

HTTPCLIENT object

The HTTPCLIENT object allows an OpenInsight application to issue HTTP and HTTPS requests from Basic+. It can retrieve text or binary data, submit request bodies, supply authentication and proxy settings, and download response content directly to a file.



Developer Notes

1. HTTPCLIENT is a Presentation Server control implemented using the Windows WinINet API. It transfers data; it does not display or execute web-page content. Use a WEBVIEW object when an embedded browser is required
2. Equates for this object are provided by PS_HTTPCLIENT_EQUATES, which also includes MSWIN_WININET_EQUATES and PS_STATIC_EQUATES.

Request sequence

Configure the request while the object is inactive, call OPEN to prepare it, then call SEND. After completion, inspect RESPONSESTATUS and retrieve the body through GETRESPONSECONTENT, RESPONSETEXT or the file specified by RESPONSEFILE.

Most request properties ignore changes while ACTIVE is TRUE\$. OPEN makes the object active, so configure headers, timeouts, proxy settings and the response destination before opening it. One HTTPCLIENT object processes one request at a time; use separate controls for independent simultaneous requests. Synchronous requests

Synchronous requests

In synchronous mode, SEND returns after response reception finishes or an error occurs. Check the method result first and then the HTTP status. A successful method return does not imply an HTTP 2xx status.

```
Declare Function Exec_Method, Get_Property
$insert Logical

CtrlEntID = @Window : ".HCL_REQUEST"
Call Set_Property_Only( CtrlEntID, "ASYNCMODE", FALSE$ )
Call Set_Property_Only( CtrlEntID, "TIMEOUT", 60000 )
Call Set_Property_Only( CtrlEntID, "ACCEPTTYPES", "*/*" )
Call Set_Property_Only( CtrlEntID, "RESPONSEFILE", "" )

Success = Exec_Method( CtrlEntID, "OPEN", "GET", |
    "https://api.example.com/resource" )

If Success Then
    Success = Exec_Method( CtrlEntID, "SEND" )
End

If Success Then
    Status = Get_Property( CtrlEntID, "RESPONSESTATUS" )
    If Status<1> >= 200 And Status<1> < 300 Then
        Response = Exec_Method( CtrlEntID, "GETRESPONSECONTENT" )
        // Decode or process Response for the expected media type.
    End Else
        // Handle the HTTP status and any error response body.
    End
End Else
    ErrorText = Get_Property( CtrlEntID, "ERRORTEXT" )
    // Report or record ErrorText here.
End
```

Response content and character encoding

GETRESPONSECONTENT returns the response bytes without text conversion and is suitable for binary content and explicitly decoded text. RESPONSETEXT decodes the received bytes using the charset declared in Content-Type, with Windows ANSI fallback when the charset is absent. UTF-8 and UTF-16 are supported. Unsupported charsets or failed conversions return null and set ERRORTEXT; decoded embedded

null characters also return null. When RESPONSEFILE is set, the body is written to disk instead of being retained in memory.

Asynchronous requests

Set ASYNCMODE to TRUE\$ before OPEN. An asynchronous SEND returns TRUE\$ after queuing the operation; this return does not confirm that the server accepted the request or that the download succeeded. Return from the initiating event handler so the Presentation Server can run the operation and its callbacks.

Use READystateCHANGED to detect normal completion and PROGRESS to update a progress display. BytesExpected can be zero when the response has no usable Content-Length, so a progress handler must allow an unknown total. OPEN itself can wait for connection setup even in asynchronous mode.

```
$Insert Logical

Call Set_Property_Only( CtrlEntID, "ASYNCMODE", TRUE$ )
Call Set_Property_Only( CtrlEntID, "TIMEOUT", 60000 )
Success = Exec_Method( CtrlEntID, "OPEN", "GET", |
                      "https://api.example.com/resource" )

If Success Then
    Queued = Exec_Method( CtrlEntID, "SEND" )
End Else
    ErrorText = Get_Property( CtrlEntID, "ERRORTEXT" )
End

// Return to the Presentation Server to process the request.
```

Errors and cancellation

Keep transport errors separate from HTTP status codes. ERRORTEXT describes local or WinINet failures. RESPONSESTATUS describes a response received from the server and also records the local cancellation code 477. A server-side 4xx or 5xx response does not by itself raise REQUESTERROR.

REQUESTERROR reports asynchronous WinINet completion errors, immediate send/receive failures and response-file write/close failures. TIMEOUT reports expiry of an asynchronous control wait. Handles and open response files are closed before these events, allowing handlers to reopen the client. Also check OPEN and SEND return values for immediate setup, validation or queuing failures.

ABORT closes the request and resets an active READystate to 0, making the client configurable again. Configure any changed settings, then use OPEN and SEND to retry. OPEN also aborts an existing request before applying its optional arguments. Cancellation does not raise a completion event; event return values do not cancel the transfer.

HTTPCLIENT Properties

The HTTPCLIENT object supports the following properties. Common inherited properties are described in the Common Object API and Common GUI Object API chapters.

Name	Description
ACCEPTTYPES	Specifies the media types accepted in the response.
ACCESSTYPE	Specifies how the client obtains its connection and proxy settings.
ACTIVE	Indicates whether the client is in an active request state.
ASYNCMODE	Specifies whether requests are sent using asynchronous processing.
ERRORTEXT	Returns diagnostic text from the most recent operation that recorded an error.
PASSWORD	Specifies the password used for server authentication.
PORT	Specifies the port used to connect to the target server.
PROXYBYPASSNAMES	Specifies the destinations that should bypass the named proxy.
PROXYNAMES	Specifies the proxy servers used by the client.
PROXYPASSWORD	Specifies the password used to authenticate with the proxy server.
PROXYUSERNAME	Specifies the user name used to authenticate with the proxy server.
READYSTATE	Returns the current stage of the request.
REFERRER	Specifies the referring URL supplied with the request.
REQUESTHANDLES	Returns the native WinINet handles owned by the control.
REQUESTHEADERS	Specifies additional HTTP headers to send with the request.
REQUESTOPTIONS	Specifies the WinINet options used when opening a request.
RESPONSEFILE	Specifies a file into which the response body is downloaded.
RESPONSEHEADERS	Returns the response headers, or the value of a named response header.
RESPONSESTATUS	Returns the response status and advertised content length.
RESPONSETEXT	Returns the in-memory response body as text.
TIMEOUT	Specifies the timeout interval used by the client.
URL	Specifies the target URL for the request.
USERAGENT	Specifies the User-Agent identification used for the request.
USERNAME	Specifies the user name used for server authentication.
VERB	Specifies the HTTP method used for the request.
WORKOFFLINE	Specifies whether the client should use offline operation.

ACCEPTTYPES property

Description

Specifies the media types accepted in the response.

Property Value

An @Fm-delimited list of media type strings, such as "application/json", "text/plain" or "*/*". The native default is null.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

Set this property before calling OPEN. Changes are ignored while ACTIVE is TRUE\$, including the interval between OPEN and SEND. Empty list entries are ignored. Setting null clears the list. Use "*/*" when any response type is acceptable; this setting does not decode the response body.

Example

```
Types = "application/json" : @Fm : "text/plain"  
Call Set_Property_Only( CtrlEntID, "ACCEPTTYPES", Types )
```

See Also

REQUESTHEADERS property, RESPONSETEXT property, GETRESPONSECONTENT method.

ACCESSTYPE property

Description

Specifies how the client obtains its connection and proxy settings.

Property Value

An integer value from the following table. The default is 0. Unsupported numeric values are changed to 0.

Value	Description
0	Preconfigured. Use Windows connection settings.
1	Direct. Connect without a configured proxy.
3	Proxy. Use the named proxy settings.
4	Preconfigured without automatic configuration scripts.

Property Traits

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

Remarks

Set this property before calling OPEN. Changes are ignored while ACTIVE is TRUE\$, including the interval between OPEN and SEND. PROXYNAMES and PROXYBYPASSNAMES are used when ACCESSTYPE is 3. The Windows constants are available through MSWIN_WININET_EQUATES, which is included by PS_HTTPCLIENT_EQUATES.

For more information on this property please refer to the InternetOpenW function on the Microsoft Website.

Example

```
Call Set_Property_Only( CtrlEntID, "ACCESSTYPE", 3 )
Call Set_Property_Only( CtrlEntID, "PROXYNAMES", "proxy.example.com:8080" )
```

See Also

PROXYNAMES property, PROXYBYPASSNAMES property, OPEN method.

ACTIVE property

Description

Indicates whether the client is in an active request state.

Property Value

A Boolean value: TRUE\$ for READYSTATE 1, 2 or 3; FALSE\$ for state 0 or 4.

Property Traits

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

Remarks

ACTIVE becomes TRUE\$ when OPEN succeeds, before SEND is called. It describes the request state, not current network traffic. Completion returns it to FALSE\$. Cancelling or closing a failed active request resets READYSTATE to 0 and makes configuration properties writable again. ACTIVE does not indicate whether the HTTP response status is successful.

Example

```
IsActive = Get_Property( CtrlEntID, "ACTIVE" )
State = Get_Property( CtrlEntID, "READYSTATE" )
```

See Also

READYSTATE property, ERRORTXT property, ABORT method.

ASYNCMODE property

Description

Specifies whether requests are sent using asynchronous processing.

Property Value

A Boolean value of TRUE\$ or FALSE\$. The native default is FALSE\$.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

Set this property before calling OPEN. Changes are ignored while ACTIVE is TRUE\$, including the interval between OPEN and SEND. In synchronous mode SEND waits for the response body. In asynchronous mode SEND queues the operation and returns; handle READYSTATECHANGED and PROGRESS to follow it. OPEN may still wait while preparing its connection and request handles. Return control to the Presentation Server after an asynchronous SEND so it can process the queued operation.

Example

```
$Insert Logical  
Call Set_Property_Only( CtrlEntID, "ASYNCMODE", TRUE$ )
```

See Also

SEND method, READYSTATECHANGED event, PROGRESS event, REQUESTERROR event.

ERRORTEXT property

Description

Returns diagnostic text from the most recent operation that recorded an error.

Property Value

A string, initially null. Invalid request sequencing can return "Invalid HTTP Request State". Other failures can return a Windows error message.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
N/a	Get	No	No	No

Remarks

Read this property immediately after a failed operation. OPEN resets it; SEND, response-header queries, response reception and TESTCONNECTION also clear it before proceeding. A subsequent call can therefore replace or erase an earlier diagnostic. Reading ERRORTEXT itself does not clear it. HTTP response codes such as 404 or 500 are reported separately by RESPONSESTATUS.

Example

```
Success = Exec_Method( CtrlEntID, "SEND" )  
  
If Success Else  
    ErrorText = Get_Property( CtrlEntID, "ERRORTEXT" )  
End
```

See Also

RESPONSESTATUS property, REQUESTERROR event, OPEN method, SEND method.

PASSWORD property

Description

Specifies the password used for server authentication.

Property Value

A string, initially null. It is paired with USERNAME.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

Set this property before calling OPEN. Changes are ignored while ACTIVE is TRUE\$, including the interval between OPEN and SEND. Setting null clears the stored password. A non-empty Password argument to OPEN updates this property before opening the request. An omitted or empty OPEN argument leaves the existing value unchanged; clear it with SET_PROPERTY while inactive when required.

Example

```
Call Set_Property_Only( CtrlEntID, "USERNAME", UserName )
Call Set_Property_Only( CtrlEntID, "PASSWORD", Password )
```

See Also

USERNAME property, PROXYPASSWORD property, OPEN method.

PORT property

Description

Specifies the port used to connect to the target server.

Property Value

An unsigned port number. The initial value is 80. URL selects the default port for its scheme (80 for HTTP, 443 for HTTPS), or the explicit port supplied in the URL, with or without a path.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

Set this property before calling OPEN. Changes are ignored while ACTIVE is TRUE\$, including the interval between OPEN and SEND. Set URL before setting a PORT override. With an explicit HTTPS URL, changing PORT preserves secure mode; with an explicit HTTP URL, PORT clears secure mode even for port 443. With a schemeless URL, port 443 enables secure mode and other ports clear it. A later REQUESTOPTIONS assignment can change the secure flag; see that property for ordering details.

Example

```
Call Set_Property_Only( CtrlEntID, "USERNAME", UserName )
Call Set_Property_Only( CtrlEntID, "PASSWORD", Password )
```

See Also

URL property, REQUESTOPTIONS property, OPEN method.

PROXYBYPASSNAMES property

Description

Specifies the destinations that should bypass the named proxy.

Property Value

An @Fm-delimited list of host names, IP addresses or wildcard patterns. The default is null.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

Set this property before calling OPEN. Changes are ignored while ACTIVE is TRUE\$, including the interval between OPEN and SEND. This list is used only when ACCESTYPE is 3. The special value "<local>" bypasses the proxy for host names without a period. Empty entries are ignored; setting null clears the list.

Example

```
Bypass = "<local>" : @Fm : "*.example.com"  
Call Set_Property_Only( CtrlEntID, "PROXYBYPASSNAMES", Bypass )
```

See Also

ACCESTYPE property, PROXYNAMES property.

PROXYNAMES property

Description

Specifies the proxy servers used by the client.

Property Value

An @Fm-delimited list of proxy specifications accepted by WinINet, for example "proxy.example.com:8080". The default is null.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

Set this property before calling OPEN. Changes are ignored while ACTIVE is TRUE\$, including the interval between OPEN and SEND. Set ACCESTYPE to 3 to use this list. Setting null clears it. The list is joined with spaces when passed to WinINet. Server credentials and proxy credentials are separate settings.

Example

```
Call Set_Property_Only( CtrlEntID, "ACCESTYPE", 3 )
Call Set_Property_Only( CtrlEntID, "PROXYNAMES", "proxy.example.com:8080" )
```

See Also

ACCESTYPE property, PROXYBYPASSNAMES property, PROXYUSERNAME property.

PROXYPASSWORD property

Description

Specifies the password used to authenticate with the proxy server.

Property Value

A string, initially null. It is paired with PROXYUSERNAME.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

Set this property before calling OPEN. Changes are ignored while ACTIVE is TRUE\$, including the interval between OPEN and SEND. Setting null clears the stored value. This property does not change PASSWORD, which supplies credentials for the target server.

Example

```
Call Set_Property_Only( CtrlEntID, "PROXYUSERNAME", ProxyUser )
Call Set_Property_Only( CtrlEntID, "PROXYPASSWORD", ProxyPassword )
```

See Also

PROXYUSERNAME property, PROXYNAMES property, PASSWORD property.

PROXYUSERNAME property

Description

Specifies the user name used to authenticate with the proxy server.

Property Value

A string, initially null.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

Set this property before calling OPEN. Changes are ignored while ACTIVE is TRUE\$, including the interval between OPEN and SEND. Setting null clears the stored value. Set PROXYPASSWORD separately. These credentials are applied to the connection during OPEN.

Example

```
Call Set_Property_Only( CtrlEntID, "PROXYUSERNAME", ProxyUser )
```

See Also

PROXYPASSWORD property, PROXYNAMES property, USERNAME property.

READYSTATE property

Description

Returns the current stage of the request.

Property Value

An integer value from 0 to 4, as shown below. The initial value is 0.

Value	Description
0	Unsent. Initial state, initialization in progress, or an active request closed by cancellation or failure.
1	Opened. OPEN succeeded; SEND may now be called.
2	Headers received. Response status and headers are available.
3	Loading. The response body is being downloaded.
4	Done. Response reception has completed.

Property Traits

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

Remarks

State 4 indicates that reception completed, not that the server returned a successful HTTP status. Inspect RESPONSESTATUS separately. Closing a failed or cancelled active request resets the state to 0 without raising READYSTATECHANGED during cleanup. ABORT after completion leaves state 4 unchanged. PS_HTTPCLIENT_EQUATES defines PS_HCL_RS_UNSENT\$, PS_HCL_RS_OPENED\$, PS_HCL_RS_HEADERS_RECEIVED\$, PS_HCL_RS_LOADING\$ and PS_HCL_RS_DONE\$.

Example

```
$Insert PS_HTTPCLIENT_EQUATES

State = Get_Property( CtrlEntID, "READYSTATE" )

If State = PS_HCL_RS_DONE$ Then
    Status = Get_Property( CtrlEntID, "RESPONSESTATUS" )
End
```

See Also

ACTIVE property, RESPONSESTATUS property, READYSTATECHANGED event.

REFERRER property

Description

Specifies the referring URL supplied with the request.

Property Value

A URL string. The default is null, which means no referrer is supplied through this property.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

Set this property before calling OPEN. Changes are ignored while ACTIVE is TRUE\$, including the interval between OPEN and SEND. Set the URL of the resource from which the request originated when the receiving application requires this information. Setting null clears the value.

Example

```
Call Set_Property_Only( CtrlEntID, "REFERRER", "https://app.example.com/" )
```

See Also

URL property, REQUESTHEADERS property, OPEN method.

REQUESTHANDLES property

Description

Returns the native WinINet handles owned by the control.

Property Value

An @Fm-delimited array of three handle values.

Value	Description
<1>	Session handle returned by InternetOpen.
<2>	Connection handle returned by InternetConnect.
<3>	Request handle returned by HttpOpenRequest.

Property Traits

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

Remarks

This is an advanced, read-only runtime property included in the HTTPCLIENT property list. Handles belong to the control and can be closed or replaced by OPEN, ABORT, errors or object destruction. Do not close these handles yourself or retain them across requests.

Example

```
Handles      = Get_Property( CtrlEntID, "REQUESTHANDLES" )
RequestHandle = Handles<3>
```

See Also

OPEN method, ABORT method, ACTIVE property.

REQUESTHEADERS property

Description

Specifies additional HTTP headers to send with the request.

Property Value

An @Fm-delimited array. Each field contains a header name in value 1 and its value in value 2, separated by @Vm. The default is null.

Value	Description
<n,1>	Header name.
<n,2>	Header value.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

Set this property before calling OPEN. Changes are ignored while ACTIVE is TRUE\$, including the interval between OPEN and SEND. Setting this property replaces the entire header list. Setting null clears it. Supply the header name without a colon; the control inserts the separator when sending. REQUESTHEADERS describes the configured additional headers, not all headers generated by WinINet.

Example

```
Headers          = ""
Headers<1,1>     = "Content-Type"
Headers<1,2>     = "application/json"
Headers<2,1>     = "Accept"
Headers<2,2>     = "application/json"
Call Set_Property_Only( CtrlEntID, "REQUESTHEADERS", Headers )
```

See Also

ACCEPTTYPES property, USERAGENT property, SEND method.

REQUESTOPTIONS property

Description

Specifies the WinINet options used when opening a request.

Property Value

At runtime this is an @Fm-delimited array of Boolean flags, not a numeric bit mask. The field positions are listed below. Native flags initially are zero; URL and PORT can enable secure mode.

Field	Option	Meaning
1	UseCacheIfNetFail	Use cached content for certain connection failures.
2	HyperLink	Reload when expiry and modification information are absent.
3	IgnoreCertCNInvalid	Skip certificate host-name checking.
4	IgnoreCertDateInvalid	Skip certificate validity-date checking.
5	IgnoreRedirectToHTTP	Allow HTTPS-to-HTTP redirects transparently.
6	IgnoreRedirectToHTTPS	Allow HTTP-to-HTTPS redirects transparently.
7	KeepConnection	Request connection keep-alive.
8	AllowTempFile	Allow a temporary file for non-cacheable content.
9	NoAuth	Disable automatic authentication.
10	NoAutoRedirect	Disable automatic redirect handling.
11	NoCacheWrite	Do not cache the returned resource.
12	NoCookies	Disable automatic cookie handling.
13	NoUI	Disable the cookie dialog.
14	PragmaNoCache	Resolve through the origin rather than a proxy cache.
15	Reload	Fetch from the origin rather than local cache.
16	Resync	Reload a modified resource.
17	UseSecure	Enable a secure HTTP request.

Each field is a Boolean value. Fields 3 and 4 disable certificate checks; field 5 permits a redirect to an unencrypted HTTP URL. Enable these only when that behavior is intended. AllowTempFile controls WinINet temporary storage and is separate from RESPONSEFILE.

Property Traits

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

Remarks

Set this property before calling OPEN. Changes are ignored while ACTIVE is TRUE\$, including the interval between OPEN and SEND. A non-empty array replaces the options. Omitted trailing flags are cleared; a null property value leaves them unchanged. The getter returns 17 fields; the setter ignores fields beyond position 17. When this array is assigned, port 443 forces field 17 on. At other ports, include field 17 when secure mode is required. Assigning URL or PORT afterwards recalculates the

secure flag from the scheme and port. The Form Designer stores its options as a bit mask and converts their display separately.

For more information on this property please refer to the `HttpOpenRequestW` function on the Microsoft Website.

Example

```
// Construct the documented array directly.  
Options = ""  
Options<11> = 1 ; // NoCacheWrite  
Options<15> = 1 ; // Reload  
Options<17> = 1 ; // UseSecure  
Call Set_Property_Only( CtrlEntID, "REQUESTOPTIONS", Options )
```

See Also

ACCEPTTYPES property, USERAGENT property, SEND method.

RESPONSEFILE property

Description

Specifies a file into which the response body is downloaded.

Property Value

A file path. The default is null, which stores the body in memory.

Property Traits

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

Remarks

Set this property before calling OPEN. Changes are ignored while ACTIVE is TRUE\$, including the interval between OPEN and SEND. OPEN opens the destination for binary output and truncates an existing file to zero length. Choose a path whose existing contents may be replaced. If the file cannot be opened, OPEN returns FALSE\$ and ERRORTTEXT is "Cannot open response output file". The file is closed on completion, cancellation or request failure. Write and final-close failures set ERRORTTEXT and prevent successful completion; asynchronous failures raise REQUESTERROR. A failed or cancelled transfer may leave a partial file. With this property set, the body is not retained for GETRESPONSECONTENT or RESPONSETEXT.

Example

```
// The destination directory must already exist.  
Call Set_Property_Only( CtrlEntID, "RESPONSEFILE", "C:\Downloads\report.pdf" )
```

See Also

GETRESPONSECONTENT method, RESPONSETEXT property, PROGRESS event.

RESPONSEHEADERS property

Description

Returns the response headers, or the value of a named response header.

Property Value

Without an index, returns the raw response-header block with CR/LF pairs replaced by @Fm. The HTTP status line and trailing empty fields may be present. With a header-name index, returns the value of that header.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
N/a	Get	Yes	No	No

Remarks

Use this property after READystate reaches 2 and while the request handle remains valid. The full list contains text lines such as "Content-Type: application/json"; it does not use the @vm'd name/value format of REQUESTHEADERS. Query failures return null and set ERRORTXT. A named lookup is equivalent to GETRESPONSEHEADER.

Example

```
Headers      = Get_Property( CtrlEntID, "RESPONSEHEADERS" )
ContentType = Get_Property( CtrlEntID, "RESPONSEHEADERS", "Content-Type" )
```

See Also

GETRESPONSEHEADER method, RESPONSESTATUS property, ERRORTXT property.

RESPONSESTATUS property

Description

Returns the response status and advertised content length.

Property Value

An @Fm-delimited array with the following fields. OPEN clears the previous values to code 0, null text and length 0.

Value	Description
<1>	Numeric HTTP status code.
<2>	Status description returned by the server.
<3>	Advertised response length in bytes.

Property Traits

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

Remarks

Inspect the status code even when SEND succeeds. HTTP error responses are still valid HTTP exchanges. Content length comes from the response Content-Length header; zero can mean that no length was supplied and does not prove that the body is empty. ABORT records code 477 and text "User Cancelled". This is a local status, not a response from the server.

Example

```
Status      = Get_Property( CtrlEntID, "RESPONSESTATUS" )
StatusCode   = Status<1>
StatusText   = Status<2>
ContentLength = Status<3>
```

See Also

ERRORTXT property, READYSTATE property, ABORT method, PROGRESS event.

RESPONSETEXT property

Description

Returns the in-memory response body as text.

Property Value

A string decoded from the received bytes using the charset declared in the response Content-Type header. When the charset is absent or empty, the Windows ANSI code page is used. The result is null when no in-memory body is available or the decoded text contains a null character.

Property Traits

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

Remarks

Read after successful completion; this property does not start or finish a download. Charset names are case-insensitive and may be quoted. UTF-8, UTF-16 and Windows MIME charset aliases are supported. UTF-8/UTF-16 byte-order marks are removed when using those encodings. Only received bytes are decoded; unused buffer space is ignored.

An unsupported declared charset, a conversion failure or an incomplete UTF-16 byte sequence returns null and sets ERRORTEXT. Unsupported charsets do not fall back to ANSI. The response charset remains available after handles close. RESPONSEFILE bypasses the in-memory body. Use GETRESPONSECONTENT for original bytes or an application-selected decoder.

Example

```
// Uses the declared charset, for example UTF-8.  
ResponseText = Get_Property( CtrlEntID, "RESPONSETEXT" )
```

See Also

GETRESPONSECONTENT method, RESPONSEFILE property, RESPONSEHEADERS property.

TIMEOUT property

Description

Specifies the timeout interval used by the client.

Property Value

An integer interval in milliseconds. The native default and the Form Designer default PS_HCL_DFLT_TIMEOUT\$ are both 180000 (180 seconds). Zero or Windows INFINITE selects unlimited control waits. Negative runtime values are converted to INFINITE.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

Set this property before calling OPEN. Changes are ignored while ACTIVE is TRUE\$, including the interval between OPEN and SEND. The value is used for the WinINet connection-timeout option and each asynchronous control wait; it is not a total-request deadline. Zero is passed as INFINITE for the connection-timeout option. Expiry of a control wait closes the request, sets ERRORTXT and raises TIMEOUT with its source identifier. Underlying network operations can still fail or time out independently; an asynchronous WinINet error is reported through REQUESTERROR.

Example

```
// Set a timeout of 120 seconds.  
Call Set_Property_Only( CtrlEntID, "TIMEOUT", 120000 )
```

See Also

TIMEOUT event, ERRORTXT property, ASYNCMODE property.

URL property

Description

Specifies the target URL for the request.

Property Value

A URL string, initially null. Use an HTTP or HTTPS URL with forward slashes, for example "https://api.example.com/resource". A missing path is treated as "/".

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

Set this property before calling OPEN. Changes are ignored while ACTIVE is TRUE\$, including the interval between OPEN and SEND. Each assignment replaces the previous parsed host, path, port and secure setting. HTTPS enables secure mode, including on an explicit custom port; HTTP clears it. Explicit ports are accepted with or without a path. Query strings are sent and URL fragments are omitted; bracketed IPv6 hosts are supported. Schemeless URLs default to HTTP, with the existing port-443 secure-mode convention. Set any PORT override after URL. No request is sent until OPEN and SEND are called.

Example

```
Call Set_Property_Only( CtrlEntID, "URL", "https://api.example.com:8443/" )
```

See Also

PORT property, REQUESTOPTIONS property, OPEN method.

USERAGENT property

Description

Specifies the User-Agent identification used for the request.

Property Value

A string. The native default is "RevPS HttpClient". If the stored string is null, the getter attempts to obtain the Windows URLMON user-agent string.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

Set this property before calling OPEN. Changes are ignored while ACTIVE is TRUE\$, including the interval between OPEN and SEND. GET_PROPERTY returns the effective value, so after clearing this property it may return a non-empty Windows user-agent string. Set an explicit application identifier when a stable value is required.

Example

```
Call Set_Property_Only( CtrlEntID, "USERAGENT", "ExampleApplication/1.0" )
```

See Also

REQUESTHEADERS property, OPEN method.

USERNAME property

Description

Specifies the user name used for server authentication.

Property Value

A string, initially null. It is paired with PASSWORD.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

Set this property before calling OPEN. Changes are ignored while ACTIVE is TRUE\$, including the interval between OPEN and SEND. A non-empty UserName argument to OPEN updates this property. An omitted or empty OPEN argument preserves the existing value. Setting null with SET_PROPERTY clears the stored value. Proxy authentication uses PROXYUSERNAME and PROXYPASSWORD instead.

Example

```
Call Set_Property_Only( CtrlEntID, "USERNAME", UserName )
Call Set_Property_Only( CtrlEntID, "PASSWORD", Password )
```

See Also

PASSWORD property, PROXYUSERNAME property, OPEN method.

VERB property

Description

Specifies the HTTP method used for the request.

Property Value

A string. The native default is "GET". Common values include GET, HEAD, POST, PUT, DELETE, OPTIONS and PATCH.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

Set this property before calling OPEN. Changes are ignored while ACTIVE is TRUE\$, including the interval between OPEN and SEND. Use the uppercase method required by the server. The runtime stores the supplied string rather than restricting it to the Form Designer list. Changing VERB does not send a request. For methods with a body, pass the encoded body to SEND and set the appropriate Content-Type header before OPEN.

Example

```
Call Set_Property_Only( CtrlEntID, "VERB", "POST" )
```

See Also

REQUESTHEADERS property, OPEN method, SEND method.

WORKOFFLINE property

Description

Specifies whether the client should use offline operation.

Property Value

A Boolean value of TRUE\$ or FALSE\$. The native default is FALSE\$.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

Set this property before calling OPEN. Changes are ignored while ACTIVE is TRUE\$, including the interval between OPEN and SEND. TRUE\$ enables the WinINet offline flag when the session is opened. This uses cached resources rather than establishing normal online access. A resource may be unavailable when it is not present in the cache. Cache and reload behavior can also be affected by REQUESTOPTIONS.

Example

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

See Also

REQUESTOPTIONS property, ACCESTYPE property, OPEN method.

HTTPCLIENT Methods

The HTTPCLIENT object supports the following methods:

Name	Description
ABORT	Attempts to cancel the current request and closes its WinINet handles.
GETRESPONSECONTENT	Returns the response body as a Basic+ byte string without text conversion.
GETRESPONSEHEADER	Returns the value of a named response header.
OPEN	Initializes a new request and prepares its session, connection and request handles.
SEND	Sends the prepared HTTP request and an optional request body.
TESTCONNECTION	Asks WinINet to attempt a general Internet connection.

ABORT Method

Description

Attempts to cancel the current request and closes its WinINet handles.

Syntax

```
Success = Exec_Method( CtrlEntID, "ABORT" )
```

Parameters

N/a.

Returns

TRUE\$. The method cancels an active request; this return value does not indicate a completed download.

Remarks

If READystate is not 0, the method closes the session, connection and request handles, closes any open response file, and records status 477 with text "User Cancelled". An active READystate (1, 2 or 3) becomes 0 and ACTIVE becomes FALSE\$; a completed state 4 remains unchanged. Received bytes are retained, and a partial output file may remain. Cleanup does not raise READystateCHANGED or a completion event. Calling ABORT in state 0 does nothing and still returns TRUE\$.

Example

```
Success = Exec_Method( CtrlEntID, "ABORT" )  
Status  = Get_Property( CtrlEntID, "RESPONSESTATUS" )
```

See Also

ACTIVE property, READystate property, RESPONSESTATUS property, OPEN method.

GETRESPONSECONTENT Method

Description

Returns the response body as a Basic+ byte string without text conversion.

Syntax

```
Response = Exec_Method( CtrlEntID, "GETRESPONSECONTENT" )
```

Parameters

N/a.

Returns

The bytes held in the response buffer. Before response headers are available, or when no in-memory body is present, the result is null.

Remarks

Normally call this after successful synchronous SEND or after READYSTATE reaches 4. If headers are available and reception has not completed, this method attempts to receive the remaining body; it is not merely a passive buffer read in that state. A failed receive can leave partial content, so check ERRORTXT and completion before treating the result as a complete response. The method does not parse JSON, decode a character set or interpret HTTP status. With RESPONSEFILE set, read the file instead.

Example

```
Response = Exec_Method( CtrlEntID, "GETRESPONSECONTENT" )  
ByteCount = BLen( Response )
```

See Also

RESPONSETEXT property, RESPONSEFILE property, RESPONSESTATUS property, SEND method.

GETRESPONSEHEADER Method

Description

Returns the value of a named response header.

Syntax

```
HeaderValue = Exec_Method( CtrlEntID, "GETRESPONSEHEADER", HeaderName )
```

Parameters

Name	Required	Description
HeaderName	Yes	Name of the response header, for example Content-Type.

Returns

The header value, or null if the name is empty, the header cannot be obtained, or the request is in an invalid state.

Remarks

READYSTATE must be at least 2 and the request handle must still be valid. A lookup failure records ERRORTXT. A missing or empty HeaderName returns null without performing a query. This method is equivalent to an indexed RESPONSEHEADERS lookup. It returns a header value, not a parsed media type or a name/value array.

Example

```
ContentType = Exec_Method( CtrlEntID, "GETRESPONSEHEADER", "Content-Type" )  
  
If ContentType = "" Then  
    ErrorText = Get_Property( CtrlEntID, "ERRORTXT" )  
End
```

See Also

RESPONSEHEADERS property, ERRORTXT property, READYSTATE property.

OPEN Method

Description

Initializes a new request and prepares its session, connection and request handles.

Syntax

```
Success = Exec_Method( CtrlEntID, "OPEN", Verb, URL, |  
                        Async, UserName, Password )
```

Parameters

Name	Required	Description
Verb	No	HTTP method. Non-empty values update VERB.
URL	No	Target URL. Non-empty values update URL.
Async	No	TRUE\$ or FALSE\$. Updates ASYNCMODE when supplied as a non-empty value.
UserName	No	Server user name. Non-empty values update USERNAME.
Password	No	Server password. Non-empty values update PASSWORD.

Returns

TRUE\$ when initialization succeeds and the request remains open, or FALSE\$ on failure or cancellation. Read ERRORTXT immediately on failure. Successful initialization leaves READYSTATE at 1.

Remarks

Arguments omitted or passed as null leave their properties unchanged. Pass 0 (FALSE\$), rather than null, to explicitly select synchronous mode. Configure other request properties before OPEN. OPEN clears the previous response, status and error information, and prepares RESPONSEFILE if set. It does not transmit the HTTP request. ASYNCMODE does not make OPEN entirely nonblocking. If an earlier request remains active, OPEN aborts it before applying the supplied replacement arguments. To change properties without OPEN arguments, call ABORT, set the properties and then call OPEN. Callback-registration failure returns FALSE\$.

Example

```
Success = Exec_Method( CtrlEntID, "OPEN", "GET", |  
                      "https://api.example.com/resource", 0 )
```

See Also

ACTIVE property, URL property, VERB property, ASYNCMODE property, SEND method.

SEND Method

Description

Sends the prepared HTTP request and an optional request body.

Syntax

```
Success = Exec_Method( CtrlEntID, "SEND", RequestContent )
```

Parameters

Name	Required	Description
RequestContent	No	Raw request-body bytes. Omit or pass null for an empty body.

Returns

In synchronous mode, TRUE\$ if sending and response reception succeed, or FALSE\$ if either reports a failure. In asynchronous mode, TRUE\$ means the operation was queued, not completed.

Remarks

Call OPEN successfully before SEND. The send operation requires READystate 1. The body is sent as supplied; set its encoding and Content-Type before OPEN. SEND does not serialize Basic+ arrays to JSON or perform form encoding. In asynchronous mode the body is copied for later transmission; return to the Presentation Server to allow processing. Use READystateCHANGED and PROGRESS for normal progress, REQUESTERROR for asynchronous failures, and TIMEOUT for expired control waits. In either mode, check RESPONSESTATUS to determine whether the HTTP operation succeeded at the application level.

Example

```
// Prepare headers and body before OPEN.
Headers = ""
Headers<1,1> = "Content-Type"
Headers<1,2> = "application/json"
Call Set_Property_Only( CtrlEntID, "REQUESTHEADERS", Headers )
Body = '{"name":"Example"}' ; // ASCII JSON payload
Success = Exec_Method( CtrlEntID, "OPEN", "POST", "https://api.example.com/items", 0 )

If Success Then
    Success = Exec_Method( CtrlEntID, "SEND", Body )
End
```

See Also

OPEN method, REQUESTHEADERS property, RESPONSESTATUS property, GETRESPONSECONTENT method.

TESTCONNECTION Method

Description

Asks WinINet to attempt a general Internet connection.

Syntax

```
Success = Exec_Method( CtrlEntID, "TESTCONNECTION" )
```

Parameters

N/a.

Returns

TRUE\$ if the object is already active or WinINet reports success; FALSE\$ if the connection attempt fails.

Remarks

This is not an HTTP request to URL and does not validate a particular host, path, certificate or HTTP status. It clears ERRORTTEXT before testing and records an error message on failure. An active object returns TRUE\$ immediately without a fresh network test. Use OPEN and SEND to test the actual endpoint.

Example

```
Success = Exec_Method( CtrlEntID, "TESTCONNECTION" )  
  
If Success Else  
    ErrorText = Get_Property( CtrlEntID, "ERRORTTEXT" )  
End
```

See Also

ACTIVE property, ERRORTTEXT property, OPEN method, SEND method.

HTTPCLIENT Events

The following events describe HTTP request activity. PROGRESS and READystateCHANGED are generated in asynchronous mode. REQUESTERROR reports asynchronous operation failures and TIMEOUT reports expired control waits. Their entries describe cleanup and recovery behaviour.

Name	Description
PROGRESS	Reports the bytes received during an asynchronous response download.
READystateCHANGED	Reports a request-state assignment during asynchronous processing.
REQUESTERROR	Reports an asynchronous request or response-output failure.
TIMEOUT	Reports expiry of an asynchronous control wait.

PROGRESS event

Description

Reports the bytes received during an asynchronous response download.

Syntax

```
bForward = PROGRESS( CtrlEntID, CtrlClassID, |  
                     BytesReceived, BytesDownloaded, BytesExpected )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the HTTPCLIENT object receiving the event.
CtrlClassID	Type of object receiving the event (always "HTTPCLIENT").
BytesReceived	Bytes received by the most recent read.
BytesDownloaded	Cumulative bytes downloaded so far.
BytesExpected	Advertised Content-Length, or 0 when unavailable.

Returns

TRUE\$ to forward normal event processing, or FALSE\$ to stop forwarding. The transfer does not use this value as a cancellation flag.

Remarks

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

This event is raised after non-empty response reads in asynchronous mode. For RESPONSEFILE downloads, BytesDownloaded is the current output-file position; otherwise it is the in-memory byte count. BytesExpected is not guaranteed to be known or to match the eventual downloaded size. Do not divide by it unless it is greater than zero. A PROGRESS return value does not cancel reception; use ABORT for cancellation.

Example

```
Function PROGRESS( CtrlEntID, CtrlClassID, BytesReceived, |
                  BytesDownloaded, BytesExpected )

    If BytesExpected > 0 Then
        Percent = Int( BytesDownloaded * 100 / BytesExpected )
        // Update the application's progress display.
    End Else
        // Show BytesDownloaded without a percentage.
    End

    Return 1
```

See Also

ASYNCMODE property, RESPONSEFILE property, RESPONSESTATUS property, READYSTATECHANGED event.

READYSTATECHANGED event

Description

Reports a request-state assignment during asynchronous processing.

Syntax

```
bForward = READYSTATECHANGED( CtrlEntID, CtrlClassID, NewState )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the HTTPCLIENT object receiving the event.
CtrlClassID	Type of object receiving the event (always "HTTPCLIENT").
NewState	The new READYSTATE value, from 0 to 4.

Returns

TRUE\$ to forward normal event processing, or FALSE\$ to stop forwarding. The transfer does not use this value as a cancellation flag.

Remarks

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

Normal processing reports states 0, 1, 2, 3 and 4 as the request is initialized, opened, sent and received. The state is assigned before the event is raised. State 4 is the normal completion notification; inspect RESPONSESTATUS before accepting the response as successful. State 0 can be reported during initialization even if the previous value was already 0. The event is not raised for synchronous requests, and a failure or ABORT need not produce state 4. Cleanup of an active request resets READYSTATE to 0 without raising this event. If a handler aborts or reopens the control, the interrupted operation stops using the previous request.

Example

```
Function READYSTATECHANGED( CtrlEntID, CtrlClassID, NewState )

    Declare Function Get_Property, Exec_Method
    $Insert PS_HTTPCLIENT_EQUATES

    If NewState = PS_HCL_RS_DONE$ Then
        Status = Get_Property( CtrlEntID, "RESPONSESTATUS" )
        If Status<1> >= 200 And Status<1> < 300 Then
            Response = Exec_Method( CtrlEntID, |
                                    "GETRESPONSECONTENT" )
            // Process Response, or the RESPONSEFILE destination.
        End Else
            // Handle the returned HTTP status.
        End
    End

    Return 1
```

See Also

READYSTATE property, ACTIVE property, RESPONSESTATUS property, SEND method.

REQUESTERROR event

Description

Reports an asynchronous request or response-output failure.

Syntax

```
bForward = REQUESTERROR( CtrlEntID, CtrlClassID, ErrorText )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the HTTPCLIENT object receiving the event.
CtrlClassID	Type of object receiving the event (always "HTTPCLIENT").
ErrorText	Error description recorded for the failed request.

Returns

TRUE\$ to forward normal event processing, or FALSE\$ to stop forwarding. The transfer does not use this value as a cancellation flag.

Remarks

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

In asynchronous mode this event reports failed pending WinINet operations, immediate send/receive errors, failed control waits, and response-file write/close errors. Handles and any open response file are closed before dispatch; an active request becomes inactive. A handler may configure and reopen the client.

Expiry of a control wait raises TIMEOUT instead. HTTP 4xx and 5xx status codes are not by themselves REQUESTERROR conditions. Check method return values for setup or validation failures that occur before the asynchronous operation. Preserve ErrorText before another operation can clear ERRORTXT. The event return value does not retry the request.

Example

```
Function REQUESTERROR( CtrlEntID, CtrlClassID, ErrorText )  
  
    // Save the supplied diagnostic for application reporting.  
    Call Set_Property_Only( @Window, "@HTTP_ERROR", ErrorText )  
  
    Return 1
```

See Also

ERRORTXT property, RESPONSESTATUS property, TIMEOUT event, SEND method.

TIMEOUT event

Description

Reports expiry of an asynchronous control wait.

Syntax

```
bForward = TIMEOUT ( CtrlEntID, CtrlClassID, StatusID )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the HTTPCLIENT object receiving the event.
CtrlClassID	Type of object receiving the event (always "HTTPCLIENT").
StatusID	Timeout source identifier, as shown below. 0 : Unknown 1 : Connection Wait 2 : Request-handle creation wait. 3 : Request-completion wait, including send/read operations.

Returns

TRUE\$ to forward normal event processing, or FALSE\$ to stop forwarding. The transfer does not use this value as a cancellation flag.

Remarks

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

Raised when the control wait for connection setup, request-handle creation or request completion expires. ERRORTTEXT receives the Windows timeout diagnostic. Handles and any open response file are closed before dispatch, and an active request becomes inactive. A handler may configure and reopen the client.

TIMEOUT measures an individual control wait, not total request duration. Zero or INFINITE disables this wait limit. An underlying WinINet timeout is reported as an operation error through REQUESTERROR. Local wait expiry does not set RESPONSESTATUS to HTTP 408. The event return value does not retry the request.

Example

```
Function TIMEOUT( CtrlEntID, CtrlClassID, StatusID )  
  
    // Record the expired wait source for application reporting.  
    Call Set_Property_Only( @Window, "@HTTP_TIMEOUT", StatusID )  
  
Return 1
```

See Also

TIMEOUT property, ERRORTTEXT property, REQUESTERROR event.