEW2_WEB_ERROR_STATUS_ERROR_HTTP_INVALID_SERVER_RESPONSE	Indicates that the server returned an invalid or unrecognized response.
COREWEBVIEW2_WEB_ERROR_STATUS_CONNECTION_ABORTED	Indicates that the connection was stopped.
COREWEBVIEW2_WEB_ERROR_STATUS_CONNECTION_RESET	Indicates that the connection was reset.
COREWEBVIEW2_WEB_ERROR_STATUS_DISCONNECTED	Indicates that the Internet connection has been lost.

Values	Descriptions
COREWEBVIEW2_WEB_ERROR_STATUS_CANNOT_CONNECT	Indicates that a connection to the destination was not established.
COREWEBVIEW2_WEB_ERROR_STATUS_HOST_NAME_NOT_RESOLVED	Indicates that the provided host name was not able to be resolved.
COREWEBVIEW2_WEB_ERROR_STATUS_OPERATION_CANCELED	Indicates that the operation was canceled.
COREWEBVIEW2_WEB_ERROR_STATUS_REDIRECT_FAILED	Indicates that the request redirect failed.
COREWEBVIEW2_WEB_ERROR_STATUS_UNEXPECTED_ERROR	Indicates that an unexpected error occurred.
COREWEBVIEW2_WEB_ERROR_STATUS_VALID_AUTHENTICATION_CREDENTIALS_REQUIRED	Indicates that user is prompted with a login, waiting on user action.
COREWEBVIEW2_WEB_ERROR_STATUS_VALID_PROXY_AUTHENTICATION_REQUIRED	Indicates that user lacks proper authentication credentials for a proxy server.

Indicates the error status values for web navigations.

## COREWEBVIEW2\_WEB\_RESOURCE\_CONTEXT

enum [COREWEBVIEW2\\_WEB\\_RESOURCE\\_CONTEXT](#)

[Expand table](#)

Values	Descriptions
COREWEBVIEW2_WEB_RESOURCE_CONTEXT_ALL	Specifies all resources.
COREWEBVIEW2_WEB_RESOURCE_CONTEXT_DOCUMENT	Specifies a document resource.
COREWEBVIEW2_WEB_RESOURCE_CONTEXT_STYLESHEET	Specifies a CSS resource.
COREWEBVIEW2_WEB_RESOURCE_CONTEXT_IMAGE	Specifies an image resource.
COREWEBVIEW2_WEB_RESOURCE_CONTEXT_MEDIA	Specifies another media resource such as a video.
COREWEBVIEW2_WEB_RESOURCE_CONTEXT_FONT	Specifies a font resource.
COREWEBVIEW2_WEB_RESOURCE_CONTEXT_SCRIPT	Specifies a script resource.
COREWEBVIEW2_WEB_RESOURCE_CONTEXT_XML_HTTP_REQUEST	Specifies an XML HTTP request, Fetch and EventSource API communication.
COREWEBVIEW2_WEB_RESOURCE_CONTEXT_FETCH	Specifies a Fetch API communication.
COREWEBVIEW2_WEB_RESOURCE_CONTEXT_TEXT_TRACK	Specifies a TextTrack resource.
COREWEBVIEW2_WEB_RESOURCE_CONTEXT_EVENT_SOURCE	Specifies an EventSource API communication.

Values	Descriptions
COREWEBVIEW2_WEB_RESOURCE_CONTEXT_WEBSOCKET	Specifies a WebSocket API communication.
COREWEBVIEW2_WEB_RESOURCE_CONTEXT_MANIFEST	Specifies a Web App Manifest.
COREWEBVIEW2_WEB_RESOURCE_CONTEXT_SIGNED_EXCHANGE	Specifies a Signed HTTP Exchange.
COREWEBVIEW2_WEB_RESOURCE_CONTEXT_PING	Specifies a Ping request.
COREWEBVIEW2_WEB_RESOURCE_CONTEXT_CSP_VIOLATION_REPORT	Specifies a CSP Violation Report.
COREWEBVIEW2_WEB_RESOURCE_CONTEXT_OTHER	Specifies an other resource.

Specifies the web resource request contexts.

## COREWEBVIEW2\_WEB\_RESOURCE\_REQUEST\_SOURCE\_KINDS

enum [COREWEBVIEW2\\_WEB\\_RESOURCE\\_REQUEST\\_SOURCE\\_KINDS](#)

[Expand table](#)

Values	Descriptions
COREWEBVIEW2_WEB_RESOURCE_REQUEST_SOURCE_KINDS_NONE	
COREWEBVIEW2_WEB_RESOURCE_REQUEST_SOURCE_KINDS_DOCUMENT	Indicates that web resource is requested from main page including dedicated workers and iframes.
COREWEBVIEW2_WEB_RESOURCE_REQUEST_SOURCE_KINDS_SHARED_WORKER	Indicates that web resource is requested from shared worker.
COREWEBVIEW2_WEB_RESOURCE_REQUEST_SOURCE_KINDS_SERVICE_WORKER	Indicates that web resource is requested from service worker.
COREWEBVIEW2_WEB_RESOURCE_REQUEST_SOURCE_KINDS_ALL	Indicates that web resource is requested from any supported source.

Specifies the source of `WebResourceRequested` event.

## COREWEBVIEW2\_COLOR

A value representing RGBA color (Red, Green, Blue, Alpha) for WebView2.

typedef [COREWEBVIEW2\\_COLOR](#) Each component takes a value from 0 to 255, with 0 being no intensity and 255 being the highest intensity.

## CompareBrowserVersions

public STDAPI [CompareBrowserVersions](#)(PCWSTR version1, PCWSTR version2, int \* result)

This method is for anyone want to compare version correctly to determine which version is newer, older or same.

Use it to determine whether to use webview2 or certain feature based upon version. Sets the value of result to `-1`, `0` or `1` if `version1` is less than, equal or greater than `version2` respectively. Returns `E_INVALIDARG` if it fails to parse any of the version strings or any input parameter is `null`. Directly use the `versionInfo` obtained from `GetAvailableCoreWebView2BrowserVersionString` with input, channel information is ignored.

## CreateCoreWebView2Environment

```
public STDAPI  
CreateCoreWebView2Environment(ICoreWebView2CreateCoreWebView2EnvironmentCompletedHandler *  
environmentCreatedHandler)
```

Creates an evergreen WebView2 Environment using the installed WebView2 Runtime version.

This is equivalent to running `CreateCoreWebView2EnvironmentWithOptions` with `nullptr` for `browserExecutableFolder`, `userDataFolder`, `additionalBrowserArguments`. For more information, navigate to `CreateCoreWebView2EnvironmentWithOptions`.

## CreateCoreWebView2EnvironmentWithOptions

```
public STDAPI CreateCoreWebView2EnvironmentWithOptions(PCWSTR browserExecutableFolder,  
PCWSTR userDataFolder, ICoreWebView2EnvironmentOptions * environmentOptions,  
ICoreWebView2CreateCoreWebView2EnvironmentCompletedHandler * environmentCreatedHandler)
```

DLL export to create a WebView2 environment with a custom version of WebView2 Runtime, user data folder, and with or without additional options.

When WebView2 experimental APIs are used, make sure to provide a valid `environmentOptions` so that WebView2 runtime knows which version of the SDK that the app is using. Otherwise, WebView2 runtime assumes that the version of the SDK being used is the latest version known to it, which might not be the version of the SDK being used. This wrong SDK version assumption could result in some experimental APIs not being available.

The WebView2 environment and all other WebView2 objects are single threaded and have dependencies on Windows components that require COM to be initialized for a single-threaded apartment. The app is expected to run `CoInitializeEx` before running `CreateCoreWebView2EnvironmentWithOptions`.

```
text  
  
CoInitializeEx(nullptr, COINIT_APARTMENTTHREADED);
```

If `CoInitializeEx` did not run or previously ran with `COINIT_MULTITHREADED`, `CreateCoreWebView2EnvironmentWithOptions` fails with one of the following errors.

```
text  
  
CO_E_NOTINITIALIZED - if CoInitializeEx was not called  
RPC_E_CHANGED_MODE - if CoInitializeEx was previously called with  
COINIT_MULTITHREADED
```

Use `browserExecutableFolder` to specify whether WebView2 controls use a fixed or installed version of the WebView2 Runtime that exists on a user machine. To use a fixed version of the WebView2 Runtime, pass the folder path that contains the fixed version of the WebView2 Runtime to `browserExecutableFolder`. BrowserExecutableFolder supports both relative (to the application's executable) and absolute files paths. To create WebView2 controls that use the installed version of the WebView2 Runtime that exists on user machines, pass a `null` or empty string to `browserExecutableFolder`. In this scenario, the API tries to find a compatible version of the WebView2 Runtime that is installed on the user machine (first at the machine level, and then per user) using the selected channel preference. The path of fixed version of the WebView2 Runtime should not contain `\Edge\Application\`. When such a path is used, the API fails with `HRESULT_FROM_WIN32(ERROR_NOT_SUPPORTED)`.

The default channel search order is the WebView2 Runtime, Beta, Dev, and Canary. When an override `WEBVIEW2_RELEASE_CHANNEL_PREFERENCE` environment variable or applicable `releaseChannelPreference` registry value is set to `1`, the channel search order is reversed.

You may specify the `userDataFolder` to change the default user data folder location for WebView2. The path is either an absolute file path or a relative file path that is interpreted as relative to the compiled code for the current process. For UWP apps, the default user data folder is the app data folder for the package. For non-UWP apps, the default user data (`{Executable File Name}.WebView2`) folder is created in the same directory next to the compiled code for the app. WebView2 creation fails if the compiled code is running in a directory in which the process does not have permission to create a new directory. The app is responsible to clean up the associated user data folder when it is done.

#### ① Note

As a browser process may be shared among WebViews, WebView creation fails with `HRESULT_FROM_WIN32(ERROR_INVALID_STATE)` if the specified options does not match the options of the WebViews that are currently running in the shared browser process.

`environmentCreatedHandler` is the handler result to the async operation that contains the `WebView2Environment` that was created.

The `browserExecutableFolder`, `userDataFolder` and `additionalBrowserArguments` of the `environmentOptions` may be overridden by values either specified in environment variables or in the registry.

When creating a `WebView2Environment` the following environment variables are verified.

text

WEBVIEW2\_BROWSER\_EXECUTABLE\_FOLDER  
WEBVIEW2\_USER\_DATA\_FOLDER  
WEBVIEW2\_ADDITIONAL\_BROWSER\_ARGUMENTS  
WEBVIEW2\_RELEASE\_CHANNEL\_PREFERENCE

If you find an override environment variable, use the `browserExecutableFolder` and `userDataFolder` values as replacements for the corresponding values in `CreateCoreWebView2EnvironmentWithOptions` parameters. If `additionalBrowserArguments` is specified in environment variable or in the registry, it is appended to the corresponding values in `CreateCoreWebView2EnvironmentWithOptions` parameters.

While not strictly overrides, additional environment variables may be set.

text

WEBVIEW2\_WAIT\_FOR\_SCRIPT\_DEBUGGER

When found with a non-empty value, this indicates that the WebView is being launched under a script debugger. In this case, the WebView issues a `Page.waitForDebugger` CDP command that runs the script inside the WebView to pause on launch, until a debugger issues a corresponding `Runtime.runIfWaitingForDebugger` CDP command to resume the runtime.

 **Note**

The following environment variable does not have a registry key equivalent:

`WEBVIEW2_WAIT_FOR_SCRIPT_DEBUGGER`.

When found with a non-empty value, it indicates that the WebView is being launched under a script debugger that also supports host apps that use multiple WebViews. The value is used as the identifier for a named pipe that is opened and written to when a new WebView is created by the host app. The payload should match the payload of the `remote-debugging-port` JSON target and an external debugger may use it to attach to a specific WebView instance. The format of the pipe created by the debugger should be `\.\.\pipe\WebView2\Debugger\{app_name}\{pipe_name}`, where the following are true.

- `{app_name}` is the host app exe file name, for example, `WebView2Example.exe`
- `{pipe_name}` is the value set for `WEBVIEW2_PIPE_FOR_SCRIPT_DEBUGGER`

To enable debugging of the targets identified by the JSON, you must set the

`WEBVIEW2_ADDITIONAL_BROWSER_ARGUMENTS` environment variable to send `--remote-debugging-port={port_num}`, where the following is true.

- `{port_num}` is the port on which the CDP server binds.

 **Warning**

If you set both `WEBVIEW2_PIPE_FOR_SCRIPT_DEBUGGER` and `WEBVIEW2_ADDITIONAL_BROWSER_ARGUMENTS` environment variables, the WebViews hosted in your app and associated contents may be exposed to 3rd party apps such as debuggers.

 **Note**

The following environment variable does not have a registry key equivalent:

`WEBVIEW2_PIPE_FOR_SCRIPT_DEBUGGER`.

If none of those environment variables exist, then the registry is examined next. The following registry values are verified.

text

`[{Root}]\Software\Policies\Microsoft\Edge\WebView2\BrowserExecutableFolder`  
`"{AppId}"=""`

```
[{Root}]\Software\Policies\Microsoft\Edge\WebView2\ReleaseChannelPreference  
"{AppId}=""  
  
[{Root}]\Software\Policies\Microsoft\Edge\WebView2\AdditionalBrowserArguments  
"{AppId}=""  
  
[{Root}]\Software\Policies\Microsoft\Edge\WebView2\UserDataFolder  
"{AppId}=""
```

Use a group policy under **Administrative Templates > Microsoft Edge WebView2** to configure `browserExecutableFolder` and `releaseChannelPreference`.

In the unlikely scenario where some instances of WebView are open during a browser update, the deletion of the previous WebView2 Runtime may be blocked. To avoid running out of disk space, a new WebView creation fails with `HRESULT_FROM_WIN32(ERROR_DISK_FULL)` if it detects that too many previous WebView2 Runtime versions exist.

The default maximum number of WebView2 Runtime versions allowed is `20`. To override the maximum number of the previous WebView2 Runtime versions allowed, set the value of the following environment variable.

text

`COREWEBVIEW2_MAX_INSTANCES`

If the Webview depends upon an installed WebView2 Runtime version and it is uninstalled, any subsequent creation fails with `HRESULT_FROM_WIN32(ERROR_PRODUCT_UNINSTALLED)`.

First verify with Root as `HKLM` and then `HKCU`. `AppId` is first set to the Application User Model ID of the process, then if no corresponding registry key, the `AppId` is set to the compiled code name of the process, or if that is not a registry key then `*`. If an override registry key is found, use the `browserExecutableFolder` and `userDataFolder` registry values as replacements and append `additionalBrowserArguments` registry values for the corresponding values in `CreateCoreWebView2EnvironmentWithOptions` parameters.

The following summarizes the possible error values that can be returned from `CreateCoreWebView2EnvironmentWithOptions` and a description of why these errors occur.

[Expand table](#)

Error value	Description
<code>CO_E_NOTINITIALIZED</code>	ColInitializeEx was not called.
<code>RPC_E_CHANGED_MODE</code>	ColInitializeEx was previously called with COINIT_MULTITHREADED.
<code>HRESULT_FROM_WIN32(ERROR_NOT_SUPPORTED)</code>	\Edge\Application path used in browserExecutableFolder.
<code>HRESULT_FROM_WIN32(ERROR_INVALID_STATE)</code>	Specified options do not match the options of the WebViews that are currently running in the shared browser process.
<code>HRESULT_FROM_WIN32(ERROR_DISK_FULL)</code>	In the unlikely scenario where some instances of WebView are open during a browser update, the deletion of the previous WebView2 Runtime may be blocked. To avoid running out of disk space, a new WebView creation fails with <code>HRESULT_FROM_WIN32(ERROR_DISK_FULL)</code> if it detects that too many previous WebView2 Runtime versions exist.

Error value	Description
<code>HRESULT_FROM_WIN32(ERROR_PRODUCT_UNINSTALLED)</code>	If the Webview depends upon an installed WebView2 Runtime version and it is uninstalled.
<code>HRESULT_FROM_WIN32(ERROR_FILE_NOT_FOUND)</code>	Could not find Edge installation.
<code>HRESULT_FROM_WIN32(ERROR_FILE_EXISTS)</code>	User data folder cannot be created because a file with the same name already exists.
<code>E_ACCESSDENIED</code>	Unable to create user data folder, Access Denied.
<code>E_FAIL</code>	Edge runtime unable to start.

## CreateWebViewEnvironmentWithOptionsInternal

```
public STDAPI CreateWebViewEnvironmentWithOptionsInternal(bool checkRunningInstance, int
runtimeType, PCWSTR userDataFolder, IUnknown * environmentOptions,
ICoreWebView2CreateCoreWebView2EnvironmentCompletedHandler *
webViewEnvironmentCreatedHandler)
```

This is a DLL export out of `EmbeddedBrowserWebView.dll` which can be found in the installation folder of the WebView2 Runtime you wish to use.

### ⓘ Note

This function may be modified or removed in future versions. It is recommended you use `CreateCoreWebView2EnvironmentWithOptions` instead of this function.

This function creates a WebView2 environment with a specified version of WebView2 Runtime, user data folder, and with or without additional options.

This is an internal method used by `CreateCoreWebView2EnvironmentWithOptions` that acts similar to `CreateCoreWebView2EnvironmentWithOptions`, but it will only create an `ICoreWebView2Environment` from the WebView2 Runtime of the module on which you call `CreateWebViewEnvironmentWithOptionsInternal`. This is unlike `CreateCoreWebView2EnvironmentWithOptions` which handles many other cases including: falling back to other non-stable WebView2 Runtime channels when the stable WebView2 Runtime is not available, handling developer environment variables to change the runtime, handling policy registry keys to change the runtime, and others. You should use `CreateCoreWebView2EnvironmentWithOptions` rather than this method.

If `checkRunningInstance` is set then `CreateWebViewEnvironmentWithOptionsInternal` will forward the creation call to a different WebView2 Runtime if there is already a different WebView2 Runtime running for the specified user data folder. This matches `CreateCoreWebView2EnvironmentWithOptions` behavior. If not set, then this forwarding will not occur and creation will fail if there is already a different WebView2 Runtime running for the specified user data folder.

The `runtimeType` parameter is used to indicate if the WebView2 Runtime is fixed version with value `1`, evergreen with value `0`, or unknown with value `-1`.

See the `CreateCoreWebView2EnvironmentWithOptions` documentation for information on the `userDataFolder`, `environmentOptions`, and `webViewEnvironmentCreatedHandler` parameters which match the parameters from

`CreateCoreWebView2EnvironmentWithOptions`.

## GetAvailableCoreWebView2BrowserVersionString

```
public STDAPI GetAvailableCoreWebView2BrowserVersionString(PCWSTR browserExecutableFolder,  
LPWSTR * versionInfo)
```

Get the browser version info including channel name if it is not the WebView2 Runtime.

Channel names are Beta, Dev, and Canary. If an override exists for the `browserExecutableFolder` or the channel preference, the override is used. If an override is not specified, then the parameter value passed to `GetAvailableCoreWebView2BrowserVersionString` is used. Returns `HRESULT_FROM_WIN32(ERROR_FILE_NOT_FOUND)` if it fails to find an installed WebView2 runtime or non-stable Microsoft Edge installation.

The caller must free the returned string with `CoTaskMemFree`. See [API Conventions](#).

---

## Feedback

Was this page helpful?



# interface

## ICoreWebView2AcceleratorKeyPressedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2AcceleratorKeyPressedEventHandler
: public IUnknown
```

Receives `AcceleratorKeyPressed` events.

## Summary

Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2Controller * sender,
ICoreWebView2AcceleratorKeyPressedEventArgs * args)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2AddScriptToExecuteOnDocumentCreatedCompletedHandler

Article • 02/26/2024

```
interface ICoreWebView2AddScriptToExecuteOnDocumentCreatedCompletedHandler
: public IUnknown
```

Receives the result of the `AddScriptToExecuteOnDocumentCreated` method.

## Summary

Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provide the completion status and result of the corresponding asynchronous method.

## Applies to

Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### Invoke

Provide the completion status and result of the corresponding asynchronous method.

```
public HRESULT Invoke(HRESULT errorCode, LPCWSTR id)
```

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2BasicAuthenticationRequestedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2BasicAuthenticationRequestedEventHandler
: public IUnknown
```

The caller implements this interface to handle the BasicAuthenticationRequested event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Called to provide the implementer with the event args for the corresponding event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1150.38
WebView2 Win32 Prerelease	1.0.1133

## Members

### Invoke

Called to provide the implementer with the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2 * sender,
ICoreWebView2BasicAuthenticationRequestedEventArgs * args)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2BrowserExtensionEnableCompletedHandler

Article • 02/26/2024

```
interface ICoreWebView2BrowserExtensionEnableCompletedHandler
: public IUnknown
```

The caller implements this interface to receive the result of setting the browser Extension as enabled or disabled.

## Summary

Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the result of the browser extension enable operation.

If enabled, the browser Extension is running in WebView instances. If disabled, the browser Extension is not running in WebView instances.

## Applies to

Expand table

Product	Introduced
WebView2 Win32	1.0.2210.55
WebView2 Win32 Prerelease	1.0.2194

## Members

### Invoke

Provides the result of the browser extension enable operation.

```
public HRESULT Invoke(HRESULT errorCode)
```

---

## Feedback

Was this page helpful?



# interface

## ICoreWebView2BrowserExtensionRemoveCompletedHandler

Article • 02/26/2024

```
interface ICoreWebView2BrowserExtensionRemoveCompletedHandler
: public IUnknown
```

The caller implements this interface to receive the result of removing the browser Extension from the Profile.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the result of the browser extension Remove operation.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.2210.55
WebView2 Win32 Prerelease	1.0.2194

## Members

### Invoke

Provides the result of the browser extension Remove operation.

```
public HRESULT Invoke(HRESULT errorCode)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2BrowserProcessExitedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2BrowserProcessExitedEventHandler
: public IUnknown
```

Receives `BrowserProcessExited` events.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.992.28
WebView2 Win32 Prerelease	1.0.1010

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2Environment * sender,
ICoreWebView2BrowserProcessExitedEventArgs * args)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2BytesReceivedChangedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2BytesReceivedChangedEventHandler  
: public IUnknown
```

Implements the interface to receive `BytesReceivedChanged` event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

Use the `ICoreWebView2DownloadOperation.BytesReceived` property to get the received bytes count.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.902.49
WebView2 Win32 Prerelease	1.0.902

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2DownloadOperation * sender, IUnknown *  
args)
```

No event args exist and the `args` parameter is set to `null`.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2CallDevToolsProtocolMethodCompletedHandler

Article • 02/26/2024

```
interface ICoreWebView2CallDevToolsProtocolMethodCompletedHandler
: public IUnknown
```

Receives `CallDevToolsProtocolMethod` completion results.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the completion status and result of the corresponding asynchronous method.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### Invoke

Provides the completion status and result of the corresponding asynchronous method.

```
public HRESULT Invoke(HRESULT errorCode, LPCWSTR returnObjectAsJson)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2CapturePreviewCompletedHandler

Article • 02/26/2024

```
interface ICoreWebView2CapturePreviewCompletedHandler
: public IUnknown
```

Receives the result of the `CapturePreview` method.

## Summary

Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the completion status of the corresponding asynchronous method.

The result is written to the stream provided in the `CapturePreview` method.

## Applies to

Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### Invoke

Provides the completion status of the corresponding asynchronous method.

```
public HRESULT Invoke(HRESULT errorCode)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2ClearBrowsingDataCompletedHandler

Article • 02/26/2024

```
interface ICoreWebView2ClearBrowsingDataCompletedHandler
: public IUnknown
```

The caller implements this interface to receive the ClearBrowsingData result.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provide the completion status of the corresponding asynchronous method.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1245.22
WebView2 Win32 Prerelease	1.0.1248

## Members

### Invoke

Provide the completion status of the corresponding asynchronous method.

```
public HRESULT Invoke(HRESULT errorCode)
```

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2ClearServerCertificateErrorActionsCompletedHandler

Article • 02/26/2024

```
interface ICoreWebView2ClearServerCertificateErrorActionsCompletedHandler
: public IUnknown
```

Receives the result of the `clearServerCertificateErrorActions` method.

## Summary

Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the result of the corresponding asynchronous method.

## Applies to

Expand table

Product	Introduced
WebView2 Win32	1.0.1245.22
WebView2 Win32 Prerelease	1.0.1248

## Members

### Invoke

Provides the result of the corresponding asynchronous method.

```
public HRESULT Invoke(HRESULT errorCode)
```

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2ClientCertificateRequestedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2ClientCertificateRequestedEventHandler
: public IUnknown
```

An event handler for the `ClientCertificateRequested` event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.961.33
WebView2 Win32 Prerelease	1.0.955

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2 * sender,
ICoreWebView2ClientCertificateRequestedEventArgs * args)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2ContainsFullScreenElementChangedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2ContainsFullScreenElementChangedEventHandler
: public IUnknown
```

Receives `ContainsFullScreenElementChanged` events.

## Summary

Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2 * sender, IUnknown * args)
```

No event args exist and the `args` parameter is set to `null`.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2ContentLoadingEventHandler

Article • 02/26/2024

```
interface ICoreWebView2ContentLoadingEventHandler
: public IUnknown
```

Receives `ContentLoading` events.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2 * sender,
ICoreWebView2ContentLoadingEventArgs * args)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2ContextMenuRequestedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2ContextMenuRequestedEventHandler
: public IUnknown
```

Receives `ContextMenuRequested` events.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Called to provide the event args when a context menu is requested on a WebView element.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1185.39
WebView2 Win32 Prerelease	1.0.1189

## Members

### Invoke

Called to provide the event args when a context menu is requested on a WebView element.

```
public HRESULT Invoke(ICoreWebView2 * sender,  
ICoreWebView2ContextMenuRequestedEventArgs * args)
```

---

## Feedback

Was this page helpful?



# interface

## ICoreWebView2CreateCoreWebView2CompositionControllerCompletedHandler

Article • 02/26/2024

```
interface  
ICoreWebView2CreateCoreWebView2CompositionControllerCompletedHandler  
: public IUnknown
```

The caller implements this interface to receive the CoreWebView2Controller created via CreateCoreWebView2CompositionController.

## Summary

Expand table

Members	Descriptions
Invoke	Called to provide the implementer with the completion status and result of the corresponding asynchronous method call.

## Applies to

Expand table

Product	Introduced
WebView2 Win32	1.0.774.44
WebView2 Win32 Prerelease	1.0.790

## Members

### Invoke

Called to provide the implementer with the completion status and result of the corresponding asynchronous method call.

```
public HRESULT Invoke(HRESULT errorCode, ICoreWebView2CompositionController  
* webView)
```

---

## Feedback

Was this page helpful?



# interface ICoreWebView2CreateCoreWebView2ControllerCompletedHandler

Article • 02/26/2024

```
interface ICoreWebView2CreateCoreWebView2ControllerCompletedHandler
: public IUnknown
```

Receives the `CoreWebView2Controller` created using `CreateCoreWebView2Controller`.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the completion status and result of the corresponding asynchronous method.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.488
WebView2 Win32 Prerelease	0.9.488

## Members

### Invoke

Provides the completion status and result of the corresponding asynchronous method.

```
public HRESULT Invoke(HRESULT errorCode, ICoreWebView2Controller *  
createdController)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2CreateCoreWebView2EnvironmentCompletedHandler

Article • 02/26/2024

```
interface ICoreWebView2CreateCoreWebView2EnvironmentCompletedHandler
: public IUnknown
```

Receives the `WebView2Environment` created using `CreateCoreWebView2Environment`.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the completion status and result of the corresponding asynchronous method.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### Invoke

Provides the completion status and result of the corresponding asynchronous method.

```
public HRESULT Invoke(HRESULT errorCode, ICoreWebView2Environment *  
createdEnvironment)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2CursorChangedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2CursorChangedEventHandler
: public IUnknown
```

The caller implements this interface to receive CursorChanged events.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Called to provide the implementer with the event args for the corresponding event.

Use the Cursor property to get the new cursor.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.774.44
WebView2 Win32 Prerelease	1.0.790

## Members

### Invoke

Called to provide the implementer with the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2CompositionController * sender, IUnknown * args)
```

There are no event args and the args parameter will be null.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2CustomItemSelectedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2CustomItemSelectedEventHandler
: public IUnknown
```

Raised to notify the host that the end user selected a custom `ContextMenuItem`.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

`CustomItemSelected` event is raised on the specific `ContextMenuItem` that the end user selected.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1185.39
WebView2 Win32 Prerelease	1.0.1189

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2ContextMenuItem * sender, IUnknown * args)
```

No event args exist and the `args` parameter is set to `null`.

---

## Feedback

Was this page helpful?



# interface

## ICoreWebView2DevToolsProtocolEventReceivedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2DevToolsProtocolEventReceivedEventHandler
: public IUnknown
```

Receives `DevToolsProtocolEventReceived` events from the WebView.

## Summary

Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2 * sender,
ICoreWebView2DevToolsProtocolEventArgs * args)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2DocumentTitleChangedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2DocumentTitleChangedEventHandler
: public IUnknown
```

Receives `DocumentTitleChanged` events.

## Summary

Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

Use the `DocumentTitle` property to get the modified title.

## Applies to

Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2 * sender, IUnknown * args)
```

No event args exist and the `args` parameter is set to `null`.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2DOMContentLoadedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2DOMContentLoadedEventHandler
: public IUnknown
```

Receives `DOMContentLoaded` events.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.705.50
WebView2 Win32 Prerelease	1.0.721

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2 * sender,
ICoreWebView2DOMContentLoadedEventArgs * args)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2DownloadStartingEventHandler

Article • 02/26/2024

```
interface ICoreWebView2DownloadStartingEventHandler
: public IUnknown
```

Add an event handler for the `DownloadStarting` event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.902.49
WebView2 Win32 Prerelease	1.0.902

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2 * sender,
ICoreWebView2DownloadStartingEventArgs * args)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2EstimatedEndTimeChangedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2EstimatedEndTimeChangedEventHandler
: public IUnknown
```

Implements the interface to receive `EstimatedEndTimeChanged` event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

Use the `ICoreWebView2DownloadOperation.EstimatedEndTime` property to get the new estimated end time.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.902.49
WebView2 Win32 Prerelease	1.0.902

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2DownloadOperation * sender, IUnknown *  
args)
```

No event args exist and the `args` parameter is set to `null`.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2ExecuteScriptCompletedHandler

Article • 02/26/2024

```
interface ICoreWebView2ExecuteScriptCompletedHandler
: public IUnknown
```

Receives the result of the `ExecuteScript` method.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provide the implementer with the completion status and result of the corresponding asynchronous method.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### Invoke

Provide the implementer with the completion status and result of the corresponding asynchronous method.

```
public HRESULT Invoke(HRESULT errorCode, LPCWSTR resultObjectAsJson)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2ExecuteScriptWithResultCompletedHandler

Article • 02/26/2024

```
interface ICoreWebView2ExecuteScriptWithResultCompletedHandler
: public IUnknown
```

This is the callback for ExecuteScriptWithResult.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the result of ExecuteScriptWithResult.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.2277.86
WebView2 Win32 Prerelease	1.0.2357

## Members

### Invoke

Provides the result of ExecuteScriptWithResult.

```
public HRESULT Invoke(HRESULT errorCode, ICoreWebView2ExecuteScriptResult *  
result)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2FaviconChangedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2FaviconChangedEventHandler
: public IUnknown
```

This interface is a handler for when the `Favicon` is changed.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Called to notify the favicon changed. The event args are always null.

The sender is the ICoreWebView2 object the top-level document of which has changed favicon and the eventArgs is nullptr. Use the FaviconUri property and GetFavicon method to obtain the favicon data. The second argument is always null. For more information see [add\\_FaviconChanged](#).

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1293.44
WebView2 Win32 Prerelease	1.0.1305

## Members

### Invoke

Called to notify the favicon changed. The event args are always null.

```
public HRESULT Invoke(ICoreWebView2 * sender, IUnknown * args)
```

---

## Feedback

Was this page helpful?



# interface

## ICoreWebView2FocusChangedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2FocusChangedEventArgs
: public IUnknown
```

Receives `GotFocus` and `LostFocus` events.

## Summary

Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2Controller * sender, IUnknown * args)
```

No event args exist and the `args` parameter is set to `null`.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2FrameContentLoadingEventHandler

Article • 02/26/2024

```
interface ICoreWebView2FrameContentLoadingEventHandler
: public IUnknown
```

Receives `ContentLoading` events for iframe.

## Summary

Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

Expand table

Product	Introduced
WebView2 Win32	1.0.1108.44
WebView2 Win32 Prerelease	1.0.1133

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2Frame * sender,
ICoreWebView2ContentLoadingEventArgs * args)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2FrameCreatedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2FrameCreatedEventHandler
: public IUnknown
```

Receives `FrameCreated` event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the result for the iframe created event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.902.49
WebView2 Win32 Prerelease	1.0.902

## Members

### Invoke

Provides the result for the iframe created event.

```
public HRESULT Invoke(ICoreWebView2 * sender,
ICoreWebView2FrameCreatedEventArgs * args)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2FrameDestroyedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2FrameDestroyedEventHandler
: public IUnknown
```

Receives `FrameDestroyed` event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the result for the iframe destroyed event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.902.49
WebView2 Win32 Prerelease	1.0.902

## Members

### Invoke

Provides the result for the iframe destroyed event.

```
public HRESULT Invoke(ICoreWebView2Frame * sender, IUnknown * args)
```

No event args exist and the `args` parameter is set to `null`.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2FrameDOMContentLoadedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2FrameDOMContentLoadedEventHandler
: public IUnknown
```

Receives `DOMContentLoaded` events for iframe.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1108.44
WebView2 Win32 Prerelease	1.0.1133

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2Frame * sender,
ICoreWebView2DOMContentLoadedEventArgs * args)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2FrameNameChangedEvent Handler

Article • 02/26/2024

```
interface ICoreWebView2FrameNameChangedEventHandler
: public IUnknown
```

Receives `FrameNameChanged` event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the result for the iframe name changed event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.902.49
WebView2 Win32 Prerelease	1.0.902

## Members

### Invoke

Provides the result for the iframe name changed event.

```
public HRESULT Invoke(ICoreWebView2Frame * sender, IUnknown * args)
```

No event args exist and the `args` parameter is set to `null`.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2FrameNavigationCompletedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2FrameNavigationCompletedEventHandler
: public IUnknown
```

Receives `NavigationCompleted` events for iframe.

## Summary

Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

Expand table

Product	Introduced
WebView2 Win32	1.0.1108.44
WebView2 Win32 Prerelease	1.0.1133

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2Frame * sender,
ICoreWebView2NavigationCompletedEventArgs * args)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2FrameNavigationStartingEventHandler

Article • 02/26/2024

```
interface ICoreWebView2FrameNavigationStartingEventHandler
: public IUnknown
```

Receives `NavigationStarting` events for iframe.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1108.44
WebView2 Win32 Prerelease	1.0.1133

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2Frame * sender,
ICoreWebView2NavigationStartingEventArgs * args)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2FramePermissionRequestedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2FramePermissionRequestedEventHandler
: public IUnknown
```

Receives `PermissionRequested` events for iframes.

## Summary

Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

Expand table

Product	Introduced
WebView2 Win32	1.0.1185.39
WebView2 Win32 Prerelease	1.0.1158

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2Frame * sender,
ICoreWebView2PermissionRequestedEventArgs2 * args)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2FrameWebMessageReceivedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2FrameWebMessageReceivedEventHandler
: public IUnknown
```

Receives `WebMessageReceived` events for iframe.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1108.44
WebView2 Win32 Prerelease	1.0.1133

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2Frame * sender,
ICoreWebView2WebMessageReceivedEventArgs * args)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2GetCookiesCompletedHandler

Article • 02/26/2024

```
interface ICoreWebView2GetCookiesCompletedHandler
: public IUnknown
```

Receives the result of the `GetCookies` method.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the completion status of the corresponding asynchronous method call.

The result is written to the cookie list provided in the `GetCookies` method call.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.705.50
WebView2 Win32 Prerelease	1.0.721

## Members

### Invoke

Provides the completion status of the corresponding asynchronous method call.

```
public HRESULT Invoke(HRESULT result, ICoreWebView2CookieList * cookieList)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2GetFaviconCompletedHandler

Article • 02/26/2024

```
interface ICoreWebView2GetFaviconCompletedHandler
: public IUnknown
```

This interface is a handler for the completion of the population of `imageStream`.

## Summary

Expand table

Members	Descriptions
<a href="#">Invoke</a>	Called to notify the favicon has been retrieved.

`errorCode` returns `S_OK` if the API succeeded. The image is returned in the `faviconStream` object. If there is no image then no data would be copied into the `imageStream`. For more details, see the [GetFavicon](#) API.

## Applies to

Expand table

Product	Introduced
WebView2 Win32	1.0.1293.44
WebView2 Win32 Prerelease	1.0.1305

## Members

### Invoke

Called to notify the favicon has been retrieved.

```
public HRESULT Invoke(HRESULT errorCode, IStream * faviconStream)
```

---

## Feedback

Was this page helpful?



# interface

## ICoreWebView2GetNonDefaultPermissionSettingsCompletedHandler

Article • 02/26/2024

```
interface ICoreWebView2GetNonDefaultPermissionSettingsCompletedHandler
: public IUnknown
```

The caller implements this interface to handle the result of [GetNonDefaultPermissionSettings](#).

## Summary

[+] [Expand table](#)

Members	Descriptions
<a href="#">Invoke</a>	Provides the permission setting collection for the requested permission kind.

## Applies to

[+] [Expand table](#)

Product	Introduced
WebView2 Win32	1.0.1661.34
WebView2 Win32 Prerelease	1.0.1671

## Members

### Invoke

Provides the permission setting collection for the requested permission kind.

```
public HRESULT Invoke(HRESULT errorCode,  
ICoreWebView2PermissionSettingCollectionView * collectionView)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2GetProcessExtendedInfosCompletedHandler

Article • 02/26/2024

```
interface ICoreWebView2GetProcessExtendedInfosCompletedHandler
: public IUnknown
```

Receives the result of the `GetProcessExtendedInfos` method.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the process extended info list for the <code>GetProcessExtendedInfos</code> .

The result is written to the collection of `ProcessExtendedInfos` provided in the `GetProcessExtendedInfos` method call.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.2210.55
WebView2 Win32 Prerelease	1.0.2357

## Members

### Invoke

Provides the process extended info list for the `GetProcessExtendedInfos`.

```
public HRESULT Invoke(HRESULT errorCode,  
ICoreWebView2ProcessExtendedInfoCollection * value)
```

---

## Feedback

Was this page helpful?



# interface

## ICoreWebView2HistoryChangedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2HistoryChangedEventHandler
: public IUnknown
```

Receives `HistoryChanged` events.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2 * sender, IUnknown * args)
```

No event args exist and the `args` parameter is set to `null`.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2IsDefaultDownloadDialogOpenChangedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2IsDefaultDownloadDialogOpenChangedEventHandler
: public IUnknown
```

Implements the interface to receive `IsDefaultDownloadDialogOpenChanged` events.

## Summary

Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

Expand table

Product	Introduced
WebView2 Win32	1.0.1072.54
WebView2 Win32 Prerelease	1.0.1083

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2 * sender, IUnknown * args)
```

No event args exist and the `args` parameter is set to `null`.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2IsDocumentPlayingAudioChangedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2IsDocumentPlayingAudioChangedEventHandler
: public IUnknown
```

Implements the interface to receive `IsDocumentPlayingAudioChanged` events.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

Use the `IsDocumentPlayingAudio` property to get the audio playing state.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1072.54
WebView2 Win32 Prerelease	1.0.1083

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2 * sender, IUnknown * args)
```

No event args exist and the `args` parameter is set to `null`.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2IsMutedChangedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2IsMutedChangedEventHandler
: public IUnknown
```

Implements the interface to receive `IsMutedChanged` events.

## Summary

Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

Use the `IsMuted` property to get the mute state.

## Applies to

Expand table

Product	Introduced
WebView2 Win32	1.0.1072.54
WebView2 Win32 Prerelease	1.0.1083

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2 * sender, IUnknown * args)
```

No event args exist and the `args` parameter is set to `null`.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2LaunchingExternalUriSchemeEventHandler

Article • 02/26/2024

```
interface ICoreWebView2LaunchingExternalUriSchemeEventHandler
: public IUnknown
```

Event handler for the `LaunchingExternalUriScheme` event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Receives the event args for the corresponding event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1823.32
WebView2 Win32 Prerelease	1.0.1905

## Members

### Invoke

Receives the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2 * sender,
ICoreWebView2LaunchingExternalUriSchemeEventArgs * args)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2MoveFocusRequestedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2MoveFocusRequestedEventHandler
: public IUnknown
```

Receives `MoveFocusRequested` events.

## Summary

Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2Controller * sender,
ICoreWebView2MoveFocusRequestedEventArgs * args)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2NavigationCompletedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2NavigationCompletedEventHandler
: public IUnknown
```

Receives `NavigationCompleted` events.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2 * sender,
ICoreWebView2NavigationCompletedEventArgs * args)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2NavigationStartingEventHandler

Article • 02/26/2024

```
interface ICoreWebView2NavigationStartingEventHandler
: public IUnknown
```

Receives `NavigationStarting` events.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2 * sender,
ICoreWebView2NavigationStartingEventArgs * args)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2NewBrowserVersionAvailableEventHandler

Article • 02/26/2024

```
interface ICoreWebView2NewBrowserVersionAvailableEventHandler
: public IUnknown
```

Receives `NewBrowserVersionAvailable` events.

## Summary

Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2Environment * sender, IUnknown * args)
```

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2NewWindowRequestedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2NewWindowRequestedEventHandler
: public IUnknown
```

Receives `NewWindowRequested` events.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2 * sender,
ICoreWebView2NewWindowRequestedEventArgs * args)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2PermissionRequestedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2PermissionRequestedEventHandler
: public IUnknown
```

Receives `PermissionRequested` events.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2 * sender,
ICoreWebView2PermissionRequestedEventArgs * args)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2PrintCompletedHandler

Article • 02/26/2024

```
interface ICoreWebView2PrintCompletedHandler
: public IUnknown
```

Receives the result of the `Print` method.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the result of the corresponding asynchronous method.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1518.46
WebView2 Win32 Prerelease	1.0.1549

## Members

### Invoke

Provides the result of the corresponding asynchronous method.

```
public HRESULT Invoke(HRESULT errorCode, COREWEBVIEW2_PRINT_STATUS
printStatus)
```

# Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2PrintToPdfCompletedHandler

Article • 02/26/2024

```
interface ICoreWebView2PrintToPdfCompletedHandler
: public IUnknown
```

Receives the result of the `PrintToPdf` method.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the result of the corresponding asynchronous method.

If the print to PDF operation succeeds, `isSuccessful` is true. Otherwise, if the operation failed, `isSuccessful` is set to false. An invalid path returns `E_INVALIDARG`.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1020.30
WebView2 Win32 Prerelease	1.0.1056

## Members

### Invoke

Provides the result of the corresponding asynchronous method.

```
public HRESULT Invoke(HRESULT errorCode, BOOL.isSuccessful)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2PrintToPdfStreamCompletedHandler

Article • 02/26/2024

```
interface ICoreWebView2PrintToPdfStreamCompletedHandler
: public IUnknown
```

Receives the result of the `PrintToPdfStream` method.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the result of the corresponding asynchronous method.

`errorCode` returns `S_OK` if the `PrintToPdfStream` operation succeeded. The printable pdf data is returned in the `pdfStream` object.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1518.46
WebView2 Win32 Prerelease	1.0.1549

## Members

### Invoke

Provides the result of the corresponding asynchronous method.

```
public HRESULT Invoke(HRESULT errorCode, IStream * pdfStream)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2ProcessFailedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2ProcessFailedEventHandler
: public IUnknown
```

Receives `ProcessFailed` events.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2 * sender,
ICoreWebView2ProcessFailedEventArgs * args)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2ProcessInfosChangedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2ProcessInfosChangedEventHandler
: public IUnknown
```

An event handler for the `ProcessInfosChanged` event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1108.44
WebView2 Win32 Prerelease	1.0.1133

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2Environment * sender, IUnknown * args)
```

No event args exist and the `args` parameter is set to `null`.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2ProfileAddBrowserExtensionCompletedHandler

Article • 02/26/2024

```
interface ICoreWebView2ProfileAddBrowserExtensionCompletedHandler
: public IUnknown
```

The caller implements this interface to receive the result of loading an browser Extension.

## Summary

Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the result of the <code>AddBrowserExtension</code> operation.g.

## Applies to

Expand table

Product	Introduced
WebView2 Win32	1.0.2210.55
WebView2 Win32 Prerelease	1.0.2194

## Members

### Invoke

Provides the result of the `AddBrowserExtension` operation.g.

```
public HRESULT Invoke(HRESULT errorCode, ICoreWebView2BrowserExtension *  
extension)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface ICoreWebView2ProfileDeletedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2ProfileDeletedEventHandler
: public IUnknown
```

Receives the `CoreWebView2Profile.Deleted` event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Called to provide the implementer with the event args for the profile deleted event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.2210.55
WebView2 Win32 Prerelease	1.0.2194

## Members

### Invoke

Called to provide the implementer with the event args for the profile deleted event.

```
public HRESULT Invoke(ICoreWebView2Profile * sender, IUnknown * args)
```

No event args exist and the `args` parameter is set to `null`.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2ProfileGetBrowserExtensionsCompletedHandler

Article • 02/26/2024

```
interface ICoreWebView2ProfileGetBrowserExtensionsCompletedHandler
: public IUnknown
```

The caller implements this interface to receive the result of getting the browser Extensions.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the browser extension list for the requested user profile.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.2210.55
WebView2 Win32 Prerelease	1.0.2194

## Members

### Invoke

Provides the browser extension list for the requested user profile.

```
public HRESULT Invoke(HRESULT errorCode, ICoreWebView2BrowserExtensionList *  
extensionList)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2RasterizationScaleChangedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2RasterizationScaleChangedEventHandler
: public IUnknown
```

Receives `RasterizationScaleChanged` events.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Called to provide the implementer with the event args for the corresponding event.

Use the `RasterizationScale` property to get the modified scale.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.774.44
WebView2 Win32 Prerelease	1.0.824

## Members

### Invoke

Called to provide the implementer with the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2Controller * sender, IUnknown * args)
```

There are no event args and the args parameter will be null.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2ScriptDialogOpeningEventHandler

Article • 02/26/2024

```
interface ICoreWebView2ScriptDialogOpeningEventHandler
: public IUnknown
```

Receives `ScriptDialogOpening` events.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2 * sender,
ICoreWebView2ScriptDialogOpeningEventArgs * args)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2ServerCertificateErrorDetectedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2ServerCertificateErrorDetectedEventHandler
: public IUnknown
```

An event handler for the `ServerCertificateErrorDetected` event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1245.22
WebView2 Win32 Prerelease	1.0.1248

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2 * sender,
ICoreWebView2ServerCertificateErrorDetectedEventArgs * args)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2SetPermissionStateCompletedHandler

Article • 02/26/2024

```
interface ICoreWebView2SetPermissionStateCompletedHandler
: public IUnknown
```

The caller implements this interface to handle the result of `SetPermissionState`.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provide the completion status of the corresponding asynchronous method.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1661.34
WebView2 Win32 Prerelease	1.0.1671

## Members

### Invoke

Provide the completion status of the corresponding asynchronous method.

```
public HRESULT Invoke(HRESULT errorCode)
```

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2SourceChangedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2SourceChangedEventArgs
: public IUnknown
```

Receives `SourceChanged` events.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2 * sender,
ICoreWebView2SourceChangedEventArgs * args)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2StateChangedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2StateChangedEventHandler
: public IUnknown
```

Implements the interface to receive `StateChanged` event.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

Use the `ICoreWebView2DownloadOperation.State` property to get the current state, which can be in progress, interrupted, or completed. Use the `ICoreWebView2DownloadOperation.InterruptReason` property to get the interrupt reason if the download is interrupted.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.902.49
WebView2 Win32 Prerelease	1.0.902

## Members

## Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2DownloadOperation * sender, IUnknown *  
args)
```

No event args exist and the `args` parameter is set to `null`.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2StatusBarTextChangedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2StatusBarTextChangedEventHandler
: public IUnknown
```

Receives `StatusTextChanged` events.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Called to provide the implementer with the event args for the corresponding event.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.1185.39
WebView2 Win32 Prerelease	1.0.1189

## Members

### Invoke

Called to provide the implementer with the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2 * sender, IUnknown * args)
```

No event args exist and the `args` parameter is set to `null`.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2TrySuspendCompletedHandler

Article • 02/26/2024

```
interface ICoreWebView2TrySuspendCompletedHandler
: public IUnknown
```

The caller implements this interface to receive the TrySuspend result.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the result of the TrySuspend operation.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.774.44
WebView2 Win32 Prerelease	1.0.790

## Members

### Invoke

Provides the result of the TrySuspend operation.

```
public HRESULT Invoke(HRESULT errorCode, BOOL.isSuccessful)
```

See [Sleeping Tabs FAQ](#) for conditions that might prevent WebView from being suspended. In those situations, `isSuccessful` will be false and `errorCode` is `S_OK`.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2WebMessageReceivedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2WebMessageReceivedEventHandler
: public IUnknown
```

Receives `WebMessageReceived` events.

## Summary

Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2 * sender,
ICoreWebView2WebMessageReceivedEventArgs * args)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2WebResourceRequestedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2WebResourceRequestedEventHandler
: public IUnknown
```

Runs when a URL request (through network, file, and so on) is made in the webview for a Web resource matching resource context filter and URL specified in [AddWebResourceRequestedFilter](#).

## Summary

[ ] [Expand table](#)

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

The host views and modifies the request or provide a response in a similar pattern to HTTP, in which case the request immediately completed. This may not contain any request headers that are added by the network stack, such as an [Authorization](#) header.

## Applies to

[ ] [Expand table](#)

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

## Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2 * sender,  
ICoreWebView2WebResourceRequestedEventArgs * args)
```

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2WebResourceResponseReceivedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2WebResourceResponseReceivedEventHandler
: public IUnknown
```

Receives `WebResourceResponseReceived` events.

## Summary

Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

Expand table

Product	Introduced
WebView2 Win32	1.0.705.50
WebView2 Win32 Prerelease	1.0.721

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2 * sender,
ICoreWebView2WebResourceResponseReceivedEventArgs * args)
```

---

# Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2WebResourceResponseViewGetContentCompletedHandler

Article • 02/26/2024

```
interface ICoreWebView2WebResourceResponseViewGetContentCompletedHandler
: public IUnknown
```

Receives the result of the [ICoreWebView2WebResourceResponseView::GetContent](#) method.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the completion status and result of the corresponding asynchronous method call.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	1.0.705.50
WebView2 Win32 Prerelease	1.0.721

## Members

### Invoke

Provides the completion status and result of the corresponding asynchronous method call.

```
public HRESULT Invoke(HRESULT errorCode, IStream * content)
```

A failure `errorCode` will be passed if the content failed to load. `E_ABORT` means the response loading was blocked (e.g., by CORS policy); `ERROR_CANCELLED` means the response loading was cancelled. `ERROR_NO_DATA` means the response has no content data, `content` is `null` in this case. Note content (if any) is ignored for redirects, 204 No Content, 205 Reset Content, and HEAD-request responses.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2WindowCloseRequestedEventHandler

Article • 02/26/2024

```
interface ICoreWebView2WindowCloseRequestedEventHandler
: public IUnknown
```

Receives `WindowCloseRequested` events.

## Summary

Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

## Applies to

Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2 * sender, IUnknown * args)
```

No event args exist and the `args` parameter is set to `null`.

---

## Feedback

Was this page helpful?

 Yes

 No

# interface

## ICoreWebView2ZoomFactorChangedEvent Handler

Article • 02/26/2024

```
interface ICoreWebView2ZoomFactorChangedEventHandler
: public IUnknown
```

Implements the interface to receive `ZoomFactorChanged` events.

## Summary

[+] Expand table

Members	Descriptions
<a href="#">Invoke</a>	Provides the event args for the corresponding event.

Use the `ICoreWebView2Controller.ZoomFactor` property to get the modified zoom factor.

## Applies to

[+] Expand table

Product	Introduced
WebView2 Win32	0.9.430
WebView2 Win32 Prerelease	0.9.488

## Members

### Invoke

Provides the event args for the corresponding event.

```
public HRESULT Invoke(ICoreWebView2Controller * sender, IUnknown * args)
```

No event args exist and the `args` parameter is set to `null`.

---

## Feedback

Was this page helpful?

 Yes

 No