WEBVIEW object

The WEBVEW object is a control that wraps the Microsoft WebView2 Edge Browser control and allows you to embed web technologies (HTML, CSS and JavaScript) in your OpenInsight forms.

Developer Notes

1. Equated constants for the WEBVIEW object can be found in the PS_WEBVIEW_EQUATES insert record.
2. The WEBVIEW object relies on the installation of the "WebView2 Runtime platform" in order to run (see Deployment Notes below).
3. It is recommended that developers review the Microsoft documentation for the WebView2 Edge Browser along with the details of the OpenInsight WEBVIEW object here. The Microsoft documentation can be found at:

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

Deployment Notes

The WEBVIEW object relies on the "WebView2 Runtime platform" which is essentially the same rendering engine that powers the Chromium-based Microsoft Edge Browser – it contains modified Microsoft Edge binaries that are fine-tuned and tested for WebView2-based applications.

When you distribute an app that includes WEBVIEW objects (like the OpenInsight IDE itself), you need to consider how the WebView2 Runtime is distributed and updated on client machines, as there are two options available:

- The automatically updated "Evergreen" Runtime
- A "Fixed Version" Runtime

The Evergreen Runtime

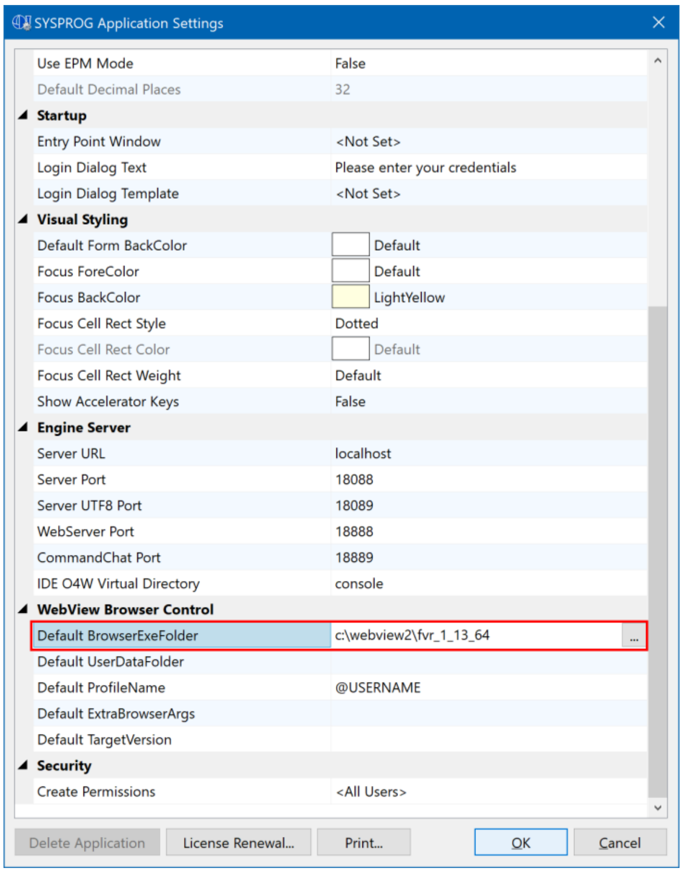
Current versions of Windows have an Evergreen version of the WebView2 platform installed as part of the normal OS, so it is always available, and by default the WEBVIEW object will use this when creating a new instance. The Evergreen runtime is always kept up to date by Windows as part of the normal Windows Update cycle (earlier versions of Windows 10 needed to have the Evergreen version installed separately, but it's been a core part of Windows 10 since 2022 now). This is by far the easiest way to use the WebView2 platform.

The Fixed Version Runtime

It is also possible to create a WEBVIEW object from a "Fixed Version", which means that you may choose and supply a specific version of the WebView2 Runtime for use with your application. This may be desirable when you need full control over the versioning of the platform, where you know you have fully tested against it, and where it meets your requirements. A Fixed Version Runtime will not be updated automatically by Windows and you are responsible for installing and maintaining it on client machines.

To use a Fixed Version Runtime with the WEBVIEW object:

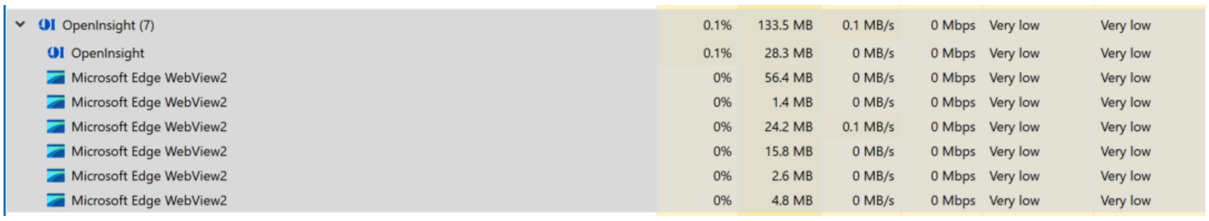
- Ensure that it is installed correctly.
- Ensure the WEBVIEW object uses Fixed Version Runtime via the BROWSEREXEFOLDER property:
  - To specify a Fixed Version Runtime for a *specific* WEBVIEW object set it's BROWSEREXEFOLDER property at design time.
  - To specify a Fixed Version Runtime for *all* WEBVIEW objects in your application set the "Default BrowserExeFolder" parameter in the Application Settings dialog box:



Use EPM Mode	False
Default Decimal Places	32
Startup	
Entry Point Window	<Not Set>
Login Dialog Text	Please enter your credentials
Login Dialog Template	<Not Set>
Visual Styling	
Default Form BackColor	Default
Focus ForeColor	Default
Focus BackColor	LightYellow
Focus Cell Rect Style	Dotted
Focus Cell Rect Color	Default
Focus Cell Rect Weight	Default
Show Accelerator Keys	False
Engine Server	
Server URL	localhost
Server Port	18088
Server UTF8 Port	18089
WebServer Port	18888
CommandChat Port	18889
IDE O4W Virtual Directory	console
WebView Browser Control	
Default BrowserExeFolder	c:\webview2\fvr_1_13_64
Default UserDataFolder	
Default ProfileName	@USERNAME
Default ExtraBrowserArgs	
Default TargetVersion	
Security	
Create Permissions	<All Users>

WebView2 Runtime instances

It should be noted that when WEBVIEW objects are created by the Presentation Server, the WebView2 Runtime platform launches several instances of the "Microsoft Edge WebView2" process in order to provide web-browsing functionality. This can be seen in the example Task Manager screen shot below:



OpenInsight (7)	0.1%	133.5 MB	0.1 MB/s	0 Mbps	Very low	Very low
OpenInsight	0.1%	28.3 MB	0 MB/s	0 Mbps	Very low	Very low
Microsoft Edge WebView2	0%	56.4 MB	0 MB/s	0 Mbps	Very low	Very low
Microsoft Edge WebView2	0%	1.4 MB	0 MB/s	0 Mbps	Very low	Very low
Microsoft Edge WebView2	0%	24.2 MB	0.1 MB/s	0 Mbps	Very low	Very low
Microsoft Edge WebView2	0%	15.8 MB	0 MB/s	0 Mbps	Very low	Very low
Microsoft Edge WebView2	0%	2.6 MB	0 MB/s	0 Mbps	Very low	Very low
Microsoft Edge WebView2	0%	4.8 MB	0 MB/s	0 Mbps	Very low	Very low

Like any other Chromium-based web browser this is part of the WEBVIEW object's normal operation and does not represent an issue. These instances are managed and closed as required.

More information on deploying the WebView2 Runtime can be found on the Microsoft website here:

https://learn.microsoft.com/en-us/microsoft-edge/webview2/concepts/distribution

Asynchronous Programming

When working with the WEBVIEW object is it very important to bear in mind that it is built around an asynchronous programming model, i.e., when commands are executed the results of those commands are normally returned via an event, not by a value returned directly to the caller. The WEBVIEW object generally resists attempts to force it into a synchronous paradigm and this should be avoided where possible.

Some common examples of the asynchronous model are:

- Navigating to a page (via the NAVIGATE method): This results in several different events being fired as the WEBVIEW object loads the requested content.

- Executing JavaScript (via the EXECUTESCRIPT method): The results are returned via the WEBSCRIPTRESULT event (note however, that we *do* provide a synchronous version of this method, but the normal asynchronous version is preferred).

- Creating the object: WEBVIEW objects are created asynchronously and may not be ready for navigation as soon as the parent form is ready. Rather than wait in a loop checking the READYSTATE property, your application should listen for the WEBVIEWCREATED event to determine when it is safe to begin using the control.

Communicating with the content in a WEBVIEW object

There are several different ways to communicate with the content hosted inside a WEBVIEW object, and for the hosted content to communicate with your application too.

Script Execution - Calling web-side code from Basic+

This is the most common and easiest method of interacting with WEBVIEW content and involves executing JavaScript statements and functions via the EXECUTESCRIPT method. You may also use the ADDINITSCRIPT method to add JavaScript that is executed as pages are loaded into the WEBVIEW object as well.

WebMessaging - Calling web-side code from Basic+

The POSTJSONMESSAGE and POSTTEXTMESSAGE methods may be used to post data to the hosted content in a WEBVIEW object. The content itself needs to add a JavaScript "message" event listener to the "window.chrome.webview" object in order to receive the message, so this method is for use with content that is designed to be hosted within a WEBVIEW control (rather than say, hosted in a normal browser).

WebMessaging - Calling a Basic+ event from web-side code

Content hosted inside a WEBVIEW object may post data to its host by using the JavaScript "window.chrome.webview.postMessage" method. This in turn triggers the WEBVIEW object's WEBMESSAGE event, along with the posted data.

(Note: The EnableWebMessages parameter in the WEBVIEW SETTINGS property must be set to TRUE$ for WebMessaging to work.)

More information on WebView2 interop can be found on the Microsoft website here:

> https://learn.microsoft.com/en-us/microsoft-edge/webview2/how-to/communicate-btwn-web-native

User Interface Integration

There are some UI elements managed by the WEBVIEW object that may be integrated more closely with the hosting OpenInsight application to give a more consistent look and feel. These are:

- Context menus
- Dialog boxes
- Opening new windows
- Permission requests
- Authentication requests

Context menus

The WEBVIEW object supports the standard OpenInsight CONTEXTMENU property, so it may be assigned a normal context menu and it will be displayed using the same visual styling as the rest of your application.

If the CONTEXTMENU property is not assigned then the WEBVIEW object displays a default context menu which normally exhibits the same look and feel as the menus in the standard Edge Browser. However, it is possible to override this behavior by setting the CONTEXTMENU property and merging the items from the default WEBVIEW context menu with the items from the OpenInsight context menu, thereby using the same visual styling as your application. This process takes place in the WEBINITCONTEXTMENU and WEBCONTEXTMENU events and is detailed more fully below.

Dialog boxes

The JavaScript language supports several standard dialog boxes:

- Alert
- Confirm
- BeforeUnload
- Prompt

As with the Context Menus these are normally displayed using the same look and feel as the JavaScript dialog boxes in the standard Edge Browser, but this behavior can be changed using the "EnableScriptDialogs" option in the WEBVIEW object's SETTINGS property. If set to False then the WEBVIEW uses normal OpenInsight dialogs instead, which will match the visual styling of your application. See the WEBSHOWDIALOG event for more details.

Opening new windows

Links in an HTML page can open content in a new window", as can the JavaScript "window.open" method. In this case there are three options available to the OpenInsight application:

- Launch a secondary PS form containing another WEBVIEW object and direct the current WEBVIEW object to load the new content into that.
- Allow the new content to be loaded into a new top-level WebView2 "WindowProxy" form – this is a minimal form (supplied by the WebView2 Runtime itself) that is designed to host a WEBVIEW object but offers less control than the first option. It is not an OpenInsight form so cannot be accessed via Basic+.
- Deny the new window request.

More information on this can be found in the documentation for the WEBOPENWINDOW event below.

Permission requests

Some content loaded into a WEBVIEW object may require user permission before an action can be taken. Examples include:

- Accessing the microphone
- Accessing the camera
- Accessing location data
- Reading the clipboard

In cases such as this a WEBPERMISSIONREQUEST event is raised, displaying an OpenInsight dialog to allow the user to decide if permission is granted. See the WEBPERMISSIONREQUEST event documentation for more details.

Authentication requests

When a WEBVIEW object receives an Authentication request from the server it normally shows a dialog box to ask the user for their credentials. It is possible to override this behavior by changing the AUTHENTICATIONMODE property and handling the request in the WEBAUTHREQUEST event that is raised instead. See the WEBAUTHREQUEST event documentation for more details.

WEBVIEW Properties

The WEBVIEW object supports the following properties

Name	Description
ALLOWSINGLESIGNON	Specifies if the object can use Single-Sign-On with Azure AD and MS Account resources.
AUDIOPLAYING	Specifies if the currently loaded document is playing audio.
AUTHENTICATIONMODE	Specifies if the WEBAUTHREQUEST event is used to handle authentication requests.
BACKCOLOR	Specifies the default background for documents.
BROWSEREXEFOLDER	Specifies the folder containing the executable files for the browser process.
BROWSERVISIBLE	Specifies if the WebView browser component itself is visible.
CANGOBACK	Specifies if the object can navigate to a previous page in its history.
CANGOFORWARD	Specifies if the object can navigate to a next page in its history.
COLORSCHEME	Specifies the preferred color scheme for browser UI elements.
DOCUMENTTITLE	Returns the title for the currently loaded top-level document.
ERRORTEXT	Returns error information arising from the most recently executed property or method operation.
EXCLUSIVEUDFACCESS	Specifies if other processes can create a browser session from the same user data environment.
EXTRABROWSERARGS	Specifies extra arguments passed to the browser process at startup.
HISTORY	Returns a dynamic array of visited sites when history-tracking is enabled.
INITIALIZED	Specifies if the object is loaded and ready to accept navigation requests.
INPRIVATEMODE	Specifies if the object is operating in “InPrivate” mode.
LANGUAGE	Specifies the default language used by the object.
MUTED	Specifies if the audio output of the object is muted.
PDFTOOLBARSETTINGS	Hides or shows variable items on the PDF toolbar.
PROCESSID	Returns the ID of the object’s browser process.
PROFILENAME	Specifies the profile name used for the object.
READYSTATE	Specifies the status of the current navigation request.
SETTINGS	Specifies the configuration options for the object.
SUSPENDED	Returns TRUE$ if the object’s browser process is suspended.
SYNCSTATUSLINE	Specifies if the parent form’s STATUSLINE property is automatically updated by the object.
SYNCTITLE	Specifies if the parent form’s TEXT property is automatically updated by the object.

TARGETVERSION	Specifies the minimum version of the browser process required by the object.
TRACKHISTORY	Specifies if the WebView tracks the sites that the user has visited.
URI	Specifies the URI for the object to navigate to.
USERAGENT	Specifies a custom User-Agent string for the object.
USERDATAFOLDER	Specifies the folder used to store the user's browsing data.
VERSION	Returns the version string of the browser component.
ZOOMFACTOR	Specifies the zoom factor for the object.

The following Common GUI Object properties are not supported:

- COMPOSITED
- CURSOR
- ECHO
- FONT
- FORECOLOR
- TEXT
- TOOLTIP

ALLOWSINGLESIGNON property

Description

Specifies if the object can use single sign on with Azure Active Directory (AAD) and personal Microsoft Account (MSA) resources.

Property Value

The ALLOWSINGLESIGNON property is a Boolean value of TRUE$ or FALSE$. This property defaults to FALSE$.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
Get/Set	Get	No	No	No

Remarks

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.

For more information on this property please refer to the Windows WebView2 documentation regarding the ICoreWebView2EnvironmentOptions get_AllowSingleSignOnUsingOSPrimaryAccount method on the Microsoft website.

Example

```
// Example: Check to see if Single-Sign-On is enabled.

SSOAllowed = Get_Property( CtrlEntID, "ALLOWSINGLESIGNON" )
```

See Also

N/a.

AUDIOPLAYING property

Description
Specifies if the currently loaded document is playing audio.

Property Value
The AUDIOPLAYING property is a Boolean value of TRUE$ or FALSE$.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
N/a	Get	No	No	No

Remarks
This property will return TRUE$ if audio is playing, even if the MUTED property is TRUE$.

For more information on this property please refer to the Windows WebView2 documentation regarding the ICoreWebView2_8 get_IsDocumentPlayingAudio method on the Microsoft website.

Example

```
// Example: Check to see audio is playing in the WEBVIEW control.

IsAudioPlaying = Get_Property( CtrlEntID, "AUDIOPLAYING" )
```

See Also
WEBVIEW MUTED property, WEBVIEW WEBAUDIOCHANGED event, WEBVIEW WEBMUTEDCHANGED event.

AUTHENTICATIONMODE property

Description

Specifies if the WEBAUTHREQUEST event handler is used to handle authentication requests.

Property Value

The AUTHENTICATIONMODE property is a numeric value defined as follows:

Value	Description
0	Default. Use the WEBVIEW object's default authentication handler. This is the default [sic] mode.
1	Custom. Use the WEBAUTHREQUEST handler.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
Get/Set	Get/Set	No	No	No

Remarks

The default WEBVIEW behavior for authentication requests displays a dialog box with the server's Challenge string and prompts to enter a username and password. This can be changed by setting the AUTHENTICATION MODE to 1 (Custom) and handling it in the WEBAUTHREQUEST event.

Equates for use with this property can be found in the PS_WEBVIEW_EQUATES insert record.

For more information on this property please refer to the Windows WebView2 documentation regarding the ICoreWebView2_10 add_BasicAuthenticationRequested method on the Microsoft website.

Example

```
// Example: Set the WEBVIEW object to use the WEBAUTHREQUEST event handler
$Insert PS_WebView_Equates

Call Set_Property_Only( CtrlEntID, "AUTHENTICATIONMODE", |
                        WBV_AUTHMODE_CUSTOM$ )
```

See Also

WEBVIEW AUTHENTICATE method, WEBVIEW WEBAUTHREQUEST event.

BACKCOLOR property

Description

Specifies the default background color for documents loaded in the WEBVIEW object. This color is used 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.

Property Value

The BACKCOLOR property is a single numeric RGB value. Note that unlike the normal BACKCOLOR property gradient values are not supported. The default color is white.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
Get/Set	Get/Set	No	No	No

Remarks

For more information on this property please refer to the Windows WebView2 documentation regarding the ICoreWebView2Controller2 get_DefaultBackgroundColor method on the Microsoft website.

Example

```
// Example: Set the default background color to Red
$Insert Colors

Call Set_Property_Only( CtrlEntID, "BACKCOLOR", RED$ )
```

See Also

Common GUI BACKCOLOR property.

BROWSEREXEFOLDER property

Description

Specifies the location of the WebView2 Runtime executable files. If this is null then the default BrowserExeFolder value specified in the SYSTEM WEBVIEWCONFIG property is used instead. If that default value is also null then the WEBVIEW object assumes that the Microsoft WebView2 "Evergreen" runtime version is installed.

Property Value

This property value may be null or may contain a relative or absolute folder path to the location of the WebView2 runtime executable files.

When set in the Form Designer it may contain OS environment variable strings in the form: %variableName% (*not* case-sensitive). These will be expanded with their actual values when the WEBVIEW control is created at runtime.

This property may also contain the following OpenInsight environment variables (case-sensitive) that are expanded with their Basic+ values when the control is created at runtime:

```
@APPID    - Expands to @AppID<1>
@USERNAME – Expands to @UserName
```

Using Get_Property at runtime returns the fully expanded version of the string.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
Get/Set	Get	No	No	No

Remarks

See the Deployment Notes section earlier in this document for further details on how the WebView2 control handles the BROWSEREXEFOLDER property.

A default value for the BROWSEREXEFOLDER can be set via the SYSTEM WEBVIEWCONFIG property. Note that the property value returned at runtime is the *effective* value – i.e. if the object property is null, but the WEBVIEWCONFIG property specifies a default value *then that default value will be returned*.

For more information on this topic please refer to the Windows WebView2 documentation regarding the CreateCoreWebView2EnvironmentWithOptions function on the Microsoft website.

Example

```
// Example: Assuming the following:
//
// 1) Windows has an environment variable LOCALAPPDATA set like so:
//
//   LOCALAPPDATA=C:\Users\AgentC\AppData\Local
//
// 2) OpenInsight is logged into the EXAMPLES app with a username
//    of A_TEST
//
// 3) The BrowserExeFolder property is set in the Form Designer to
//    a value of:
//
//       %LocalAppData%\@APPID_WebView2

FolderVal = Get_Property( CtrlEntID, "BROWSEREXEFOLDER" )

// FolderVal now contains the value:
//
//   C:\Users\AgentC\AppData\Local\EXAMPLES_WebView2
```

See Also

WEBVIEW Deployment Notes, SYSTEM WEBVIEWCONFIG property.

BROWSERVISIBLE property

Description

Specifies if the actual browser component of the WEBVIEW object is visible. The WEBVIEW object is comprised of two parts: The embedded Edge WebView2 browser component, and a very basic wrapper control that the browser uses as a surface to display to.

The BROWSERVISIBLE property is synchronized to the normal VISIBLE property, i.e. setting VISIBLE will also update BROWSERVISIBLE (but not vice-versa). The BROWSERVISIBILE property can be used to control the visibility of the embedded browser component separately if desired, although this is not normally necessary.

There are CPU and memory benefits when the WEBVIEW object (and therefore the browser) is hidden. For instance, Chromium has code that throttles activities on the page like animations and some tasks are run less frequently. Similarly, the browser component will purge some caches to reduce memory usage.

The SUSPEND method can only be used when the WEBVIEW object is hidden.

Property Value

The BROWERVISIBLE property is a Boolean value of TRUE$ or FALSE$.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
N/a	Get/Set	No	No	No

Remarks

For more information on this property please refer to the Windows WebView2 documentation regarding the ICoreWebView2Controller_8 get_IsVisible method on the Microsoft website.

Example

```
// Example: Check if the browser component of the WEBVIEW object is visible…
$Insert Logical

IsBrowserVisible = Get_Property( CtrlEntID, "BROWSERVISIBLE" )
```

See Also

Common GUI VISIBLE property, WEBVIEW SUSPEND method, WEBVIEW SUSPENDED event.

CANGOBACK property

Description
Specifies if the WEBVIEW control can navigate back to a previous page in its navigation history.

Property Value
The CANGOBACK property is a Boolean value of TRUE$ or FALSE$.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
N/a	Get	No	No	No

Remarks
For more information on this property please refer to the Windows WebView2 documentation regarding the ICoreWebView2 get_CanGoBack method on the Microsoft website.

Example

```
// Example: Check to see if the Back button should be enabled.

EnableBackButton = Get_Property( CtrlEntID, "CANGOBACK" )
```

See Also
WEBVIEW CANGOFORWARD property, WEBVIEW HISTORY property, WEBVIEW BACK method, WEBVIEW CLEARBROWSINGDATA method, WEBVIEW FORWARD method, WEBVIEW WEBDATACLEARED event, WEBVIEW WEBHISTORYCHANGED event.

CANGOFORWARD property

Description

Specifies if the WEBVIEW control can navigate forward to a page in its navigation history.

Property Value

The CANGOFORWARD property is a Boolean value of TRUE$ or FALSE$.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
N/a	Get	No	No	No

Remarks

For more information on this property please refer to the Windows WebView2 documentation regarding the ICoreWebView2 get_CanGoForward method on the Microsoft website.

Example

```
// Example: Check to see if the Forward button should be enabled.

EnableForwardButton = Get_Property( CtrlEntID, "CANGOFORWARD" )
```

See Also

WEBVIEW CANGOBACK property, WEBVIEW HISTORY property, WEBVIEW BACK method, WEBVIEW CLEARBROWSINGDATA method, WEBVIEW FORWARD method, WEBVIEW WEBDATACLEARED event, WEBVIEW WEBHISTORYCHANGED event.

COLORSCHEME property

Description

Specifies the preferred color scheme for browser UI elements like context menus and dialogs in the WEBVIEW object.

Property Value

The ColorScheme property is a numeric value corresponding to one of the following:

Value	Description
0	Auto (default) – use whatever theme the OS is currently set to.
1	Light
2	Dark

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
Get/Set	Get/Set	No	No	No

Remarks

Equates for use with this property can be found in the PS_WEBVIEW_EQUATES insert record.

For more information on this property please refer to the Windows WebView2 documentation regarding the ICoreWebView2Profile get_PreferredColorScheme method on the Microsoft website.

Example

```
// Example: Set the color scheme to Dark
$Insert PS_WebView_Equates

Call_Set_Property_Only( CtrlEntID, "COLORSCHEME", WBV_COLORSCHEME_DARK$ )
```

See Also

N/a.

DOCUMENTTITLE property

Description
Returns the title of the current loaded top-level document in the WEBVIEW object.

Property Value
The DOCUMENTTITLE property value is a string containing the title for the current top-level document.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
N/a	Get	No	No	No

Remarks
For more information on this property please refer to the Windows WebView2 documentation regarding the ICoreWebView2 get_DocumentTitle method on the Microsoft website.

Example

```
// Example: Set the caption of the parent form to the same value as the
// document loaded in the WEBVIEW.

DocTitle = Get_Property( CtrlEntID, "DOCUMENTTITLE" )
Call Set_Property_Only( @Window, "TEXT", DocTitle )
```

See Also
WEBVIEW SYNCTITLE property, WEBVIEW NAVIGATE method, WEBVIEW WEBTITLECHANGED event.

ERRORTEXT property

Description

Returns any error information arising from the most recently executed property or method operation.

Property Value

The ERRORTEXT property value is a string containing any error details pertaining to the most recent property access or method execution. This value is an empty string if no errors have been recorded.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
N/a	Get	No	No	No

Remarks

The ERRORTEXT property is cleared before a new property or method operation is executed.

N/a.

Example

```
// Example: Call the EXECUTESCRIPT method to get the document title and check
// the ERRORTEXT variable if it indicates a failure

If Exec_Method( CtrlEntID, "EXECUTESCRIPT", "document.title", FALSE$, TRUE$ ) Else
   // Get the error details...
   ErrorText = Get_Property( CtrlEntID, "ERRORTEXT" )
End
```

See Also

WEBVIEW LOGERRORS property.

EXCLUSIVEUDFACCESS property

Description

Specifies if other WEBVIEW objects can use an environment created from the same user data folder as the current WEBVIEW object.

Property Value

The EXCLUSIVEUDFACCESS property value is a boolean value. When set to TRUE$ the current WEBVIEW object has exclusive access to the user data folder. Other WEBVIEW controls are prevented from accessing the folder and therefore cannot be created unless they specify a different USERDATAFOLDER property. The default value is FALSE$.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
Get/Set	Get	No	No	No

Remarks

For more information on this property please refer to the Windows WebView2 documentation regarding the ICoreWebView2EnvironmentOptions2 get_ExclusiveUserDataFolderAccess method on the Microsoft website.

Example

```
// Example: Check if the current WEBVIEW has exclusive access to the
// user data folder.

IsExclusiveUDF = Get_Property( CtrlEntID, "EXCLUSIVEUDFACCESS" )
```

See Also

SYSTEM WEBVIEWCONFIG property, WEBVIEW USERDATAFOLDER property.

EXTRABROWSERARGS property

Description
Specifies the extra arguments that can be passed to the underlying Chromium browser process when the WEBVIEW object is created.

Property Value
This property value is space-delimited string containing one or more command-line switches in the format:

```
"—" <switchName> "=" <switchValue>
```

E.g.

```
--disable-the-thing=1 --phase-converter-level=max
```

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
Get/Set	Get	No	No	No

Remarks
Default switches can be set via the SYSTEM WEBVIEWCONFIG property. Note that the property value returned at runtime is the *effective* value – i.e. if the object property is null, but the WEBVIEWCONFIG property specified a default value *then the default value will be returned*.

If you specify a switch that conflicts with WebView functionality, it is ignored. Specific features are disabled internally and blocked from being enabled. If a switch is specified multiple times, only the last instance is used.

For more information on this property please refer to the Windows WebView2 documentation regarding the ICoreWebView2EnvironmentOptions put_AdditionalBrowserArguments method on the Microsoft website.

Example

```
// Example: Get the Extra Browser Arguments for the WEBVIEW

BrowserArgs = Get_Property( CtrlEntID, "EXTRABROWSERARGS" )
```

See Also
SYSTEM WEBVIEWCONFIG property.

HISTORY property

Description

Returns a list of sites that have been navigated to by the WEBVIEW object in the current session.

Property Value

The HISTORY property is an @fm-delimited dynamic array of sites with each entry having an @vm-delimited format like so:

```
<0,1> Document Title
<0,2> URI
```

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
N/a	Get	No	No	No

Remarks

The TRACKHISTORY property must be set to TRUE$ for this property to be active. Setting the TRACKHISTORY property to FALSE$ will clear the data held in this property, as will using the CLEARBROWSINGDATA method.

Note that this property is a simple in-memory cache of navigated sites and is not saved when the WEBVIEW object is destroyed. It is provided to allow easy runtime access to the object's current browsing history and it is *not* the same as the standard full browsing history that is stored to disk in the User Data Folder as part of normal web-browsing operations. It is not affected by the INPRIVATEMODE property.

Example

```
// Example: Get the browsing history and load it into the EDT_HISTORY EditTable
// control

HistList = Get_Property( CtrlEntID, "HISTORY" )
Call Set_Property_Only( @Window : ".EDT_HISTORY", "LIST", HistList )


// Check to see if the WEBVIEW is saving browser data.

IsPrivate = Get_Property( CtrlEntID, "INPRIVATEMODE" )
```

See Also

WEBVIEW CANGOBACK property, WEBVIEW CANGOFORWARD property, WEBVIEW INPRIVATEMODE property, WEBVIEW TRACKHISTORY property, WEBVIEW BACK method, WEBVIEW CLEARBROWSINGDATA method, WEBVIEW FORWARD method, WEBVIEW WEBDATACLEARED event, WEBVIEW WEBHISTORYCHANGED, WEBVIEW WEBNAVIGATED event.

INITIALIZED property

Description

Specifies if the WEBVIEW object has been initialized successfully and is ready to begin navigation operations.

Property Value

This property is a boolean value. It returns TRUE$ if the WEBVIEW object is initialized and ready for use, or FALSE$ otherwise.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
N/a	Get	No	No	No

Remarks

This property may be used in a loop during a form's CREATE event processing to check if the WEBVIEW object is ready to begin navigating. However, this requires further complications such as allowing the system to process waiting events, handling timeouts, and ensuring the form hasn't been closed (see the example below).

A better and preferred alternative is to use the WEBVIEWCREATED event, which is raised by the WEBVIEW object when it has been initialized successfully and is therefore in the appropriate state to begin navigating.

Example

```
// Example: Wait until the WBV_BROWSER WEBVIEW control is ready on the form,
// timing out after 1 minute (60000 milliseconds)

Declare Function MsWin_GetTickCount64
$Insert Logical

EndTickCount  = ( MsWin_GetTickCount64() + 60000 )
WbvReady      = FALSE$

Loop
   WbvInit = Get_Property( @Window : ".WBV_BROWSER", "INITIALIZED" )
Until WbvInit
   Call Exec_Method( "SYSTEM", "PROCESSEVENTS", TRUE$ )
While Get_Property( @Window, "HANDLE" )
While ( MsWin_GetTickCount64() < EndTickCount )
Repeat

If WbvInit Then
   // OK to start browsing...
End
```

See Also
WEBVIEW READYSTATE property, WEBVIEW WEBVIEWCREATED event.

INPRIVATEMODE property

Description

Specifies if the WebView is operating in "InPrivate" mode.

Property Value

The INPRIVATEMODE property is a Boolean value of TRUE$ or FALSE$. When set to TRUE$ browsing data such as history, temporary internet files and cookies etc) are not saved to disk once the browsing session has ended.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
Get/Set	Get	No	No	No

Remarks

This property does not affect the TRACKHISTORY and HISTORY properties because they are never saved to disk.

For more information on this property please refer to the Windows WebView2 documentation regarding the ICoreWebView2ControllerOptions get_IsInPrivateModeEnabled method on the Microsoft website.

Example

```
   // Example: Check to see if the WEBVIEW is saving browser data.

   IsPrivate = Get_Property( CtrlEntID, "INPRIVATEMODE" )
```

See Also

WEBVIEW HISTORY property, WEBVIEW TRACKHISTORY property, WEBVIEW CLEARBROWSINGDATA method.

LANGUAGE property

Description

Specifies the default display language for the WEBVIEW object, which is used for browser UI elements like context menus and dialogs. It also applies to the accept-languages HTTP header that is sent to websites.

Property Value

This property is a string. When null it defaults to the current user's language settings, otherwise it should be in the format:

```
language[-country]
```

Where language is the two-letter ISO 639 code, and the optional country part is the two-letter ISO 3166 code.

E.g.

```
en-gb
fr-ca
```

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
Get/Set	Get	No	No	No

Remarks

For more information on this property please refer to the Windows WebView2 documentation regarding the ICoreWebView2EnvironmentOptions get_Language method on the Microsoft website.

Example

```
// Example: Get the default language setting.

WebViewLanguage = Get_Property( CtrlEntID, "LANGUAGE" )
```

See Also

N/a.

MUTED property

Description

Specifies if all audio output from this WEBVIEW object is muted or not.

Property Value

The MUTED property is a Boolean value of TRUE$ or FALSE$. When TRUE$ all audio output is prevented.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
N/a	Get/Set	No	No	No

Remarks

For more information on this property please refer to the Windows WebView2 documentation regarding the ICoreWebView2_8 get_IsMuted method on the Microsoft website.

Example

```
// Example: Mute all audio output in the WEBVIEW control
$Insert Logical

Call Set_Property_Only( CtrlEntID, "MUTED", TRUE$ )
```

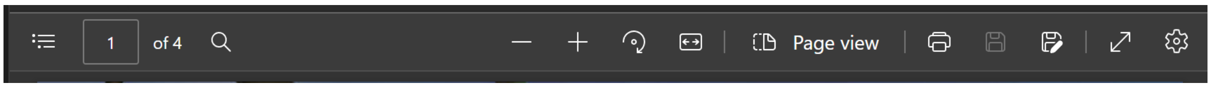
See Also

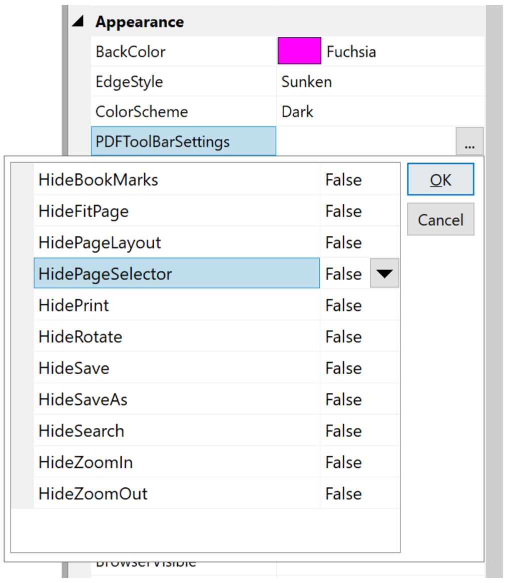
WEBVIEW AUDIOPLAYING property, WEBVIEW WEBAUDIOCHANGED event, WEBVIEW WEBMUTEDCHANGED event.

PDFTOOLBARSETTINGS property

Description

Specifies which features of the toolbar are visible when a PDF document is loaded into the WEBVIEW object.





BackColor	🟪 Fuchsia
EdgeStyle	Sunken
ColorScheme	Dark
PDFToolBarSettings	

HideBookMarks	False
HideFitPage	False
HidePageLayout	False
HidePageSelector	False
HidePrint	False
HideRotate	False
HideSave	False
HideSaveAs	False
HideSearch	False
HideZoomIn	False
HideZoomOut	False

Property Value

The PDFTOOLBARSETTINGS property is an @fm-delimited dynamic array of Boolean values with the following structure:

```
<1>  Hide Bookmarks
<2>  Hide Fit-Page
<3>  Hide Page Layout
<4>  Hide Page Selector
<5>  Hide Print
<6>  Hide Save
<7>  Hide Save-As
<8>  Hide Search
<9>  Hide Zoom-In
<10> Hide Zoom-Out
```

Setting a value to TRUE$ hides the specified toolbar feature, while setting it to FALSE$ will display it. By default all values are set to FALSE$.

Property Traits

Development	Runtime	Indexed	Scaled	Synthetic
Get/Set	Get/Set	No	No	No

Remarks

Changes to this property apply to all WEBVIEW objects in the same environment and using the same profile. Changes to this setting apply only after the next navigation.

For more information on this property please refer to the Windows WebView2 documentation regarding the ICoreWebView2Settings7 get_HiddenPdfToolbarItems method on the Microsoft website.

Example

```
// Example: Hide the Save and Save-As buttons in the WEBVIEW's PDF toolbar
$Insert PS_WebView_Equates

PDFTbrSettings = Get_Property( CtrlEntID, "PDFTOOLBARSETTINGS" )

PDFTbrSettings<WBV_PDFTBRSET_POS_HIDESAVE$>   = TRUE$
PDFTbrSettings<WBV_PDFTBRSET_POS_HIDESAVEAS$> = TRUE$

Call Set_Property_Only( CtrlEntID, "PDFTOOLBARSETTINGS", PDFTbrSettings )
```

See Also

WEBVIEW URI property, WEBVIEW NAVIGATE method, WEBVIEW PRINT2PDF method.

PROCESSID property

Description

Returns the ProcessID of the browser process hosting the WEBVIEW object.

Property Value

This property is an integer value.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
N/a	Get	No	No	No

Remarks

For more information on this property please refer to the Windows WebView2 documentation regarding the ICoreWebView2 get_BrowserProcessId method on the Microsoft website.

Example

```
// Example: Get the browser Process ID.

ProcessID = Get_Property( CtrlEntID, "PROCESSID" )
```

See Also

WEBVIEW BROWSEREXEFOLDER property.

PROFILENAME property

Description

Specifies the profile name used by the WEBVIEW object. This name is used to create the profile folder in the User Data Folder (UDF – see the USERDATAFOLDER property).

Property Value

The PROFILENAME property may be null or it may be a string value which has the following restrictions:

- It has a maximum length of 64 characters
- It is ASCII case-insensitive
- It may only contain the following characters:
  - alphabet characters: a-z and A-Z
  - digit characters: 0-9
  - '#', '@', '$', '(', ')', '+', '-', '_', '~', '.', ' ' (space)
- It must not end with a period '.' or ' ' (space)

When set in the Form Designer it may contain OS environment variable strings in the form: %variableName% (*not* case-sensitive). These will be expanded with their actual values when the WEBVIEW control is created at runtime.

This property may also contain the following OpenInsight environment variables (case-sensitive) that are expanded with their Basic+ values when the control is created at runtime:

```
@APPID    - Expands to @AppID<1>
@USERNAME – Expands to @UserName
```

Using Get_Property at runtime returns the fully expanded version of the string.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
Get/Set	Get	No	No	No

Remarks

A default value for the PROFILENAME can be set via the SYSTEM WEBVIEWCONFIG property. Note that the property value returned at runtime is the *effective* value – i.e. if the object property is null, but the WEBVIEWCONFIG property specifies a default value *then that default value will be returned*.

For more information on this property please refer to the Windows WebView2 documentation regarding the ICoreWebView2ControllerOptions get_ProfileName method on the Microsoft website.

Example

```
// Example: Get the profile name.

ProfileName = Get_Property( CtrlEntID, "PROFILENAME" )
```

See Also

SYSTEM WEBVIEWCONFIG property, WEBVIEW USERDATAFOLDER property.

READYSTATE property

Description

Specifies the status of a navigation request. As the WEBVIEW object navigates to a page it reaches certain defined stages, and this property is updated at each stage so it can be used to track the progress of the navigation.

Property Value

This property is a numeric value. As each stage in the navigation is reached it is updated to indicate its progress:

Stage	Value	Description
Uninitialized	0	The WEBVIEW object has not navigated to any pages yet.
Navigating	1	The WEBVIEW object has begun navigating to a page. The WEBNAVIGATING event is raised.
ContentLoading	2	The WEBVIEW object has begun loading content for the current page. The WEBCONTENTLOADING event is raised.
ContentLoaded	3	The WEBVIEW has loaded DOM content for the current page. The WEBCONTENTLOADED event is raised.
Navigated	4	The WEBVIEW object has finished navigating to a page. The WEBNAVIGATED event is raised.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
N/a	Get	No	No	No

Remarks

Due to the preference for asynchronous operations when using the WEBVIEW object, use of the various "Web" navigation events (WEBNAVIGATING, WEBNAVIGATED etc.) is encouraged rather waiting in a loop checking the READYSTATE property to see the status of the request.

READYSTATE is intended to be used in a loop once a navigation request has been made to check the progress of the request. However, this requires dealing with further complications such as allowing the system to process waiting events, handling timeouts, and ensuring the form hasn't been closed, thereby destroying the WEBVIEW object (see the example below).

A better and preferred alternative is to use the appropriate events, which are raised by the WEBVIEW object as it navigates:

- WEBNAVIGATING
- WEBCONTENTLOADING

- WEBCONTENTLOADED
- WEBNAVIGATED

Using these conforms to the asynchronous programming paradigm preferred by the WEBVIEW object.

Note also that the READYSTATE property simply reflects the state of the most recent navigation event – it is not linked to a specific URI and so does not identify the navigation requests when dealing with concurrent operations, unlike the events listed above.

Constants for use with the READYSTATE property can be found in the PS_WEBVIEW_EQUATES insert record.

Example

```
// Example: Wait in a loop, checking the READYSYATE until navigation has finished.

Declare Function MsWin_GetTickCount64
$insert PS_WebView_Equates
$Insert Logical

Call Exec_Method( @Window : ".WBV_BROWSER",     |
                  "NAVIGATE",                   |
                  "https://www.revelation.com" )

TimeOut = ( MsWin_GetTickCount64() + 10000 ) ; // 10 seconds

Loop
   Call Exec_Method( "SYSYTEM", "PROCESSEVENTS", TRUE$ )
   RS = Get_Property( @Window : ".WBV_BROWSER", "READYSTATE" )
Until ( RS >= WBV_READYSTATE_CONTENTLOADED$ )
Until ( MsWin_GetTickCount64() > TimeOut )
While Get_Property( @Window : ".WBV_BROWSER", "HANDLE" )
Repeat
```

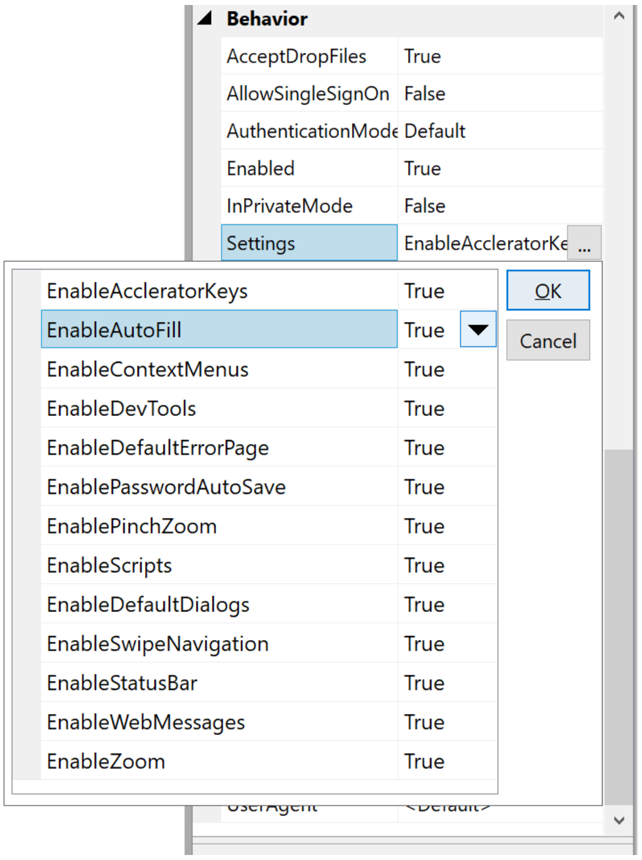
See Also

WEBVIEW INITIALIZED property, WEBVIEW URL property, WEBVIEW NAVIGATE method, WEBVIEW WEBCONTENTLOADED event, WEBVIEW WEBCONTENTLOADING event, WEBVIEW WEBNAVIGATED event, WEBVIEW WEBNAVIGATING event, WEBVIEW WEBVIEWCREATED event.

SETTINGS property

Description

Specifies the configuration options for the WEBVIEW object.



Behavior	
AcceptDropFiles	True
AllowSingleSignOn	False
AuthenticationMode	Default
Enabled	True
InPrivateMode	False
Settings	EnableAccleratorKe ...

EnableAccleratorKeys	True
EnableAutoFill	True
EnableContextMenus	True
EnableDevTools	True
EnableDefaultErrorPage	True
EnablePasswordAutoSave	True
EnablePinchZoom	True
EnableScripts	True
EnableDefaultDialogs	True
EnableSwipeNavigation	True
EnableStatusBar	True
EnableWebMessages	True
EnableZoom	True

Property Value

The SETTINGS property value is an @fm-delimited dynamic array of boolean flags that control the options for the WEBVIEW object:

```
<1>  EnableAccleratorKeys
<2>  EnableAutoFill
<3>  EnableContextMenus
<4>  EnableDevTools
<5>  EnableDefaultErrorPage
<6>  EnablePasswordAutoSave
<7>  EnablePinchZoom
<8>  EnableScripts
<9>  EnableScriptDialogs
<10> EnableSwipeNavigation
<11> EnableStatusBar
<12> EnableWebMessages
<13> EnableZoom
```

Setting	Description
EnableAcceleratorKeys	When this setting is set to FALSE$ it disables all accelerator keys that access features specific to a web browser,. Defaults to TRUE$.
EnableAutoFill	Specifies whether autofill for information like names, street and email addresses, phone numbers, and arbitrary input is enabled. Defaults to TRUE$.
EnableContextMenus	Setting this option to FALSE$ prevents the default context menus from being shown to the user. Defaults to TRUE$.
EnableDevTools	Specifies if the user can use the context menu or keyboard shortcuts to open the DevTools window. Defaults to TRUE$.
EnableDefaultErrorPage	Specifies if the built-in error pages for navigation failure and render process failure are used. Defaults to TRUE$.
EnablePasswordAutoSave	Specifies if autosave for password information is enabled. Defaults to FALSE$.
EnablePinchZoom	Enables or disables the ability of the end user to use a pinching motion on touch input enabled devices to scale the web content. Defaults to TRUE$.
EnableScripts	Specifies if JavaScript is enabled in future navigations. Note that This only affects scripts in the document. Scripts injected with EXECUTESCRIPT method run even if script is disabled. The default value is TRUE$.
EnableScriptDialogs	Specifies if the default JavaScript dialogs (alert, confirm, prompt and beforeunload) are used by the WebView control. When set to FALSE$ OpenInsight dialogs are used instead (See the WEBSHOWDIALOG event for more details). Defaults to TRUE$.
EnableSwipeNavigation	Enables or disables the ability of the end user to use swiping gesture on touch input enabled devices to navigate. Defaults to TRUE$.
EnableStatusBar	Specifies if the status bar is displayed. Defaults to TRUE$.
EnableWebMessages	Specifies communication from the host to the top-level HTML document of the WEBVIEW object is allowed using the POSTJSONMESSAGE and POSTTEXTMESSAGE methods, and the postMessage function of the JavaScript window.chrome.webview object (see the WEBMESSAGE event for more details). Defaults to TRUE$.
EnableZoom	When TRUE$ the user may zoom the web content using the keyboard and mouse. Defaults to TRUE$.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
Get/Set	Get/Set	No	No	No

Remarks

Constants for use with the SETTINGS property can be found in the PS_WEBVIEW_EQUATES insert record.

For more information on this property please refer to the Windows WebView2 documentation regarding the ICoreWebView2Settings interfaces and methods on the Microsoft website.

Example

```
// Example: Ensure that we are using OpenInsight script dialogs rather than the
// default JavaScript ones.
$Insert PS_WebView_Equates

WBVSettings = Get_Property( CtrlEntID, "SETTINGS" )

WBVSettings<WBV_SET_POS_SCRIPTDIALOGS$> = FALSE$

Call Set_Property_Only( CtrlEntID, "SETTINGS", WBVSettings )
```

See Also

WEBVIEW ZOOMFACTOR property, WEBVIEW EXECUTESCRIPT method, WEBVIEW OPENDEVTOOLS method, WEBVIEW POSTJSONMESSAGE method, WEBVIEW POSTTEXTMESSAGE method, WEBVIEW WEBINITCONTEXTMENU event, WEBVIEW WEBMESSAGE event, WEBVIEW SHOWDIALOG event, WEBVIEW STATUSTEXTCHANGED event.

SUSPENDED property

Description

Indicates if the WEBVIEW object is currently suspended.

Property Value

The SUSPENDED property is a Boolean value of TRUE$ or FALSE$.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
N/a	Get	No	No	No

Remarks

For more information on this property please refer to the Windows WebView2 documentation regarding the ICoreWebView2_3 get_ get_IsSuspended method on the Microsoft website.

Example

```
// Example: Check to see if the WEBVIEW control is suspended.

IsSuspended = Get_Property( CtrlEntID, "SUSPENDED" )
```

See Also

WEBVIEW BROWSERVISIBLE property, WEBVIEW RESUME method, WEBVIEW SUSPEND method, WEBVIEW WEBSUPENDED event.

SYNCSTATUSLINE property

Description

Specifies if the parent form’s STATUSLINE property is automatically updated when the WEBVIEW object’s status text changes.

Property Value

The SYNCSTATUSLINE property is a Boolean value of TRUE$ or FALSE$.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
Get/Set	Get/Set	No	No	No

Remarks

N/a.

Example

```
// Example: Ensure the STATUSLINE control in the parent form is updated
// from the WEBVIEW object when it’s status text changes.

Call Set_Property_Only( CtrlEntID, "SYNCSTATUSLINE", TRUE$ )
```

See Also

WINDOW STATUSLINE property, WEBVIEW WEBSTATUSTEXTCHANGED event.

SYNCTITLE property

Description

Specifies if the parent form's caption text (TEXT property) is automatically updated when the title of the top-level document is changed in the WEBVIEW object.

Property Value

The SYNCTITLE property is a Boolean value of TRUE$ or FALSE$.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
Get/Set	Get/Set	No	No	No

Remarks

N/a.

Example

```
// Example: Ensure the parent form’s caption is updated when the document
// title attribute is changed in the WEBVIEW object.

Call Set_Property_Only( CtrlEntID, "SYNCTITLE", TRUE$ )
```

See Also

WEBVIEW DOCUMENTTITLE property, WINDOW TEXT property, WEBVIEW NAVIGATE method, WEBVIEW WEBTITLECHANGED event.

TARGETVERSION property

Description

Specifies the minimum version of the Microsoft WebView2 runtime libraries that are compatible with the WEBVIEW object.

Property Value

The TARGETVERSION property is four-part period (".") delimited string formatted as follows:

```
    <majorNo> "." <minorNo> "." <buildNo> "." <releaseNo>
```

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
Get/Set	Get	No	No	No

Remarks

A default value for the TARGETVERSION can be set via the SYSTEM WEBVIEWCONFIG property. Note that the property value returned at runtime is the *effective* value – i.e. if the object property is null, but the WEBVIEWCONFIG property specifies a default value *then that default value will be returned*.

For more information on this topic please refer to the Windows WebView2 documentation regarding the CreateCoreWebView2EnvironmentWithOptions function on the Microsoft website.

Example

```
// Example: Check to see if we have a target version defined for the WEBVIEW control

TargetVersion = Get_Property( CtrlEntID, "TARGETVERSION" )
```

See Also

WEBVIEW Deployment Notes, WEBVIEW VERSION property, SYSTEM WEBVIEWCONFIG property.

TRACKHISTORY property

Description

Specifies if the WEBVIEW object keeps a list of sites that it has navigated to during the current session (exposed as the HISTORY property).

Property Value

The TRACKHISTORY property is a Boolean value of TRUE$ or FALSE$. When set to TRUE$ the HISTORY property may be used to see a list of sites that the WEBVIEW object has navigated to. Defaults to FALSE$.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
Get/Set	Get/Set	No	No	No

Remarks

Setting the TRACKHISTORY property to FALSE$ will clear the data held in the HISTORY property, as will using the CLEARBROWSINGDATA method.

Example

```
// Example: Ensure the WEBVIEW object is updating the HISTORY property during
// navigation

Call Set_Property_Only( CtrlEntID, "TRACKHISTORY", TRUE$ )
```

See Also

WEBVIEW HISTORY property, WEBVIEW INPRIVATEMODE property, WEBVIEW NAVIGATED event.

URI property

Description

Specifies the URI (Uniform Resource Identifier) of the top-level document in the WEBVIEW object.

Property Value

The URI property value is a string containing the identifier of the current top-level document. When this property is set the WEBVIEW object will navigate to the new value.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
Get/Set	Get/Set	No	No	No

Remarks

For more information on this property please refer to the Windows WebView2 documentation regarding the ICoreWebView2 get_Source and Navigate methods on the Microsoft website.

Example

```
// Example: Navigate to the Revelation Software home page.

Call Set_Property_Only( CtrlEntID, "URI", "https://www.revelation.com" )
```

See Also

WEBVIEW NAVIGATE method, WEBVIEW WEBNAVIGATING event, WEBVIEW NAVIGATED event.

USERAGENT property

Description
Specifies a default “User-Agent” string sent by the WEBVIEW object in a request.

Property Value
The USERAGENT property contains the string sent in the “User-Agent” header when the WEBVIEW object makes a request to a server. It defaults to the same value as the “User-Agent” of the Microsoft Edge web-browser.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
Get/Set	Get/Set	No	No	No

Remarks
For more information on this property please refer to the Windows WebView2 documentation regarding the ICoreWebView2Settings2 get_UserAgent and put_UserAgent methods on the Microsoft website.

Example

```
// Example: Specify a custom User-Agent string for the WEBVIEW object

Call Set_Property_Only( CtrlEntID, "USERAGENT", "OpenInsight WebView 10.2" )
```

See Also
WEBVIEW NAVIGATE method.

USERDATAFOLDER property

Description

Specifies the location of the WEBVIEW object's user data folder (UDF). The UDF is a folder stored on the user's machine that stores browser data such as cookies, permissions, and cached resources. Each instance of the WEBVIEW object is associated with a UDF (Multiple WEBVIEW objects may share a UDF unless it is set to exclusive access as per the EXCLUSIVEUDFACCESS property).

Property Value

This property value may be null or may contain a relative or absolute folder path to the location of the UDF. If this is null the default UserDataFolder value specified in the SYSTEM WEBVIEWCONFG property is used instead.

When set in the Form Designer it may contain OS environment variable strings in the form: %variableName% (*not* case-sensitive). These will be expanded with their actual values when the WEBVIEW control is created at runtime.

This property may also contain the following OpenInsight environment variables (case-sensitive) that are expanded with their Basic+ values when the control is created at runtime:

```
@APPID    - Expands to @AppID<1>
@USERNAME – Expands to @UserName
```

Using Get_Property at runtime returns the fully expanded version of the string.

Note that a UDF should meet the following requirements:

- The custom UDF location must have appropriate Read/Write permissions
- Avoid creating a UDF on a network drive. This can result in slowdowns, crashes, or loss of data.

If no value is set for this property in either the WEBVIEW object itself or via the SYSTEM WEBVIEWCONFIG property, then the object would use/create a default folder in the directory that OpenInsight is running in called "OpenInsight.exe.WebView2". Because many OpenInsight systems are executed from a network they may not have appropriate permissions in this folder so this not an ideal scenario and must be avoided. For this reason, the WEBVIEWCONFIG property *always* returns a default value of:

```
%localappdata%\RevSoft\WBV\@APPID_@USERNAME
```

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
Get/Set	Get	No	No	No

Remarks

See the Deployment Notes section earlier in this document for further details on how the WebView2 control handles the USERDATAFOLDER property.

Note that the property value returned at runtime is the *effective* value – i.e., if the object property is null, but the WEBVIEWCONFIG property specifies a default value, *then that default value will be returned*. A default value for the USERDATAFOLDER is always set via the SYSTEM WEBVIEWCONFIG property as noted above.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2Environment7 get_UserDataFolder method, and the WebView2 "Manage user data folders" page on the Microsoft website.

Example

```
// Example: Assuming the following:
//
// 1) Windows has an environment variable LOCALAPPDATA set like so:
//
//   LOCALAPPDATA=C:\Users\AgaentC\AppData\Local
//
// 2) OpenInsight is logged into the EXAMPLES app with a username
//    of A_TEST
//
// 3) The UserDataFolder property is set in the Form Designer to
//    a value of:
//
//       %LocalAppData%\@APPID_@USERNAME_WebView2UDF

FolderVal = Get_Property( CtrlEntID, "USERDATAFOLDER" )

// FolderVal now contains the value:
//
//   C:\Users\AgaentC\AppData\Local\EXAMPLES_A_TEST_WebView2UDF
```

See Also

WEBVIEW Deployment Notes, WEBVIEW EXCLUSIVEUDFACCESS property, SYSTEM WEBVIEWCONFIG property.

VERSION property

Description

Returns the version of the Microsoft WebView2 runtime libraries that are being used by the WEBVIEW object.

Property Value

The VERSION property is four-part period (".") delimited string formatted as follows:

```
<majorNo> "." <minorNo> "." <buildNo> "." <releaseNo>
```

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
N/a	Get	No	No	No

Remarks

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2Environment get_BrowserVersionString method on the Microsoft website.

Example

```
// Get the version of the WEBVIEW control

Version = Get_Property( CtrlEntID, "VERSION" )
```

See Also

WEBVIEW Deployment Notes, WEBVIEW TARGETVERSION property, SYSTEM WEBVIEWCONFIG property.

ZOOMFACTOR property

Description

Specifies the default zoom factor for the WEBVIEW object.

Property Value

The ZOOMFACTOR property is a numeric value between 0.25 and 5.0

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
Get/Set	Get/Set	No	Yes	No

Remarks

A zoom factor that is applied via this property becomes the new default zoom for the WEBVIEW object. This zoom factor applies across all navigations and is the zoom factor that the WEBVIEW object is returned to when the user presses Ctrl+0. When the zoom factor is changed by the user that zoom applies only to the current page.

Setting this property does not trigger a WEBZOOMCHANGED event.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2Controller get_ZoomFactor and put_ZoomFactor methods on the Microsoft website.

Example

```
// Set the default Zoom Factor to 200% (2.0)

Call Set_Property_Only( CtrlEntID, "ZOOMFACTOR", 2.0 )
```

See Also

WEBVIEW WEBZOOMCHANGED event.

WEBVIEW Methods

The WEBVIEW object supports the following methods:

Name	Description
ADDCDPEVENT	Allows the WEBVIEW object to receive notifications for a specified CDP event.
ADDINITSCRIPT	Adds a script to the WEBVIEW object that is run before documents are loaded.
ALLOWOPENWINDOW	Allows the WEBVIEW object to open a new window from the WEBOPENWINDOW event.
AUTHENTICATE	Allows the WEBVIEW to return credentials or cancel a Basic Authentication challenge.
BACK	Navigates to the previous page in the navigation history.
CANCELCONTEXTMENU	Cancels a context menu request and releases any associated resources.
CANCELDIALOG	Cancels a "show dialog" request and releases any associated resources.
CANCELPERMISSIONREQUEST	Cancels and denies any pending permission request and releases any associated resources.
CLEARBROWSINGDATA	Clears browsing data stored by the WEBVIEW object.
CONFIRMDIALOG	Notifies the WEBVIEW object that the "show dialog" request was processed.
COPY	Copies the current selection in the WEBVIEW object to the clipboard.
CUT	Removes the current selection from the WEBVIEW object and copies it to the clipboard.
DELETECOOKIES	Removes one or more cookies from the WEBVIEW object.
DENYOPENWINDOW	Prevents the WEBVIEW object from opening a new window in the WEBOPENWINDOW event.
EXECUTECDPMETHOD	Executes a specified CDP method.
EXECUTESCRIPT	Executes the specified JavaScript.
FORWARD	Navigates to the next page in the navigation history.
GETCOOKIES	Returns a list of cookies from the WEBVIEW object that match a specified URI.
MAPHOSTNAMETOFOLDER	Creates a mapping between a virtual host name and a local folder path
NAVIGATE	Navigates the WEBVIEW object to the specified URI.
OPENBROWSERTASKMANAGER	Opens the browser task manager window for the WEBVIEW object.
OPENDEVTOOLS	Opens the DevTools window for the current document in the WEBVIEW object.

PASTE	Pastes the clipboard contents at the insertion point (replaces current selection) in the WEBVIEW object.
POSTJSONMESSAGE	Posts a JSON-formatted WebMessage to the top-level document in the WEBVIEW control.
POSTTEXTMESSAGE	Posts a text-formatted WebMessage to the top-level document in the WEBVIEW control.
PRINT	Opens a dialog box to print the current document in the WEBVIEW object.
PRINTTOPDF	Print the current page to PDF asynchronously with the provided settings.
REDO	Discards the results of the last Undo command.
RELOAD	Reloads the current top-level document in the WEBVIEW object.
REMOVECDPEVENT	Stops the WEBVIEW object from receiving notifications for a specified CDP event.
REMOVEINITSCRIPT	Removes the specified JavaScript previously added with the ADDINTISCRIPT method.
RESUME	Resumes a suspended WEBVIEW object.
SAVETOFILE	Saves the contents of the WEBVIEW object to disk using the MHTML format.
SELECTALL	Selects all the content of the editable region of the WEBVIEW object.
SETCOOKIE	Adds or updates a cookie in the WEBVIEW object.
SETHTML	Loads a HTML document into the WEBVIEW object as a string.
SETPERMISSION	Allows or denies access to privileged resources from the content in the WEBVIEW object.
STOP	Stops all navigations and pending resource fetches in the WEBVIEW object.
SUSPEND	Suspends a WEBVIEW object, forcing it to consume less memory and resources.
UNDO	Undoes the last command.
UNMAPHOSTNAME	Removes a mapping between a virtual host name and a local folder path.

ADDCDPEVENT method

Description

Allows the WEBVIEW object to receive notifications for a specified Chrome DevTools Protocol (CDP) event. The Chrome DevTools Protocol allows for tools to instrument, inspect, debug and profile Chromium, Chrome and other Blink-based browsers.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID,      |
                           "ADDCDPEVENT",  |
                           EventName )
```

Parameters

Name	Required	Description
EventName	Yes	Specifies the name of the CDP event to listen for. This is case-sensitive.

Returns

"1" (TRUE$) if the event was added successfully, "0" (FALSE$) otherwise.

Remarks

Use the REMOVECDPEVENT method to stop receiving notifications for the CDP event.

More information on the CDP can be found on here along with the events that are supported:

```
https://chromedevtools.github.io/devtools-protocol/
```

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2 GetDevToolsProtocolEventReceiver method on the Microsoft website.

Example

```
// Example: Listen for "CDP Log" events from the WEBVIEW object. The
// CDP offers a logging service - to monitor log entries in OpenInsight
// we can do the following:
//
//    1) Listen for the "Log.entryAdded" event
//    2) Begin logging by using the CDP "Log.enable" method
//    3) Handle the new log entry in the WEBCDPEVENT event
//    4) Stop logging by using the CDP "Log.disable" method.

$Insert Logical

//  Listen for the "Log.entryAdded" event
IsOk = Exec_Method( CtrlEntID, "ADDCDPEVENT", "Log.entryAdded" )

// Begin logging by using the CDP "Log.enable" method
IsOK = Exec_Method( CtrlEntID, "EXECUTECDPMETHOD", "Log.enable", "", TRUE$ )

// In the WEBCDPEVENT we now receive notifications when a log entry
// is added along with a JSON object containing the details.

...

// Stop logging by using the CDP "Log.disable" method
IsOK = Exec_Method( CtrlEntID, "EXECUTECDPMETHOD", "Log.disable", "", TRUE$ )
```

See Also

WEBVIEW EXECUTECDPMETHOD method, WEBVIEW REMOVECDPEVENT method, WEBVIEW WEBCDPEVENT event, WEBVIEW WEBCDPMETHODRESULT event.

ADDINITSCRIPT method

Description

Adds 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 (The script runs on all top-level document and child frame page navigations).

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID,        |
                           "ADDINITSCRIPT",  |
                           Script )
```

Parameters

Name	Required	Description
Script	Yes	Specifies the JavaScript to add.

Returns

"1" (TRUE$) if the script was added successfully, "0" (FALSE$) otherwise.

Remarks

The script is added asynchronously and given a unique identifier. The actual result of the method and the identifier can be accessed in the WEBINITSCRIPTADDED event.

The identifier can be used with the REMOVEINITSCRIPT method to remove it if desired.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2 AddScriptToExecuteOnDocumentCreated method on the Microsoft website.

Example

```
// Example: Display a message containing the document URI each
// time a document is loaded.

Script = "alert( window.location.href );"

IsOK = Exec_Method( CtrlEntID, "ADDINITSCRIPT", Script )

// The results of this method call are passed in the
// WEBINITSCRIPTADDED event.
```

See Also
WEBVIEW REMOVEINITSCRIPT method, WEBVIEW WEBINITSCRIPTADDED event.

ALLOWOPENWINDOW method

Description

This method should be called when handling a WEBOPENWINDOW event to allow the WEBVIEW control to open the default WebView2 window or use another specified WEBVIEW object instead.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID,          |
                           "ALLOWOPENWINDOW",  |
                           OpenID,             |
                           WebViewID )
```

Parameters

Name	Required	Description
OpenID	Yes	ID of the “open window” request as passed from the WEBOPENWINDOW event.
WebViewID	No	Name of an existing WEBVIEW object to use instead of the default WebView2 Window when opening a new window.

Returns

"1" (TRUE$) if the method was executed successfully, "0" (FALSE$) otherwise.

Remarks

When the content inside the WEBVIEW object requests to open a new window (e.g. via the JavaScript “window.open” method) then a WEBOPENWINDOW event is raised. In this event the ALLOWOPENWINDOW method may be called to allow the window to be opened or redirected to another existing WEBVIEW object, or the DENYOPENWINDOW method may called to prevent it (The system prompted WEBOPENWINDOW handler simply allows a new WebView2 to be opened).

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2 add_NewWindowRequested method on the Microsoft website.

Example

```
Function WEBOPENWINDOW( CtrlEntID, CtrlClassID, OpenID, URI, WindowInfo, |
                                                           UserInitiated )

  // Example: A WEBOPENWINDOW event handler that creates a new OpenInsight
  // form (TEST_WEBVIEW_OPENWIN) with a WEBVIEW control called WBV_BROWSER,
  // and directs the system to use that for displaying the content.

  WinID = Start_Window( "TEST_WEBVIEW_OPENWIN", "", "" )
  If BLen( WinID ) Then

     Call Exec_Method( CtrlEntID,                |
                       "ALLOWOPENWINDOW",        |
                       OpenID,                   |
                       WinID : ".WBV_BROWSER" )

  End

Return FALSE$
```

See Also

WEBVIEW DENYOPENWINDOW method, WEBVIEW WEBOPENWINDOW event.

AUTHENTICATE method

Description

This method should be called when handling a WEBAUTHREQUEST event to return the credentials for a Basic Authentication challenge, or to cancel it.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "AUTHENTICATE", URI, UserName, |
                                                Password, CancelFlag )
```

Parameters

Name	Required	Description
URI	Yes	URI that triggered the authentication request.
UserName	No	Username to return to the server.
Password	No	Password to return to the server.
CancelFlag	No	Set to TRUE$ to cancel the request. Defaults to FALSE$.

Returns

"1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

Remarks

A WEBAUTHREQUEST event is raised to handle a Basic Authentication request from the server when the AUTHENTICATIONMODE property is set to "Custom". At this point the handler should respond to the request by using the AUTHENTICATE method to return the credentials or set the cancel flag (so that the authentication fails).

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2_10 add_BasicAuthenticationRequested method on the Microsoft website.

Example

```
Function WEBAUTHREQUEST( CtrlEntID, CtrlClassID, URI, Challenge )

  // Example: A WEBAUTHREQUEST handler that uses a dialog box to ask the user
  // for the credentials for the passed URL.  The AUTHENTICATE method is
  // executed  to return an answer to the server.

  // Assume we have a dialog box called GET_WEB_CREDENTIALS that takes
  // the passed URI and Challenge string returned from the server.

  DlgParams = URL : @fm : Challenge
  Credentials  = Dialog_Box( "GET_WEB_CREDENTIALS", @Window, DlgParams )

  If BLen( Credentials ) Then
     // Assume the dialog passed back the UserName and Password as
     // an @fm-delimited array

     UN = Credentials<1>
     PW = Credentials<2>

     Call Exec_Method( CtrlEntID, "AUTHENTICATE", UN, PW, FALSE$ )

  End Else
     // User Cancelled - stop the request
     Call Exec_Method( CtrlEntID, "AUTHENTICATE", "", "", TRUE$ )
  End

Return FALSE$
```

See Also

WEBVIEW AUTHENTICATIONMODE property, WEBVIEW WEBAUTHREQUEST event.

BACK method

Description
Navigates the WEBVIEW object to the previous page in the navigation history.

Syntax
```
SuccessFlag = Exec_Method( CtrlEntID, "BACK" )
```

Parameters
N/a.

Returns
"1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

Remarks
For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2 GoBack method on the Microsoft website.

Example
```
// Example: Navigate to the previous page in the WEBVIEW object's
// navigation history

If Get_Property( CtrlEntID, "CANGOBACK" ) Then
   Call Exec_Method( CtrlEntID, "BACK" )
End
```

See Also
WEBVIEW CANGOBACK property, WEBVIEW CANGOFORWARD property, WEBVIEW HISTORY property, WEBVIEW CLEARBROWSINGDATA method, WEBVIEW FORWARD method, WEBVIEW WEBDATACLEARED event, WEBVIEW WEBHISTORYCHANGED event.

CANCELCONTEXTMENU method

Description

Cancels a context menu request and releases any associated resources.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "CANCELCONTEXTMENU", MenuID )
```

Parameters

Name	Required	Description
MenuID	Yes	The ID of the context menu. This is the value passed in the MenuID parameter of the WEBINITCONTEXTMENU and WEBCONTEXTMENU events.

Returns

"1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

Remarks

This method can only be used within the context of the WEBINITCONTEXTMENU and WEBCONTEXTMENU events.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2_11 add_ContextMenuRequested method on the Microsoft website.

Example

See the WEBVIEW WEBCONTEXTMENU event for an example of using the CANCELCONTEXTMENU method.

See Also

WEBVIEW WEBCONTEXTMENU event, WEBVIEW WEBINITCONTEXTMENU event.

CANCELDIALOG method

Description

Cancels a “show dialog” request and releases any associated resources.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "CANCELDIALOG", DialogID )
```

Parameters

Name	Required	Description
DialogID	Yes	A unique ID for the show dialog request. This is the value passed in the DialogID parameter of the WEBSHOWDIALOG event.

Returns

"1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

Remarks

This method can only be used within the context of a WEBSHOWDIALOG event.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2 add_ScriptDialogOpening method on the Microsoft website.

Example

See the WEBVIEW WEBSHOWDIALOG event for an example of using the CANCELDIALOG method.

See Also

WEBVIEW CONFIRMDIALOG request, WEBVIEW WEBSHOWDIALOG event.

CANCELPERMISSIONREQUEST method

Description

Cancels and denies any pending permission request and releases any associated resources.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "CANCELPERMISSIONREQUEST" )
```

Parameters

N/a.

Returns

"1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

Remarks

This method can only be used within the context of a WEBPERMISSIONREQUEST event. Ideally permission requests should be handled by using the SETPERMISSION method, but this method can be used to simply deny a request if desired. (It is used in the system promoted WEBPERMISSIONREQUEST event handler as a "catch-all" to ensure any resources are released).

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2 add_PermissionRequested method on the Microsoft website.

Example

```
 Function WEBPERMISSIONREQUEST( CtrlEntID, CtrlClassID, URI, RequestType, |
                                UserInitiated )

   // Example: A WEBPERMISSIONREQUEST event handler that refuses permission for all
   //          requests and prevents the system promoted event handler from executing.
   $Insert Logical

   Call Exec_Method( CtrlEntID, "CANCELPERMISSIONREQUEST" )

Return FALSE$
```

See Also

WEBVIEW SETPERMISSION method, WEBVIEW WEBPERMISSIONREQUEST event.

CLEARBROWSINGDATA method

Description

Clears browsing data stored by the WEBVIEW object based on the type of data and/or a date range.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "CLEARBROWSINGDATA", DataTypes, |
                                      dateTimeFrom, dateTimeFrom )
```

Parameters

Name	Required	Description

DataTypes	Yes	An @fm-delimited list of boolean flags representing the types of data to clear, or "*" for all.
		
		`<1>  All Profile Data (same as "*")`
		`<2>  All Site Data`
		`<3>  All DOM Storage`
		`<4>  File Systems`
		`<5>  Indexed DB`
		`<6>  Local Storage`
		`<7>  Web SQL`
		`<8>  Cache Storage`
		`<9>  Cookies`
		`<10> Disk Cache`
		`<11> Download History`
		`<12> General AutoFill`
		`<13> Password AutoSave`
		`<14> Browsing History`
		`<15> Settings`
		
		Note that this array represents a hierarchy of types to clear – see the Remarks section below for more details.

DateTimeFrom	No	Date from which data should be cleared (expressed in internal Revelation DateTime (DT) format).
DateTimeTo	No	Date up to which data should be cleared (expressed in internal Revelation DateTime (DT) format).

Returns

"1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

Remarks

The WEBVIEW object raises a WEBDATACLEARED event to return the actual result of the CLEARBROWSINGDATA method.

Calling this method also clears the data held by the HISTORY property if the DataTypes argument includes the "Browsing History" type (field <14>).

Note that the DataTypes parameter represents a hierarchical structure like so:

```
All Profile Data
    All Site Data
        All DOM Storage
            File Systems
            Indexed DB
            Local Storage
            Web SQL
            Cache Storage
        Cookies
    Disk Cache
    Download History
    General Autofill
    Password Autosave
    Browsing History
    Settings
```

E.g., if you specify "All DOM Storage" then you implicitly specify "File Systems", "Indexed DB", "Local Storage", "Web SQL" and "Cache Storage" as well, and so on.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2Profile ClearBrowsingDataInTimeRange method on the Microsoft website.

Example

```
// Example: Clear "All Browsing History" in the previous week
$Insert PS_WebView_Equates

DataTypes = ""
DataTypes<WBV_CBD_POS_BROWSINGHISTORY$> = TRUE$

DateFrom = ( Date() - 7 ) : ".0" ; // DT format

Call Exec_Method( CtrlEntID, "CLEARBROWSINGDATA", DataTypes, DateFrom, "" )

// Example: Clear "All Site Data" and "Download History".
//
// Because we have selected "All Site Data" this will also clear:
//
//   1) All DOM Storage (File Systems, Indexed Local Storage, Web SQL and
//                       Cache Storage)
//   2) Cookies
//

DataTypes = ""
DataTypes<WBV_CBD_POS_ALLSITEDATA$> = TRUE$

Call Exec_Method( CtrlEntID, "CLEARBROWSINGDATA", DataTypes, "", "" )
```

See Also

WEBVIEW CANGOBACK property, WEBVIEW CANGOFORWARD property, WEBVIEW HISTORY property, WEBVIEW BACK method, WEBVIEW FORWARD method, WEBVIEW WEBDATACLEARED event.

CONFIRMDIALOG method

Description

Notifies the WEBVIEW object that the "show dialog" request was processed and optionally returns a value for a "prompt" type dialog.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "CONFIRMDIALOG", DialogID, ResponseText )
```

Parameters

Name	Required	Description
DialogID	Yes	A unique ID for the show dialog request. This is the value passed in the DialogID parameter of the WEBSHOWDIALOG event.
ResponseText	No	The value the user entered for a "prompt" type dialog. Only for use with the latter type.

Returns

"1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

Remarks

This method can only be used within the context of a WEBSHOWDIALOG event.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2 add_ScriptDialogOpening method on the Microsoft website.

Example

See the WEBVIEW WEBSHOWDIALOG event for an example of using the CONFIRMDIALOG method.

See Also

WEBVIEW CANCELDIALOG request, WEBVIEW WEBSHOWDIALOG event.

COPY method

Description

Copies the current selection in the WEBVIEW object to the clipboard.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "COPY" )
```

Parameters

N/a.

Returns

"1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

Remarks

The WEBVIEW COPY method uses the EXECUTESCRIPT method to call the underlying JavaScript `document.execCommand( "copy" )` method.

For more information on the JavaScript `document.execCommand` method please refer to the documentation on the Mozilla Developer website at:

https://developer.mozilla.org/en-US/docs/Web/API/Document/execCommand

Example

```
// Example: Copy the current selection

Call Exec_Method( CtrlEntID, "COPY" )
```

See Also

WEBVIEW CUT method, WEBVIEW EXECUTESCRIPT method, WEBVIEW PASTE method, WEBVIEW REDO method, WEBVIEW UNDO method.

CUT method

Description

Removes the current selection from the WEBVIEW object and copies it to the clipboard.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "CUT" )
```

Parameters

N/a.

Returns

"1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

Remarks

The WEBVIEW CUT method uses the EXECUTESCRIPT method to call the underlying JavaScript `document.execCommand( "cut" )` method.

For more information on the JavaScript `document.execCommand` method please refer to the documentation on the Mozilla Developer website at:

https://developer.mozilla.org/en-US/docs/Web/API/Document/execCommand

Example

```
// Example: Cut the current selection

Call Exec_Method( CtrlEntID, "CUT" )
```

See Also

WEBVIEW COPY method, WEBVIEW EXECUTESCRIPT method, WEBVIEW PASTE method, WEBVIEW REDO method, WEBVIEW UNDO method.

DELETECOOKIES method

Description

This method deletes one or more cookies from the WEBVIEW object, based on a combination of the name URI, domain, and path.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID,       |
                           "DELETECOOKIES", |
                           Name,            |
                           URI,             |
                           Domain,          |
                           Path )
```

Parameters

Name	Required	Description
Name	Yes	The name of the cookie to delete, or "*" to specify all cookies.
URI	No	If passed only cookies matching this name are deleted and the domain and path arguments are ignored.
Domain	No	If passed only cookies matching this domain are deleted.
Path	No	If passed only cookies matching this path are deleted.

Returns

"1" (TRUE$) if the method was executed successfully, "0" (FALSE$) otherwise.

Remarks

Note that this method could affect other WEBVIEW objects that are using the same UDF (User Data Folder) and profile name.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2CookieManager DeleteAllCookies, DeleteCookies, and DeleteCookiesWithDomainAndPath methods on the Microsoft website.

Example

```
// Example: Delete all cookies from "www.revelation.com"

Call Exec_Method( CtrlEntID, "DELETECOOKIES", "*", "", "revelation.com", "" )
```

See Also

WEBVIEW USERDATAFOLDER property, WEVIEW PROFILENAME property, WEBVIEW GETCOOKIES method, WEBVIEW SETCOOKIE method.

DENYOPENWINDOW method

Description

This method should be called when handling a WEBOPENWINDOW event to prevent the WEBVIEW object from opening a new default WebView2 window.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID,          |
                           "DENYOPENWINDOW",  |
                           OpenID )
```

Parameters

Name	Required	Description
OpenID	Yes	ID of the "open window" request as passed from the WEBOPENWINDOW event.

Returns

"1" (TRUE$) if the method was executed successfully, "0" (FALSE$) otherwise.

Remarks

When the content inside the WEBVIEW object requests to open a new window (e.g. via the JavaScript "window.open" method) then a WEBOPENWINDOW event is raised. In this event the DENYOPENWINDOW method may be called to prevent the new window from being opened.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2 add_NewWindowRequested method on the Microsoft website.

Example

```
 Function WEBOPENWINDOW( CtrlEntID, CtrlClassID, OpenID, URI, WindowInfo, |
                                                           UserInitiated )

  // Example: A WEBOPENWINDOW handler that prevents a new window from opening
  // from the "example.com" domain (not a rigorous test!)

  If IndexC( URI, "example.com", 1 ) then
     Call Exec_Method( CtrlEntID, "DENYOPENWINDOW", OpenID )
  End

Return FALSE$
```

See Also
WEBVIEW ALLOWOPENWINDOW method, WEBVIEW WEBOPENWINDOW event.

EXECUTECDPMETHOD method

Description

Executes a specified Chrome DevTools Protocol (CDP) method for the WEBVIEW object (The Chrome DevTools Protocol allows for tools to instrument, inspect, debug and profile Chromium, Chrome and other Blink-based browsers).

CDP methods are always executed asynchronously.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "EXECUTECDPMETHOD", MethodName, |
                                      MethodParams, IgnoreResult )
```

Parameters

Name	Required	Description
MethodName	Yes	Name of the CDP method to execute in the format: `{domain}.{method}` This is case-sensitive.
MethodParams	No	JSON-encoded parameter to pass to the CDP method. Defaults to null.
IgnoreResult	No	If TRUE$ then a WEBCDPMETHODRESULT event is *not* raised to return the results of the method call. Defaults to FALSE$.

Returns

"1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

Remarks

The actual results of the method call are returned via a WEBCDPMETHODRESULT event. Note that 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.

More information on the CDP can be found on here along with the methods that are supported:

```
https://chromedevtools.github.io/devtools-protocol/
```

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2 CallDevToolsProtocolMethod method on the Microsoft website.

Example

```
// Example:  Execute the "Browser.getVersion" CDP method to get the
// browser version information.  This method takes no parameters.
//
// The version information is returned via the WEBCDPMETHODRESULT
// event.

Call Exec_Method( CtrlEntID, "EXECUTECDPMETHOD",   |
                  "Browser.getVersion", "", FALSE$ )


// Example:  Execute the "DOM.getDocument" CDP method to get the
// root DOM node and the subtree
//
// The DOM node is returned via the WEBCDPMETHODRESULT
// event.

// To get the entire subtree we need to pass a "depth" parameter
// with an integer value of -1, encoded as a JSON object.
JsonParam = '{ "depth" : -1 }'

Call Exec_Method( CtrlEntID, "EXECUTECDPMETHOD",   |
                  "DOM.getDocument", JsonParam, FALSE$ )
```

See Also

WEBVIEW ADDCDPEVENT method, WEBVIEW REMOVECDPEVENT property, WEBVIEW WEBCDPEVENT event, WEBVIEW WEBCDPMETHODRESULT event.

EXECUTESCRIPT method

Description

Executes the specified JavaScript in the context of the top-level document of the WEBVIEW object.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID,        |
                           "EXECUTESCRIPT",  |
                           Script,           |
                           IgnoreResult,     |
                           SyncMode,         |
                           SyncTimeout )
```

Parameters

Name	Required	Description
Script	Yes	A string containing the JavaScript to execute.
IgnoreResult	No	When TRUE$ the script is executed asynchronously and no WEBSCRIPTRESULT event is raised. Defaults to FALSE$. (This parameter overrides SyncMode – i.e. if IgnoreResult is TRUE$ then SyncMode is forced to FALSE$).
SyncMode	No	When TRUE$ the script is executed in synchronous mode, i.e. the result of the script is returned directly from the Exec_Method call. When set to FALSE$ (the default) The result of the script Is returned via the WEBSCRIPTRESULT event.
SyncTimeout	No	If the SyncMode parameter is TRUE$ then this parameter specifies the time in milliseconds to wait for an answer. Defaults to 10000ms (10 seconds).

Returns

If SyncMode is FALSE$ then this method returns "1" (TRUE$) if executed successfully, "0" (FALSE$) otherwise. If SyncMode is TRUE$ then the result of the executed script is returned. If SyncMode is TRUE$ and the script times out then null is returned.

Remarks

As stated previously the WEBVIEW object is designed to be run in an asynchronous fashion and running the EXECUTESCRIPT method in the default asynchronous mode is always the preferred solution (i.e., SyncMode is FALSE$).

However, OpenInsight provides a synchronous solution (when SyncMode is set to TRUE$) which essentially waits in a timed loop for the answer to be returned via the WEBSYNCSCRIPTRESULT event and passed back to the caller. This can be used for small fast requests – anything lengthy should be processed asynchronously.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2 ExecuteScript method on the Microsoft website.

Example

```
$Insert Logical

// Example: Set the document title using JavaScript, ignoring any result

Script = "document.title = 'New Title';"
Call Exec_Method( CtrlEntID, "EXECUTESCRIPT", Script, TRUE$ )

// Example: Get the document title using a synchronous EXECUTESCRIPT
// call

Script = "document.title"
DocTitle = Exec_Method( CtrlEntID,        |
                        "EXECUTESCRIPT", |
                        Script,          |
                        FALSE$,          |
                        TRUE$ )


// Example: Get the contents of the document body using an asynchronous
// EXECUTESCRIPT call.  The results of the script will be returned via
// the WEBSCRIPTRESULT event.

Script = "document.body.innerHTML"
Call Exec_Method( CtrlEntID, "EXECUTESCRIPT", Script, FALSE$ )
```

See Also

WEBVIEW POSTJSONMESSAGE method, WEBVIEW POSTTEXTMESSAGE method, WEBVIEW WEBSCRIPTRESULT event, WEBVIEW WEBSYNCSCRIPTRESULT event.

FORWARD method

Description

Navigates the WEBVIEW object to the next page in the navigation history.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "FORWARD" )
```

Parameters

N/a.

Returns

"1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

Remarks

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2 GoForward method on the Microsoft website.

Example

```
// Example: Navigate to the next page in the WEBVIEW object's
// navigation history

If Get_Property( CtrlEntID, "CANGOFORWARD" ) Then
   Call Exec_Method( CtrlEntID, "FORWARD" )
End
```

See Also

WEBVIEW CANGOBACK property, WEBVIEW CANGOFORWARD property, WEBVIEW HISTORY property, WEBVIEW BACK method, WEBVIEW CLEARBROWSINGDATA method, WEBVIEW WEBDATACLEARED event, WEBVIEW WEBHISTORYCHANGED event.

GETCOOKIES method

Description

This method returns an array of cookies from the WEBVIEW object that match a specified URI.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "GETCOOKIES", URI )
```

Parameters

Name	Required	Description
URI	Yes	Specifies the URI to use for matching the cookies. If "*" then all cookies under the same profile are returned.

Returns

Returns an @fm-delimited array of matching cookies. Each cookie has the following @vm-delimited structure:

```
<0,1> Name
<0,2> Value
<0,3> Domain
<0,4> Path
<0,5> Expires     (Internal DT format)
<0,6> Secure      (TRUE$/FALSE$)
<0,7> HTTPOnly    (TRUE$/FALSE$)
<0,8> SameSite    ("None","Lax","Strict")
<0,9> SessionOnly (TRUE$/FALSE$)
```

Remarks

Equates for use with the GETCOOKIES method can be found in the PS_WEBVIEW_EQUATES insert record.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2CookieManager GetCookies method on the Microsoft website.

Example

```
// Example: Get all cookies from the Revelation web site

RevURI     = "https://www.revelation.com"
RevCookies = Exec_Method( CtrlEntID, "GETCOOKIES", RevURI )
```

See Also

WEBVIEW DELETECOOKIES method, WEBVIEW SETCOOKIE method.

MAPHOSTNAMETOFOLDER method

Description

This method creates a mapping between a virtual host name and a local folder path to make content in that folder available to via that host name.

Due to security issues, the WEBVIEW object may refuse to load local content such as images using the "file//" protocol. In this case it is possible to map the folder containing the local content to a "virtual host name", and then refer to the content using that hostname and the normal "http://" or https:// protocol instead.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "MAPHOSTNAMETOFOLDER", HostName, |
                                                 FolderPath, AccessType )
```

Parameters

Name	Required	Description
HostName	Yes	Specifies the virtual host name to set. After setting the mapping, documents loaded in the WEBVIEW object can use HTTP or HTTPS URLs at the specified host name to access files in the local folder specified by the FolderPath parameter.
FolderPath	Yes	Specifies the local folder path to map to. Both absolute and relative paths are supported for folderPath. Relative paths are interpreted as relative to the folder where OpenInsight.exe is located. This parameter must not exceed the Windows MAX_PATH limit (260 characters).

AccessType	Yes	Specifies the level of access to resources under the virtual host from other sites. Can be one of the following values:
		
		`0 : Deny`
		`1 : Allow`
		`2 : DenyCORS`
		
		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 "0" (Deny).
		
		Cross-origin resource access types are documented on the Microsoft website under the topic:
		
		"CoreWebView2HostResourceAccessKind"

Returns

Returns "1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

Remarks

Equates for use with the AccessType parameter can be found in the PS_WEBVIEW_EQUATES insert record.

For more information on this topic please refer to the Windows WebView2 documentation regarding the following topics on the Microsoft website:

- ICoreWebView2_3 SetVirtualHostNameToFolderMapping method
- CoreWebView2HostResourceAccessKind Enum

Example

```
// Example: Map the "c:\web-images" folder to the virtual host
//          "myAppImages" so it can be used with the https
//          protocol

$Insert PS_WebView_Equates

HostName    = "myAppImages"
LocalFolder = "c:\web-images"
AccessType  = WBV_MH2F_ACCESS_TYPE_DENY$

SuccessFlag = Exec_Method( CtrlEntID, "MAPHOSTTOFOLDERNAME", |
                           HostName, LocalFolder, AccessType )
```

See Also

WEBVIEW UNMAPHOSTNAME method.

NAVIGATE method

Description

Navigates the top-level document in the WEBVIEW object to the specified URI.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "NAVIGATE", URI, Method, Headers, |
                           Content )
```

Parameters

Name	Required	Description
URI	Yes	The location to navigate to.
Method	No	HTTP method to use (GET,POST,DELETE etc). Defaults to GET.
Headers	No	A list of HTTP headers to send to the server in the usual format: `<headerName> "=" <headerValue>` Each header should be delimited with CRLF.
Content	No	If making a HTTP POST request this parameter contains the data to send to the server.

Returns

Returns "1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

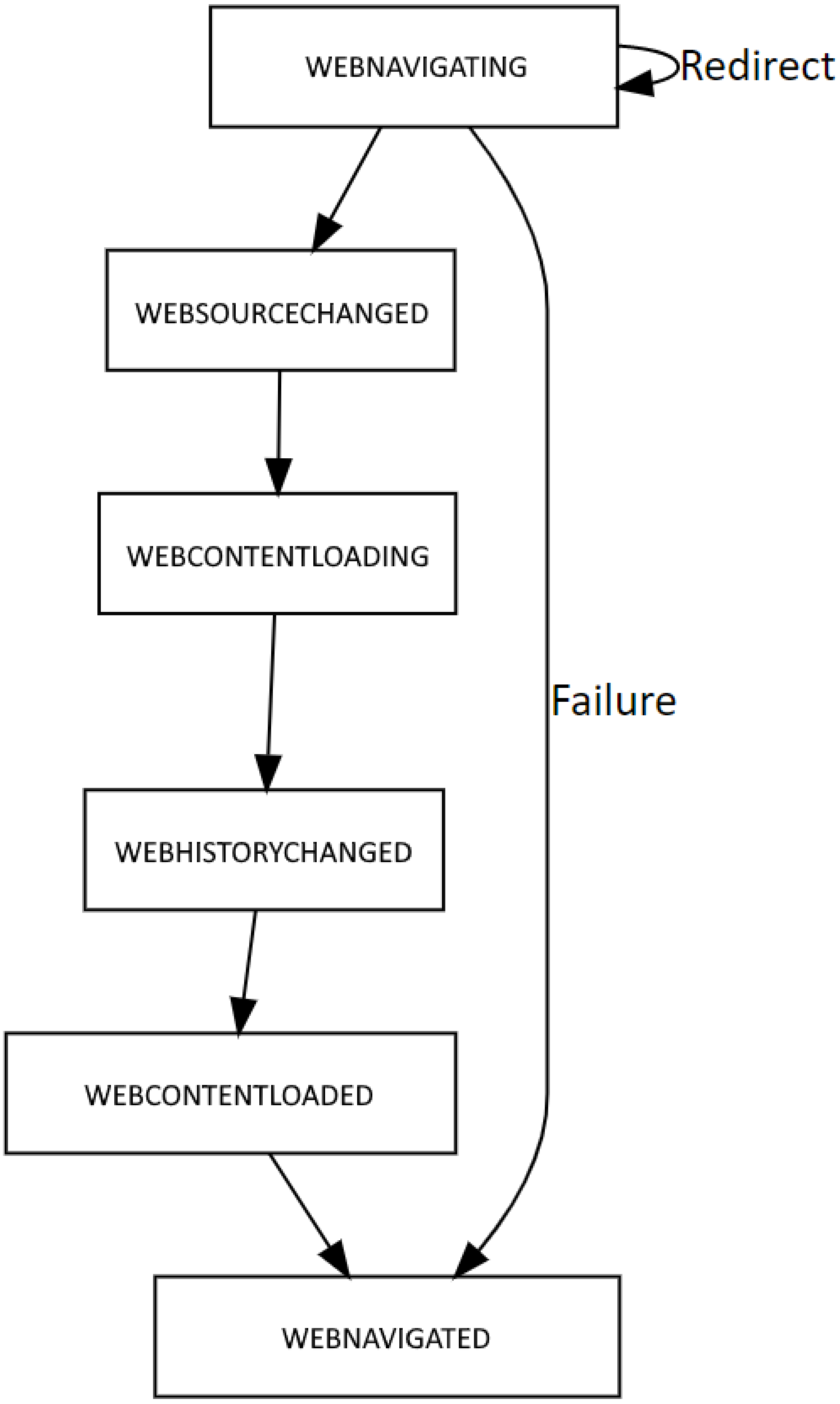
Remarks

Navigation is always an asynchronous operation, and it triggers a series of events in the WEBVIEW object as the content is loaded:

- WEBNAVIGATING
- WEBSOURCECHANGED
- WEBCONTENTLOADING
- WEBHISTORYCHANGED
- WEBCONTENTLOADED
- WEBNAVIGATED

(The diagram below shows the sequence in which these events are fired).

Each navigation attempt is assigned a unique ID that can be used to track its progress. This ID is passed to the beginning WEBNAVIGATING event and can be used from that point onwards.


WEBNAVIGATING
Redirect
WEBSOURCECHANGED
WEBCONTENTLOADING
Failure
WEBHISTORYCHANGED
WEBCONTENTLOADED
WEBNAVIGATED

For more information on WEBVIEW navigation please refer to the Windows WebView2 documentation on the Microsoft website regarding the following topics:

- ICoreWebView2 Navigate method.
- ICoreWebView2 NavigateWithWebResourceRequest method

Example

```
// Example: Navigate to a web site

SuccessFlag = Exec_Method( CtrlEntID, "NAVIGATE", "https://www.revelation.com" )


// Example: Send data to a web-server using the POST method

$Insert RTI_Text_Equates

URI     = "https://www.example.com/cgi-bin/oecgi.exe/set_name"

Content = "FNAME=John&SNAME=Doe"

Headers =           "Content-Type=application/x-www-form-urlencoded"
Headers := CRLF$ : "Content-Length=" : bLen( Content )
Headers := CRLF$ : "Content-Language=en-US"
Headers := CRLF$ : "Charset=utf-8"

// etc...

SuccessFlag = Exec_Method( CtrlEntID, "NAVIGATE", URI, "POST", Headers, Content )
```

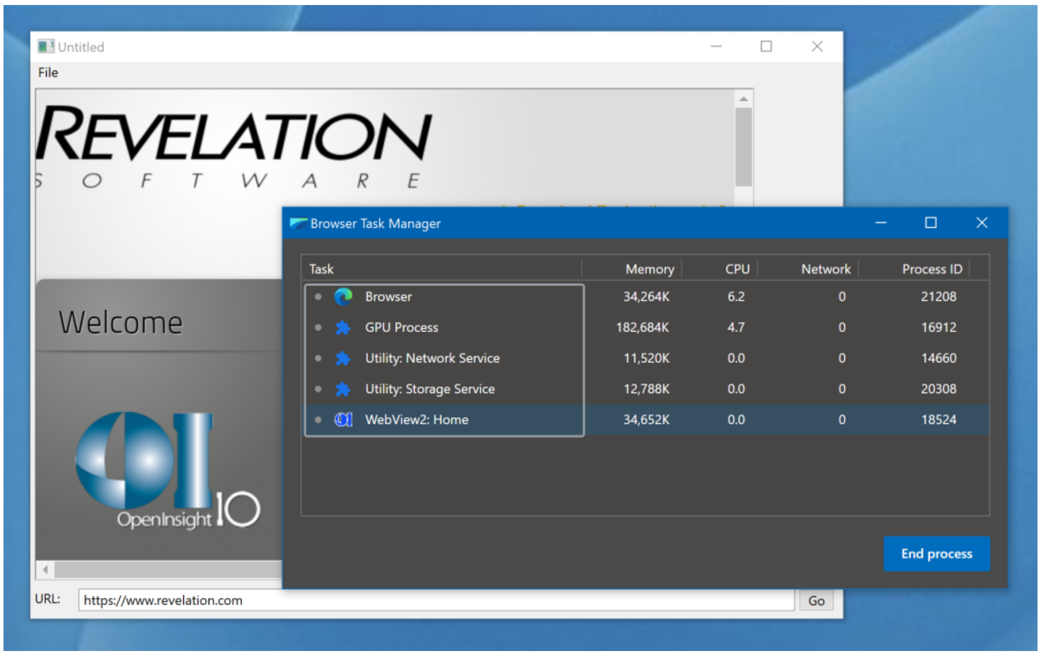
See Also

WEBVIEW URI property, WEBVIEW RELOAD method, WEBVIEW SETHTML method, WEBVIEW STOP method, WEBVIEW WEBNAVIGATING event, WEBVIEW WEBRESOURCECHANGED event, WEBVIEW WEBCONTENTLOADING event, WEBVIEW WEBHISTORYCHANGED event, WEBVIEW WEBCONTENTLOADED event, WEBVIEW WEBNAVIGATED event.

OPENBROWSERTASKMANAGER method

Description

Opens the browser task manager window for the WEBVIEW object.



Task	Memory	CPU	Network	Process ID
Browser	34,264K	6.2	0	21208
GPU Process	182,684K	4.7	0	16912
Utility: Network Service	11,520K	0.0	0	14660
Utility: Storage Service	12,788K	0.0	0	20308
WebView2: Home	34,652K	0.0	0	18524

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "OPENBROWSERTASKMANAGER" )
```

Parameters

N/a.

Returns

"1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

Remarks

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2_6 OpenTaskManagerWindow method on the Microsoft website.

Example

```
// Example: Open the Browser Task Manage window

Call Exec_Method( CtrlEntID, "OPENBROWSERTASKMANAGER" )
```

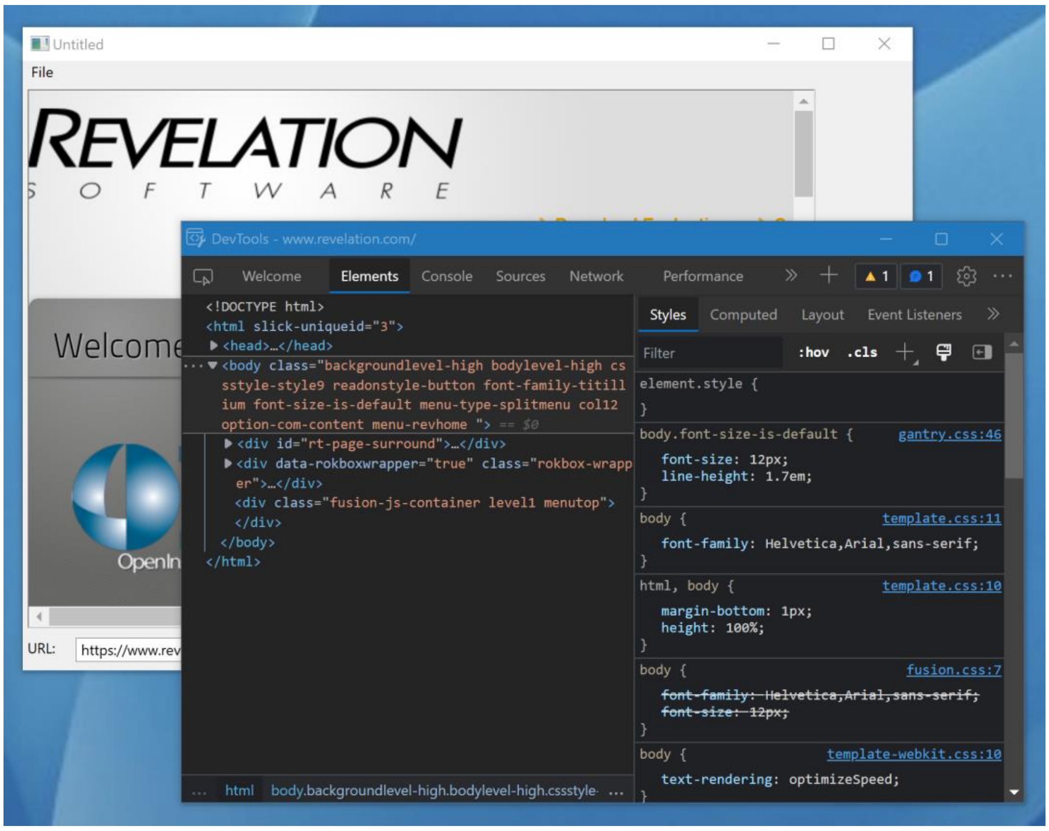
See Also

N/a.

OPENDEVTOOLS method

Description

Opens the DevTools window for the current document in the WEBVIEW object.



```
<!DOCTYPE html>
<html slick-uniqueid="3">
 ▶<head>…</head>
···▼<body class="backgroundlevel-high bodylevel-high cs
    sstyle-style9 readonstyle-button font-family-titill
    ium font-size-is-default menu-type-splitmenu col12
    option-com-content menu-revhome "> == $0
    ▶<div id="rt-page-surround">…</div>
    ▶<div data-rokboxwrapper="true" class="rokbox-wrapp
      er">…</div>
      <div class="fusion-js-container level1 menutop">
      </div>
    </body>
</html>
```

```
element.style {
}
body.font-size-is-default {                gantry.css:46
   font-size: 12px;
   line-height: 1.7em;
}
body {                                   template.css:11
   font-family: Helvetica,Arial,sans-serif;
}
html, body {                             template.css:10
   margin-bottom: 1px;
   height: 100%;
}
body {                                      fusion.css:7
   ~~font-family: Helvetica,Arial,sans-serif;~~
   ~~font-size: 12px;~~
}
body {                            template-webkit.css:10
   text-rendering: optimizeSpeed;
}
```

… html body.backgroundlevel-high.bodylevel-high.cssstyle- …

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "OPENDEVTOOLS" )
```

Parameters

N/a.

Returns

"1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

Remarks
For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2 OpenDevToolsWindow method on the Microsoft website.

Example
```
// Example: Open the DevTools window

Call Exec_Method( CtrlEntID, "OPENDEVTOOLS" )
```

See Also
N/a.

PASTE method

Description

Pastes the clipboard contents at the insertion point (replaces current selection) in the WEBVIEW object.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "PASTE" )
```

Parameters

N/a.

Returns

"1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

Remarks

The WEBVIEW PASTE method uses the EXECUTESCRIPT method to call the underlying JavaScript `document.execCommand( "paste" )` method.

For more information on the JavaScript `document.execCommand` method please refer to the documentation on the Mozilla Developer website at:

`https://developer.mozilla.org/en-US/docs/Web/API/Document/execCommand`

Example

```
// Example: Paste the current selection

Call Exec_Method( CtrlEntID, "PASTE" )
```

See Also

WEBVIEW COPY method, WEBVIEW CUT method, WEBVIEW EXECUTESCRIPT method, WEBVIEW REDO method, WEBVIEW UNDO method.

POSTJSONMESSAGE method

Description

Posts a JSON-formatted WebMessage to the top-level document in the WEBVIEW object.

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

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "POSTJSONMESSAGE", JsonMessage )
```

Parameters

Name	Required	Description
JsonMessage	Yes	A string containing a JSON formatted message.

Returns

"1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

Remarks

The EnableWebMessages field in the WEBVIEW SETTINGS property must be set to TRUE$ for this method to work.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2 PostWebMessageAsJson method on the Microsoft website.

Example

```
// Example JavaScript for the HTML document to listen for the
// "message" event on the "window.chrome.webview" object.
//
// When we get the message we examine the passed event object's
// data looking for an object called "msgText".  When we find
// it we use a simple alert() message to display the text.

// This should be placed in the HTML document...
<script>

    window.chrome.webview.addEventListener( "message", objEvent => {
         alert( objEvent.data.msgText );
    });

</script>
```

```
// Example: Post a message to the document using a JSON formatted object

JsonMessage = '{ "msgText" : "Test Message from OI" }'

Call Exec_Method( CtrlEntID, "POSTJSONMESSAGE", JsonMessage )

// "Test Message from OI" will now be shown in an alert message
// in the WEBVIEW object.
```

See Also

WEBVIEW SETTINGS property, WEBVIEW EXECUTESCRIPT method, WEBVIEW POSTTEXTMESSAGE method, WEBVIEW WEBMESSAGE event.

POSTTEXTMESSAGE method

Description

Posts a text WebMessage to the top-level document in the WEBVIEW object.

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

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "POSTTEXTMESSAGE", TextMessage )
```

Parameters

Name	Required	Description
TextMessage	Yes	A string containing the message.

Returns

"1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

Remarks

The EnableWebMessages field in the WEBVIEW SETTINGS property must be set to TRUE$ for this method to work.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2 PostWebMessageAsString method on the Microsoft website.

Example

```
// Example JavaScript for the HTML document to listen for the
// "message" event on the "window.chrome.webview" object.
//
// When we get the message we examine the passed event object's
// data property and use a simple alert() message to display it.

// This should be placed in the HTML document...
<script>

    window.chrome.webview.addEventListener( "message", objEvent => {
         alert( objEvent.data );
    });

</script>
```

```
// Example: Post a message to the document using a a simple text string

Call Exec_Method( CtrlEntID, "POSTTEXTMESSAGE", "Test Message from OI"  )

// "Test Message from OI" will now be shown in an alert message
// in the WEBVIEW object.
```

See Also

WEBVIEW SETTINGS property, WEBVIEW EXECUTESCRIPT method, WEBVIEW POSTTEXTMESSAGE method, WEBVIEW WEBMESSAGE event.

PRINT method

Description
Opens a dialog box to print the current document in the WEBVIEW object.

Syntax
```
SuccessFlag = Exec_Method( CtrlEntID, "PRINT" )
```

Parameters
N/a.

Returns
"1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

Remarks
The WEBVIEW PRINT method uses the EXECUTESCRIPT method to call the underlying JavaScript `window.print()` method to print the document.

For more information on the JavaScript `window.print()` method please refer to the documentation on the Mozilla Developer website at:

    https://developer.mozilla.org/en-US/docs/Web/API/Window/print

Example
```
// Example: Print the current document

Call Exec_Method( CtrlEntID, "PRINT" )
```

See Also
WEBVIEW EXECUTESCRIPT method, WEBVIEW PRINTTOPDF method.

PRINTTOPDF method

Description

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

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID,     |
                           "PRINTTOPDF",  |
                           FileName,      |
                           PrintSettings )
```

Parameters

Name	Required	Description
FileName	Yes	Specifies the absolute name and path of the PDF file to write to. If the path points to an existing file, the file will be overwritten.

PrintSettings	No	Specifies an @fm-delimited array of settings to use when printing the PDF:
		`<1>  PrintHeaderAndFooter`
		`<2>  PrintSelectedOnly`
		`<3>  PrintBackgrounds`
		`<4>  HeaderTitle`
		`<5>  FooterURI`
		`<6>  Landscape`
		`<7>  PageWidth`
		`<8>  PageHeight`
		`<9>  LeftMargin`
		`<10> TopMargin`
		`<11> RightMargin`
		`<12> BottomMargin`
		`<13> ScaleFactor`
		See “Remarks” below for more details on these settings.

Returns

"1" (TRUE$) if the script was added successfully, "0" (FALSE$) otherwise.

Remarks

As this is an asynchronous operation the results are returned via the WEBPDFPRINTED event.

The PrintSettings parameter has the following options:

Name	Description
PrinterHeaderAndFooter	If TRUE$ then print the header and footer. 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. Defaults to FALSE$.
PrintSelectedOnly	If TRUE$ then only the current selection of HTML in the document is printed. Defaults to FALSE$.
PrintBackgrounds	If TRUE$ then background colors and images are printed. Defaults to FALSE$.
HeaderTitle	Specifies the text for the title to put in the document header. Use the string "<no-title>" to prevent a title from being printed. Defaults to the document title.
FooterURI	Specifies the text for the URI to put in the document footer. Use the string "<no-uri>" to prevent a URI from being printed. Defaults to the document URI.
Landscape	If TRUE$ then print the document using Landscape orientation. Defaults to FALSE$.
PageWidth	Specifies the width of the page in inches. Must be greater than 0. Defaults to 8.5 inches.
PageHeight	Specifies the height of the page in inches. Must be greater than 0. Defaults to 11 inches.
LeftMargin	Specifies the width of the left margin in inches. The default is 0.4 inches (1 cm).
TopMargin	Specifies the height of the top margin in inches. The default is 0.4 inches (1 cm).
RightMargin	Specifies the width of the right margin in inches. The default is 0.4 inches (1 cm).
BottomMargin	Specifies the height of the bottom margin in inches. The default is 0.4 inches (1 cm).
ScaleFactor	Specifies the scaling applied to the document when printing. This is a numeric value between 0.1 (10%) and 2.0 (200%). Defaults to 1.0.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2_7 PrintToPdf method, and the ICoreWebView2PrintSettings interface on the Microsoft website.

Example

```
// Example: Ask the user for an PDF file name and save
// the contents of the WEBVIEW object to it.

$Insert PS_ChooseFile_Equates
$Insert PS_FileSystem_Equates
$Insert PS_WebView_Equates
$Insert Logical

Equ INVALID_CHARS$ to '*."/\[]:;|,'
Equ WEBVIEW_CTRL$  to @Window : ".WBV_BROWSER"

// Show PDF and All Files ...
Filter =  "PDF Files(*.pdf)/*.pdf/"
Filter := "All Files (*.*)/*.*/"

Flags = 0;
Flags = BitOr( Flags, OFN_NOCHANGEDIR$ )
Flags = BitOr( Flags, OFN_PATHMUSTEXIST$ )
Flags = BitOr( Flags, OFN_OVERWRITEPROMPT$ )

DfltName = Get_Property( WEBVIEW_CTRL$, "DOCUMENTTITLE" )

Spaces = Space( Len( INVALID_CHARS$ ) )
Convert INVALID_CHARS$ To Spaces In DfltName

// Default to "My Documents"
StartDir = Exec_Method( "FILESYSYEM", "GETSPECIALDIR", |
                        PS_GSD_PERSONAL$ )

CFOpt = ""
CFOpt<CHFILE_POS_MODE$>         = CHFILE_MODE_SAVEAS$
CFOpt<CHFILE_POS_FILTERSTRING$> = Filter
CFOpt<CHFILE_POS_FILTERINDEX$>  = 1
CFOpt<CHFILE_POS_DFLTNAME$>     = DfltName : ".pdf"
CFOpt<CHFILE_POS_FLAGS$>        = Flags
CFOpt<CHFILE_POS_INITDIR$>      = StartDir
CFOpt<CHFILE_POS_TITLE$>        = "Print WebPage As"

PdfName = Exec_Method( "SYSTEM", "CHOOSEFILE", @Window, CFOpt )
If BLen( PDFName ) Then

   PrintSettings = ""
   PrintSettings<WBV_P2PDFSET_POS_HDRANDFTR$> = TRUE$
   PrintSettings<WBV_P2PDFSET_POS_FTRURI$>    = WBV_P2PDFSET_NO_FTRURI$

   IsOK = Exec_Method( WEBVIEW_CTRL$, "PRINTTOPDF", PDFName, |
                       PrintSettings )
   If IsOK Then
      // Use the WEBPDFPRINTED event to see if the print
      // operation was successful.
   End
End
```

See Also

WEBVIEW PRINT method, WEBVIEW WEBPDFPRINTED event.

REDO method

Description

Discards the results of the last Undo command performed on the content in the WEBVIEW object.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "REDO" )
```

Parameters

N/a.

Returns

"1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

Remarks

The WEBVIEW REDO method uses the EXECUTESCRIPT method to call the underlying JavaScript `document.execCommand( "redo" )` method.

For more information on the JavaScript `document.execCommand` method please refer to the documentation on the Mozilla Developer website at:

https://developer.mozilla.org/en-US/docs/Web/API/Document/execCommand

Example

```
// Example: Perform a Redo operation on the current item in the WEBVIEW control

Call Exec_Method( CtrlEntID, "REDO" )
```

See Also

WEBVIEW COPY method, WEBVIEW CUT method, WEBVIEW EXECUTESCRIPT method, WEBVIEW PASTE method, WEBVIEW UNDO method.

RELOAD method

Description

Reloads the current top-level document in the WEBVIEW object.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "RELOAD" )
```

Parameters

N/a.

Returns

"1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

Remarks

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

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2 Reload method on the Microsoft website.

Example

```
// Example: Reload the document in the WEBVIEW object

Call Exec_Method( CtrlEntID, "RELOAD" )
```

See Also

WEBVIEW URI property, WEBVIEW NAVIGATE method.

REMOVECDPEVENT method

Description

Stops the WEBVIEW object from receiving notifications for a specified Chrome DevTools Protocol (CDP) event. The Chrome DevTools Protocol allows for tools to instrument, inspect, debug and profile Chromium, Chrome and other Blink-based browsers.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID,         |
                           "REMOVECDPEVENT",  |
                           EventName )
```

Parameters

Name	Required	Description
EventName	Yes	Specifies the name of the CDP event to stop listening for. This is case-sensitive.

Returns

"1" (TRUE$) if the event was removed successfully, "0" (FALSE$) otherwise.

Remarks

More information on the CDP can be found on here along with the events that are supported:

```
https://chromedevtools.github.io/devtools-protocol/
```

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2 GetDevToolsProtocolEventReceiver method on the Microsoft website.

Example

```
// Example: Stop listening for the "Log.entryAdded" CDP event

IsOk = Exec_Method( CtrlEntID, "REMOVECDPEVENT", "Log.entryAdded" )
```

See Also

WEBVIEW EXECUTECDPMETHOD method, WEBVIEW ADDCDPEVENT method, WEBVIEW WEBCDPEVENT event, WEBVIEW WEBCDPMETHODRESULT event.

REMOVEINITSCRIPT method

Description

Removes the specified JavaScript previously added to the WEBVIEW object with the ADDINTISCRIPT method.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "REMOVEINITSCRIPT", ScriptID )
```

Parameters

Name	Required	Description
ScriptID	Yes	Specifies the ID of JavaScript to remove. This ID was returned via the WEBINITSCRIPTADDED event.

Returns

"1" (TRUE$) if the script was removed successfully, "0" (FALSE$) otherwise.

Remarks

The script is added asynchronously by the ADDINITSCRIPT method and given a unique identifier. The actual result of the method and the identifier are returned in the WEBINITSCRIPTADDED event. The identifier may be saved for later use with the REMOVEINITSCRIPT method.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2 RemoveScriptToExecuteOnDocumentCreated method on the Microsoft website.

Example

```
// Example: Remove a script added earlier with the ADDINITSCRIPT method.

IsOK = Exec_Method( CtrlEntID, "REMOVEINITSCRIPT", ScriptID )
```

See Also

WEBVIEW ADDINITSCRIPT method, WEBVIEW WEBINITSCRIPTADDED event.

RESUME method

Description

Resumes a suspended WEBVIEW object.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "RESUME" )
```

Parameters

N/a.

Returns

"1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

Remarks

Note that when a suspended hidden WEBVIEW object is made visible again the browser renderer is automatically resumed – there is no need to explicitly call the RESUME method.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2_3 Resume method on the Microsoft website.

Example

```
// Resume a suspended WEBVIEW object

If Get_Property( CtrlEntID, "SUSPENDED" ) Then
   Call Exec_Method( CtrlEntID, "RESUME" )
End
```

See Also

WEBVIEW SUSPENDED property, WEBVIEW SUSPEND method, WEBVIEW SUSPENDED event.

SAVETOFILE method

Description

Saves the contents of the specified WEBVIEW control to disk using the MHTML format.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "SAVETOFILE", FileName )
```

Parameters

Name	Required	Description
FileName	Yes	Name and path to save the file as. The file will be saved in the MHTML format.

Returns

"1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

Remarks

This method is executed asynchronously and the actual result of the save operation is returned in the WEBSAVEDTOFILE event.

For more information on this topic please refer to the “Page.captureSnapshot” method in the Chrome DevTools Protocol website here:

https://chromedevtools.github.io/devtools-protocol/tot/Page/#method-captureSnapshot

Example

```
// Example: Ask the user for an MHTML file name and save
// the contents of the WEBVIEW object to it.

$Insert PS_ChooseFile_Equates
$Insert PS_FileSystem_Equates

Equ INVALID_CHARS$ to '*."/\[]:;|,'
Equ WEBVIEW_CTRL$  to @Window : ".WBV_BROWSER"

// Show MHTML and All Files ...
Filter =  "MIME HTML Files(*.mhtml)/*.mhtml/"
Filter := "All Files (*.*)/*.*/"

Flags = 0;
Flags = BitOr( Flags, OFN_NOCHANGEDIR$ )
Flags = BitOr( Flags, OFN_PATHMUSTEXIST$ )
Flags = BitOr( Flags, OFN_OVERWRITEPROMPT$ )

DfltName = Get_Property( WEBVIEW_CTRL$, "DOCUMENTTITLE" )

Spaces = Space( Len( INVALID_CHARS$ ) )
Convert INVALID_CHARS$ To Spaces In DfltName

// Default to "My Documents"
StartDir = Exec_Method( "FILESYSYEM", "GETSPECIALDIR", |
                        PS_GSD_PERSONAL$ )

CFOpt = ""
CFOpt<CHFILE_POS_MODE$>         = CHFILE_MODE_SAVEAS$
CFOpt<CHFILE_POS_FILTERSTRING$> = Filter
CFOpt<CHFILE_POS_FILTERINDEX$>  = 1
CFOpt<CHFILE_POS_DFLTNAME$>     = DfltName : ".mhtml"
CFOpt<CHFILE_POS_FLAGS$>        = Flags
CFOpt<CHFILE_POS_INITDIR$>      = StartDir
CFOpt<CHFILE_POS_TITLE$>        = "Save WebPage As"

SaveName = Exec_Method( "SYSTEM", "CHOOSEFILE", @Window, CFOpt )
If BLen( SaveName ) Then

   If Exec_Method( WEBVIEW_CTRL$, "SAVETOFILE", SaveName ) Then
      // Use the WEBSAVEDTOFILE event to see if the save
      // operation was successful.
   End
End
```

See Also

WEBVIEW WEBSAVEDTOFILE event.

SELECTALL method

Description

Selects all the content of the editable region of the WEBVIEW object.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "SELECTALL" )
```

Parameters

N/a.

Returns

"1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

Remarks

The WEBVIEW SELECTALL method uses the EXECUTESCRIPT method to call the underlying JavaScript `document.execCommand( "selectAll" )` method.

For more information on the JavaScript `document.execCommand` method please refer to the documentation on the Mozilla Developer website at:

https://developer.mozilla.org/en-US/docs/Web/API/Document/execCommand

Example

```
// Example: Select all of the text

Call Exec_Method( CtrlEntID, "SELECTALL" )
```

See Also

WEBVIEW COPY method, WEBVIEW CUT method, WEBVIEW EXECUTESCRIPT method, WEBVIEW PASTE method.

SETCOOKIE method

Description

Adds or updates a cookie in the WEBVIEW object with the specified cookie details.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "SETCOOKIE", Cookie )
```

Parameters

Name	Required	Description

Cookie	Yes	Cookie details to set. This is an @vm-delimited array with the following structure:
		
		`<0,1> Name      (Required)`
		`<0,2> Value`
		`<0,3> Domain    (Required)`
		`<0,4> Path`
		`<0,5> Expires   (Internal DT format)`
		`<0,6> Secure    (TRUE$/FALSE$)`
		`<0,7> HTTPOnly  (TRUE$,FALSE$)`
		`<0,8> SameSite  ("None","Lax", or "Strict"`
		`                 -  Defaults to "Lax")`

Returns

"1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

Remarks

Cookies are session cookies and will not be persistent if Expires is set to -1.0 (Any negative value set other than -1.0 is treated as -1.0).

Cookies may also be set via the RequestHeaders parameter in the WEBVIEW NAVIGATE method.

Equates for use with the SETCOOKIE method can be found in the PS_WEBVIEW_EQUATES insert record.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2CookieManager CreateCookie and AddOrUpdateCookie methods, and the ICoreWebView2Cookie interface on the Microsoft website.

Information on cookies and the Set-Cookie HTTP header can be found on the Mozilla developer site at:

```
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie
```

Example

```
// Example set a secure cookie for the Examples.Com domain
$Insert PS_WebView_Equates

equ ONE_DAY_SECS$ to 84600

Cookie = ""
Cookie<WBV_COOKIE_POS_NAME$>     = "UserID"
Cookie<WBV_COOKIE_POS_VALUE$>    = "AgentC"
Cookie<WBV_COOKIE_POS_DOMAIN$>   = "example.com"

// Expires 24 hours from now
Expires = ( Date() + 1 ) + ( Time() / ONE_DAY_SECS$ )
Cookie<WBV_COOKIE_POS_EXPIRES$> =  Expires

Cookie<WBV_COOKIE_POS_SECURE$>   = TRUE$

IsOK = Exec_Method( CtrlEntID, "SETCOOKIE", Cookie )
```

See Also

WEBVIEW DELETECOOKIES method, WEBVIEW GETCOOKIES method, WEBVIEW NAVIGATE method.

SETHTML method

Description

Loads a string containing HTML content into the WEBVIEW object as a top-level document.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "SETHTML", HTMLContent )
```

Parameters

Name	Required	Description
HTMLContent	Yes	Specifies the HTML string to load into the WEBVIEW object. This parameter may not be larger than 2MB (2 * 1024 * 1024 bytes) in total size. The origin of the new page is "about:blank".

Returns

"1" (TRUE$) if the navigation was started successfully, "0" (FALSE$) otherwise.

Remarks

The normal sequence of navigation events fire when using this method. The URI passed to the navigation events is a Base64-encoded representation of the HTMLContent argument.

Note that the WEBVIEW object imposes stricter security measures on content loaded via this method. For example, links in the content that represent "file://" URIs are not allowed. A better way of loading local content is to use a HTTPSERVER control and navigate to that instead.

For more information on this method please refer to the Windows WebView2 documentation regarding the ICoreWebView2 NavigateToString method on the Microsoft website.

Example

```
// Example: Read an html file from disk and load it into the
// WBV_BROWSER WEBVIEW control

OSRead HtmlFile From ".\html\customer_entry.html" Then

   Call Exec_Method( @Window : ".WBV_BROWSER", "SETHTML", HtmlFile )

End
```

See Also

WEBVIEW NAVIGATE method.

SETPERMISSION method

Description

Allows or denies access to privileged resources from the content in the WEBVIEW object.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "SETPERMISSION", URI, RequestType, |
                           UserInitiated, AllowRequest )
```

Parameters

Name	Required	Description
URI	Yes	Specifies the source of the permission request.

RequestType	Yes	Specifies the type of resource that access was requested for. Can be one of the following:
		`0 : Unknown`
		`1 : Microphone`
		`2 : Camera`
		`3 : Geolocation`
		`4 : Notification`
		`5 : Other Sensor`
		`6 : Read Clipboard`

UserInitiated	Yes	A boolean value that specifies if a user gesture initiated the request.
AllowRequest	Yes	Set to TRUE$ to allow access to be granted to the requested resourced, or FALSE$ to deny it.

Returns

"1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

Remarks

This method can only be used within the context of a WEBPERMISSIONREQUEST event. The first three parameters passed to the method should match the same parameters passed to the WEBPERMISSIONREQUEST.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2 add_PermissionRequested method on the Microsoft website.

Example

See the WEBPERMISSIONREQUEST event for an example of using the SETPERMISSION method.

See Also

WEBVIEW SETPERMISSION method, WEBVIEW WEBPERMISSIONREQUEST event.

STOP method

Description

Stops all navigations and pending resource fetches (executing scripts are not stopped however).

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "STOP" )
```

Parameters

N/a.

Returns

"1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

Remarks

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2 Stop method on the Microsoft website.

Example

```
// Example: Stop a navigating WEBVIEW object

Call Exec_Method( CtrlEntID, "STOP" )
```

See Also

WEBVIEW URI property, WEBVIEW RELOAD method, WEBVIEW NAVIGATE method, WEBVIEW WEBNAVIGATING event.

SUSPEND method

Description

Suspends a WEBVIEW object, forcing it to consume less memory and resources, in effect putting it to sleep. Can only be called if the WEBVIEW object is NOT visible.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "SUSPEND" )
```

Parameters

N/a.

Returns

"1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

Remarks

Suspending is similar to putting a tab to sleep in the Edge browser. Suspending pauses 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 this method is "best efforts" only - there may be cases where the WEBVIEW object continues to run in the background. Use the WEBSUSPENDED event to listen for the actual result of the SUSPEND call.

When a suspended hidden WEBVIEW object is made visible again the browser renderer is automatically resumed – there is no need to explicitly call the RESUME method.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2_3 TrySuspend method on the Microsoft website.

Example

```
// Example: Hide the WEBVIEW object and then suspend it. When we
// make the WEBVIEW visible again it will be automatically resumed.
//
// Note that hiding the WEBVIEW also hides the browser rendering
// component (See the BROWSERVISIBLE property) so it may now be
// suspended

$Insert MSWin_ShowWindow_Equates

Call Set_Property_Only( CtrlEntID, "VISIBLE", SW_HIDE$ )

Call Exec_Method( CtrlEntID, "SUSPEND" )

// We could listen for the success of the SUSPEND call via the
// WEBSUSPENDED event if we wished

...

// Showing the WEBVIEW object again automatically resumes it
Call Set_Property_Only( CtrlEntID, "VISIBLE", SW_NORMAL$ )
```

See Also

Common GUI VISIBLE property, WEBVIEW BROWSERVISIBLE property, WEBVIEW SUSPENDED property, WEBVIEW RESUME method, WEBVIEW WEBSUSPENDED event.

UNDO method

Description

Undoes the last command performed on the content in the WEBVIEW object.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "UNDO" )
```

Parameters

N/a.

Returns

"1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

Remarks

The WEBVIEW UNDO method uses the EXECUTESCRIPT method to call the underlying JavaScript `document.execCommand( "undo" )` method.

For more information on the JavaScript `document.execCommand` method please refer to the documentation on the Mozilla Developer website at:

https://developer.mozilla.org/en-US/docs/Web/API/Document/execCommand

Example

```
// Example: Perform an Undo operation on the current item in the WEBVIEW object

Call Exec_Method( CtrlEntID, "UNDO" )
```

See Also

WEBVIEW COPY method, WEBVIEW CUT method, WEBVIEW EXECUTESCRIPT method, WEBVIEW PASTE method, WEBVIEW REDO method.

UNMAPHOSTNAME method

Description

Clears a host name mapping for local folder that was added by the MAPHOSTNAMETOFOLDER method.

Syntax

```
SuccessFlag = Exec_Method( CtrlEntID, "UNMAPHOSTNAME", HostName )
```

Parameters

Name	Required	Description
HostName	Yes	Specifies the virtual host name to remove.

Returns

Returns "1" (TRUE$) if the method was executed successfully, or "0" (FALSE$) otherwise.

Remarks

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2_3 ClearVirtualHostNameToFolderMapping method on the Microsoft website.

Example

```
// Example: Remove the “myAppImages” host name mapping

SuccessFlag = Exec_Method( CtrlEntID, "UNMAPHOSTNAME", "myAppImages" )
```

See Also

WEBVIEW MAPHOSTNAMETOFOLDER method.

WEBVIEW Events

The WEBVIEW object supports the following events:

Name	Description
WEBAUDIOCHANGED	Occurs when the "audio playing" status is changed.
WEBAUTHREQUEST	Occurs when the WEBVIEW object encounters an authentication request.
WEBCDPEVENT	Occurs when the WEBVIEW object receives notification that a tracked CDP event is fired.
WEBCDPMETHODRESULT	Returns the results of a previous asynchronous call to the EXECUTECDPMETHOD method.
WEBCLOSEWINDOW	Occurs when content inside the WEBVIEW object requests to close the window.
WEBCONTENTLOADED	Occurs when the initial HTML document has been parsed during navigation.
WEBCONTENTLOADING	Occurs when the WEBVIEW object begins loading new content.
WEBCONTEXTMENU	Occurs when the WEBVIEW object wishes to display a context menu.
WEBDATACLEARED	Returns the results of CLEARBROWSINGDATA method.
WEBHISTORYCHANGED	Occurs when a navigation updates the history of the WEBVIEW object.
WEBINITCONTEXTMENU	Occurs when the WEBVIEW object wishes to initialize a context menu.
WEBINITSCRIPTADDED	Returns the results of a call to the ADDINITSCRIPT method.
WEBMESSAGE	Occurs when the WEBVIEW object receives a WebMessage from the hosted content.
WEBMUTEDCHANGED	Occurs when the "audio muted" status is changed.
WEBNAVIGATED	Occurs when the WEBVIEW object has completely loaded.
WEBNAVIGATING	Occurs when the WEBVIEW object is navigating to a different URI.
WEBOPENWINDOW	Occurs when the WEBVIEW object attempts to open a new window.
WEBPDFPRINTED	Returns the results of a previous call to the PRINTTOPDF method.
WEBPERMISSIONREQUEST	Occurs when the content in the WEBVIEW object requests permission to access some privileged resources.
WEBSAVEDTOFILE	Returns the results of the SAVETOFILE method.
WEBSCRIPTRESULT	Returns the results of an asynchronous EXECUTESCRIPT method.
WEBSHOWDIALOG	Occurs when a the WEBVIEW object needs to display a custom dialog in response to a JavaScript dialog statement.
WEBSOURCECHANGED	Occurs when the WEBVIEW object's source property changes during navigation.

WEBSUSPENDED	Occurs when the SUSPEND method is executed and returns the results of the suspend operation.
WEBSTATUSTEXTCHANGED	Occurs when the WEBVIEW object is showing a status message, a URL, or an empty string.
WEBSYNCSCRIPTRESULT	System-level event to support the EXECUTESCRIPT method in synchronous mode.
WEBTITLECHANGED	Occurs when the title of the top-level document in the WEBVIEW object changes.
WEBVIEWCREATED	Occurs when the WEBVIEW object has been created and is ready to navigate.
WEBZOOMCHANGED	Occurs when the user changes the Zoom Factor via the mouse or keyboard.

The following Common GUI Object events are also supported:

- GOTFOCUS
- HELP
- LOSTFOCUS
- NOTES
- OMNIEVENT
- TIMER
- WINMSG

WEBAUDIOCHANGED event

Description

Occurs when the “audio playing” status is changed.

Syntax

```
bForward = WEBAUDIOCHANGED( CtrlEntID, CtrlClassID, AudioPlaying )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the WEBVIEW object receiving the event.
CtrlClassID	Type of object receiving the event (always "WEBVIEW").
AudioPlaying	Set to TRUE$ if the WEBVIEW object is playing audio (even if muted), or FALSE$ otherwise.

Returns

TRUE$ or FALSE$. If FALSE$, the program execution returns to the calling procedure. If TRUE$, the event processing goes to the next level.

Remarks

There is no system-level promoted event handler for the WEBAUDIOCHANGED event.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2_8 add_IsDocumentPlayingAudioChanged method on the Microsoft website.

Example

```
Function WEBAUDIOCHANGED( CtrlEntID, CtrlClassID, AudioPlaying )

  // Example: Display a message with the audio status

  If AudioPlaying Then
     MsgText = "Playing some tunes"
  End Else
     MsgText = "In Basic+, no one can hear you scream"
  End

  Call Msg( @Window, MsgRec )

Return TRUE$
```

See Also

WEBVIEW AUDIOPLAYING property, WEBVIEW MUTED property, WEBVIEW WEBMUTEDCHANGED event.

WEBAUTHREQUEST event

Description

Occurs when the WEBVIEW object encounters a Basic HTTP Authentication request (as described in https://developer.mozilla.org/docs/Web/HTTP/Authentication), an NTLM authentication or a Proxy Authentication request.

A WEBAUTHREQUEST event handler should provide a response via the AUTHENTICATE method with the appropriate credentials or cancel the request.

Syntax

```
bForward = WEBAUTHREQUEST( CtrlEntID, CtrlClassID, URI, Challenge )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the WEBVIEW object receiving the event.
CtrlClassID	Type of object receiving the event (always "WEBVIEW").
URI	URI of the request that triggered the authentication request.
Challenge	The authentication challenge string returned from the server.

Returns

TRUE$ or FALSE$. If FALSE$, the program execution returns to the calling procedure. If TRUE$, the event processing goes to the next level.

Remarks

A WEBAUTHREQUEST event will only be triggered when the AUTHENTICATIONMODE property is set to "Custom". If it is set to "Default" the WEBVIEW object displays a default authentication challenge dialog prompt to the user.

The WEBAUTHREQUEST event has a system-level promoted event handler that performs the following tasks:

- Executes the WEBAUTHREQUEST quick event handler (if defined) and checks the event status – if it returns anything other than FALSE$ the event is cancelled.
- Calls the AUTHENTICATE method to cancel the request.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2_10 add_BasicAuthenticationRequested method on the Microsoft website.

Example

```
Function WEBAUTHREQUEST( CtrlEntID, CtrlClassID, URI, Challenge )

  // Example: A WEBAUTHREQUEST handler that uses a dialog box to ask the user
  // for the credentials for the passed URL.  The AUTHENTICATE method is
  // executed  to return an answer to the server.

  // Assume we have a dialog box called GET_WEB_CREDENTIALS that takes
  // the passed URI and Challenge string returned from the server.

  DlgParams = URL : @fm : Challenge
  Credentials  = Dialog_Box( "GET_WEB_CREDENTIALS", @Window, DlgParams )

  If BLen( Credentials ) Then
     // Assume the dialog passed back the UserName and Password as
     // an @fm-delimited array

     UN = Credentials<1>
     PW = Credentials<2>

     Call Exec_Method( CtrlEntID, "AUTHENTICATE", UN, PW, FALSE$ )

  End Else
     // User Cancelled - stop the request
     Call Exec_Method( CtrlEntID, "AUTHENTICATE", "", "", TRUE$ )
  End

Return FALSE$
```

See Also

WEBVIEW AUTHENTICATIONMODE property, WEBVIEW AUTHENTICATE method.

WEBCDPEVENT event

Description

Occurs when the WEBVIEW object receives notification that a tracked CDP event is fired. CDP events can be tracked by using the ADDCDPEVENT method. The Chrome DevTools Protocol allows for tools to instrument, inspect, debug and profile Chromium, Chrome and other Blink-based browsers.

Syntax

```
bForward = WEBCDPEVENT( CtrlEntID, CtrlClassID, EventName, JsonEventInfo, |
                                                                SessionID )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the WEBVIEW object receiving the event.
CtrlClassID	Type of object receiving the event (always "WEBVIEW").
EventName	Name of the CDP event that has triggered the WEBCDPEVENT notification.
JsonEventInfo	A JSON object containing information about the CDP event. The contents of this argument depend on the CDP event and it may be null.
SessionID	The session ID of the target where the CDP event originates from. This parameter may be null if the event comes from the default session for the top page.

Returns

TRUE$ or FALSE$. If FALSE$, the program execution returns to the calling procedure. If TRUE$, the event processing goes to the next level.

Remarks

There is no system-level promoted event handler for the WEBCDPEVENT event.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2DevToolsProtocolEventReceiver add_DevToolsProtocolEventReceived method on the Microsoft website.

Example

```
// Example: Listen for "CDP Log" events from the WEBVIEW object. The
// CDP offers a logging service - to monitor log entries in OpenInsight
// we can do the following:
//
//    1) Listen for the "Log.entryAdded" event
//    2) Begin logging by using the CDP "Log.enable" method
//    3) Handle the new log entry in the WEBCDPEVENT event
//    4) Stop logging by using the CDP "Log.disable" method.

$Insert Logical

//  Listen for the "Log.entryAdded" event
IsOk = Exec_Method( CtrlEntID, "ADDCDPEVENT", "Log.entryAdded" )

// Begin logging by using the CDP "Log.enable" method
IsOK = Exec_Method( CtrlEntID, "EXECUTECDPMETHOD", "Log.enable", "", TRUE$ )

// In the WEBCDPEVENT we now receive notifications when a log entry
// is added along with a JSON object containing the details (see below )

...

// Stop logging by using the CDP "Log.disable" method
IsOK = Exec_Method( CtrlEntID, "EXECUTECDPMETHOD", "Log.disable", "", TRUE$ )
```

```
Function WEBCDPEVENT( CtrlEntID, CtrlClassID, EventName, JsonEventInfo, SessionID )

   // Example: When we get a "Log.entryAdded" event then output the
   // JsonEventInfo to the Sytem Monitor
   $Insert Logical

   bForward = TRUE$

   Begin Case
      Case ( EventName == "Log.entryAdded" )

         MsgText = EventName : ": " : JsonEventInfo
         Call Exec_Method( "SYSTEMMONITOR", "OUTPUT", MsgText )

         bForward = FALSE$ ; // Handled

   End Case

Return bForward
```

See Also

WEBVIEW ADDCDPEVENT method, WEBVIEW REMOVECDPEVENT method.

WEBCDPMETHODRESULT event

Description

Returns the results of a previous asynchronous call to the EXECUTECDPMETHOD method.

Syntax

```
bForward = WEBCDPMETHODRESULT( CtrlEntID, CtrlClassID, ErrorCode, JsonResult )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the WEBVIEW object receiving the event.
CtrlClassID	Type of object receiving the event (always "WEBVIEW").
ErrorCode	Error code returned from the method call (if appropriate).
JsonResult	A JSON object containing results of the method call. The contents of this argument depend on the CDP method and may be parsed and processed using the Basic+ JSON stored procedures like RTI_RJSON.

Returns

TRUE$ or FALSE$. If FALSE$, the program execution returns to the calling procedure. If TRUE$, the event processing goes to the next level.

Remarks

There is no system-level promoted event handler for the WEBCDPMETHODRESULT event.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2 CallDevToolsProtocolMethod method on the Microsoft website.

Example

```
Function WEBCDPMETHODRESULT( CtrlEntID, CtrlClassID, ErrorCode, JsonResult )

  // Assume we have executed the "Browser.getVersion" method - we are
  // expecting a Json object that looks like this:
  //
  // {
  //   "jsVersion":"11.9.17.11",
  //   "product":"Edg/119.0.2151.97",
  //   "protocolVersion":"1.3",
  //   "revision":"@42f50dec8b7a97d2cd39634128418ccfb4ef77b8",
  //   "userAgent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) ... <more>
  // }
  //
  // So we can use RTI_RJSON to parse out the members

  $Insert RTI_RJSON_Equates
  $Insert Logical

  Product = ""

  If ( ErrorCode == 0 ) Then

     ObjVersion = RTI_RJSON( RJSON_MTD_PARSE$, JsonResult )
     If ObjVersion Then

        ItemCount = RTI_RJSON( RJSON_MTD_GETITEMCOUNT$, ObjVersion )
        If ItemCount Then
           ItemNames = RTI_RJSON( RJSON_MTD_GETOBJECTMEMBERS$, ObjVersion )

           Locate "product" In ItemNames Using @Fm Setting Pos Then
              Product = RTI_RJSON( RJSON_MTD_GETITEMVALUE$, ObjVersion, "product" )
           End
        End

        Call RTI_RJSON( RJSON_MTD_DELETE$, ObjVersion )
     End
  End

  If BLen( Product) Then
     Call Msg( @Window, Product )
  End

Return FALSE$
```

See Also

WEBVIEW ADDCDPEVENT method, WEBVIEW EXECCDPMETHOD method, WEBVIEW EXECUTESCRIPT method, WEBVIEW REMOVECDPEVENT method.

WEBCLOSEWINDOW event

Description

Occurs when content inside the WEBVIEW object requests to close the window, such as after the JavaScript window.close() is executed.

Syntax

```
bForward = WEBCLOSEWINDOW( CtrlEntID, CtrlClassID )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the WEBVIEW object receiving the event.
CtrlClassID	Type of object receiving the event (always "WEBVIEW").

Returns

TRUE$ or FALSE$. If FALSE$, the program execution returns to the calling procedure. If TRUE$, the event processing goes to the next level.

Remarks

The application should close the WEBVIEW and its parent form or “tab” if that makes sense to the application.

There is no system-level promoted event handler for the WEBCLOSEWINDOW event.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2 add_WindowCloseRequested method on the Microsoft website.

Example

```
 Function WEBCLOSEWINDOW( CtrlEntID, CtrlClassID )

   // Example: Ask the user if it's OK to close the parent form before closing...
   $Insert Msg_Equates
   $Insert Logical

   MsgRec = ""
   MsgRec<MTEXT$>     = "The document is requesting to close the form - OK to close?"
   MsgRec<MTYPE$>     = "BNY"
   MsgRec<MJUST$>     = "C"
   MsgRec<MICON$>     = "?"
   MsgRec<MCAPTION$> = "Close form"

   MsgVal = Msg( @Window, MsgRec )

   If ( MsgVal == TRUE$ )
      // Async CLOSE...
      Call Exec_Method( @Window, "CLOSE", FALSE$, TRUE$ )
   End

Return FALSE$
```

See Also

N/a.

WEBCONTENTLOADED event

Description

Occurs when the initial HTML document has been parsed during navigation.

Syntax

```
bForward = WEBCONTENTLOADED( CtrlEntID, CtrlClass, NavID, URI )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the WEBVIEW object receiving the event.
CtrlClassID	Type of object receiving the event (always "WEBVIEW").
NavID	Unique identifier of the navigation process that triggered the WEBCONTENTLOADED event.
URI	Location that is being navigated to.

Returns

TRUE$ or FALSE$. If FALSE$, the program execution returns to the calling procedure. If TRUE$, the event processing goes to the next level.

Remarks

There is no system-level promoted event handler for the WEBCONTENTLOADED event.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2_2 add_DOMContentLoaded method on the Microsoft website.

Example

```
Function WEBCONTENTLOADED( CtrlEntID, CtrlClassID, NavID, URI )

  // Example: Update a ListBox log tracking navigation events

  LogText = "Loaded content for " : URI
  Call Exec_Method( @Window : ".LST_NAVLOG", "INSERT", -1, LogText )

Return TRUE$
```

See Also

WEBVIEW URI property, WEBVIEW NAVIGATE method, WEBVIEW WEBCONTENTLOADING event, WEBVIEW WEBHISTORYCHANGED event, WEBVIEW WEBNAVIGATING event, WEBSOURCECHANGED event.

WEBCONTENTLOADING event

Description

Occurs when a navigation process begins loading content into the WEBVIEW object.

Syntax

```
bForward = WEBCONTENTLOADING( CtrlEntID, CtrlClass, NavID, URI, ErrorPage )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the WEBVIEW object receiving the event.
CtrlClassID	Type of object receiving the event (always "WEBVIEW").
NavID	Unique identifier of the navigation process that triggered the WEBCONTENTLOADING event.
URI	Location that is being navigated to.
ErrorPage	Set to TRUE$ if the loaded content is an Error Page, or FALSE$ otherwise.

Returns

TRUE$ or FALSE$. If FALSE$, the program execution returns to the calling procedure. If TRUE$, the event processing goes to the next level.

Remarks

There is no system-level promoted event handler for the WEBCONTENTLOADING event.

This event does not trigger if a same page navigation occurs.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2 add_ContentLoading method on the Microsoft website.

Example

```
Function WEBCONTENTLOADING( CtrlEntID, CtrlClassID, NavID, URI, ErrorPage )

  // Example: Alert the user to an error condition

  If ErrorPage Then
     MsgText = "Error when loading " : quote( URI )
     Call Msg( @Window, MsgText )
  End

Return TRUE$
```

See Also

WEBVIEW URI property, WEBVIEW NAVIGATE method, WEBVIEW WEBCONTENTLOADED event, WEBVIEW WEBHISTORYCHANGED event, WEBVIEW WEBNAVIGATING event, WEBSOURCECHANGED event.

WEBCONTEXTMENU event

Description

Occurs when the WEBVIEW object wishes to display a context menu after a user right-clicks somewhere on the hosted content. It is the WEBVIEW equivalent of the Common GUI CONTEXTMENU event.

Syntax

```
bForward = WEBCONTEXTMENU( CtrlEntID,
                           CtrlClassID,
                           MenuID,
                           MenuStruct,
                           XPos,
                           YPos,
                           TargetInfo,
                           DefaultStruct,
                           AttachOnly )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the object receiving the event.
CtrlClassID	Type of object receiving the event.
MenuID	ID of the ContextMenu entity to display.
MenuStruct	A dynamic array containing the executable structure of the menu.
XPos	Client area X-coordinate where the user clicked.
YPos	Client area Y-coordinate where the user clicked.

TargetInfo	An @vm-delimited array of information about the item that the user has clicked on. See the Remarks section below for more details.
	
	`<0,1> Type`
	`<0,2> IsEditable`
	`<0,3> PageUri`
	`<0,4> SourceUri`
	`<0,5> LinkUri`
	`<0,6> LinkText`
	`<0,7> FrameUri`
	`<0,8> SelectionText`
	`<0,9> MainFrameRequest`

DefaultStruct	A dynamic array containing the executable structure of the default context menu as supplied by the WEBVIEW object. This is the menu that the WEBVIEW would have displayed if the CONTEXTMENU property was not set.
AttachOnly	If TRUE$ then the TRACKPOPUPMENU method is called with the AttachOnly flag so that the menu is parsed, created and attached, but not displayed. *This argument should not be changed by a WEBCONTEXTMENU event handler*.

Returns

TRUE$ or FALSE$. If FALSE$, the program execution returns to the calling procedure. If TRUE$, the event processing goes to the next level.

Remarks

The WEBVIEW object will not display any context menus if the "EnableContextMenus" option in the SETTINGS property is FALSE$.

The WEBVIEW object supplies information about the item that has been clicked on via the TargetInfo parameter. This is an @vm-delimited array with the following structure:

Index	Name	Description

<0,1>	Type	Specifies the type of the item. Can be one of the following:
		
		`"0" : Page`
		`"1" : Image`
		`"2" : SelectedText`
		`"3" : Audio`
		`"4" : Video`

<0,2>	IsEditable	Set to TRUE$ if the context menu is requested on an editable item, or FALSE$ otherwise.
<0,3>	PageURI	Contains the URI of the page.
<0,4>	SourceURI	Contains the source URI of the item (can be null).
<0,5>	LinkURI	Contains the link URI of the item (can be null).
<0,6>	LinkText	Contains the link text of the item is a link (can be null).
<0,7>	FrameURI	Contains the URI of the frame.
<0,8>	SelectionText	Contains the text selected if appropriate (can be null).
<0,9>	MainFrameRequest	Set to TRUE$ if the context menu was requested on the main frame, or FALSE$ if on another frame.

This event has a system-level handler which performs the following tasks:

- Adds the default system menus to the MenuStruct if appropriate.
- Calling a WEBCONTEXTMENU quick event, if defined.
    - If the event status returns anything other than 0 then the context menu display is cancelled via the CANCELCONTEXTMENU method.
- Displaying the context menu via the TRACKPOPUPMENU method.

Note that it is possible to merge items from the passed default WEBVIEW context menu ("DefaultStruct") into the OpenInsight context menu to be displayed ("MenuStruct"). When doing this it is important to ensure that the details of any items copied from DefaultStruct into MenuStruct are preserved because they contain special flags and identifiers that are needed by the WEBVIEW object if the item is selected by the user.

Equates constants for working with menu structures can be found in the PS_MENU_EQUATES insert record.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2_11 add_ContextMenuRequested method on the Microsoft website.

Example

```
Function WEBCONTEXTMENU( CtrlEntID, CtrlClassID, MenuID, MenuStruct, xPos, yPos, |
                                            TargetInfo, DefaultStruct, AttachOnly )

  // Example: Prevent a context menu from being displayed for images

  $Insert PS_WebView_Equates
  $Insert Logical

  If ( TargetInfo<0,WBV_TARGETINFO_POS_TYPE$> == WBV_TARGET_TYPE_IMAGE$ ) then
     Call Exec_Method( CtrlEntID, "CANCELCONTEXTMENU", MenuID )
     Return FALSE$
  End

Return TRUE$
```

```
Function WEBCONTEXTMENU( CtrlEntID, CtrlClassID, MenuID, MenuStruct, xPos, yPos, |
                                            TargetInfo, DefaultStruct, AttachOnly )

  // Example: Simple trick to use the default WEBVIEW context menu and make
  //          it look like a native OI menu

  $Insert PS_WebView_Equates
  $Insert Logical

  MenuStruct = DefaultStruct

Return TRUE$
```

```
Function WEBCONTEXTMENU( CtrlEntID, CtrlClassID, MenuID, MenuStruct, xPos, yPos, |
                                            TargetInfo, DefaultStruct, AttachOnly )

  // Example: Assume that our OI menu has a "COPY" item and we want to replace it
  //          with the "COPY" item from the WEBVIEW control (if it has one)

  $Insert PS_WebView_Equates
  $Insert PS_Menu_Equates
  $Insert Logical

  CopyItem = ""
  GoSub GetWebViewCopyItem

  If BLen( CopyItem ) Then
     GoSub ReplaceOICopyItem
  End

Return TRUE$

GetWebViewCopyItem:
  XCount = FieldCount( DefaultStruct, @Vm )
  For X = 5 to XCount
     If ( DefaultStruct<0,X>[1,1] == "@" ) Then
        Null ; // Ignore - it's an imagelist header
     End Else
        If ( DefaultStruct<0,X,MENUPOS_TYPE$> == MENUTYPE_ITEM$ ) Then
           ItemName = DefaultStruct<0,X,MENUPOS_NAME$>[-1,"B."]
           Begin Case
              Case ( ItemName == "COPY" )
                 CopyItem = DefaultStruct<0,X>
                 X = XCount ; // Break;
           End Case
        End
     End
  Next

Return

ReplaceOICopyItem:
  XCount = FieldCount( MenuStruct, @Vm )
  For X = 5 to XCount
     If ( MenuStruct<0,X>[1,1] == "@" ) Then
        Null ; // Ignore - it's an imagelist header
     End Else
        If ( MenuStruct<0,X,MENUPOS_TYPE$> == MENUTYPE_ITEM$ ) Then
           ItemName = MenuStruct<0,X,MENUPOS_NAME$>[-1,"B."]
           Begin Case
              Case ( ItemName = "COPY" )
                 MenuStruct<0,X> = CopyItem
                 X = XCount ; // Break;
           End Case
        End
     End
  Next

Return
```

See Also

Common GUI CONTEXTMENU property, WEBVIEW SETTINGS property, Common GUI SHOWMENU method, Common GUI TRACKPOPUPMENU method, WEBVEW CANCELCONTEXTMENU method, Common GUI CONTEXTMENU event, Common GUI MENU event, WEBVIEW INITCONTEXTMENU event, ContextMenu stored procedure.

WEBDATACLEARED event

Description

Returns the results of a call to the CLEARBROWSINGDATA method.

Syntax

```
bForward = WEBDATACLEARED( CtrlEntID, CtrlClassID, SuccessFlag, ErrorCode )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the WEBVIEW object receiving the event.
CtrlClassID	Type of object receiving the event (always "WEBVIEW").
SuccessFlag	Set to TRUE$ if the WEBVIEW object was clear operation was successful, or FALSE$ otherwise.
ErrrorCode	If SuccessFlag is FALSE$ then ErrorCode contains the error code that describes the reason for the failure.

Returns

TRUE$ or FALSE$. If FALSE$, the program execution returns to the calling procedure. If TRUE$, the event processing goes to the next level.

Remarks

There is no system-level promoted event handler for the WEBDATACLEARED event.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2Profile ClearBrowsingDataInTimeRange method on the Microsoft website.

Example

```
Function WEBDATACLEARED( CtrlEntID, CtrlClassID, SuccessFlag, ErrorCode )

  // Example: Display an error message if necessary
  Declare Function RTI_ErrorText
  $Insert Logical

  If SuccessFlag Else

     ErrorText = RTI_ErrorText( "WIN", ErrorCode, TRUE$ )

     Call Msg( @Window, ErrorText )

  End

Return TRUE$
```

See Also

WEBVIEW CANGOBACK property, WEBVIEW CANGOFORWARD property, WEBVIEW HISTORY property, WEBVIEW BACK method, WEBVIEW FORWARD method, WEBVIEW CLEARBROWSINGDATA method.

WEBHISTORYCHANGED event

Description

Occurs when a navigation updates the history of the WEBVIEW object.

Syntax

```
bForward = WEBHISTORYCHANGED( CtrlEntID, CtrlClass )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the WEBVIEW object receiving the event.
CtrlClassID	Type of object receiving the event (always "WEBVIEW").

Returns

TRUE$ or FALSE$. If FALSE$, the program execution returns to the calling procedure. If TRUE$, the event processing goes to the next level.

Remarks

There is no system-level promoted event handler for the WEBHISTORYCHANGED event.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2 add_HistoryChanged method on the Microsoft website.

Example

```
Function WEBHISTORYCHANGED( CtrlEntID, CtrlClassID )

  // Example: Update the Back and Forward buttons when the history
  //          changes

  BackEnabled    = Get_Property( CtrlEntID, "CANGOBACK" )
  ForwardEnabled = Get_Property( CtrlEntID, "CANGOFOWARD" )

  Call Set_Property( @Window : ".BTN_BACK", "ENABLED", BackEnabled )
  Call Set_Property( @Window : ".BTN_FORWARD", "ENABLED", ForwardEnabled )

Return TRUE$
```

See Also

WEBVIEW CANGOBACK property, WEBVIEW CANGOFORWARD property, WEBVIEW HISTORY property, WEBVIEW BACK method, WEBVIEW FORWARD method.

WEBINITCONTEXTMENU event

Description

Occurs when the WEBVIEW object wishes to initialize a context menu after a user right-clicks somewhere on the hosted content. This event is responsible for initializing the context menu ready for display. It is the WEBVIEW equivalent of the Common GUI INITCONTEXTMENU event.

Syntax

```
bForward = WEBINITCONTEXTMENU( CtrlEntID,
                               CtrlClassID,
                               MenuID,
                               XPos,
                               YPos,
                               TargetInfo,
                               DefaultStruct )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the object receiving the event.
CtrlClassID	Type of object receiving the event.
MenuID	ID of the ContextMenu entity to display.
XPos	Client area X-coordinate where the user clicked.
YPos	Client area Y-coordinate where the user clicked.

TargetInfo	An @vm-delimited array of information about the item that the user has clicked on. See the Remarks section below for more details.
	`<0,1> Type`
	`<0,2> IsEditable`
	`<0,3> PageUri`
	`<0,4> SourceUri`
	`<0,5> LinkUri`
	`<0,6> LinkText`
	`<0,7> FrameUri`
	`<0,8> SelectionText`
	`<0,9> MainFrameRequest`

DefaultStruct	OIWIN structure of the default context menu as supplied by the WEBVIEW object. This is the menu that the WEBVIEW would have displayed if the CONTEXTMENU property was not set.

Returns

TRUE$ or FALSE$. If FALSE$, the program execution returns to the calling procedure. If TRUE$, the event processing goes to the next level.

Remarks

The WEBVIEW object will not display any context menus if the "EnableContextMenus" option in the SETTINGS property is FALSE$.

The WEBVIEW object supplies information about the item that has been clicked on via the TargetInfo parameter. This is an @vm-delimited array with the following structure:

Index	Name	Description

<0,1>	Type	Specifies the type of the item. Can be one of the following:
		
		`"0" : Page`
		`"1" : Image`
		`"2" : SelectedText`
		`"3" : Audio`
		`"4" : Video`

<0,2>	IsEditable	Set to TRUE$ if the context menu is requested on an editable item, or FALSE$ otherwise.
<0,3>	PageURI	Contains the URI of the page.
<0,4>	SourceURI	Contains the source URI of the item (can be null).
<0,5>	LinkURI	Contains the link URI of the item (can be null).
<0,6>	LinkText	Contains the link text of the item is a link (can be null).
<0,7>	FrameURI	Contains the URI of the frame.
<0,8>	SelectionText	Contains the text selected if appropriate (can be null).
<0,9>	MainFrameRequest	Set to TRUE$ if the context menu was requested on the main frame, or FALSE$ if on another frame.

This event has a system-level handler which performs the following tasks:

- Calling the Yield stored procedure to clear any pending events.
- Calling a WEBINITCONTEXTMENU quick event, if defined.
    - If the event status returns anything other than 0 then the context menu display is cancelled.
- Reading the context menu definition from the repository (if not already cached).
- Compiling it into an "executable" format and caching it.
- Compiling the "DefaultStruct" menu struct into an executable format.
- Firing the subsequent WEBCONTEXTMENU event.

The intent of the WEBINITCONTEXTMENU event is as a place for the Presentation Server to begin the WEBVIEW context menu process, so as such it is a system tool – it is not really intended that developers have to interact with this event, although there's nothing to prevent this if desired.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2_11 add_ContextMenuRequested method on the Microsoft website.

Example

N/a.

See Also

Common GUI CONTEXTMENU property, WEBVIEW SETTINGS property, Common GUI SHOWMENU method, Common GUI TRACKPOPUPMENU method, Common GUI CONTEXTMENU event, Common GUI MENU event, WEBVIEW CONTEXTMENU event, ContextMenu stored procedure.

WEBINITSCRIPTADDED event

Description

Returns the results of a call to the ADDINITSCRIPT method.

Syntax

```
bForward = WEBINITSCRIPTADDED( CtrlEntID, CtrlClassID, ID, ErrorCode )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the WEBVIEW object receiving the event.
CtrlClassID	Type of object receiving the event (always "WEBVIEW").
ID	Unique ID of the script that was added. This can be used with the REMOVEINITSCRIPT method.
ErrorCode	Contains an error code if a problem occurred when adding the script.

Returns

TRUE$ or FALSE$. If FALSE$, the program execution returns to the calling procedure. If TRUE$, the event processing goes to the next level.

Remarks

There is no system-level promoted event handler for the WEBINITSCRIPTADDED event.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2 AddScriptToExecuteOnDocumentCreated method on the Microsoft website.

Example

```
Function WEBINTISCRIPTADDED( CtrlEntID, CtrlClassID, ID, ErrorCode )

   // Example: Display an error message if a call to ADDINITSCRIPT failed
   $Insert Logical

   If ErrorCode Then
      ErrorText = "ADDINITSCRIPT failed - Error [" : ErrorCode : "]"
      Call Msg( @Window, ErrorText )
   End

Return TRUE$
```

See Also

WEBVIEW ADDINITSCRIPT method, WEBVIEW EXECUTESCRIPT method, WEBVIEW REMOVEINITSCRIPT method.

WEBMESSAGE event

Description

Occurs when the WEBVIEW object receives a WebMessage from the hosted content via the window.chrome.webview.postMessage() method.

Syntax

```
bForward = WEBMESSAGE( CtrlEntID, CtrlClass, URI, JsonMessage, TextMessage )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the WEBVIEW object receiving the event.
CtrlClassID	Type of object receiving the event (always "WEBVIEW").
URI	The URI of the document that sent the message.
JsonMessage	The message that was posted converted to a JSON string. This value can be parsed and processed using the Basic+ JSON stored procedures like RTI_RJSON.
TextMessage	If the posted message was a simple string type this parameter contains the raw non-JSON version of the message, otherwise it is set to null.

Returns

TRUE$ or FALSE$. If FALSE$, the program execution returns to the calling procedure. If TRUE$, the event processing goes to the next level.

Remarks

There is no system-level promoted event handler for the WEBMESSAGE event.

The EnableWebMessages field in the WEBVIEW SETTINGS property must be set to TRUE$ for this WebMessaging to work.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2 add_WebMessageReceived method on the Microsoft website.

Example

```
  // Example JavaScript for the HTML document to send a object message
  // back to the WEBVIEW object.
  //
  // We are going to get the contents of the FORENAME and SURNAME INPUT
  // elements and combine them into a JavaScript object which we then post
  // back the WEBVIEW.
  //
  // In the WEBVIEW object's WEBCHANGED event we can parse and display
  // the "forename" member in an OpenInsight message.
  //
  // Note that we are sending it as an object so it will appear in
  // the JsonMesssage parameter of the WEBMESSAGE event.

  // This should be placed in the HTML document...
  <script>
     let dataObject = {};

     dataObject.forename = document.getElementById( "FORENAME" ).value;
     dataObject.surname  = document.getElementById( "SURNAME" ).value;

     window.chrome.webview.postMessage( dataObject );

  </script>

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////

 Function WEBMESSAGE( CtrlEntID, CtrlClassID, URI, JsonMessage, TextMessage )

  // Example: Look for a JSON message, extract the "forename" value, and then
  //          display it via Msg()
  //
  // (Error checking omitted for clarity!)

  $Insert RTI_RJSON_Equates

  Forename = ""

  ObjMessage = RTI_RJSON( RJSON_MTD_PARSE$, JsonMessage )
  If ObjMessage Then

     ItemCount = RTI_RJSON( RJSON_MTD_GETITEMCOUNT$, ObjMessage )
     If ItemCount Then
        ItemNames = RTI_RJSON( RJSON_MTD_GETOBJECTMEMBERS$, ObjMessage )

        Locate "forename" In ItemNames Using @Fm Setting Pos Then
           Forename = RTI_RJSON( RJSON_MTD_GETITEMVALUE$, ObjMessage, "forename" )
        End
     End

     Call RTI_RJSON( RJSON_MTD_DELETE$, ObjMessage )
  End

  If BLen( Forename ) Then
     Call Msg( @Window, Forename )
  End

Return TRUE$
```

```
  // Example JavaScript for the HTML document to send a string message
  // back to the WEBVIEW object.
  //
  // We are going to get the contents of the FORENAME INPUT element
  // and send to the WEBVIEW object.  Then in the WEBVIEW object's
  // WEBCHANGED event we can pick this up and display it in an
  // OpenInsight message.
  //
  // Note that we are sending it as a string, so it will appear in
  // the TextMesssage parameter of the WEBMESSAGE event.

  // This should be placed in the HTML document...
  <script>

     let objFormName = document.getElementById( "FORENAME" );
     window.chrome.webview.postMessage( "-forename:" + objFormName.value );

  </script>

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////

 Function WEBMESSAGE( CtrlEntID, CtrlClassID, URI, JsonMessage, TextMessage )

  // Example: Look for a text message prefixed with "-foreName" and if found
  //          : display it via Msg()

  If BLen( TextMessage ) Then

     Prefix  = TextMessage[1,":",TRUE$]
     Message = TextMessage[BCol2()+1,\00\,TRUE$]

     If Prefix == "-forename" Then
       Call Msg( @Window, "Forename is " : Message )
     End

  End

Return TRUE$
```

See Also

WEBVIEW SETTINGS property, WEBVIEW POSTJSONMESSAGE method, WEBVIEW POSTTEXTMESSAGE method.

WEBMUTEDCHANGED event

Description

Occurs when the “audio muted” status is changed.

Syntax

```
bForward = WEBMUTEDCHANGED( CtrlEntID, CtrlClassID, AudioMuted )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the WEBVIEW object receiving the event.
CtrlClassID	Type of object receiving the event (always "WEBVIEW").
AudioMuted	Set to TRUE$ if the WEBVIEW object is audio is muted, or FALSE$ otherwise.

Returns

TRUE$ or FALSE$. If FALSE$, the program execution returns to the calling procedure. If TRUE$, the event processing goes to the next level.

Remarks

There is no system-level promoted event handler for the WEBMUTEDCHANGED event.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2_8 add_IsMutedAudioChanged method on the Microsoft website.

Example

```
Function WEBMUTEDCHANGED( CtrlEntID, CtrlClassID, AudioMuted )

  // Example: Display a message with the muted status

  If AudioMuted Then
     MsgText = "In Basic+, no one can hear you scream"
  End Else
     MsgText = "Audio On"
  End

  Call Msg( @Window, MsgRec )

Return TRUE$
```

See Also

WEBVIEW AUDIOPLAYING property, WEBVIEW MUTED property, WEBVIEW WEBAUDIOCHANGED event.

WEBNAVIGATED event

Description

Occurs when the WEBVIEW object has completely loaded, or loading has stopped with an error.

Syntax

```
bForward = WEBNAVIGATED( CtrlEntID, CtrlClass, NavID, URI, Redirected, |
                         UserInitiated, StatusInfo, FrameID )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the WEBVIEW object receiving the event.
CtrlClassID	Type of object receiving the event (always "WEBVIEW").
NavID	Unique identifier for the navigation process.
URI	Location that is being navigated to.
Redirected	Set to TRUE$ when the navigation has been redirected, or FALSE$ otherwise.
UserInitiated	Set to TRUE$ when the navigation was initiated through a user gesture as opposed to programmatic navigation by page script.

StatusInfo	Contains an @vm-delimited array of data regarding the outcome of the navigation:
	
	`<0,1> SuccessFlag (TRUE$ or FALSE$)`
	`<0,2> HTTPStatusCode`
	`<0,3> WebErrorStatus (if the navigation failed)`

FrameID	If the navigation event was triggered from an IFrame then this argument contains the name of the frame.

Returns

TRUE$ or FALSE$. If FALSE$, the program execution returns to the calling procedure. If TRUE$, the event processing goes to the next level.

Remarks

There is no system-level promoted event handler for the WEBNAVIGATED event.

The SuccessFlag member in the StatusInfo parameter is set to 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 a window.stop() JavaScript statement 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 (Equates for the WebErrorStatus codes can be found the PS_WEBVIEW_EQUATES insert record).

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2_2 add_add_NavigationCompleted method on the Microsoft website.

Example

```
Function WEBNAVIGATED( CtrlEntID, CtrlClassID, NavID, URI, Redirected, |
                        UserInitiated, StatusInfo, FrameID )

  // Example: Display a message if the navigation failed

  $Insert PS_WEBVIEW_Equates
  $Insert Msg_Equates

  If StatusInfo<0,WBV_NAV_STATUS_POS_SUCCESSFLAG$> else

     MsgText =  "WebView Navigation Error"
     MsgText := "||URI: "           : URI
     MsgText := "|HTTP Status: "     : StatusInfo<0,WBV_NAV_STATUS_POS_HTTPSTATUSCODE$>
     MsgText := "|WebError Status: " : StatusInfo<0,WBV_NAV_STATUS_POS_WEBERRORSTATUS$>

     MsgRec            = ""
     MsgRec<MTEXT$>    = MsgText
     MsgRec<MICON$>    = "!"
     MsgRec<MCAPTION$> = "WebView Navigation Error"

     Call Msg( @Window, MsgRec )

  End

Return TRUE$
```

See Also

WEBVIEW URI property, WEBVIEW NAVIGATE method, WEBVIEW WEBCONTENTLOADED event, WEBVIEW WEBCONTENTLOADING event, WEBVIEW WEBHISTORYCHANGED event, WEBVIEW WEBNAVIGATING event, WEBSOURCECHANGED event.

WEBNAVIGATING event

Description

Occurs when the WEBVIEW object is navigating to a different URI. Redirects trigger this event as well, and the NavID is the same as the original one.

Syntax

```
bForward = WEBNAVIGATING( CtrlEntID, CtrlClass, NavID, URI, Redirected, |
                          UserInitiated, FrameID )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the WEBVIEW object receiving the event.
CtrlClassID	Type of object receiving the event (always "WEBVIEW").
NavID	Unique identifier for the navigation process – this is passed to subsequent events that are triggered from the same navigation.
URI	Location that is being navigated to.
Redirected	Set to TRUE$ when the navigation has been redirected, or FALSE$ otherwise.
UserInitiated	Set to TRUE$ when the navigation was initiated through a user gesture as opposed to programmatic navigation by page script.
FrameID	If the navigation event was triggered from an IFrame then this argument contains the name of the frame.

Returns

TRUE$ or FALSE$. If FALSE$, the program execution returns to the calling procedure. If TRUE$, the event processing goes to the next level.

Remarks

There is no system-level promoted event handler for the WEBNAVIGATING event.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2_2 add_NavigationStarting and add_FrameNavigationStarting methods on the Microsoft website.

Example

```
Function WEBNAVIGATING( CtrlEntID, CtrlClassID, NavID, URI, Redirected, |
                        UserInitiated, FrameID )

  // Example: Update a ListBox log tracking navigation events

  LogText = "Navigating to " : URI
  Call Exec_Method( @Window : ".LST_NAVLOG", "INSERT", -1, LogText )

Return TRUE$
```

See Also

WEBVIEW URI property, WEBVIEW NAVIGATE method, WEBVIEW WEBCONTENTLOADED event, WEBVIEW WEBCONTENTLOADING event, WEBVIEW WEBHISTORYCHANGED event, WEBVIEW WEBNAVIGATED event, WEBSOURCECHANGED event.

WEBOPENWINDOW event

Description

Occurs when the WEBVIEW object attempts to open a new window from a clicked hyperlink or the JavaScript "window.open" method.

A WEBOPENWINDOW event handler must do one of the following:

- Allow a new "WebView2 window" to open (this is a basic browser window managed by the WEBVIEW object itself and is not an OpenInsight form). This is the default behavior.
- "Redirect" the open request to another existing WEBVIEW object. For example a WEBVIEW object in another OpenInsight form or one on another tab page.
- Deny the request. This will stop the new window from opening.

Syntax

```
bForward = WEBOPENWINDOW( CtrlEntID, CtrlClassID, OpenID, URI, WindowInfo, |
                                                          UserInitiated )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the WEBVIEW object receiving the event.
CtrlClassID	Type of object receiving the event (always "WEBVIEW").
OpenID	A unique integer value that identifies the "open window" request.
URI	URI that the new window will navigate to.

WindowInfo	Contains an @vm-delimited dynamic array of requested window features. Note that it is possible for the size and position values to be null if this wasn't specified in the originating window.open() call.
	
	`<0,1> WindowName`
	`<0,2> Left`
	`<0,3> Top`
	`<0,4> Width`
	`<0,5> Height`
	`<0,6> ShowMenuBar`
	`<0,7> ShowScrollBars`
	`<0,8> ShowStatusBar`
	`<0,9> ShowToolBar`
	
	You may ignore these options if you are redirecting to another WEBVIEW object – they are simply indications of what the opener would prefer)

UserInitiated	A boolean value set to TRUE$ if the "open window" was generated by the user (e.g. clicking a hyperlink), or FALSE$ if it was generated from a script.

Returns

TRUE$ or FALSE$. If FALSE$, the program execution returns to the calling procedure. If TRUE$, the event processing goes to the next level.

Remarks

The ALLOWOPENWINDOW method is used to allow the "open window" request or redirect it to another WEBVIEW object.

The DENYOPENWINDOW method is used to stop the new window from opening.

The WEBOPENWINDOW event has a system-level promoted event handler that performs the following tasks:

- Executes the WEBOPENWINDOW quick event handler (if defined) and checks the event status – if it returns anything other than FALSE$ the event is cancelled.
- Calls the ALLOWOPENWINDOW method to open a new "WebView2 window" with the requested WindowInfo options.

Equates for use with this event can be found in the PS_WEBVIEW_EQUATES insert record.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2 add_NewWindowRequested method on the Microsoft website.

Example

```
Function WEBOPENWINDOW( CtrlEntID, CtrlClassID, OpenID, URI, WindowInfo, |
                                                            UserInitiated )

  // Example: A WEBOPENWINDOW event handler that performs the following:
  //
  //    1) If the URI points to a localhost running on port 8888 it
  //       creates a new OpenInsight form called LOCALHOST_OPENWIN with
  //       with a WEBVIEW control called WBV_BROWSER and redirects to
  //       that.
  //
  //    2) If the URI is from "www.revelation.com" we allow it to open
  //       a new "WebView2 window".
  //
  //    3) For anything else we deny it.
  //
  // (Note we are not using a robust way of getting the host and port from
  // the URI, but this is just an example and not production code ;-) )

  $Insert Logical

  Begin Case
     Case IndexC( URI, "localhost:8888", 1 )

         WinID = Start_Window( "LOCALHOST_OPENWIN", @Window, "" )
         If BLen( WinID ) Then

           Call Exec_Method( CtrlEntID,                |
                             "ALLOWOPENWINDOW",        |
                             OpenID,                   |
                             WinID : ".WBV_BROWSER" )

     Case IndexC( URI, "www.revelation.com", 1 )

        Call Exec_Method( CtrlEntID,          |
                          "ALLOWOPENWINDOW",  |
                          OpenID )

     Case OTHERWISE$

        Call Exec_Method( CtrlEntID,          |
                          "DENYOPENWINDOW",   |
                          OpenID )

  End Case

Return FALSE$
```

See Also

WEBVIEW ALLOWOPENWINDOW method, WEBVIEW DENYOPENWINDOW method.

WEBPDFPRINTED event

Description

Returns the results of a previous call to the PRINTTOPDF method.

Syntax

```
bForward = WEBPDFPRINTED( CtrlEntID, CtrlClassID, FileName, SuccessFlag, |
                                                            ErrorCode )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the WEBVIEW object receiving the event.
CtrlClassID	Type of object receiving the event (always "WEBVIEW").
FileName	Name and path of the file that was printed.
SuccessFlag	TRUE$ if the print operation was successful, or FALSE$ if it failed.
ErrorCode	Error code returned by the WEBVIEW object if the print operation failed. This will be a Windows "HRESULT" error code, so a description may be obtained by using the "WIN" method in the RTI_ErrorText stored procedure (See example below).

Returns

TRUE$ or FALSE$. If FALSE$, the program execution returns to the calling procedure. If TRUE$, the event processing goes to the next level.

Remarks

There is no system-level promoted event handler for the WEBPDFPRINTED event.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2_7 PrintToPdf method on the Microsoft website.

Example

```
FUNCTION WEBPDFPRINTED( CtrlEntID, CtrlClassID, FileName, SuccessFlag, |
                                                              ErrorCode )

  // Example - display a message with the results of the
  // print operation

  Declare Function RTI_ErrorText
  $Insert Msg_Equates
  $Insert Logical

  MsgRec = ""

  If ( SuccessFlag ) Then
     // Display a timed "success" message
     MsgRec<MTEXT$> = Quote( FileName ) : " printed successfully"
     MsgRec<MTYPE$> = "T2" ; // Timed

  End Else
     // Display an error message

     ErrorText = RTI_ErrorText( "WIN", ErrorCode, TRUE$ )

     MsgText =  "Failed to print " : Quote( FileName )
     MsgText := "Error Code: "     : ErrorCode : "|"
     MsgText := "Error Details: "  : ErrorText : "||"

     MsgRec<MTEXT$>  = MsgText
     MsgRec<MTYPE$>  = "BO" ; // "OK"
     MsgRec<MICON$>  = "!"  ; // Warning

  End

  MsgRec<MJUST$>    = "C"
  MsgRec<MCAPTION$> = "Print WebPage To PDF"

  Call Msg( @Window, "MsgRec" )
```

See Also

WEBVIEW PRINTTOPDF method.

WEBPERMISSIONREQUEST event

Description

Occurs when the content in the WEBVIEW object requests permission to access some privileged resources.

A WEBPERMISSIONREQUEST event handler should use the SETPERMISSION method to allow or deny this request.

Syntax

```
bForward = WEBPERMISSIONREQUEST( CtrlEntID, CtrlClassID, URI, RequestType, |
                                 UserInitiated )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the WEBVIEW object receiving the event.
CtrlClassID	Type of object receiving the event (always "WEBVIEW").
URI	URI of the request that triggered the permission request.

RequestType	Identifies the type of resource requested. Can be one of the following values:
	`0 : Unknown`
	`1 : Microphone`
	`2 : Camera`
	`3 : Geolocation`
	`4 : Notification`
	`5 : Other Sensor`
	`6 : Read Clipboard`

UserInitiated	Set to TRUE$ if the request originated from a user gesture, or FALSE$ otherwise.

Returns

TRUE$ or FALSE$. If FALSE$, the program execution returns to the calling procedure. If TRUE$, the event processing goes to the next level.

Remarks

The WEBPERMISSIONREQUEST event has a system-level promoted event handler that performs the following tasks:

- Executes the WEBPERMISSIONREQUEST quick event handler (if defined) and checks the event status – if it returns anything other than FALSE$ the event is assumed to have been handled by the quick event.

- Otherwise, a dialog is displayed asking the user if they wish to grant permission to the specified resource.
- Based on the user response access is granted or denied by using the SETPERMISSION method.
- If none of the above steps have handled the request it is cancelled by using the CANCELPERMISSIONREQUEST method to ensure that any waiting resources are released.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2 add_PermissionRequested method on the Microsoft website.

Example

```
Function WEBPERMISSIONREQUEST( CtrlEntID, CtrlClassID, URI, RequestType, |
                               UserInitiated )

  // Example: Allow Clipboard Read requests without any user interaction
  //          otherwise pass it onto the default handler.

  $Insert PS_WebView_Equates
  $Insert Logical

  RetVal = TRUE$

  Begin Case
     Case ( RequestType == WBV_PERMTYPE_CLIPBOARDREAD$ )
        // Allow it - we could also check the URI or UserInitiated flag
        // for extra security ...

        bSet = Exec_Method( CtrlEntID, "SETPERMISSION", URI, RequestType, |
                            UserInitiated, TRUE$ )

        If bSet Then
           reVal = FALSE$ ; // Handled - don't pass back to the system
        End

     Case OTHERWISE$
        Null

  End Case

Return RetVal
```

See Also

WEBVIEW SETPERMISSION method, WEBVIEW CANCELPERMISSIONREQUEST method.

WEBSAVEDTOFILE event

Description

Returns the results of a previous call to the SAVETOFILE method.

Syntax

```
bForward = WEBSAVEDTOFILE( CtrlEntID, CtrlClassID, FileName, SuccessFlag, |
                                                         ErrorCode, ErrorText )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the WEBVIEW object receiving the event.
CtrlClassID	Type of object receiving the event (always "WEBVIEW").
FileName	Name and path of the file that was saved.
SuccessFlag	TRUE$ if the file save operation was successful, or FALSE$ if it failed.
ErrorCode	Error code returned by the WEBVIEW object if the save operation failed.
ErrorText	Error details returned by the WEBVIEW object if the save operation failed.

Returns

TRUE$ or FALSE$. If FALSE$, the program execution returns to the calling procedure. If TRUE$, the event processing goes to the next level.

Remarks

There is no system-level promoted event handler for the WEBSAVEDTOFILE event.

For more information on this topic please refer to the "Page.captureSnapshot" method in the Chrome DevTools Protocol website here:

https://chromedevtools.github.io/devtools-protocol/tot/Page/#method-captureSnapshot

Example

```
FUNCTION WEBSAVEDTOFILE( CtrlEntID, CtrlClassID, FileName, SuccessFlag, |
                         ErrorCode, ErrorText )


  // Example - display a message with the results of the
  // save operation

  $Insert Msg_Equates
  $Insert Logical

  MsgRec = ""

  If ( SuccessFlag ) Then
     // Display a timed "success" message
     MsgRec<MTEXT$> = Quote( FileName ) : " saved successfully"
     MsgRec<MTYPE$> = "T2" ; // Timed

  End Else
     // Display an error message

     MsgText =  "Failed to save " : Quote( FileName )
     MsgText := "Error Code: "    : ErrorCode : "|"
     MsgText := "Error Details: " : ErrorText : "||"

     MsgRec<MTEXT$>  = MsgText
     MsgRec<MTYPE$>  = "BO" ; // "OK"
     MsgRec<MICON$>  = "!"  ; // Warning

  End

  MsgRec<MJUST$>    = "C"
  MsgRec<MCAPTION$> = "Save WebPage As"

  Call Msg( @Window, "MsgRec" )

Return TRUE$
```

See Also

WEBVIEW SAVETOFILE method.

WEBSCRIPTRESULT event

Description

Returns the results of a previous asynchronous call to the EXECUTESCRIPT method.

Syntax

```
bForward = WEBSCRIPTRESULT( CtrlEntID, CtrlClassID, ErrorCode, ScriptResult )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the WEBVIEW object receiving the event.
CtrlClassID	Type of object receiving the event (always "WEBVIEW").
ErrorCode	Error code returned by the WEBVIEW object if the EXECUTESCRIPT method failed. This will be a Windows "HRESULT" error code, so a description may be obtained by using the "WIN" method in the RTI_ErrorText stored procedure (See example below).
ScriptResult	A JSON encoded string containing the result of the executed JavaScript. 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".

Returns

TRUE$ or FALSE$. If FALSE$, the program execution returns to the calling procedure. If TRUE$, the event processing goes to the next level.

Remarks

There is no system-level promoted event handler for the WEBSCRIPTRESULT event.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2ExecuteScript method on the Microsoft website.

Example

```
FUNCTION WEBSCRIPTRESULT( CtrlEntID, CtrlClassID, ErrorCode, ScriptResult )

  // Example - display a message with the results of the
  // executed script

  Declare Function RTI_ErrorText
  $Insert Msg_Equates
  $Insert Logical

  MsgRec = ""

  If ( ErrorCode == 0 ) Then
     // Display message with the script results

     MsgRec<MTEXT$> = ScriptResult
     MsgRec<MTYPE$>  = "BO" ; // "OK"
     MsgRec<MICON$>  = "*"  ; // Info

  End Else
     // Display an error message

     ErrorText = RTI_ErrorText( "WIN", ErrorCode, TRUE$ )

     MsgText =  "Script Failed!||"
     MsgText := "Error Code: "     : ErrorCode : "|"
     MsgText := "Error Details: "  : ErrorText : "||"

     MsgRec<MTEXT$>  = MsgText
     MsgRec<MTYPE$>  = "BO" ; // "OK"
     MsgRec<MICON$>  = "!"  ; // Warning

  End

  MsgRec<MJUST$>    = "C"
  MsgRec<MCAPTION$> = "ExecuteScript Result"

  Call Msg( @Window, "MsgRec" )
```

See Also

WEBVIEW EXECUTESCRIPT method, WEBVIEW WEBSYNCSCRIPTRESULT event.

WEBSHOWDIALOG event

Description

Occurs when a the WEBVIEW object needs to display a custom dialog in response to a JavaScript dialog statement (i.e. “window.alert”, “window.prompt”, “window.confirm” and “beforeunload”).

By default, JavaScript dialogs are displayed in the WEBVIEW object using it’s default visual style. However, if the “EnableScriptDialogs” option in the SETTINGS property is set to FALSE$ then the WEBVIEW object triggers a WEBSHOWDIALOG event instead.

A WEBSHOWDIALOG event handler should display an appropriate OpenInsight dialog box with the passed details. After the dialog is closed the CONFIRMDIALOG method should be used to end the dialog request, and optionally return a value back to the calling script for a “prompt” type dialog.

Note however, that the CANCELDIALOG method may also be used to prevent a dialog from being displayed if desired.

Syntax

```
bForward = WEBSHOWDIALOG( CtrlEntID, CtrlClass, DialogID, URI, DialogType,
                          MessageText, DefaultResponse )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the WEBVIEW object receiving the event.
CtrlClassID	Type of object receiving the event (always "WEBVIEW").
DialogID	A unique ID for the show dialog request. This is used with the CONFIRMDIALOG and CANCELDIALOG methods.
URI	URI of the content requesting the dialog.

DialogType	Type of dialog requested. Can be one of the follow values (these map onto the four standard JavaScript dialog types):
	`0 : alert`
	`1 : confirm`
	`2 : prompt`
	`3 : beforeunload`

MessageText	Text to display in the dialog.
DefaultResponse	The response text passed to a “prompt” type dialog.

Returns

TRUE$ or FALSE$. If FALSE$, the program execution returns to the calling procedure. If TRUE$, the event processing goes to the next level.

Remarks

The WEBSHOWDIALOG event has a system-level promoted event handler that performs the following tasks:

- Executes the WEBSHOWDIALOG quick event handler (if defined) and checks the event status – if it returns anything other than FALSE$ the event is assumed to have been fully handled.

- Displays one of the following OpenInsight message boxes based on the dialog type:
    - OIWIN_WEBSHOWDIALOG_ALERT
    - OIWIN_WEBSHOWDIALOG_CONFIRM
    - OIWIN_WEBSHOWDIALOG_PROMPT
    - OIWIN_WEBSHOWDIALOG_BEFOREUNLOAD

- Depending on the user's response the CONFIRMDIALOG or the CANCELDIALOG methods are used to end the request as appropriate and ensure that all resources are released properly.

Equates for use with this event can be found in the PS_WEBVIEW_EQUATES insert record.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2 add_ScriptDialogOpening method on the Microsoft website.

Example

```
Function WEBSHOWDIALOG( CtrlEntID, CtrlClassID, DialogID, URI, DialogType, |
                                              MessageText, DefaultResponse )

  // Example: 1) Use the Msg() function for an "alert" style dialog
  //          2) Use the Msg() function for a "response" style dialog
  //          3) Don't let a "beforeunload" dialogs be shown
  //          4) Otherwise, let the system show the defaults

  $Insert PS_WebView_Equates
  $Insert Msg_Equates
  $Insert Logical

  RetVal = FALSE$ ; // Assume handled - don't pass onto the system

  Begin Case
     Case ( DialogType == WBV_DLGTYPE_ALERT$ )
        Call Msg( @Window, MessageText )
        Call Exec_Method( "CONFIRMDIALOG", DialogID, "" )

     Case ( DialogType == WBV_DLGTYPE_PROMPT$ )
        MsgRec = ""
        MsgRec<MTEXT$>      = MessageText
        MsgRec<MTYPE$>      = "R"
        MsgRec<MICON$>      = "?"
        MsgRec<MDEFINPUT$> = DefaultResponse

        MsgVal = Msg( @Window, MsgRec )
        If ( MsgVal == \0B\ ) Then
           Call Exec_Method( "CANCELDIALOG", DialogID )
        End Else
           Call Exec_Method( "CONFIRMDIALOG", DialogID, MsgVal )
        End

     Case ( DialogType == WBV_DLGTYPE_BEFOREUNLOAD$ )
        Call Exec_Method( "CANCELDIALOG", DialogID )

     Case OTHERWISE$
        RetVal = TRUE$

  End Case

Return RetVal
```

See Also

WEBVIEW SETTINGS property, WEBVIEW CANCELDIALOG method, WEBVIEW CONFIRMDIALOG method, User Interface Integration notes.

WEBSOURCECHANGED event

Description

Occurs when the WEBVIEW object's source property changes during navigation.

Syntax

```
bForward = WEBSOURCECHANGED( CtrlEntID, CtrlClass, NewDocument )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the WEBVIEW object receiving the event.
CtrlClassID	Type of object receiving the event (always "WEBVIEW").
NewDocument	Set to TRUE$ if the page being navigated to is a new document, or FALSE$ otherwise.

Returns

TRUE$ or FALSE$. If FALSE$, the program execution returns to the calling procedure. If TRUE$, the event processing goes to the next level.

Remarks

There is no system-level promoted event handler for the WEBSOURCECHANGED event.

This event only runs when navigating to a different site or for 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.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2_2 add_SourceChanged method on the Microsoft website.

Example

```
Function WEBSOURCECHANGED( CtrlEntID, CtrlClassID, NewDocument )

  // Example: Update a ListBox log tracking navigation events

  LogText = "SourceChanged"
  If NewDocument Then
     LogText := " (New Document)"
  End

  Call Exec_Method( @Window : ".LST_NAVLOG", "INSERT", -1, LogText )

Return TRUE$
```

See Also

WEBVIEW URI property, WEBVIEW NAVIGATE method, WEBVIEW WEVVIEW WEBCONTENTLOADED event, WEBCONTENTLOADING event, WEBVIEW WEBHISTORYCHANGED event, WEBVIEW WEBNAVIGATING event.

WEBSTATUSTEXTCHANGED event

Description

Occurs when the WEBVIEW object is showing a status message, a URL, or an empty string.

Syntax

```
bForward = WEBSTATUSTEXTCHANGED( CtrlEntID, CtrlClass, StatusText )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the WEBVIEW object receiving the event.
CtrlClassID	Type of object receiving the event (always "WEBVIEW").
StatusText	The status message text.

Returns

TRUE$ or FALSE$. If FALSE$, the program execution returns to the calling procedure. If TRUE$, the event processing goes to the next level.

Remarks

There is no system-level promoted event handler for the WEBSTATUSTEXTCHANGED event.

The visibility of the WEBVIEW object's own intrinsic status line is controlled by the "EnableStatusBar" setting in the SETTINGS property.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2_12 add_StatusBarTextChanged method on the Microsoft website.

Example

```
Function WEBSTATUSTEXTCHANGED( CtrlEntID, CtrlClassID, StatusText )

  // Example: Sets the LBL_STATUS control TEXT property to the WEBVIEW
  //          object's status text

  Call Set_Property( @Window : ".LBL_STATUS", "TEXT", StatusText )

Return TRUE$
```

See Also

WEBVIEW SETTINGS property, WEBVIEW SYNCSTATUSLINE property, WEBVIEW NAVIGATE method, WEBVIEW NAVIGATED event.

WEBSYNCSCRIPTRESULT event

Description

Returns the results of a previous synchronous call to the EXECUTESCRIPT method.

Note: This event is considered to be an "internal" system event, and as such is not intended for developer use – it is included here for documentation purposes only (see the Remarks section below).

Syntax

```
  bForward = WEBSYNCSCRIPTRESULT( CtrlEntID, CtrlClassID, ErrorCode,    |
                                                       ScriptResult )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the WEBVIEW object receiving the event.
CtrlClassID	Type of object receiving the event (always "WEBVIEW").
ErrorCode	Error code returned by the WEBVIEW object if the EXECUTESCRIPT method failed. This will be a Windows "HRESULT" error code, so a description may be obtained by using the "WIN" method in the RTI_ErrorText stored procedure (See example below).
ScriptResult	A JSON encoded string containing the result of the executed JavaScript. 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".

Returns

TRUE$ or FALSE$. If FALSE$, the program execution returns to the calling procedure. If TRUE$, the event processing goes to the next level.

Remarks

The WEBVIEW EXECUTESCRIPT method is a wrapper around the actual low-level script execution API provided by Microsoft which is implemented in a fully asynchronous fashion. This offers much better performance when running large and complex scripts, but it also makes using the method complex and tedious for small and simple requests, where a synchronous API would be much more preferable.

In order to provide such a synchronous interface for OpenInsight developers, the PS uses an internal process to wait for the results of script execution before returning control back to Basic+ from a synchronous EXECUTESCRIPT method call. One part of this process uses the WEBSYNCSCRIPTRESULT event to transfer the script results to the waiting caller and is implemented via a system promoted event handler. It has no other purpose beyond this function and is considered to be an internal system event that is not intended to be overridden by developers.

Example

N/a.

See Also

WEBVIEW EXECUTESCRIPT method, WEBVIEW WEBSCRIPTRESULT event.

WEBSUSPENDED event

Description

Occurs when the SUSPEND method has been executed and returns the results of the suspend operation.

Syntax

```
bForward = WEBSUSPENDED( CtrlEntID, CtrlClassID, SuccessFlag, ErrorCode )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the WEBVIEW object receiving the event.
CtrlClassID	Type of object receiving the event (always "WEBVIEW").
SuccessFlag	Set to TRUE$ if the WEBVIEW object was suspended successfully, or FALSE$ otherwise.
ErrrorCode	If SuccessFlag is FALSE$ then ErrorCode contains the error code that describes the reason for the failure.

Returns

TRUE$ or FALSE$. If FALSE$, the program execution returns to the calling procedure. If TRUE$, the event processing goes to the next level.

Remarks

There is no system-level promoted event handler for the WEBSUSPENDED event.

Under certain circumstances the WEBVIEW object may be prevented from being suspended but does not return an error code (SuccessFlag is FALSE$ and ErrorCode is 0). Full details of the conditions that cause this can be found in the "Sleeping Tabs FAQ" on the Microsoft website but some common reasons for this are:

- The WEBVIEW object is currently visible
- The page is currently holding a Web Lock or an IndexedDB transaction
- The page is sharing its BrowsingInstance with another page
- The page is currently being inspected by DevTools
- The page is currently playing audio
- The page is currently capturing a window or screen
- The page is currently capturing user media (webcam, microphone, etc)
- The page is currently being mirrored (casting, etc)
- The page is currently using WebUSB

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2_3 TrySuspend method on the Microsoft website.

Example

```
Function WEBSUSPENDED( CtrlEntID, CtrlClassID, SuccessFlag, ErrorCode )

  // Example: Display a message if the WEBVIEW could not be suspended.
  Declare Function RTI_ErrorText
  $Insert Logical

  Equ S_OK$ to 0 ; // From Windows "intsafe.h"

  If SuccessFlag Else

     If ( ErrorCode == S_OK$ ) Then
        // Not an "error"
        ErrorText = "Suspend Operation prevented due to browser activity"
     End Else
        ErrorText = RTI_ErrorText( "WIN", ErrorCode, TRUE$ )
     End

     Call Msg( @Window, ErrorText )

  End

Return TRUE$
```

See Also

WEBVIEW ISSUSPENDED property, WEBVIEW RESUME method, WEBVIEW SUSPEND method.

WEBTITLECHANGED event

Description

Occurs when the title of the top-level document in the WEBVIEW object changes and may run before or after the WEBNAVIGATED event.

Syntax

```
bForward = WEBTITLECHANGED( CtrlEntID, CtrlClass, DocumentTitle )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the WEBVIEW object receiving the event.
CtrlClassID	Type of object receiving the event (always "WEBVIEW").
DocumentTitle	New value for the document title.

Returns

TRUE$ or FALSE$. If FALSE$, the program execution returns to the calling procedure. If TRUE$, the event processing goes to the next level.

Remarks

There is no system-level promoted event handler for the WEBTITLECHANGED event.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView add_DocumentTitleChanged method on the Microsoft website.

Example

```
Function WEBTITLECHANGED( CtrlEntID, CtrlClassID, DocumentTitle )

   // Example: Sets the parent window caption to the WEBVIEW object's
   //          document title (i.e. emulates the SYNCTITLE property).

   Call Set_Property( @Window, "TEXT", DocumentTitle )

Return TRUE$
```

See Also

WEBVIEW DOCUMENTTITLE property, WEBVIEW SYNCTITLE property, WEBVIEW NAVIGATE method, WEBVIEW NAVIGATED event.

WEBVIEWCREATED event

Description

Occurs when the WEBVIEW object has been created and is ready to navigate.

Syntax

```
bForward = WEBVIEWCREATED( CtrlEntID, CtrlClassID )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the WEBVIEW object receiving the event.
CtrlClassID	Type of object receiving the event (always "WEBVIEW").

Returns

TRUE$ or FALSE$. If FALSE$, the program execution returns to the calling procedure. If TRUE$, the event processing goes to the next level.

Remarks

Due to the asynchronous programming model favored by the WEBVIEW object, it is preferable to use this event to begin navigation rather than wait in a loop for the INITIALIZED property to be set.

Example

```
 Function WEBVIEWCREATED( CtrlEntID, CtrlClassID )

  // Example: When this event is raised we know that the WEBVIEW control
  // is ready to navigate so we can go to our specified URL without a
  // problem now.

  $Insert Logical

  Call Exec_Method( CtrlEntID, "NAVIGATE", "https://www.revelation.com" )

Return TRUE$
```

See Also

WEBVIEW INITIALIZED property, WEBVIEW READYSTATE property, WEBVIEW NAVIGATE method.

WEBZOOMCHANGED event

Description

Occurs when the user changes the Zoom Factor via the mouse or keyboard.

Syntax

```
bForward = WEBZOOMCHANGED( CtrlEntID, CtrlClassID, ZoomFactor )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the WEBVIEW object receiving the event.
CtrlClassID	Type of object receiving the event (always "WEBVIEW").
ZoomFactor	The value that the zoom factor was changed to.

Returns

TRUE$ or FALSE$. If FALSE$, the program execution returns to the calling procedure. If TRUE$, the event processing goes to the next level.

Remarks

When the zoom factor is changed by the user that zoom applies only to the current page. Setting the ZOOMFACTOR property does not trigger a WEBZOOMCHANGED event.

For more information on this topic please refer to the Windows WebView2 documentation regarding the ICoreWebView2Controller add_ZoomFactorChanged method on the Microsoft website.

Example

```
Function WEBZOOMCHANGED( CtrlEntID, CtrlClassID, ZoomFactor )

   // Example: A WEBZOOMCHANGED event handler that sets the text of a STATIC control
   //  called "TXT_ZOOM" to show the current ZoomFactor as a percentage
   $Insert Logical

   // The ZoomFactor can be a value between 0.25 (25%) and 5.0 (500%)
   ZoomPct = "Zoom: " : ( ZoomFactor * 100 ) : "%"

   Call Set_Property_Only( @Window : ".TXT_ZOOM", ZoomPct )

Return FALSE$
```

See Also

WEBVIEW ZOOMFACTOR property.