Creating Mashups with 4D Live Window

By Joseph Resuello, Technical Support Engineer, 4D Inc.
Technical Note 07-07

Abstract

This Technical Note introduces the concept of Mashups and the way in which Mashups enhance the 4th Dimension experience through the use of the 4D Live Window plug-in, which is available in the 4D Web 2.0 Pack. An example database is provided revealing the plug-in's versatility and it gives insight into other possible uses of the plug-in.

What is a Mashup?

In the exciting and ever-growing realm of the World Wide Web we are witnessing a new age in the way information is processed and presented. This new age, deemed as Web 2.0, provides a richer user experience by allowing website content and functionality open to third-party developers. Public application programming interfaces (API's) by big name companies such as Google, eBay, Yahoo, and Amazon are now making it possible to create custom websites and applications that are at the creative whim of the developer.

These new websites and applications often use "Mashups". Mashups are a way of reusing existing information and presenting it in ways that can be customized to meet any need. 4D has now made it possible for 4D developers to Mashup their own databases with this new brand of richer and more readily-available content.

http://en.wikipedia.org/wiki/Mashup_(web_application_hybrid)

What is 4D Live Window?

As part of the 4D Web 2.0 pack, 4D Live Window is a 4D plug-in that allows the embedding of an internet browser into a 4D form. This unlocks the world of adding web-based content to 4D databases, allowing rich and up-to-date information in forms.

4D Live Window allows the 4D developer to drive the URL of the webpage specified using data from records in the database. One example, which will be described in detail later, is using a customer's address information to load street maps from Google Maps. Thanks to 4D Live Window, a customer

record can have the look and feel of Google Maps in that same 4D form. Instead of having to switch applications in order to view pertinent information, things are done more efficiently by having web-browser functionality in 4D.

Another possible use of 4D Live Window Mashups is displaying real-time package tracking information by using, for example, tracking numbers from shipped inventory in the database. Stock ticker information can be loaded in the database as well. The power of live up-to-date information is now in the hands of the 4D developer, and they are only limited by the power of the Web itself.

4D Live Window Methods

There are currently six methods in the 4D Live Window plug-in. They are:

Web_Back
Web_Forward
Web_SetPreferences
Web_GetURL
Web_GetContent
Web_SetURL

For the purposes of this Technical Note only the functionality of the **Web_SetURL** method will be focused on. (More detailed information about this and the other methods can be found at the Command List Reference at the end of this document.)

Web_SetURL

Web_SetURL is where all the magic happens. This is the method that loads a web URL into the 4D Live Window plug-in area. The developer can choose to load an ordinary predefined URL or a URL driven with data from the 4D database.

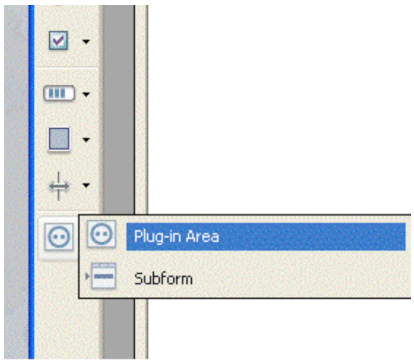
4D Live Window Examples

In this section example uses of 4D Live Window will be presented. At this point, feel free to reference the sample database that comes attached with this Technical Note.

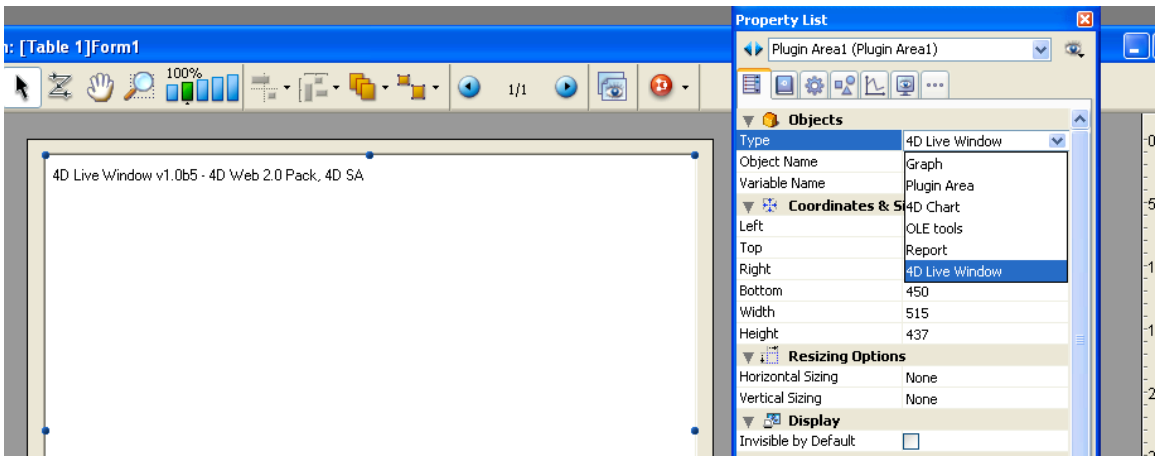
The first example, involving Google Maps, will go over how to load a URL driven with customer address information from the database.

Getting Started

A 4D Live Window plug-in area can be added onto a 4D form in the same way any other 4D plug-in would be added. First, make sure that the '4D Live Window.bundle' file is in a folder named 'Plugins' next to the database structure. Once the database is opened, go to the Form Editor in Design mode. Be sure to select the Plug-in Area option (as shown below):



Now draw the 4D Live Window area on the form. Under Property List, make sure that the plug-in area Type is set to '4D Live Window' (as shown below):



The stage is now set to enter some code to bring functionality to the 4D Live Window plug-in area.

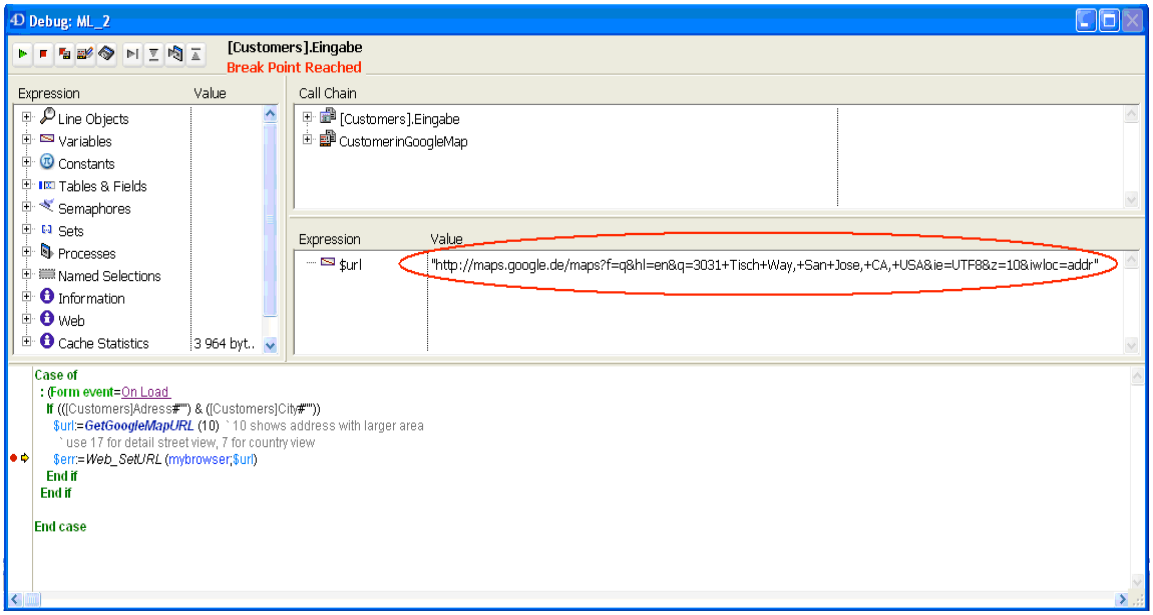
Example 1: Google Maps

Content can now be loaded into the 4D Live Window area by using the **Web_SetURL** command. In the sample database, **WebSetURL** is called during the Form Event "On Load." Typically, the "On Load" event is the best time to have the 4D Live Window loaded. By that time the URL should be all ready to go. In this example the URL has address information from the

database, and it is correctly formatted in a string to represent a Google Maps search. Here is the code for the form in regards to the "On Load" form event:

```
Case of
:(Form event=OnLoad)
        If(([Customer]Address#"")&([Customers]City#""))
                $url:=GetGoogleMapsURL(10)
                $err:=WebSetURL(mybrowser;$url)
        End if
End if
End case
```

Below is a snapshot of the debugger in 4D revealing the value of $url variable as the call to **WebSetURL** is about to be made:



Notice that URL contains the Google Maps server (maps.google.com, or maps.google.de in this case), customer address information with "+" characters between words, and special characters such as "hl=en&" in various parts of the URL string to make the Google Map URL valid.

How did the URL get properly formatted with data from our database? Please take a look at project method *GetGoogleMapsURL* which gets called right before **Web_SetURL**. *GetGoogleMapsURL* is the project method that does all the dirty work of correctly formatting the URL with domain information and data from the database. He is a closer look at *GetGoogleMapsURL*.

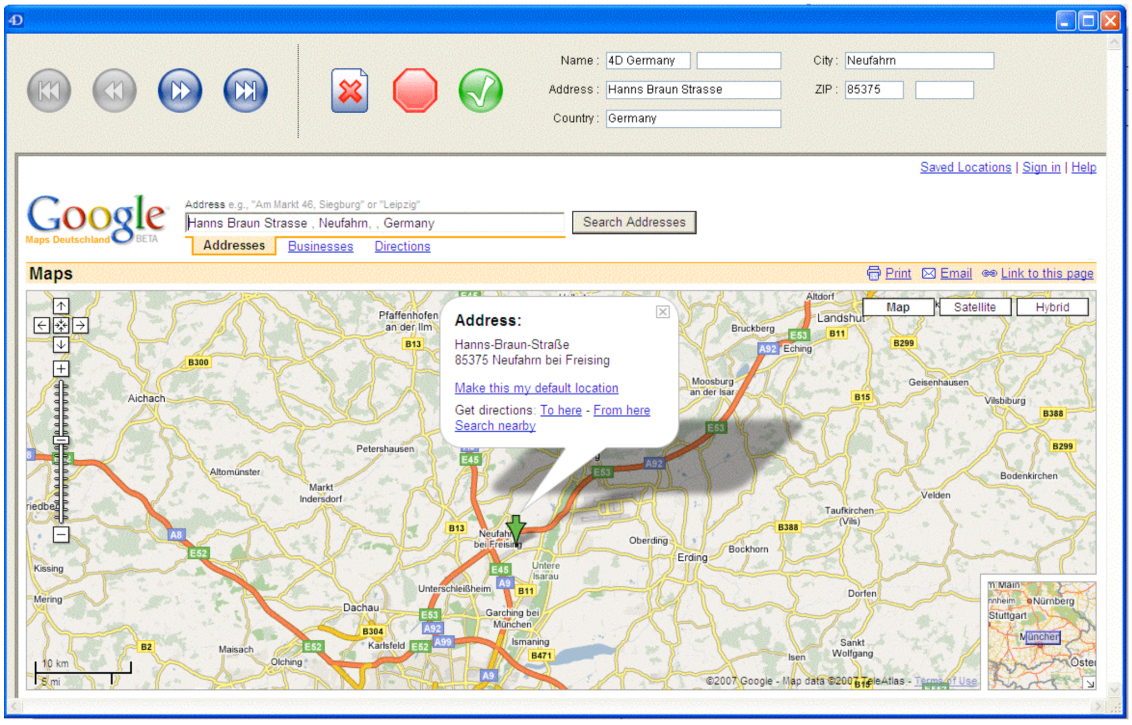
```
If (Count parameters>=1)    `   zoom value
        $z:=$1    ` 10 shows address with larger area, 17 for detail street view, 7 for
country view
Else
        $z:=15
```

```
End if

$codelang:=Get indexed string(<>StrLang;39)    `fr,  de  ou en...
If ($codelang="")
        $codelang:="en"
End if
$URL:="http://maps.google.de/maps?f=q&"
$URL:=$URL+"hl="+$codelang+"&"
$address:=[Customers]Address+",+"+[Customers]City+",+"
If ([Customers]State#"")
        $address:=$address+[Customers]State
End if
$address:=$address+",+"+[Customers]Country
$address:=Replace string($address;"  ";"+")
$URL:=$URL+"q="+$address+"&ie=UTF8&z="+String($z)    ` zoom
$URL:=$URL+"&iwloc=addr"

$0:=$URL
```

The *GetGoogleMapsURL* method formats the Google Maps URL string with customer address data in the database. It then returns the correctly formatted URL which is fully qualified to be called by **WebSetURL**. The image below is an example of a 4D form with a 4D Live Window plug-in area containing the Google Maps result:



Thus, the pseudo code for utilizing 4D Live Window involves two main procedures:

1. Formatting data from our database into a valid string to be used as a URL
2. Calling the resulting URL string using **WebSetURL**

This is the technique employed for the above Google Maps example, and it will be the same technique used for the examples that follow.

Example 2: Fedex Tracking Information

The exact same concepts from the Google Maps example can be applied to the next example of loading up the Fedex website with tracking information based on tracking numbers in the database.

Just like the Google Maps example above, a project method will do all the dirty work of returning a qualified URL based on Fedex's URL information and our tracking numbers. This method will be called *GetFedexURL* and its code follows:

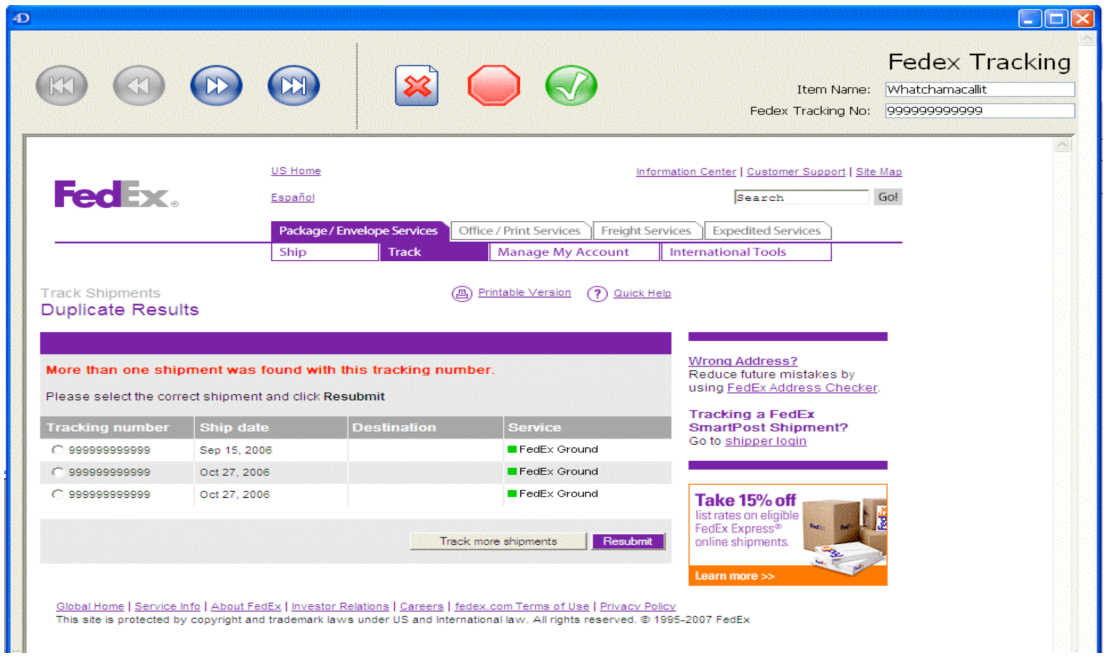
```
$URL:="http://fedex.com/Tracking?ascend_header=1&clienttype=dotcom&cntry_code=us&la
nguag"+"e=english&tracknumbers="

$address:=[Inventory]TrackingNo
$URL:=$URL+$address

$0:=$URL
```

The method *GetFedexURL* is short because the process of combining the the link from Fedex with data from the database is an easy one. It is only necessary to append the tracking number to the end of the Fedex URL. Notice the simplicity here in method *GetFedexURL* compared to example in method *GetGoogleMapsURL*.

With a valid URL from *GetFedexURL* a call to **WebSetURL** is made to load the 4D Live Window area and this is what is seen:



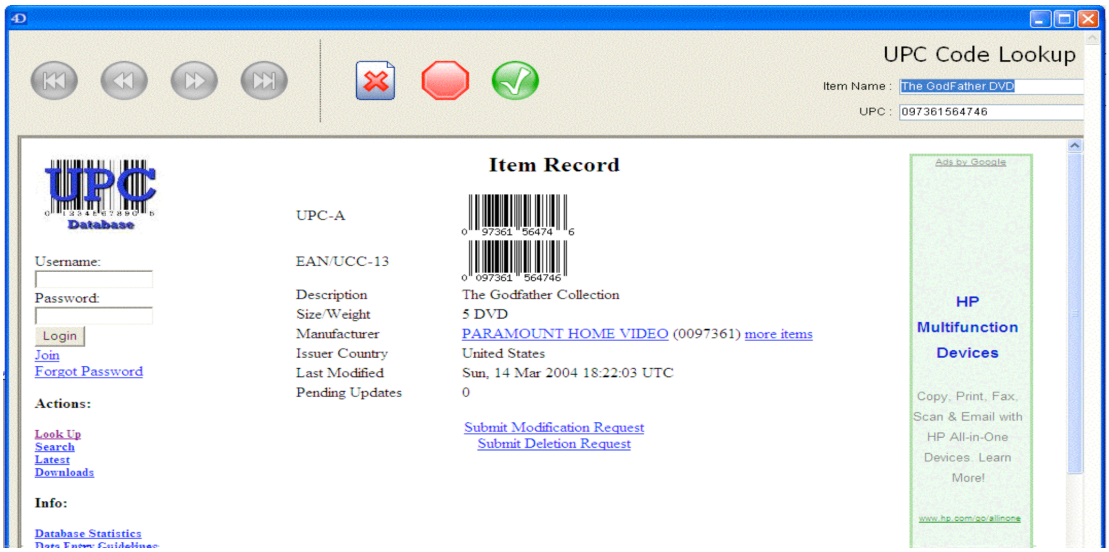
Example 3: UPC Lookup

With an inventory of goods, being able to reference goods by barcode can be a useful task. With this next example a search at UPCDatabase.com will be performed using a barcode number from the database as the query. *GetUpcURL,* the project method that correctly formats the URL string, looks like this:

```
$URL:="http://www.upcdatabase.com/item.asp?upc="
$UPC:=[Inventory]UPC
$URL:=$URL+$UPC
$0:=$URL
```

Much like the Fedex example above, the project method for formatting the URL is simple as well. The barcode number only has to be appended at the end of the UPCdatabase.com URL.

After a call to **Web_SetURL**, this is the result:



Conclusion

Creating Mashups with 4D Live Window can be an easy task as long as the website being called to has standard and understandable URL string conventions. The more explicit the string conventions, the easier it would be to append data from the database. The examples in this Technical Note display practical proof-of-concept ways of embedding a live browser window in 4D database forms. The developer is more than encouraged to explore the world of Mashups and discover more creative and innovative ways of using the 4D Live Window plug-in.

Command List Reference

Web_SetURL

Web_SetURL (Area; URL; MacOSPath) _ error code

Parameter	Type	Description
Area	Longint	_ 4D Live Window area
URL	Text	_ Full URL (http://www.4d.com)
MacOSPath	Longint	_ 0 for URL, 1 for local file path, Unused on Windows
Function Result	Longint	_ Error code (0 = no error)

Web_SetURL is usually the first command we need to apply to an area. It specifies the URL (or document) to be opened. The URL must be passed as fully qualified URL (i.e. http://www.4D.com or file:///C:/my %20Document.pdf).

On Mac OS a file name must be valid Unix path name, in the form:

file://localhost/Users/thomas/Documents/image%C3.jpg.

8-bit characters and special character like blanks must be encoded. Invalid encoded characters may crash the application.

The parameter *MacOSPath* allows automatic file path encoding. Pass 1 to use an HFS path like"MyDisk:Users:thomas:Documents:imageö.jpg”.

On Windows real file names are accepted, like "C:\my Document.pdf" or "Straße.jpg" with no encoded special characters. Also simple URL's like "www.4D.com" are accepted as well. Still, it is recommended to use fully qualified names.

Function result
If the call was successful it returns 0. Otherwise it was an error.

Web_GetURL

Web_GetURL (Area; URL) _ error code

Parameter	Type	Description
Area	Longint	_ 4D Live Window area
URL	Text	_ Current URL
Function Result	Longint	_ Error code (0 = no error)

The command Web_SetURL returns the current displayed URL.

Function result
If the call was successful it returns 0. Otherwise it was an error.

Web_Back

Web_Back (Area) _ error code

Parameter	Type	Description
Area	Longint	_ 4D Live Window area
Function Result	Longint	_ Error code (0 = no error)

The command Web_Back calls the **Back** function of the browser, going to the previously displayed URL.

Function result
If the call was successful it returns 0. Otherwise it was an error.

Web_Forward

Web_ Forward (Area) _ error code

Parameter	Type	Description
Area	Longint	_ 4D Live Window area
Function Result	Longint	_ Error code (0 = no error)

The command Web_Forward calls the **Forward** function of the browser, reverting the usage of the **Back** function. If the **Back** function was not used before, the call has no result.

Function result
If the call was successful it returns 0. Otherwise it was an error.

Web_GetContent

Web_GetContent (Area; content) _ error code

Parameter	Type	Description
Area	Longint	_ 4D Live Window area
Content	Blob	_ Page Content as Blob
Function Result	Longint	_ Error code (0 = no error)

The command Web_GetContent returns the content of the main frame as blob.

Function result
If the call was successful it returns 0. Otherwise it was an error.

Web_SetPreferences

Web_SetPreferences (Area; Selector; NumValue, StringValue) _ error code

Parameter	Type	Description
Area	Longint	_ 4D Live Window area
Selector	Longint	_ Preferences Selector
NumValue	Longint	_ Numerical value to set
StringValue	Alpha	_ String value to set
Function Result	Longint	_ Error code (0 = no error)

The command Web_SetPreferences allows to change the behavior of the 4D Live Window plug-in.

In Release 1.0 the 4th parameter StringValue is not used, always pass "" for future compatibility.

Notes: (1) Web_kResize

Mac OS only: if the plugin area is resized, some web sites (specially maps.google.com) acts on the resize by using Javascript to change the position of some elements. This may lead to a crash if the site was already displayed. To avoid this crash the developer is supposed to set the browser area to a fixed size (not resizeable).

(2) In any case the browser area is not automatically resized on Mac except for the first display (opening a window before loading a web site). If you display HTML content (or PDF images) without Javascript, you may enable the resize functionality by using this option. In that case you should not allow the end user to manually enter a URL. Using this option has no result on Windows. (3) Web_kVisible The 4D Command SET VISIBLE does not fully hide the browser because the browser responds directly to events such as mouse-over or Javascript triggered redraws. So it is necessary to hide/show the browser area by using both SET VISIBLE and Web_SetPreferences.

Function result
If the call was successful it returns 0. Otherwise it was an error.

Example
$err:=Web_SetPreferences(mybrowser; 1; 1; "") ` to enable resizing

Error Codes

All commands of the *4D Live Window* plug-in return 0 if the call was successful.
Possible error codes are:

-15001	The specified area is not a 4D Live Window area
-15002	Invalid parameter passed
-15003	Internal error