Winwap Technologies Oy

WinWAP Browser ActiveX SDK

Application Programming Interface

v3.0 based on WinWAP Browser 4.0
WAP specification version: 2.0
Document dated: 7-Feb-2007

Notice of Confidentiality

This document contains proprietary and confidential information that belongs to Winwap Technologies Oy. The recipient agrees to maintain this information in confidence and to not reproduce or otherwise disclose this information to any person outside of the group directly responsible for the evaluation of the content.

Revision history

Date	Author	Description
13-Feb-2006	S Markelov	Starting. . .

Preamble

The reader should be familiar with at least one of the following in order to fully understand and evaluate the information given in this document:

- Basic knowledge in programming techniques.
- Basic programming knowledge in a language supporting ActiveX controls.
- Basic understanding of networking connections and client/server architecture where user-agents retrieve and render information (e.g. Internet browsers and servers with services).
- The interaction between a WAP user-agent, a WAP Gateway and a WEB Server providing mobile content in Wireless Markup Language (WML) or in Extensible HyperText Markup Language (XHTML).
- Basic knowledge in ActiveX technique.

Contents

1 How to use the WinWAP Browser ActiveX (WinWapBrowserX) control

Below is a short guide on how to use the WinWAP Browser ActiveX control:

1. Set the WinWapBrowserX connection mode to either HTTP connection mode or WAP Gateway connection mode.
2. If you set the HTTP connection mode you should also set the correct proxy settings. Leave proxy address empty if you wish use direct connections to WEB servers. The Navigate [3.2.20, p. 18] method and other methods used to navigate between WML and between XHTML pages can be called after the correct proxy settings have been set.
3. If you set the WAP Gateway connection mode you should also set the gateway IP address and gateway port number. After that the Navigate [3.2.20, p. 18] method and other methods used to navigate WML and XHTML pages can be called. If the WPSRedirect [**??**, p. **??**] event occurred, a newly provided gateway IP address and port should be set.

WinWapBrowserX library represents a WinWAP Browser ActiveX control that is created via a class factory and used as an ActiveX control in ActiveX host applications.

2 Migration from the previous versions

Beginning from the version 3.0 of the WinWAP Browser ActiveX control we changed its Type Library: API and GUID.

There are two reasons that constrain us to do so:

- The API contained large number of methods and events, that was a matter of difficulties during the Type Library compilation for some kinds of previous Microsoft Windows operating systems.
- The development platform was changed and it was a matter of impossibility to keep a syntax of a couple of methods. It causes also impossibility to use applications, that were compiled for using the previous versions of the WinWAP Browser ActiveX control, with the newest API.

The new name of the Type Library is WinWapBrowserX, accordingly the new interfaces are named IWin-WapXBrowser and IWinWapXBrowserEvents.

The Visible and AxBorderStyle properties are removed from the new API. These functionalities should be implemented in the client application, if those are needed.

The GoToURI method is not more needed and it is removed from the new API, since it duplicates the functionality of the Navigate [3.2.20, p. 18] method.

The HomePage property and GoHome method are removed from the new API, since the client application can itself implement the needed functionality.

The Reload method is renamed to Refresh [3.2.5, p. 14].

The ConnectToGateway method is not more needed to be called by a client application, since the Navigate [3.2.20, p. 18] causes connecting to a WAP Gateway automatically. But it can be called to force connecting to a WAP Gateway without resource requesting.

The ImageTypeRecognitionEnabled property was removed from the new API, since it is not needed for the newest graphical library used.

The WAP_WSPHeaderEncodingVersion property is named now as WSPHeaderEncodingVersion [3.1.28, p. 12].

The WMLSource and WMLSource2 properties are renamed to Source [3.1.12, p. 8] and Source2 [3.1.15, p. 9], since the WinWAP Browser ActiveX control shows HTML, XHTML as well as WML.

The OnHTTPAbort event was removed from new the API, since it is never raised and was never raised in the previous WinWAP Browser ActiveX versions.

The OnWPSReconnectQuery event was removed from the new API, since WinWAP Browser ActiveX controls connections to a WAP Gateway itself.

To be continued…

3 WinWapBrowserX API

3.1 Interface IWinWapXBrowser, properties

3.1.1 Enabled

The following property controls how the WinWAP Browser ActiveX control responds to mouse and keyboard events. By using Enabled the availability of the control can be changed. In case Enabled is set to false, the control is disabled. In this case the control ignores mouse and keyboard events. In order to enable the control, Enabled must be set to `VARIANT_TRUE`.

```
HRESULT Enabled([out, retval] VARIANT_BOOL *Value);
HRESULT Enabled([in] VARIANT_BOOL Value);
```

3.1.2 ConnectMode

The following property specifies the connection mode for the WinWAP Browser ActiveX control. The ConnectMode can take one of the following values:

wwmHTTP	use HTTP protocol;
wwmWPS_CL	connectionless mode using WAP gateway;
wwmWPS_CO	connection-oriented mode using WAP gateway;
wwmWPS_SCL	connectionless mode using WAP gateway;
wwmWPS_SCO	secure connection-oriented mode using WAP gateway.

Below is an example of the syntax.

```
HRESULT ConnectMode([out, retval] TxWWMode *Value);
HRESULT ConnectMode([in] TxWWMode Value);
```

3.1.3 ProxyHost, ProxyPort, ProxyUsr, ProxyPwd — Proxy settings

Specify the proxy settings. In case of direct connection (without HTTP proxy) the value of the ProxyHost must be empty (zero-length string). If user authentication is not required by the HTTP proxy the value of the ProxyUsr must be empty (zero-length string).

Below is an example of the syntax.

```
HRESULT ProxyHost([out, retval] BSTR *Value);
HRESULT ProxyHost([in] BSTR Value);

HRESULT ProxyPort([out, retval] long *Value);
HRESULT ProxyPort([in] long Value);

HRESULT ProxyUsr([out, retval] BSTR *Value);
HRESULT ProxyUsr([in] BSTR Value);

HRESULT ProxyPwd([in] BSTR Value);
```

3.1.4 GatewayAddr, GatewayPort — WAP Gateway settings

These properties specify the gateway IP address and port. The address should have the format `A.B.B.B` where `A` is a number from 1 to 255 and `B` is a number from 0 to 255. The first digits of `A` and `B`, in case if `B` is greater than 0, shall not be zero. The port is a number ranging from 0 to 65535.

Below is an example of the syntax.

```
HRESULT GatewayAddr([out, retval] BSTR *Value);
HRESULT GatewayAddr([in] BSTR Value);

HRESULT GatewayPort([out, retval] long *Value);
HRESULT GatewayPort([in] long Value);
```

3.1.5 ImgPicColor, ImgBgColor

These properties specify the WBMP images picture and background colors. Below is an example of the syntax.

```
HRESULT ImgPicColor([out, retval] OLE_COLOR *Value);
HRESULT ImgPicColor([in] OLE_COLOR Value);

HRESULT ImgBgColor([out, retval] OLE_COLOR *Value);
HRESULT ImgBgColor([in] OLE_COLOR Value);
```

3.1.6 BgStyle

This property specifies the style of the background image shown. Please use the SetBgImageFromFile [3.2.9, p. 15] or SetBgImage [3.2.10, p. 15] methods to load background bitmap image. The BgStyle takes one of the following values:

bgsNoBitmap	No background image;
bgsStretched	Stretched background image;
bgsTiled	Tiled background image;
bgsTiledAndScrolled	Tiled and stretched background image;
bgsCentered	Centered background image.

Below is an example of the syntax.

```
HRESULT BgStyle([out, retval] TxBackgroundStyle *Value);
HRESULT BgStyle([in] TxBackgroundStyle Value);
```

3.1.7 BgColor

This property specifies the WinWAP Browser ActiveX background color. Below is an example of the syntax.

```
HRESULT BgColor([out, retval] OLE_COLOR *Value);
HRESULT BgColor([in] OLE_COLOR Value);
```

3.1.8 TextColor

This property specifies the WinWAP Browser ActiveX text color. Below is an example of the syntax.

```
HRESULT TextColor([out, retval] OLE_COLOR *Value);
HRESULT TextColor([in] OLE_COLOR Value);
```

3.1.9 LinkColor

This property specifies the colors of the links in WinWAP Browser ActiveX. Below is an example of the syntax.

```
HRESULT LinkColor([out, retval] OLE_COLOR *Value);
HRESULT LinkColor([in] OLE_COLOR Value);
```

3.1.10 FontFace

This property specifies the font name for text and links in WinWAP Browser ActiveX. Below is an example of the syntax.

```
HRESULT FontFace([out, retval] BSTR *Value);
HRESULT FontFace([in] BSTR Value);
```

3.1.11 FontSize

This property specifies the font size for text and links in WinWAP Browser ActiveX. The FontSize takes one of the following values:

fsTiny	tiny font size;
fsSmall	small font size;
fsNormal	normal font size;
fsLarge	large font size;
fsHuge	huge font size;
fsMega	mega huge font size.

Below is an example of the syntax.

```
HRESULT FontSize([out, retval] TxFontSize *Value);
HRESULT FontSize([in] TxFontSize Value);
```

3.1.12 Source

This property specifies the current WML, XHTML or HTML code. Below is an example of the syntax.

```
HRESULT Source([out, retval] BSTR *Value);
```

3.1.13 TransparentImage

This property specifies whether backgrounds of WAP Bitmap (WBMP) images should be transparent or not. In case the TransparentImage property is set to `VARIANT_TRUE`, all WBMP backgrounds are transparent in the WinWAP Browser ActiveX control. Below is an example of the syntax.

```
HRESULT TransparentImage([out, retval] VARIANT_BOOL *Value);
HRESULT TransparentImage([in] VARIANT_BOOL Value);
```

3.1.14 BrowserCanGoBack, BrowserCanGoForward

These properties are read-only and specify whether the WinWAP Browser ActiveX control is allowed to go back or forward. Below is an example of the syntax.

```
HRESULT BrowserCanGoBack([out, retval] VARIANT_BOOL *Value);
HRESULT BrowserCanGoForward([out, retval] VARIANT_BOOL *Value);
```

3.1.15 Source2

This property provides the current formatted/indented WML, XHTML or HTML code. Below is an example of the syntax.

```
HRESULT Source2([out, retval] BSTR *Value);
```

3.1.16 Location

This property provides the current URL location. Below is an example of the syntax.

```
HRESULT Location([out, retval] BSTR *Value);
```

3.1.17 CardTitle

This property provides the title of the current WML card. Below is an example of the syntax.

```
HRESULT CardTitle([out, retval] BSTR *Value);
```

3.1.18 EnableNavigation

This property allows WinWAP Browser ActiveX to load resources. The control is able to load resources from the WEB in case EnableNavigation is set to `VARIANT_TRUE`. In case it is set to `VARIANT_FALSE`, the control is not able to load any resource from the WEB. Please note that this property also changes a value of the InteractionLevel [3.1.21, p. 9] property to ilNoNavigation. Below is an example of the syntax.

```
HRESULT EnableNavigation([out, retval] VARIANT_BOOL *Value);
HRESULT EnableNavigation([in] VARIANT_BOOL Value);
```

3.1.19 EnableUseXOwnPopup

This property allows the user to implement an own popup menu. Please see description of the WinWAP Browser ActiveX Popup [**??**, p. **??**] event for more information. Below is an example of the syntax.

```
HRESULT UseXOwnPopup([out, retval] VARIANT_BOOL *Value);
HRESULT UseXOwnPopup([in] VARIANT_BOOL Value);
```

3.1.20 CacheEnabled

This property enables or disables caching of received resources into a disk. Below is an example of the syntax.

```
HRESULT CacheEnabled([out, retval] VARIANT_BOOL *Value);
HRESULT CacheEnabled([in] VARIANT_BOOL Value);
```

3.1.21 InteractionLevel

This property sets the level of interaction between a user and the Graphical User Interface (GUI) to the WinWAP Browser ActiveX control.

The InteractionLevel takes one of the following values:

ilFull	user can do anything
ilNoNavigation	user can't navigate
ilReadOnly	user can't input text and press button, all operations are allowed through the WinWapBrowserX API only
ilNo	all activities are prevented.

The value ilFull (set by default) allows the user to do anything.

The value ilNoNavigation prevents all downloading of information. For more information about this please refer to the EnableNavigation [3.1.18, p. 9] property.

The value ilReadOnly allows operations via the COM interface of the WinWAP Browser ActiveX control but prevents user operations via the GUI. This means that the user can not e. g. input text, select values or press buttons.

The value ilNo prevents any operations via both the COM interface and the GUI.

Below is an example of the syntax.

```
HRESULT InteractionLevel([out, retval] TxInteractionLevel *Value);
HRESULT InteractionLevel([in] TxInteractionLevel Value);
```

3.1.22 UserAgent

This property allows the user to set a personal User-Agent HTTP header. The value of UserAgent must be formatted in accordance with RFC-2616 "Hypertext Transfer Protocol – HTTP/1.1", paragraph 14.42 "User-Agent". Below is an example of the syntax.

```
HRESULT UserAgent([out, retval] BSTR *Value);
HRESULT UserAgent([in] BSTR Value);
```

If newly assigned User-Agent field is not equal to the original WinWAP User-Agent field like "WinWAP-X/3.0 (3.0.1.70; Win32)" the string like "WinWAP-X/3.0.1.70" will be appended to the newly assigned value.

3.1.23 Headers

This property allows the user to set HTTP headers that are not controlled by the WinWAP Browser ActiveX control. The HTTP headers below are controlled by the control and the user application should not set their values in order to avoid double assignments:

1. `HTTP version`: set automatically;
2. `User-Agent`: can be set through the UserAgent [3.1.22, p. 10] property;
3. `Accept`: can be set through the AcceptTypes [3.1.30, p. 12] property;
4. `Host`: set automatically;
5. `Content-Length`: set automatically;
6. `Authorization`: set through user interaction through the modal dialog;
7. `Proxy-Authorization`: can be set through the ProxyUsr [3.1.3, p. 6] and ProxyPwd [3.1.3, p. 6] properties;
8. `Cookie`: set automatically;
9. `Referrer`: set automatically.

The value of the Headers must be formatted in accordance with RFC-2616. The set value of the Headers will be used with all following requests until the new or empty value will be set. Below is an example of the syntax.

```
HRESULT Headers([out, retval] BSTR *Value);
HRESULT Headers([in] BSTR Value);
```

3.1.24 SelectionExists

This property allows the container application to determine if the content text and/or controls are selected by the end-user. Below is an example of the syntax.

```
HRESULT SelectionExists([out, retval] VARIANT_BOOL *Value);
```

3.1.25 RcvdData

This property allows the container to get an access to the collection of downloaded resources. For more information, please refer to section Interface IWinWapBrowserXRcvdData [**??**, p. **??**]. Below is an example of the syntax.

```
HRESULT RcvdData([out, retval] IWinWapXRcvdData **Value);
```

3.1.26 EnableErrorMessages

This property makes it possible to disable or enable error messages drawn in WinWAP Browser ActiveX window. By default error messages are displayed, i.e. enabled. The property is useful, for example, when the client application has to display error messages itself. Below is an example of the syntax.

```
HRESULT EnableErrorMessages([out, retval] VARIANT_BOOL *Value);
HRESULT EnableErrorMessages([in] VARIANT_BOOL Value);
```

3.1.27 CharSetEncoding

This property allows the client application to define the default character set encoding for WML and XHTML documents. The property is especially useful for 8-bit WML and XHTML documents. Currently only the values listed below are supported by WinWAP Browser ActiveX. Character set constants are defined here: `ftp://ftp.isi.edu/in-notes/iana/assignments/character-sets`.

If this property is changed, the current document being displayed is redrawn.

The CharSetEncoding takes one of the following values:

cseAuto	character set is determined automatically by the WinWAP Browser ActiveX;
cseUS_ASCII	US-ASCII (ISO646-US);
cseISO_8859_1	ISO-8859-1 (Latin1, Western);
cseISO_8859_2	ISO-8859-2 (Latin2);
cseISO_8859_3	ISO-8859-3 (Latin3);
cseISO_8859_4	ISO-8859-4 (Latin4);
cseISO_8859_5	ISO-8859-5 (Cyrillic);
cseISO_8859_6	ISO-8859-6 (Arabic);
cseISO_8859_7	ISO-8859-7 (Greek);
cseISO_8859_8	ISO-8859-8 (Hebrew);
cseISO_8859_9	ISO-8859-9 (Latin5);
cseISO_8859_15	ISO-8859-15 (Latin-9, Western);
cseSHIFT_JIS	SHIFT_JIS (Japanese);

cseUTF8	UTF-8;
cseKOI8R	KOI8-R (Cyrillic);
cseKOI8U	KOI8-U (Cyrillic/Ukrainian);
cseWin1250	windows-1250;
cseWin1251	windows-1251;
cseWin1252	windows-1252;
cseWin1253	windows-1253;
cseWin1254	windows-1254;
cseWin1255	windows-1255;
cseWin1256	windows-1256;
cseWin1257	windows-1257;
cseBig5	Chinese for Taiwan Multi-byte set.

When the CharSetEncoding value is set to cseAuto the WinWAP Browser ActiveX control examines the "Content-Type" HTTP header, WML or XHTML meta elements or XML declarations in the WML or XHTML source.

Below is an example of the syntax.

```
HRESULT CharSetEncoding([out, retval] TxCharacterSetEncoding *pVal);
HRESULT CharSetEncoding([in] TxCharacterSetEncoding newVal);
```

3.1.28 WSPHeaderEncodingVersion

This property makes it possible for the client application to define what WAP WSP header encoding version is used. The WSPHeaderEncodingVersion takes one of the following values:

wspEncVer12	WAP WSP header encoding version 1.2;
wspEncVer13	WAP WSP header encoding version 1.3;
wspEncVer14	WAP WSP header encoding version 1.4;
wspEncVer15	WAP WSP header encoding version 1.5.

The default value is wspEncVer13, which is the WAP WSP header encoding version 1.3. Some WAP gateways don't encode HTTP headers according to the newest version of WAP WSP header encoding, so therefore the client application can try decode such headers encoded with older version of the WAP-WSP header encoding. Below is an example of the syntax:

```
HRESULT WSPHeaderEncodingVersion([out, retval] long *Value);
HRESULT WSPHeaderEncodingVersion([in] long Value);
```

3.1.29 CIRCount

This property makes it possible for the client application to define the number of simultaneous requests for an images and embedded links loader. This is a count of requests that the WinWAP Browser ActiveX simultaneously sends to the WAP gateway, HTTP proxy or WEB server. The Value can range from 1 till 10. Below is an example of the syntax.

```
HRESULT CIRCount([out, retval] long *Value);
HRESULT CIRCount([in] long Value);
```

3.1.30 AcceptTypes

This property makes it possible for the client application to specify a comma separated list of content types that are accepted by the client application. WinWAP Browser ActiveX uses this list as value for the "Accept" HTTP header.

Please remember that the value must be in one line without line breaks. The shown default value below is split in several lines in order to make it easier to read. The default value is:

```
application/vnd.wap.xhtml+xml,
application/xhtml+xml,
application/wml+xml,
application/vnd.wap.wmlc,
text/vnd.wap.wml,
image/vnd.wap.wbmp,
image/png,
image/jpeg,
image/gif,
image/tiff,
*/*
```

Below is an example of the syntax.

```
HRESULT AcceptTypes([out, retval] BSTR *Value);
HRESULT AcceptTypes([in] BSTR Value);
```

3.1.31 WAPTimerSet

This property allows to select the WAP timer intervals for network bearer supporting IP, GSM USSD or GSM SMS. The retransmission timer interval is the general of the timer intervals. If the retransmission timer interval is exceeded, WinWAP Browser ActiveX re-sends the last request. If the count of re-sent requests exceeds the specified value, WinWAP Browser ActiveX aborts the current request and replies to the client application with timeout status. The WAPTimerSet property is suitable only for WAP connection-oriented modes. Please see detailed information about timers and retransmissions in the WAP WTP specification. The `Value` takes one of the following values:

Value	Retransmission interval (seconds)	Count of retransmissions before timeout report
wapTitIp	5	8
wapTitGsmUssd	20	8
wapTitGsmSms	60	8

Below is an example of the syntax.

```
HRESULT WAPTimerSet([out, retval] TxWAPTimerSet *Value);
HRESULT WAPTimerSet([in] TxWAPTimerSet Value);
```

3.2 Interface IWinWapXBrowser, methods

3.2.1 AboutBox

This method can be used to show the dialog “About WinWAP Browser ActiveX”. The syntax is described below.

```
HRESULT AboutBox(void);
```

3.2.2 GoForward, GoBack

This method can be used to show to navigate among visited WML pages. The syntax is described below.

```
HRESULT GoForward(void);
HRESULT GoBack(void);
```

3.2.3 Search

This method can be used when the user wants to search for an item in the WAP search engine available at http://www.winwap.com/search.wml. The syntax is described below.

```
HRESULT Search(void);
```

3.2.4 Stop

This method can be used to terminate downloading of a WML or XHTML page and its content. The syntax is described below.

```
HRESULT Stop(void);
```

3.2.5 Refresh

This method can be used to reload a WML or XHTML document already displayed. The syntax is described below.

```
HRESULT Reload(void);
```

3.2.6 DisconnectFromGateway

This method can be used to force disconnecting from the WAP gateway with the address and port set in the GatewayAddr [3.1.4, p. 6] and GatewayPort [3.1.4, p. 6] properties. Following Navigate [3.2.20, p. 18] call in case of WAP Gateway connection mode will cause new WAP connection establishing. The syntax is described below.

```
HRESULT DisconnectFromGateway(void);
```

3.2.7 LoadCookies

This method can be used to load Cookies from the file `bstrPath`. The returned value `vbRes` is `VARIANT_TRUE` if cookies are successfully loaded or `VARIANT_FASLE` if the file does not exist. Note, the format of the cookie cache file is new and clear text, that is different from the binary format used in the previous versions of the WinWAP Browser ActiveX. The method's syntax is described below.

```
HRESULT LoadCookies(
    [in] BSTR bstrPath,
    [out, retval] VARIANT_BOOL *vbRes);
```

3.2.8 SaveCookies

This method can be used to save cookies to the file `bstrPath`. The returned value `vbRes` is `VARIANT_TRUE` if cookies are successfully saved to the file. The syntax is described below.

```
HRESULT SaveCookies(
    [in] BSTR bstrPath,
    [out, retval] VARIANT_BOOL *vbRes);
```

3.2.9 SetBgImageFromFile

This method can be used to load a background image from the file value `vbSuccess` is `VARIANT_TRUE` if the image has been loaded successfully. In order to clear the background image `bstrPath` has to be set to empty string. The syntax is described below.

```
HRESULT SetBgImageFromFile(
    [in] BSTR bstrPath,
    [out, retval] VARIANT_BOOL *vbSuccess);
```

3.2.10 SetBgImage

This method can be used to load a background image from byte one-dimensional array (vector) that represents device-independent bitmap image in the memory. The variant type of the `vPicture` must be `VT_ARRAY | VT_UI1`. The returned value `vbSuccess` is `VARIANT_TRUE` if the image has been loaded successfully or `VARIANT_FASLE` otherwise. To clear the background image, set `vPicture` to an array with count 0 of elements.

Below is a Visual Basic sample code that demonstrates how to pass an array of bytes between Visual Basic and an ActiveX (OLE) Control. The code sample is for demonstration purposes and does not work correctly because 512 bytes of value 50 is not the correct device-independent bitmap.

```
Dim b As Boolean
Dim buf(512) As Byte
Dim cbuf(0) As Byte

For i = 0 To 511
    buf(i) = 50
Next i
b = WinWAPX.SetBgImage(buf)
If b Then WinWAPX.BgStyle = bsCentered
.....
```

```
' Reset background
b = WinWAPX.SetBgImage(cbuf)
```

The syntax is described below.

```
HRESULT SetBgImage(
    [in] VARIANT vPicture,
    [out, retval] VARIANT_BOOL *vbSuccess);
```

3.2.11 PrintPreview

This method can be used to show a print preview dialog. The user can press the "Print" button in order to print the current document or press the pages navigation and other buttons to change scale of the previewed document. To add the WML or XHTML page location and time of printing to the bottom of the page the `vbWithLocation` has to be set to `VARIANT_TRUE`. The syntax is described below.

```
HRESULT PrintPreview([in] VARIANT_BOOL vbWithLocation);
```

3.2.12 SelectAll

This method selects all shown content in the viewable area of the WinWAP Browser ActiveX control. The syntax is described below.

```
HRESULT SelectAll(void);
```

3.2.13 CopySelected

This method copies selected content to the system clipboard. The syntax is described below.

```
HRESULT CopySelected(void);
```

3.2.14 Find

This method shows the find dialog. The end-user can search based on input text in the currently displayed document. The syntax is described below.

```
HRESULT Find(void);
```

3.2.15 FindExecute

This method searches the substring `FindString` in the currently displayed document and selects this substring if it is found. The syntax is described below.

```
HRESULT FindExecute(
  [in] BSTR FindString,
  [in] long FindOptions,
  [out, retval] VARIANT_BOOL *RetVal);
```

The parameter FindOptions can be 0 or the following values can be combined with the help of the bitwise operator OR:

fo2Up	Search upwards
fo2CaseSensitive	Search with match case
fo2WholeWord	Search whole word

The function returns VARIANT_TRUE if the substring is found and VARIANT_FALSE otherwise.

3.2.16 CacheLoad

This method sets a directory path of caching and attempts to load the cache index from the file cache.idx in this directory. If the file doesn't exist it will be created. Note, format of cached files and cache index are different from the formats of the previous versions of the WinWAP Browser ActiveX. The syntax of the method is described below.

```
HRESULT CacheLoad(
  [in] BSTR bstrDirName,
  [out, retval] VARIANT_BOOL *bSuccess);
```

The parameter bstrDirName shall contain the absolute path to the directory. If the file is locked or the user does not have write permission to this directory, the cache mechanism will be disabled and bstrDirName will return VARIANT_FALSE.

3.2.17 CacheSave

This method saves the cache index into the directory set by CacheLoad [3.2.16, p. 17] method. The parameter bSuccess returns VARIANT_TRUE if cache is saved successfully. The syntax is described below.

```
HRESULT CacheSave([out, retval] VARIANT_BOOL *bSuccess);
```

3.2.18 CacheClear

This method clears the cache in the directory set by CacheLoad [3.2.16, p. 17] method: deletes the cached files and the cache index cache.idx. If any of the cached files or cache.idx could not be deleted, the parameter bSucces returns VARIANT_FALSE. The syntax is described below.

```
HRESULT CacheClear([out, retval] VARIANT_BOOL *bSuccess);
```

3.2.19 How to use cache

First of all the cache should be initialized. To initialize the cache functionality call the CacheLoad [3.2.16, p. 17] method with proper arguments: bSuccess = CacheLoad(<dirpath>), where <dirpath> is an absolute path to the directory where the cache manager should place cached data and cache index (for example, "D:\Usr\Mike\WinWAPX Temporary Files".

If the returned value bSuccess of the CacheLoad [3.2.16, p. 17] method is VARIANT_TRUE, caching will be started.

In order to save cache index, call the CacheSave [3.2.17, p. 17] method.

In order to clear the cache, call the CacheClear [3.2.18, p. 17] method.

In order to temporarily stop caching, set CacheEnabled [3.1.20, p. 9] to VARIANT_FALSE (this does not save cache index).

In order to start caching, set the CacheEnabled [3.1.20, p. 9] property value to VARIANT_TRUE. By default the CacheEnabled [3.1.20, p. 9] value is set to to VARIANT_TRUE.

After calling the CacheLoad [3.2.16, p. 17] method the WinWAP Browser ActiveX control creates or opens an existing directory and creates or loads the cache.idx file (cache index for quick search of cached data). If the CacheLoad [3.2.16, p. 17] method can't find cache.idx file or if its content is corrupted, the method searches cached data in suitable files of the given directory and rebuilds index file cache.idx. This procedure can take some time dependent on number of cached files. All cached data is saved onto disk as 00000000000000000000 files, file names contain 20 digits. The cache index is loaded into the RAM. To save the index onto the disk you should call the CacheSave [3.2.17, p. 17] method, which saves the cache index into the file cache.idx in the directory <dirpath>.

3.2.20 Navigate

This method is syntactically identical to the same method of IWebBrowser2 interface of the Microsoft WebBrowser ActiveX control. The syntax is described below.

```
HRESULT Navigate(
    [in] BSTR URL,
    [in, optional] VARIANT *Flags,
    [in, optional] VARIANT *TargetFrameName,
    [in, optional] VARIANT *PostData,
    [in, optional] VARIANT *Headers);
```

URL — name of the resource to download.

Flags, TargetFrameName — will be ignored.

PostData — pointer to data to send with HTTP POST transaction. If this parameter does not specify any post data (set to NULL), Navigate issues HTTP GET transaction.

Headers — pointer to a value that contains content type of the HTTP post data to send to the server.

Please note that the post data specified by PostData is passed as a SAFEARRAY structure. The VARIANT should be of type VT_ARRAY and point to a SAFEARRAY. The SAFEARRAY should be of element type VT_UI1, dimension one, and have an element count equal to the number of bytes of post data. The post data content type specified by Headers is passed as a BSTR. The VARIANT should be of type VT_BSTR. If Headers are set to NULL, the default content type application/x-www-form-urlencoded would be used.

3.2.21 BindOnAddress

This method binds the WAP Stack to the specified bstrIP local network interface. If the operation fails the parameter vbSucces returns VARIANT_FALSE. The parameter bstrIP must contain an Internet Protocol (Ipv4) dotted address. The syntax is described below.

```
HRESULT BindOnAddress(
    [in] BSTR bstrIP,
    [out, retval] VARIANT_BOOL *vbSuccess);
```

To bind to all system interfaces, the bstrIP should be set to "0.0.0.0". By default when the application is started, the WAP Stack is bound to all system interfaces. In result WinWapBrowserX will receive data from any interface. If the ConnectionMode property is changed it binds the WAP Stack to all interfaces (0.0.0.0). Therefore the BindOnAddress method shall be called after assignment to this property.

Please note that you should always disconnect from the WAP gateway before BindOnAddress is called.

Please note, BindOnAddress doesn't affect on outgoing network messages, since Microsoft TCP/IP stack consult always the system routing table to find appropriate outgoing interface.

3.2.22 GetLastProcessInfo

This method provides the client application with information about the last completed process. The syntax is described below.

```
HRESULT GetLastProcessInfo(
    [in] TxProcessInfo piQuery,
    [out, retval] long *pInfo);
```

The piQuery takes one of the following values:

piElapsedTime	Request for download elapsed time
piDataSize	Request for downloaded data size

In order to retrieve information about download elapsed time, piQuery should be set to piElapsedTime. In this case pInfo will return the elapsed time in milliseconds.

In order to get the size of the downloaded data, piQuery should be set to piDataSize. In this case pInfo will return the data size in bytes.

The function can be used in the events OnAfterGetRequest [**??**, p. **??**] and OnAfterGetImage [**??**, p. **??**].

Please note that if the data source was loaded from the file cache, the pInfo will be set to 0. pInfo returns the size of the received source (WML or XHTML data, image data and WML script data) and received headers. It does not show the real transferred data through network and generally it shows less then the size of the really transferred data. In WPS mode the size of transferred data is generally smaller than the size of the same data in HTTP mode. This is because data is encoded in the WPS mode.

3.2.23 GetXVersion

This method provides the application with information about the version number of the WinWAP Browser ActiveX control. he combined string of the returned values defines a version number for the WinWAP Browser ActiveX control, for example 4.0.0.60. The syntax is described below.

```
HRESULT GetXVersion(
  [out] long *pMajor,
  [out] long *pMinor,
  [out] long *pRelease,
  [out] long *pBuild);
```

3.3 Interface IWinWapBrowserXRcvdData

The interface IWinWapBrowserXRcvdData implements the standard COM-collection interface.

The interface IWinWapBrowserXRcvdData allows to get access to a collection of downloaded resources. The resources in the collection are not processed, interpreted or decoded by the WinWAP Browser ActiveX control. The container gets those as they are received from the network or the cache.

This interface provides collection of data received by the WinWAP Browser ActiveX control as a result of some navigation method execution. For example if the WinWAP Browser ActiveX has loaded a WML page, this collection contains a source of the WML page and all images and styles referenced by this page. In case of WML events, such as `onenterforward`, this rule may not apply. In other words, the first resource item in the collection is generally the last requested WML resource, and other resource items are images, scripts and styles.

The resource collection is available at any time as it is copied to the container. This happens via a request through the IWinWapXBrowser::RcvdData [**??**, p. **??**] method. It is recommended to call it in WinWAP Browser ActiveX control event handlers such as IWinWapBrowserXEvents::AfterRequest [**??**, p. **??**], IWinWapBrowserXEvents::ContentType [**??**, p. **??**] or after IWinWapBrowserXEvents::Complete [**??**, p. **??**].

3.3.1 Item

The property Item allows access to the indexed collection item. The items are indexed from 1. Please refer to section IWinWapBrowserXRcvdDataItem [3.4, p. 22] for more information about collection items. The syntax is described below.

```
HRESULT Item([in] long index, [out, retval] IWinWapXRcvdDataItem **pVal);
```

3.3.2 Count

The property Count returns the count of items in the collection. In case of processing events such as AfterRequest [**??**, p. **??**] or ContentType [**??**, p. **??**] the item corresponding to the event is indexed with Count. The syntax is described below.

```
HRESULT Count([out, retval] long *pVal);
```

3.3.3 Sample code for MS Visual Basic

```
Private Sub WinWapX_OnComplete(ByVal nFlags As Long, ByVal nData As Long)
    Dim i As Integer
    Dim s As String
    Dim m As WinWapBrowserXRcvdData ' resource collection
    Dim item As WinWapBrowserXRcvdDataItem ' resource item

    ' Display items count
    Set m = WinWapX.RcvdData
    s = "Received: " + CStr(m.Count) + " items" + Chr(13) + Chr(10)

    ' Display content types of all resources (using For Each)
    i = 1
    For Each item In m
        s = s + CStr(i) + ": " + item.ContentType + Chr(13) + Chr(10)
        i = i + 1
    Next

    s = s + "=================================" + Chr(13) + Chr(10)

    ' Display URL locations of all resources (using For To)

    For i = 1 To m.Count
        s = s + CStr(i) + ": " + m.item(i).URL + Chr(13) + Chr(10)
    Next
    TextRcvData.Text = s
End Sub
```

3.4 Interface IWinWapBrowserXRcvdDataItem

The interface IWinWapBrowserXRcvdDataItem makes it possible for the client application to get resource data and its attributes such as content type, HTTP headers, original location and real location, from which resource data was loaded.

3.4.1 ContentType

The property ContentType returns the content type of the resource. The syntax is described below.

```
HRESULT ContentType([out, retval] BSTR *pVal);
```

3.4.2 URL

The property URL returns the resource location from which the resource was really loaded. The syntax is described below.

```
HRESULT URL([out, retval] BSTR *pVal);
```

3.4.3 Data

The property Data returns the downloaded resource. The resource data is returned as `VARIANT` of type `VT_ARRAY|VT_UI1`. The syntax is described below.

```
HRESULT Data([out, retval] VARIANT *pVal);
```

3.4.4 Hdrs

The property Hdrs returns the HTTP headers sent by a WEB-server as a response on requested resource. The syntax is described below.

```
HRESULT Hdrs([out, retval] BSTR *pVal);
```

3.4.5 Src

The property Src returns the resource location originally got from the WML, XHTML source or IWinWapXBrowser::Navigate [3.2.20, p. 18] method. This address can be changed if a WEB-server redirects request to another location. The syntax is described below.

```
HRESULT Src([out, retval] BSTR *pVal);
```

Index