Web Areas

Version 11.2 of 4D allows a new type of area to be added to your forms: **Web Areas**. These areas can display any type of Web content within your 4D environment: HTML pages with static or dynamic contents, files, pictures, Javascript, Flash, PDF, etc.

Several new standard actions and numerous language commands allow the developer to control the functioning of these Web areas. Specific variables can be used to exchange information between the area and the 4D environment. This means that you can benefit from a basic Web browser in your forms.

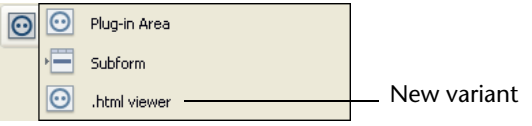
During execution, Web areas can also have a contextual menu and standard drag and drop functions.

The rendering engine of the Web area will depend on the execution platform of the application:

- Under Mac OS X, 4D uses the Apple WebKit engine (same rendering as the Safari browser),
- Under Windows, 4D uses the ActiveX Web Controls (same rendering as the Internet Explorer browser).

Creating a Web Area

A Web area is created using a new variant of the Plug-in Area/Subform button found in the object bar of the 4D Form editor:



This new variant is only available when you have the appropriate license.

To add a Web area to your form, select the **Web Area** button and then draw the area. It is possible to create several Web areas in the same form.

Like other dynamic form objects, a Web area has an object name and a variable name, which can be used to handle it by programming. The standard variable associated with a Web area object is a Longint. More specifically, you can use the SET VISIBLE and MOVE OBJECT commands with Web areas.

Mac OS Specificities

The use of Web areas under Mac OS requires specific conditions related to the system: the protocol must mandatorily be included in the requested URL and the window containing the area must be in "compositing" mode.

Protocol Included in URL

The URLs handled by programming in Web areas under Mac OS must begin with the protocol. For example, you need to pass the string "http://www.mysite.com" and not just "www.mysite.com".

Compositing Mode

In order to be displayed, the Web areas must be included in windows designed in "compositing mode." This internal mode for handling windows under Mac OS is not used in all the 4D windows.

In 4D v11 SQL, the windows designed in "compositing mode" are:

- All the windows generated by the Open form window command;
- The windows generated by the Open window command having the *type* Compositing Mode (constant value 4096);
- In Design mode, (execution via the  button), windows displaying a project form.

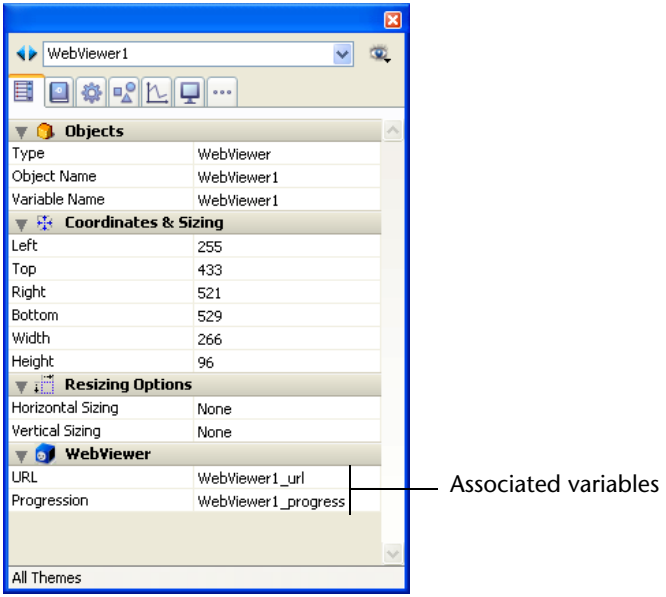
Note Certain former generation objects are not compatible with the compositing mode (for example 4D Chart areas). If they are displayed in windows in compositing mode, these objects will not work.

Associated Variables

In addition to the standard object variable, two specific variables are automatically associated with each Web area:

- The "URL" variable
- The "Progression" variable.

By default, these variables are named, respectively, areaName_url and areaName_progress. You can change these names if desired. These variables can be accessed in the Property List:



URL Variable

"URL" is a String type variable. It contains the URL loaded or being loading by the associated Web area.

The association between the variable and the Web area works in both directions:

- If the user assigns a new URL to the variable, this URL is automatically loaded by the Web area.
- Any browsing done within the Web area will automatically update the contents of the variable.

Schematically, this variable functions like the address area of a Web browser. You can represent it via a text area above the Web area.

Note For access to documents, this variable only accepts URLs that are RFC-compliant ("file://c:/MyDoc") and not system pathnames ("c:\MyDoc"). Note that the WA OPEN URL command accepts both notations (refer to the description of this command).

Progression Variable

"Progression" is a Longint type variable. It contains a value between 0 and 100, representing the percentage of loading that is complete for the page displayed in the Web area.

This variable is automatically updated by 4D. It is not possible to modify it manually.

Standard Actions

Several new standard actions are available for managing Web areas automatically. These actions can be associated with buttons or menu commands and allow quick implementation of basic Web interfaces.

- **Open Back URL**: This action opens the previous URL in the browsing sequence carried out by the user in the Web area. If there is no previous URL, in other words, if the user has only displayed a single page in the area, the associated button or menu command is disabled.
- **Open Next URL**: This action opens the next URL in the browsing sequence carried out by the user in the Web area. If there is no next URL, in other words, if the user has never gone back a page in the sequence, the associated button or menu command is disabled.
- **Refresh Current URL**: This action reloads the current contents of the Web area.
- **Stop Loading URL**: This action stops loading the page and/or objects of the current URL in the Web area.

Form Events

New form events have been added in 4D in order to allow developers to control various aspects of Web areas. In addition, Web areas are compatible with several existing form events.

New Events

New form events are available for Web areas.

- **On Begin URL Loading**
  This event is generated at the start of loading a new URL in the Web area. The "URL" variable associated with the Web area can be used to find out the URL being loaded.

Note The URL being loaded is different from the current URL (refer to the description of the WA Get current URL command).

- **On URL Resource Loading**
  This event is generated each time a new resource (picture, frame, etc.) is loaded on the current Web page.

The "Progression" variable associated with the area lets you find out the current state of the loading.

- **On End URL Loading**
  This event is generated when all the resources of the current URL have been loaded.
  You can call the WA Get current URL command in order to find out the URL that was loaded.
- **On URL Loading Error**
  This event is generated when an error is detected during the loading of a URL.
  You can call the WA GET LAST URL ERROR command in order to get information about the error.
- **On URL Filtering**
  This event is generated when the loading of a URL is blocked by the Web area because of a filter set up using the WA SET URL FILTERS command.
  You can find out the blocked URL using the WA Get last filtered URL command.
- **On Open External Link**
  This event is generated when the loading of a URL was blocked by the Web area and the URL was opened with the current system browser, because of a filter set up via the WA SET EXTERNAL LINKS FILTERS command.
  You can find out the blocked URL using the WA Get last filtered URL command.
- **On Window Opening Denied**
  This event is generated when the opening of a pop-up window is blocked by the Web area. 4D Web areas do not allow the opening of pop-up windows.
  You can find out the blocked URL using the WA Get last filtered URL command.

Compatible Events

The following form events can be used with Web areas:

- On Load
- On Unload
- On Getting Focus
- On Losing Focus
- On Drag Over

- On Drop
- On Begin Drag Over

User Interface

When the form is executed, standard browser interface functions are available to the user in the Web area, which permit interaction with other form areas:

- **Edit menu commands**: When the Web area has the focus, the **Edit** menu commands can be used to carry out actions such as copy, paste, select all, etc., according to the selection.
- **Contextual menu**: It is possible to use the standard contextual menu of the system with the Web area.
- **Drag and drop**: The user can drag and drop text and pictures within the Web area or between a Web area and the 4D form objects, according to the 4D object properties.

Web Area Commands

More than twenty new commands allow you to manage Web areas using programming. These commands are found in the new "Web Area" theme.

WA OPEN URL

WA OPEN URL({*;}object; url)

Parameter	Type		Description
*	*	→	If specified, object is an object name (string) If omitted, object is a table or a variable
object	Form object	→	Object name (if * is specified) or Table or variable (if * is omitted)
url	String	→	URL to load in Web area

The WA OPEN URL command loads the URL passed in the *url* parameter into the Web area designated by the * and *object* parameters.

If an empty string is passed in *url*, the WA OPEN URL command does nothing and no error is generated. To load a blank page into the Web area, pass the string "about:blank" in *url*.

Like the existing OPEN WEB URL command, WA OPEN URL accepts several types of syntaxes in the *url* parameter to designate the files:

- posix syntax: "file://c:/MyFile"
- system syntax: "c:\MyFolder\MyFile" (Windows) or "MyDisk:MyFolder:MyFile" (Mac OS).

This command has the same effect as modifying the value of the "URL" variable associated with the area. For example, if the variable of the area is named *MyWArea_url*:

MyWArea_url:="http://www.4d.com/"

is the same as:

WA OPEN URL(MyWArea;"http://www.4d.com/")

WA Get current URL

WA Get current URL ({*;}object) → String

Parameter	Type		Description
*	*	→	If specified, object is an object name (string) If omitted, object is a table or a variable
object	Form object	→	Object name (if * is specified) or Table or variable (if * is omitted)
Function result	String	←	URL currently loaded in the Web area

The WA Get current URL command returns the URL address of the page displayed in the Web area designated by the * and *object* parameters.

If the current URL is not available, the command returns an empty string.

If the Web page is completely loaded, the value returned by the function is the same as that of the "URL" variable associated with the Web area. If the page is in the process of being loaded, the two values will be different: the function returns the completely loaded URL and the variable contains the URL in the process of being loaded.

▸ The page displayed is the URL "www.apple.com" and the "www.4d.com" page is in the process of being loaded:

```
$url:=WA Get current URL(MyWArea) `returns "http://www.apple.com"
   `The URL variable associated contains "http://www.4d.com"
```

WA OPEN BACK URL

WA OPEN BACK URL({*;}object)

Parameter	Type		Description
*	*	→	If specified, object is an object name (string) If omitted, object is a table or a variable
object	Form object	→	Object name (if * is specified) or Table or variable (if * is omitted)

The WA OPEN BACK URL command loads the previous URL in the sequence of URLs opened into the Web area designated by the * and *object* parameters.

If there is no previous URL, the command does nothing. You can test whether a previous URL is available using the WA Back URL available command.

WA Back URL available

WA Back URL available ({*;}object) → Boolean

Parameter	Type		Description
*	*	→	If specified, object is an object name (string) If omitted, object is a table or a variable
object	Form object	→	Object name (if * is specified) or Table or variable (if * is omitted)
Function result	Boolean	←	True if there is a previous URL in the sequence of URLs opened; otherwise, False

The WA Back URL available command can be used to find out whether there is a previous URL available in the sequence of URLs opened in the Web area designated by the * and *object* parameters.

The command returns True if a URL exists and False otherwise. More particularly, this command can be used, in a custom interface, to enable or disable navigation buttons.

WA OPEN FORWARD URL

WA OPEN FORWARD URL ({*;}object)

Parameter	Type		Description
*	*	→	If specified, object is an object name (string) If omitted, object is a table or a variable
object	Form object	→	Object name (if * is specified) or Table or variable (if * is omitted)

The WA OPEN FORWARD URL command loads the forward URL in the sequence of URLs opened into the Web area designated by the * and *object* parameters.

If there is no forward URL (in other words, if the user has never returned to a previous URL), the command does nothing. You can test whether a forward URL is available using the WA Forward URL available command.

WA Forward URL available

WA Forward URL available ({*;}object) → Boolean

Parameter	Type		Description
*	*	→	If specified, object is an object name (string) If omitted, object is a table or a variable
object	Form object	→	Object name (if * is specified) or Table or variable (if * is omitted)
Function result	Boolean	←	True if there is a following URL in the sequence of URLs opened; otherwise, False

The WA Forward URL available command can be used to find out whether there is a forward URL available in the sequence of URLs opened in the Web area designated by the * and *object* parameters.

The command returns True if a URL exists and False otherwise. More particularly, this command can be used, in a custom interface, to enable or disable navigation buttons.

WA REFRESH CURRENT URL

WA REFRESH CURRENT URL ({*;}object)

Parameter	Type		Description
*	*	→	If specified, object is an object name (string) If omitted, object is a table or a variable
object	Form object	→	Object name (if * is specified) or Table or variable (if * is omitted)

The WA REFRESH CURRENT URL command reloads the current URL displayed in the Web area designated by the * and *object* parameters.

WA STOP LOADING URL

WA STOP LOADING URL ({*;}object)

Parameter	Type		Description
*	*	→	If specified, object is an object name (string) If omitted, object is a table or a variable
object	Form object	→	Object name (if * is specified) or Table or variable (if * is omitted)

The WA STOP LOADING URL command stops loading the resources of the current URL of the Web area designated by the * and *object* parameters.

WA Execute JavaScript

WA Execute JavaScript ({*;}object; jsCode) → String

Parameter	Type		Description
*	*	→	If specified, object is an object name (string) If omitted, object is a table or a variable
object	Form object	→	Object name (if * is specified) or Table or variable (if * is omitted)
jsCode	String	→	JavaScript code
Function result	String	←	Result of execution

The WA Execute JavaScript command executes, in the Web area designated by the * and *object* parameters, the JavaScript code passed in *jsCode*.

Under Mac OS, the command returns the result.
Under Windows, the command returns an empty string. Use the WA EXECUTE JAVASCRIPT FUNCTION command.

- Example

$result:=**WA Execute JavaScript**(MyWArea;"history.back()")

WA EXECUTE JAVASCRIPT FUNCTION

WA EXECUTE JAVASCRIPT FUNCTION ({*;}object; jsFunction; result|*{; param1;...;paramN})

Parameter	Type		Description
*	*	→	If specified, object is an object name (string) If omitted, object is a table or a variable
object	Form object	→	Object name (if * is specified) or Table or variable (if * is omitted)
jsFunction	String	→	Name of JavaScript function to execute
result|*	Variable	→ ←	Function result (if expected) or * for a function with no result
param	String	→	Parameter(s) to pass to function

The WA EXECUTE JAVASCRIPT FUNCTION command executes, in the Web area designated by the * and *object* parameters, the JavaScript function *jsFunction* and optionally returns its result in the *result* parameter.

If the function does not return a result, pass * in the *result* parameter.

You can pass one or more strings containing the parameters of the function in *param*.

- Calling a JavaScript function with 3 parameters:

```
$JavaScriptFunction:="TheFunctionToBeExecuted"
$Param1:="10"
$Param2:="true"
$Param3:="1,000.2" `note "," as thousands separator and "." as the decimal separator

WA EXECUTE JAVASCRIPT FUNCTION(MyWArea; $JavaScriptFunction;
            $Result; $Param1; $Param2; $Param3)
```

WA SET URL FILTERS

WA SET URL FILTERS ({*;}object; filtersArr; allowDenyArr)

Parameter	Type		Description
*	*	→	If specified, object is an object name (string) If omitted, object is a table or a variable
object	Form object	→	Object name (if * is specified) or Table or variable (if * is omitted)
filtersArr	Array string	→	Filters array
allowDenyArr	Array Boolean	→	Allow-deny array

The WA SET URL FILTERS command can be used to set up one or more filters for the Web area designated by the * and *object* parameters.

Before loading any page requested by the user, 4D consults the list of filters in order to check whether or not the target URL is allowed. The evaluation of the URL is based on the contents of the *filtersArr* and *allowDenyArr* arrays.

If the requested URL is not allowed, it is not loaded and the On URL Filtering form event is generated (see the "Form Events" paragraph on page 4).

The *filtersArr* and *allowDenyArr* arrays must be synchronized.

- Each element of the *filtersArr* array must contain a URL to be filtered. You can use the * as a wildcard to replace one or more characters.
- Each corresponding element in the *allowDenyArr* array must contain a Boolean indicating whether the URL must be allowed (True) or denied (False).

If there is a contradiction at the configuration level (the same URL is both allowed and denied), the last setting is the one taken into account.

To disable URL filtering, call the command and pass empty arrays or pass, respectively, the values "*" and True in the last elements of the *filtersArr* and *allowDenyArr* arrays.

Once the command has been executed, the filters become a property of the Web area. If the *filtersArr* and *allowDenyArr* arrays are deleted or reinitialized, the filters remain active as long as the command has not been executed again. To find out the active filters for an area, you must use the WA GET URL FILTERS command.

Important: The filtering of URLs carried out by this command only applies to the "URL" variable associated with the Web area (variable usually enterable and displayed in the form).
The filtering does not apply to the WA OPEN URL command, nor to the other navigation commands.

- You want to deny access for all the .org, .net and .fr Web sites.

```
ARRAY TEXT($filters;0)
ARRAY BOOLEAN($AllowDeny;0)
APPEND TO ARRAY($filters;"*.org")
APPEND TO ARRAY($AllowDeny;False)
APPEND TO ARRAY($filters;"*.net")
APPEND TO ARRAY($AllowDeny;False)
APPEND TO ARRAY($filters;"*.fr")
APPEND TO ARRAY($AllowDeny;False)
WA SET URL FILTERS(MyWArea;$filters;AllowDeny)
```

- You want to deny access for all Web sites except Russian ones (.ru):

```
ARRAY TEXT($filters;0)
ARRAY BOOLEAN($AllowDeny;0)

APPEND TO ARRAY($filters;"*") `Select all
APPEND TO ARRAY($AllowDeny;False) `Deny all
APPEND TO ARRAY($filters;"www.*.ru") `Select *.ru
APPEND TO ARRAY($AllowDeny;True) `Allow
```

- You want to allow access only to 4D Web sites (.com, .fr, .es, etc.):

```
APPEND TO ARRAY($filters;"*") `Select all
APPEND TO ARRAY($AllowDeny;False) `Deny all
APPEND TO ARRAY($filters;"www.4D.*") `Select 4d.fr, 4d.com...
APPEND TO ARRAY($AllowDeny;True) `Allow
```

- You want to allow local access to the documentation only (found in the folder C://doc):

```
APPEND TO ARRAY($filters;"*") `Select all
APPEND TO ARRAY($AllowDeny;False) `Deny all
APPEND TO ARRAY($filters;"file://C:/doc/*")
      `Select the path file:// allowed
APPEND TO ARRAY($AllowDeny;True) `Allow
```

- You want to allow access for all sites except one, for example the Elcaro site:

```
APPEND TO ARRAY($filters;"*")
APPEND TO ARRAY($AllowDeny;True) `Allow all
APPEND TO ARRAY(*elcaro*") `Deny all that contain elcaro
APPEND TO ARRAY($AllowDeny;False)
```

- You want to deny access to specific IP addresses:

```
APPEND TO ARRAY($filters;"*") `Select all
APPEND TO ARRAY($AllowDeny;True) `Allow all
APPEND TO ARRAY(86.83.*") `Select IP addresses beginning with 86.83.
APPEND TO ARRAY($AllowDeny;False) `Deny
APPEND TO ARRAY(86.1*") `Select IP addresses beginning with 86.1
                                        (86.10, 86.135 etc.)
APPEND TO ARRAY($AllowDeny;False) `Deny
```

(Note that the IP address of a domain may vary).

See Also: WA GET URL FILTERS

WA GET URL FILTERS

WA GET URL FILTERS({*;}object; filtersArr; allowDenyArr)

Parameter	Type		Description
*	*	→	If specified, object is an object name (string) If omitted, object is a table or a variable
object	Form object	→	Object name (if * is specified) or Table or variable (if * is omitted)
filtersArr	Array string	←	Filters array
allowDenyArr	Array Boolean	←	Allow-deny array

The WA GET URL FILTERS returns, in the *filtersArr* and *allowDenyArr* arrays, the filters that are active in the Web area designated by the * and *object* parameters. If no filter is active, the arrays are returned empty.

The filters are installed by the WA SET URL FILTERS command. If the arrays are reinitialized during the session, the WA GET URL FILTERS command can be used to find out the current settings.

See Also: WA SET URL FILTERS

WA SET EXTERNAL LINKS FILTERS

WA SET EXTERNAL LINKS FILTERS({*;}object; filtersArr; allowDenyArr)

Parameter	Type		Description
*	*	→	If specified, object is an object name (string) If omitted, object is a table or a variable
object	Form object	→	Object name (if * is specified) or Table or variable (if * is omitted)
filtersArr	Array string	→	Filters array
allowDenyArr	Array Boolean	→	Allow-deny array

The WA SET EXTERNAL LINKS FILTERS command can be used to set up one or more external link filters for the Web area designated by the * and *object* parameters. External link filters determine whether a URL associated with the current page via a link must be opened in the Web area or in the default Web browser of the machine.

When the user clicks on a link in the current page, 4D consults the list of external link filters in order to check whether the URL requested must be opened in the browser of the machine. If so, the page corresponding to the URL is displayed in the Web browser and the On Open External Link form event is generated (see the "Form Events" paragraph on page 4). Otherwise (default operation), the page corresponding to the URL is displayed in the Web area. The evaluation of the URL is based on the contents of the *filtersArr* and *allowDenyArr* arrays.

The *filtersArr* and *allowDenyArr* arrays must be synchronized.

- Each element of the *filtersArr* array must contain a URL to be filtered. You can use the * as a wildcard to replace one or more characters.
- Each corresponding element in the *allowDenyArr* array must contain a Boolean indicating whether the URL must be opened in the Web area (True) or in the Web browser (False).

If there is a contradiction at the configuration level (the same URL is both allowed and denied), the last setting is the one taken into account.

To disable URL filtering, call the command and pass empty arrays or pass, respectively, the values "*" and True in the last elements of the *filtersArr* and *allowDenyArr* arrays.

Important: The filtering established by the WA SET URL FILTERS command is taken into account before that of the WA SET EXTERNAL LINKS FILTERS command. This means that if a URL is denied because of a WA SET URL FILTERS command filter, it cannot be opened in the browser even if it is explicitly specified by the WA SET EXTERNAL LINKS FILTERS command (see example 2).

▸ This example causes sites to be opened in external browsers:

```
ARRAY STRING(0;$filters;0)
ARRAY BOOLEAN($AllowDeny;0)
APPEND TO ARRAY($filters;"*www.google.*") `Select "google"
APPEND TO ARRAY($AllowDeny;False)
            `False: this link will be opened in an external browser
APPEND TO ARRAY($filters;"*www.apple.*")
APPEND TO ARRAY($AllowDeny;False)
            `False: this link will be opened in an external browser
WA SET EXTERNAL LINKS FILTERS(MyWArea;$filters;$AllowDeny)
```

- This example combines the filtering of both sites and external links:

```
ARRAY STRING(0;$filters;0)
ARRAY BOOLEAN($AllowDeny;0)
APPEND TO ARRAY($filters;"*www.google.*") `Select "google"
APPEND TO ARRAY($AllowDeny;False) `Deny this link
WA SET URL FILTERS(MyWArea;$filters;$AllowDeny)

ARRAY STRING(0;$filters;0)
ARRAY BOOLEAN($AllowDeny;0)
APPEND TO ARRAY($filters;"*www.google.*") `Select "google"
APPEND TO ARRAY($AllowDeny;False)
   `False: this link should be opened in an external browser but this setting
   `has no effect because the link will be blocked by the URL filtering.
WA SET EXTERNAL LINKS FILTERS(MyWArea;$filters;$AllowDeny)
```

WA GET EXTERNAL LINKS FILTERS

WA GET EXTERNAL LINKS FILTERS({*;}object; filtersArr; allowDenyArr)

Parameter	Type		Description
*	*	→	If specified, object is an object name (string) If omitted, object is a table or a variable
object	Form object	→	Object name (if * is specified) or Table or variable (if * is omitted)
filtersArr	Array string	←	Filters array
allowDenyArr	Array Boolean	←	Allow-deny array

The WA GET EXTERNAL LINKS FILTERS command returns, in the *filtersArr* and *allowDenyArr* arrays, the external link filters of the Web area designated by the * and *object* parameters. If no filter is active, the arrays are returned empty.

The filters are installed by the WA SET EXTERNAL LINKS FILTERS command. If the arrays are reinitialized during the session, the WA GET EXTERNAL LINKS FILTERS command can be used to find out the current settings.

See Also: WA SET EXTERNAL LINKS FILTERS

WA GET LAST URL ERROR

WA GET LAST URL ERROR ({*;}object; url; description; errorCode)

Parameter	Type		Description
*	*	→	If specified, object is an object name (string) If omitted, object is a table or a variable
object	Form object	→	Object name (if * is specified) or Table or variable (if * is omitted)
url	String	←	URL at origin of error
description	String	←	Description of error (Mac OS)
errorCode	Longint	←	Error code

The WA GET LAST URL ERROR command can be used to recover several items of information concerning the last error occurring in the Web area designated by the * and *object* parameters.

This information is returned in three variables:

- *url*: URL causing error.
- *description* (Mac OS only): a text describing the error (if available). If it is not possible to associate a text with the error, an empty string is returned. Under Windows, this parameter is always returned empty.
- *errorCode*: the error code.
  - If the code is >=400, it is an error related to the HTTP protocol. For more information about this type of error, refer to the following address:
    http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html
  - Otherwise, it is an error returned by the WebKit (Mac OS) or ActiveX (Windows).

It is advisable to call this command within the framework of the On URL Loading Error form event in order to find out the cause of the error that just occurred. For more information, please refer to the "Form Events" paragraph on page 4.

WA Get last filtered URL

WA Get last filtered URL ({*;}object) → String

Parameter	Type		Description
*	*	→	If specified, object is an object name (string) If omitted, object is a table or a variable
object	Form object	→	Object name (if * is specified) or Table or variable (if * is omitted)
Function result	String	←	Last filtered URL

The WA Get last filtered URL command returns the last URL that was filtered in the Web area designated by the * and *object* parameters.

The URL may have been filtered for one of the following reasons:

- The URL was denied because of a filter (WA SET URL FILTERS command),
- The link is open in the default browser (WA SET EXTERNAL LINKS FILTERS command),
- The URL attempts to open a pop-up window.

It is advisable to call this command in the context of the On URL Filtering, On Open External Link and On Window Opening Denied form events in order to find out the URL that was filtered. For more information, please refer to the "Form Events" paragraph on page 4.

WA GET URL HISTORY

WA GET URL HISTORY({*;}object; urlsArr{; direction{; titlesArr}})

Parameter	Type		Description
*	*	→	If specified, object is an object name (string) If omitted, object is a table or a variable
object	Form object	→	Object name (if * is specified) or Table or variable (if * is omitted)
urlsArr	Alpha array	←	Array of URLs visited
direction	Integer	→	0 or omitted=List of previous URLs, 1=List of next URLs
titlesArr	Alpha array	←	Array of window titles

The WA GET URL HISTORY command returns one or two arrays containing the URLs visited during the session in the Web area designated by the * and *object* parameters. It can be used to build a custom navigation interface.

The information provided concerns the session; in other words, the navigation carried out in the same Web area as long as the form has not been closed.

The *urlsArr* array is filled with the list of URLs visited. Depending on the value of the *direction* parameter (if it is passed), the array recovers the list of previous URLs (default operation), or the list of next URLs. These lists correspond to the content of the standard Back and Forward buttons of browsers.

The URLs are classed by chronological order.

Pass a value indicating the list to recover in *direction*. You can use one of the following constants, found in the "Web Area" theme:

Constant	Type	Value
Previous URLs	Longint	0
Next URLs	Longint	1

If you omit the *direction* parameter, the value 0 is used.

If it is passed, the *titlesArr* parameter contains the list of window names associated with the URLs. This array is synchronized with the *urlsArr* array.

WA Create URL history menu

WA Create URL history menu ({*;}object{; direction}) → MenuRef

Parameter	Type		Description
*	*	→	If specified, object is an object name (string) If omitted, object is a table or a variable
object	Form object	→	Object name (if * is specified) or Table or variable (if * is omitted)
direction	Integer	→	0 or omitted=List of previous URLs, 1=List of next URLs
Function result	MenuRef	←	Menu reference

The WA Create URL history menu command creates and fills a menu that can be used directly for navigation among the URLs visited during the session in the Web area designated by the * and *object* parameters. It can be used to build a custom navigation interface.

The information provided concerns the session; in other words, the navigation carried out in the same Web area as long as the form has not been closed.

Pass a value indicating the list to recover in *direction*. You can use one of the following constants, found in the "Web Area" theme:

Constant	Type	Value
Previous URLs	Longint	0
Next URLs	Longint	1

If you omit the *direction* parameter, the value 0 is used.

Once the menu has been generated, you can work with it using the standard 4D menu management commands, in particular Dynamic pop up menu.

Call the RELEASE MENU command to delete the URL history menu once it is no longer useful.

- Creation of two custom history menus:

```
PrevURLMenu:=WA Create URL history menu(*;"MyWArea";
                                        Previous URLs)
NextURLMenu:=WA Create URL history menu(*;"MyWArea"; Next URLs)
```

See Also: Dynamic pop up menu, RELEASE MENU

WA Get page title

WA Get page title ({*;}object) → String

Parameter	Type		Description
*	*	→	If specified, object is an object name (string) If omitted, object is a table or a variable
object	Form object	→	Object name (if * is specified) or Table or variable (if * is omitted)
Function result	String	←	Title of current page

The WA Get page title command returns the title of the current page or the page being displayed in the Web area designated by the * and *object* parameters. The title corresponds to the HTML "Title" tag.

This command returns an empty string if there is no title available for the current URL.

WA SET PAGE CONTENT

WA SET PAGE CONTENT({*;}object; content; baseURL)

Parameter	Type		Description
*	*	→	If specified, object is an object name (string) If omitted, object is a table or a variable
object	Form object	→	Object name (if * is specified) or Table or variable (if * is omitted)
content	String	→	HTML source code
baseURL	String	→	URL for relative references (Mac OS)

The WA SET PAGE CONTENT command replaces the page displayed in the Web area designated by the * and *object* parameters by the HTML code passed in the *content* parameter.

The *baseURL* parameter can be used to specify, under Mac OS, a base URL that will be added in front of any relative links found in the page. Under Windows, this parameter has no effect and the base URL is not specified. It is therefore not possible to use relative references on this platform.

Note Under Windows, it is imperative for a page to have already been loaded into the Web area before this command can be called. If necessary, you can pass the "about:blank" URL in order to load a blank page.

- Display of the "Hello world!" phrase and definition of a "file:///" base URL (Mac OS only):

  WA SET PAGE CONTENT(MyWArea;"<html><body><h1>Hello World!</h1></body></html>";"file:///")

WA Get page content

WA Get page content({*;}object) → String

Parameter	Type		Description
*	*	→	If specified, object is an object name (string) If omitted, object is a table or a variable
object	Form object	→	Object name (if * is specified) or Table or variable (if * is omitted)
Function result	String	←	HTML source code

The WA Get page content command returns the HTML code of the current page or the page being displayed in the Web area designated by the * and *object* parameters.

This command returns an empty string if the contents of the current page is not available.

WA SET PAGE TEXT LARGER

WA SET PAGE TEXT LARGER({*;}object)

Parameter	Type		Description
*	*	→	If specified, object is an object name (string) If omitted, object is a table or a variable
object	Form object	→	Object name (if * is specified) or Table or variable (if * is omitted)

The WA SET PAGE TEXT LARGER command increases the size of the text displayed in the Web area designated by the * and *object* parameters.

Under Mac OS, the scope of this command is the 4D session: the configuration carried out by this command is not retained after the 4D application is closed.
Under Windows, the scope of this command is global: the configuration is retained after the 4D application is closed.

WA SET PAGE TEXT SMALLER

WA SET PAGE TEXT SMALLER({*;}object)

Parameter	Type		Description
*	*	→	If specified, object is an object name (string) If omitted, object is a table or a variable
object	Form object	→	Object name (if * is specified) or Table or variable (if * is omitted)

The WA SET PAGE TEXT SMALLER command reduces the size of the text displayed in the Web area designated by the * and *object* parameters.

Under Mac OS, the scope of this command is the 4D session: the configuration carried out by this command is not retained after the 4D application is closed.
Under Windows, the scope of this command is global: the configuration is retained after the 4D application is closed.

WA SET PREFERENCE

WA SET PREFERENCE({*;}object; selector; value)

Parameter	Type		Description
*	*	→	If specified, object is an object name (string) If omitted, object is a table or a variable
object	Form object	→	Object name (if * is specified) or Table or variable (if * is omitted)
selector	Longint	→	Preference to be modified
value	Boolean	→	Value of the preference

The WA SET PREFERENCE command can be used to set different preferences for the Web area designated by the * and *object* parameters.

Pass the preference to be modified in the *selector* parameter and the value to be assigned to it in the *value* parameter.

In *selector*, you can pass one of the following constants, found in the "Web Area" theme:

Constant	Type	Value
WA Java	Longint	0
WA JavaScript	Longint	1
WA PlugIns	Longint	2
WA Contextual menu	Longint	3

For each preference, pass True in *value* to activate it and False to deactivate it.

Here is the meaning of the selectors:

- WA Java:
- WA JavaScript:
- WA PlugIns:
- WA Contextual menu:

WA Get preference

WA Get preference ({*;}object; selector) → Boolean

Parameter	Type		Description
*	*	→	If specified, object is an object name (string) If omitted, object is a table or a variable
object	Form object	→	Object name (if * is specified) or Table or variable (if * is omitted)
selector	Longint	→	Preference to get
Function result	Boolean	←	Current value of the preference

The WA Get preference command can be used to get the current value of the preference in the Web area designated by the * and *object* parameters.

Pass the preference whose value you want to get in the *selector* parameter. You can pass one of the following constants, found in the "Web Area" theme:

Constant	Type	Value
WA Java	Longint	0
WA JavaScript	Longint	1
WA PlugIns	Longint	2
WA Contextual menu	Longint	3

The command returns the current value of this preference: True if the preference is active and False otherwise. For more information about these preferences, please refer to the description of the routine WA SET PREFERENCE, page 24.