

PlugIn Wizard 1.1

MacOS® and Windows® Version

Documentation



© 1995-1997 ACI SA All rights reserved
All referenced trade names are trademarks or registered trademarks
of their respective holders.

Introduction

4th Dimension possesses a robust procedural language containing hundreds of commands. However the 4D developer will require a functionality which is not part of the 4th Dimension Language.

4D Plug-Ins are one way to extend the 4th Dimension Language and provide the 4D developer with a required functionality.

Plug-In Wizard will help developers to design easily their own plug-ins for 4th Dimension on MacOS or Windows platform, or both.

Plug-In Wizard also allows cross-development using Macintosh toolbox calls in a Windows development project (see Using Mac2Win further in this documentation).

Here are the three steps to using PlugIn Wizard :

1. Via a graphic interface the developer defines a list of the Plug-In's routines and describes each one (parameters and returned value). Once the Plug-In has been created the defined routines will be available as 4th Dimension commands.

The developer can also design plug-in areas.

2. PlugIn Wizard generates:

- A resource file which contains the mandatory resources use by 4th Dimension
- A C/C++ source code file containing definitions and routine declarations
- A compiler project for use with CodeWarrior 10 or Visual C++ 4.0
- The Plug-In documentation in RTF format

3. Once the above steps have been completed the developer only needs to write the code for each of the routines, compile and link. The Plug-In is ready to use.

Installation

- On MaOS

Copy the folder containing PlugIn Wizard , PlugIn Wizard.comp and PlugIn Wizard.data to your hard disk. Once PlugIn Wizard has been copied to your hard disk it is ready to use.

- On Windows

Copy the folder containing PlugIn Wizard.4DB , PlugIn Wizard.4DC , PlugIn Wizard.RSR and PlugIn Wizard.4DD to your hard disk. Once PlugIn Wizard has been copied to your hard disk it is ready to use.

When the database is first opened it will contain a default project named "Sample".

Requirements and limits

PlugIn Wizard is a 4th Dimension version 6 database for both MacOS and Windows platforms. PlugIn Wizard requires 4th Dimension version 6.0 or later.

- On MaOS

Projects are generated for Metrowerks CodeWarrior 10 or later.

- On Windows

Projects are generated for Microsoft Visual C++ 4.0 or later.

- PlugIn Wizard limitations

- Number of Plug-In projects stored in the PlugIn Wizard database is limited by available hard disk space
- A given Plug-In may contain thousands of individual routines
- A Plug-In routine is limited to twenty-five (25) parameters
- A Plug-In routines name is limited to fifteen (15) characters

Getting started

For example, we will create a plug-in called BlobPlugIn which contains a routine to find a value in a blob (FindInBlob).

This routine will accept three parameters: the blob in which we search, the blob size and the value to search. It will returns the position of the found string.

Double-Click on your 4D application, then open the "PlugIn Wizard.comp" database on Macintosh, or the "PlugIn Wizard.4DC" database on Windows. When requested, create a new data file.

Open Plug-In Wizard, then choose New in the File menu.

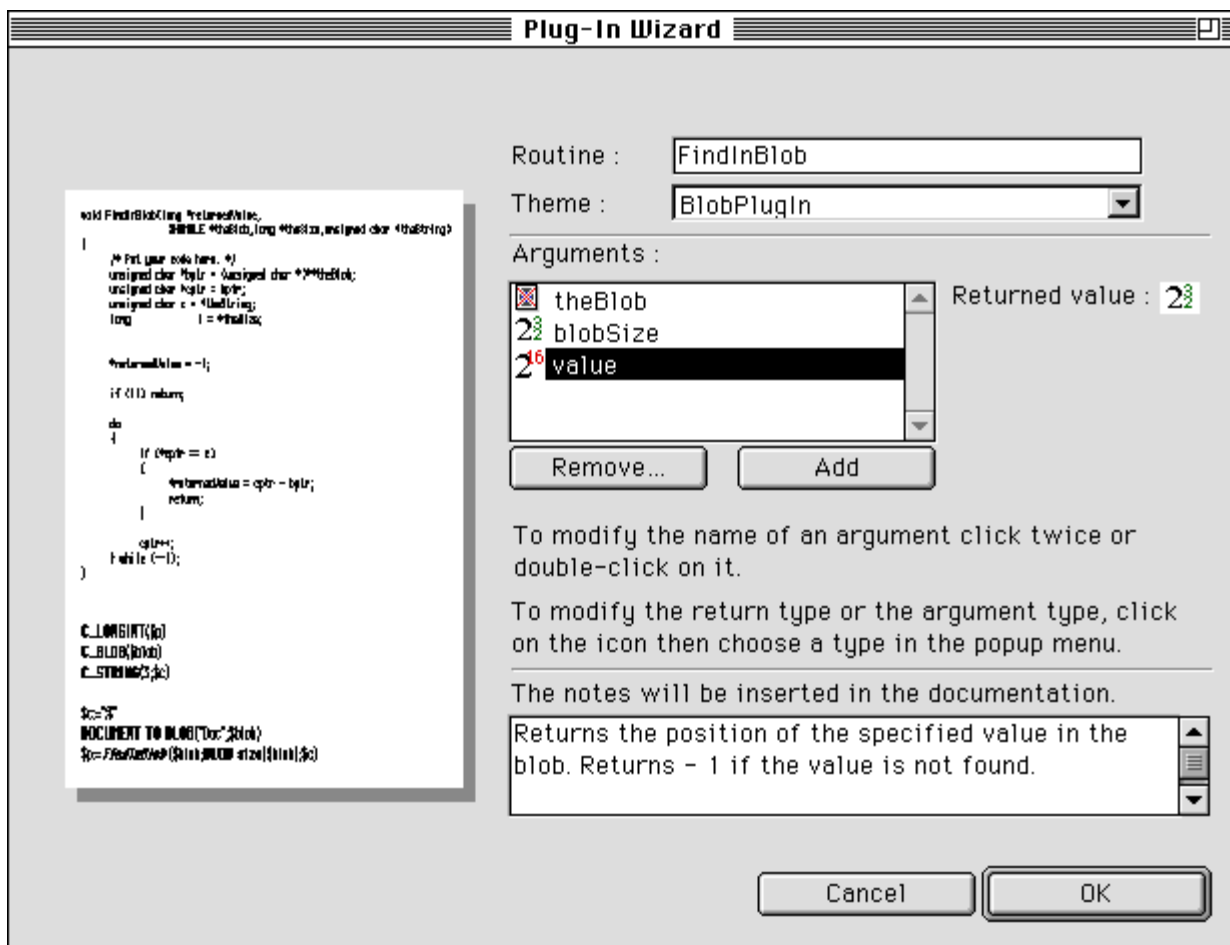
In the Plug-In pane give a name to the package. The name can only contain letters, spaces and numbers. You can put some comments if you want, they will be used in documentation and reported as comments in source code.



Click Next. The Routine Pane appears. Click on Add routine button.



The routine entry form allow you to set the parameters and the returned value of the routine :



- type "FindInBlob" as routine name;
- click on the button Add to add the first argument;
- click on the icon that is at the left of the argument name and select the type Blob in the popup menu.
- Click twice on the argument name to edit it, then type "theBlob" instead of "Arg1";
- Add the second argument, call it "blobSize" and select long integer type.
- Add the second argument, call it "value" and select short integer type.
- click on the icon at the right of Returned value and select long integer in the popup menu.

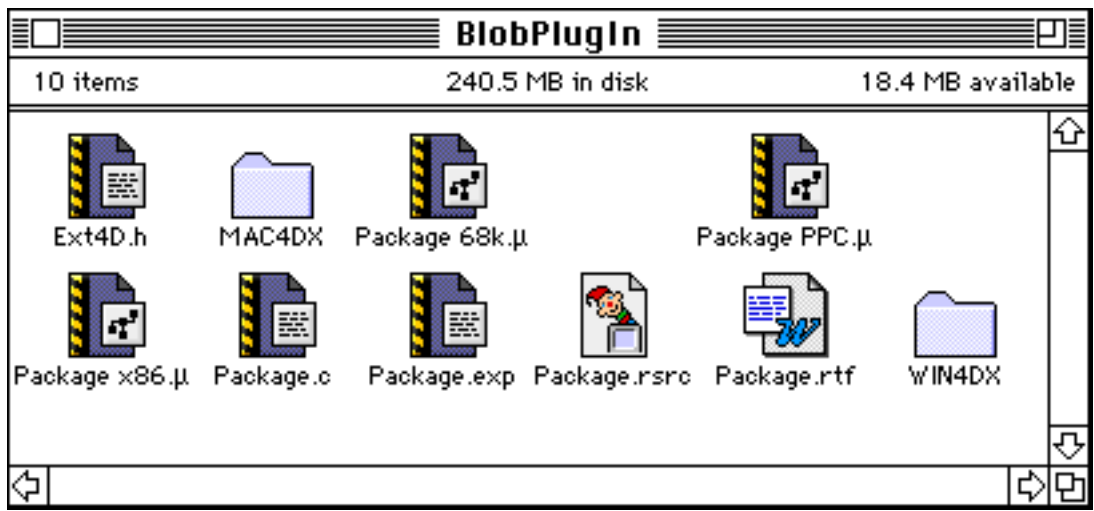
You can also type the routine description, it will be inserted in documentation, and as comments in the source code.

The name of a routine cannot exceed 15 characters, it contains only letters, spaces and numbers.

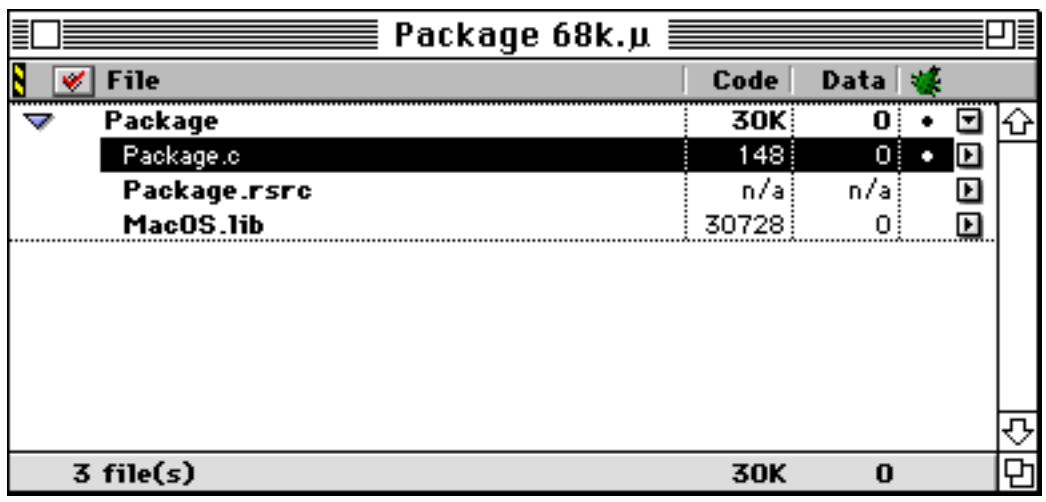
Select the Final step pane to generate the resource, the source and the documentation. Then click on Make All .



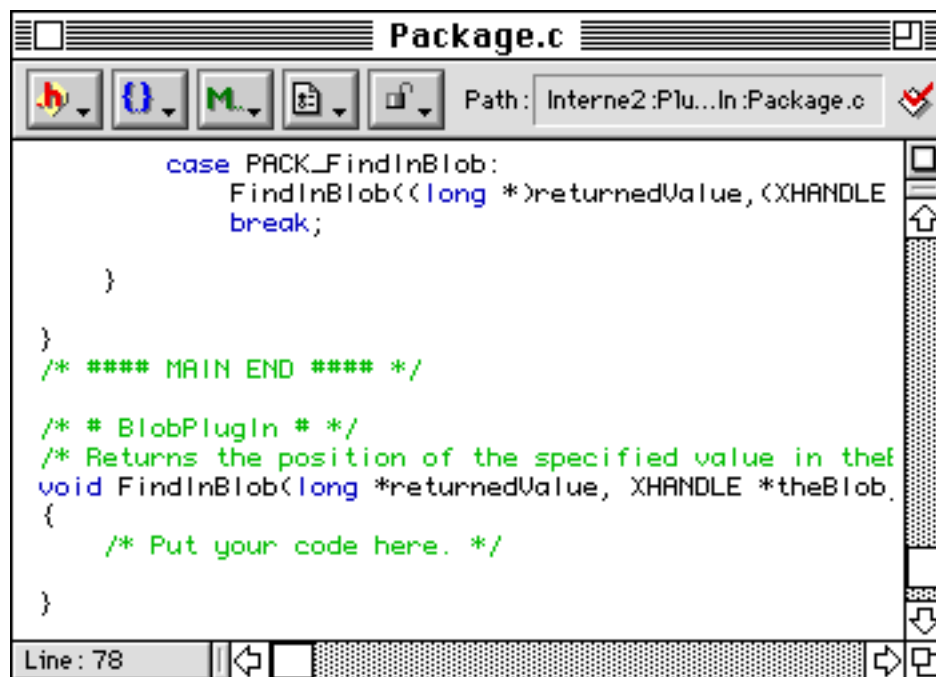
On MacOS, the folder BlobPlugIn now contains :



Double click on "Package 68k.µ" icon, Code Warrior will open the generated project:



Package.c is the C source code generated, double-click on it:



```
case PACK_FindInBlob:
    FindInBlob((long *)returnedValue, (XHANDLE
    break;

}

}
/* **** MAIN END **** */

/* # BlobPlugIn # */
/* Returns the position of the specified value in the
void FindInBlob(long *returnedValue, XHANDLE *theBlob,
{
    /* Put your code here. */
}

Line: 78
```

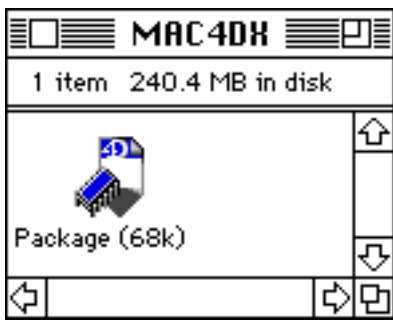
The function FindInBlob is here, ready to be written. Type the following code in the function body:

```
void FindInBlob(long *returnValue, XHANDLE *theBlob, long *blobSize, short *value )
{
    unsigned char *blobptr = (unsigned char*) **theBlob;
    long position = 0;

    *returnValue = -1;

    for ( position = 0; position < *blobSize; position++ )
    {
        if ( *blobptr++ == (unsigned char) *value)
        {
            *returnValue = position;
            break;
        }
    }
}
```

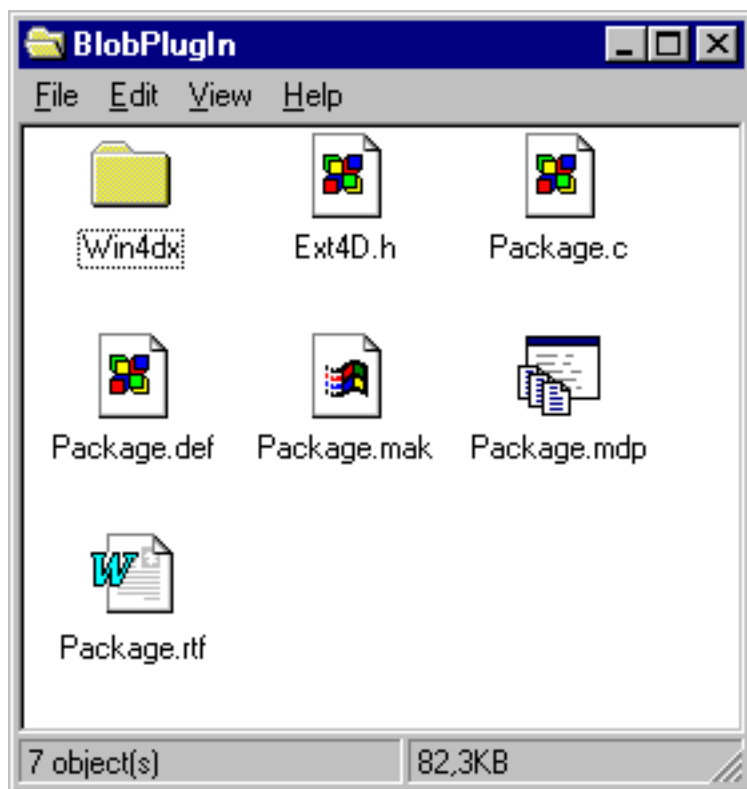
Select **Make** in the **Project** menu. Look in MAC4DX, your plug-in — named **Package (68k)** — is ready to be used.



Now copy the MAC4DX folder holding your new Plug-In in the folder of one of your 4D database, open this database and write the following method to test your plug-In:

```
SET BLOB SIZE (vBlob;1000;0)
vblob{897}:=42
pos:=FindInBlob (vBlob;1000;42)
ALERT ("found at offset :"+String (pos))
```

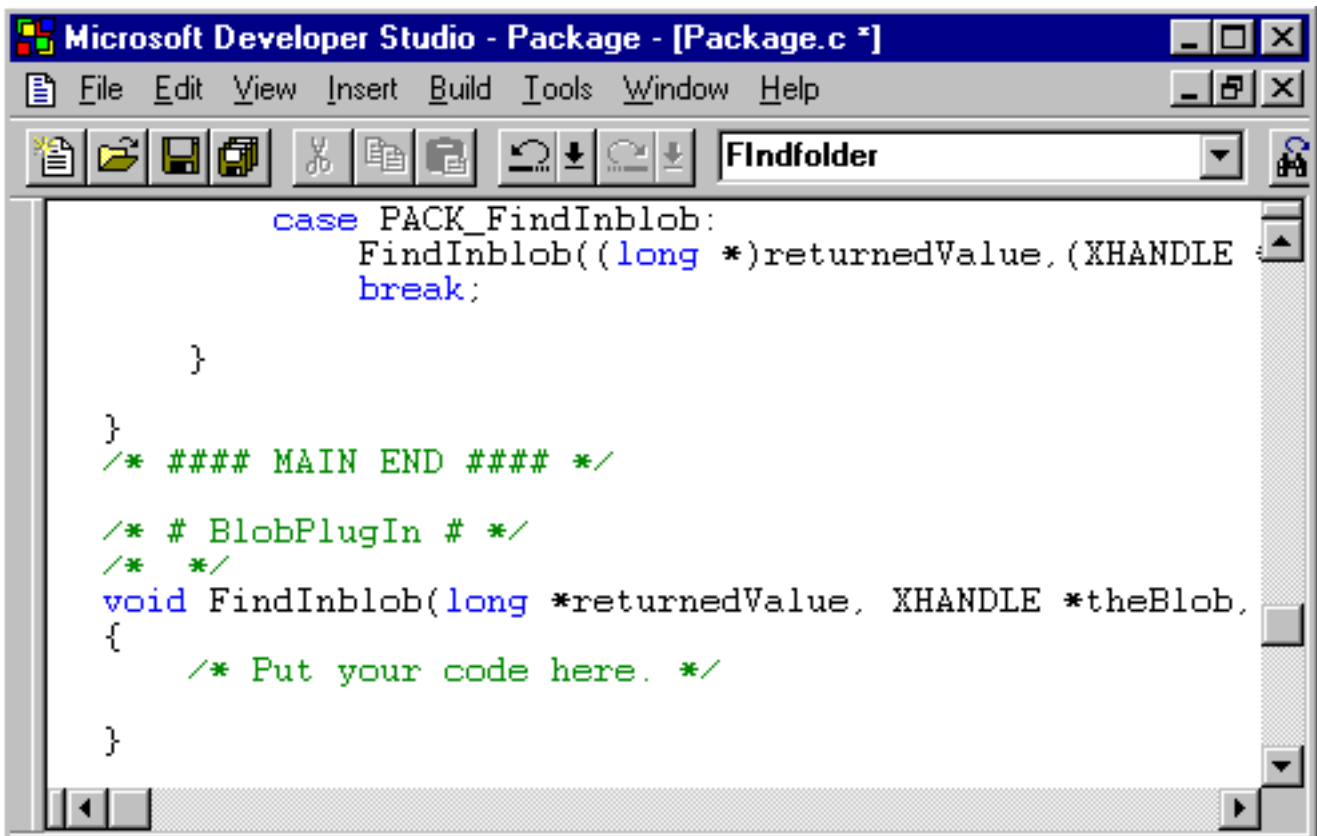
On windows, the folder BlobPlugIn contains :



Double click on "Package.mdp" icon, Visual C++ will open the generated project:



Package.c is the C source code generated, double-click on it:



```
        case PACK_FindInblob:
            FindInblob((long *)returnedValue, (XHANDLE
            break;

    }

}
/* #### MAIN END #### */

/* # BlobPlugIn # */
/* */
void FindInblob(long *returnedValue, XHANDLE *theBlob,
{
    /* Put your code here. */

}
```

The function FindInBlob is here, ready to be written. Type the following code in the function body:

```
void FindInBlob(long *returnValue, XHANDLE *theBlob, long *blobSize, short* value )
{
    unsigned char *blobptr = (unsigned char*) **theBlob;
    long position = 0;

    *returnValue = -1;

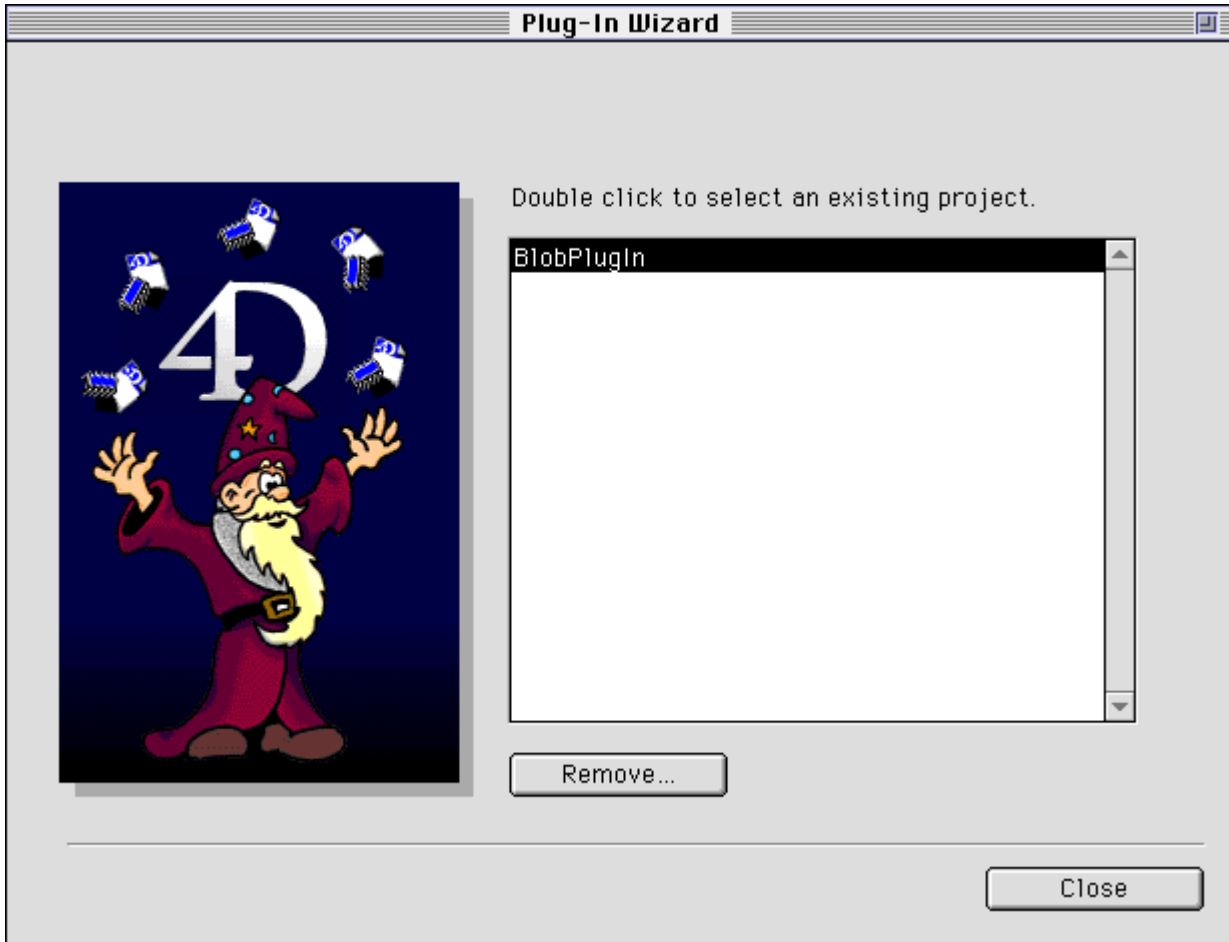
    for ( position = 0; position < *blobSize; position++ )
    {
        if ( *blobptr++ == (unsigned char) *value )
        {
            *returnValue = position;
            break;
        }
    }
}
```

Select **Build** in the **Build** menu. Look in WIN4DX, your plug-in — named **Package.4DX** — is ready to be used (it requires the file **Package.RSR**).



Adding and modifying projects

To add a plug-in project in the database, choose **New** in the **File** menu.



To modify an existing plug-in project, choose **Open** in the **File** menu, then double-click on the project name.

To remove a plug-in project click once on its name, then click on **Remove** .

Functions and themes

As with 4th Dimension commands, plug-in routines can be grouped by themes. The default theme is the plug-in name.

To add a routine in a specific theme, click on the theme name, then click on Add routine .

To change the theme (or any other information) of a routine, double-click on the icon located at left of the routine name.

In the routine entry form, select the new theme via the popup menu below the name of the routine.

Caution : when you remove a theme, all the routines linked to this theme are removed.

Generated files

- on both platforms :
 - Ext4D.h: the header containing Extension Kit definitions ;
 - Package.c (or Package.cpp **) *: the source containing the plug-in routines ;
 - Package.rtf *: the package documentation (RTF file).
- on MacOS
 - Package.rsrc *: the resources ;
 - Package 68k.µ (Package 68k++.µ **) and Package PPC.µ (Package PPC++.µ **) are CodeWarrior projects to generate the plug-in ;
 - MAC4DX: the folder where you will find the plug-in (after link) ;

 - Package x86.µ (Package x86++.µ **): CodeWarrior project to generate Windows code from a Macintosh (located in WIN4DX) ;
 - Package.exp: export definitions (referenced from Package x86.µ or Package x86++.µ) ;
 - Package.RSR *: the windows version of Package.rsrc.
 - WIN4DX: the folder where you will find the plug-in (after link) ;
- on Windows :
 - Package.RSR *: the resources (located in WIN4DX) ;
 - Package.mpd (PackageCpp.mpd **) and Package.mak (PackageCpp.mak **): the Visual C++ project to generate the plug-in ;
 - Package.def: export definitions (referenced from Package.mak or PackageCpp.mak).
 - WIN4DX: the folder where you will find the plug-in (after link) ;

* these files are overwritten when you click on Build , Generate, Write or Make All (Final step pane).

** these are the files generated when you choose Write C++ source in the Final step pane.

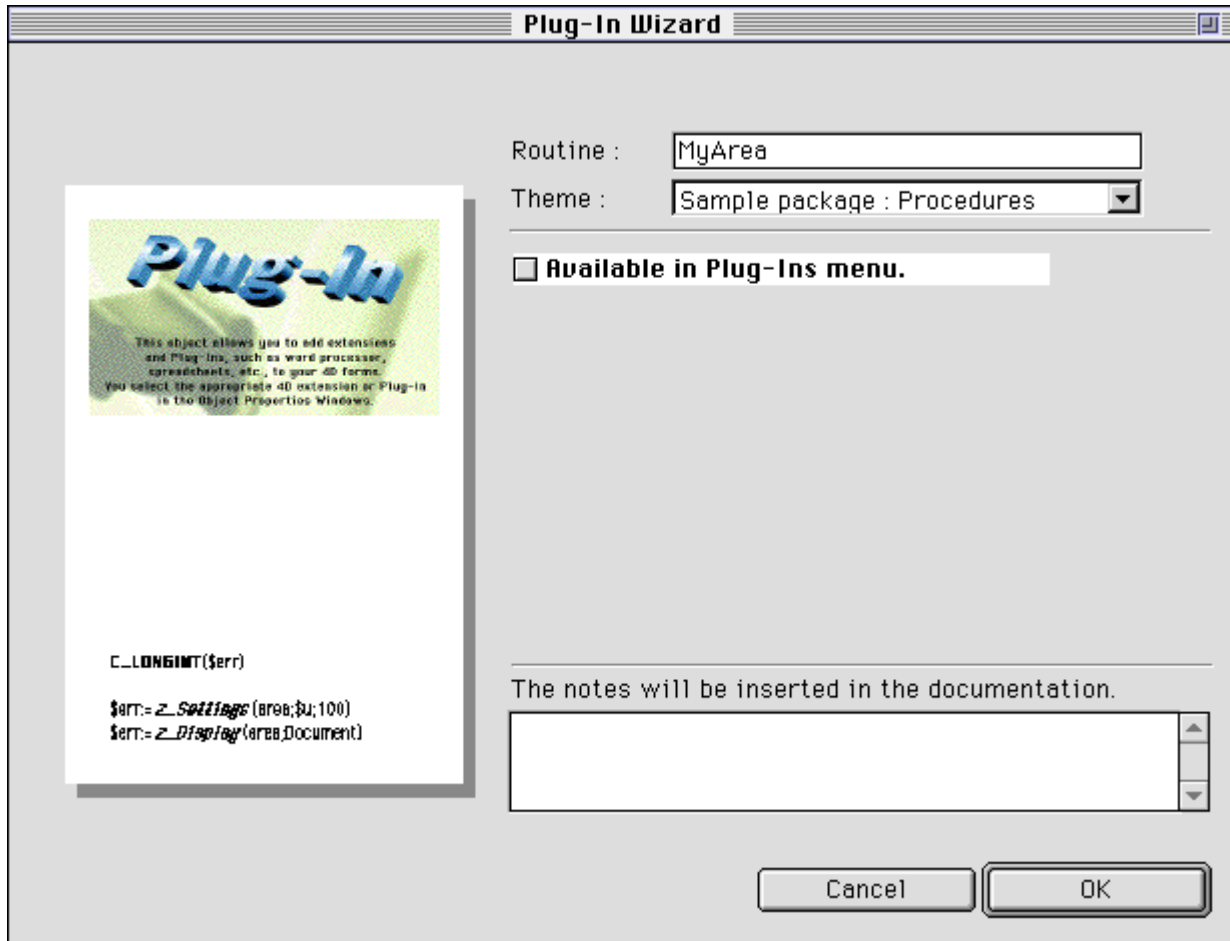
Compilers projects, headers and export definition files are not overwritten, so you can include other files in the projects.

Note : If you are using Visual C++ or CodeWarrior x86 project don't forget to put the PPC_H folder in your project folder (see chapter Using Mac2Win).

Plug-In areas

Version 1.1 of PlugIn Wizard allows developers to create plug-in areas.

In the Routine Pane, click on Add area button.



If you want that the area is available from Plug-Ins menu of 4D (User Mode), check the box Available in Plug-Ins menu .

PlugIn Wizard will automatically add the _ or % sign before the name, to indicate that the routine refers to a plug-in area.

PlugIn Wizard will generate a sample of function body managing the area, this function handle basic events sent to the area (initialization, deinitialization, update events). PlugIn Wizard also pre-defines a structure containing the private data of the area. This sample is ready to be used (a cross will be drawn inside the area).

For more information, please read the External Kit manual.

Using Mac2Win (Altura)

For Windows cross-developments, version 1.1 of PlugIn Wizard generates sources and projects referring to Mac2Win of Altura. This development kit contains a library (ASINTPPC.Lib) and headers files, which allow to use Macintosh Toolbox calls within a Windows development project.

Using Mac2Win allows you to use the same sources on Windows and Macintosh.

These calls will be usefull if you plan to perform graphic operations, or if you have to deal with Macintosh memory 'Handle'.

How to use it with projects generated by PlugIn Wizard ?

1. Design your plug-in and let PlugIn Wizard generate the files.
2. Look for the PPC_H folder located in Extensions Kit (in ACI Product Line CD).
3. Copy it into the folder containing the projects generated by PlugIn Wizard.
4. Compile and link, your plug-in is ready to be used.

This feature is available for both Visual C++ and CodeWarrior x86 projects.

For advanced users

- Because this version of PlugIn Wizard overwrites the C source, be sure to have defined all the routines of the package before generating the source.

If you plan to add or remove routines after generating the C source, store your original Package.c in a safe place, generate the new source code, then merge the old source code with the new one.

- On Macintosh, if you plan to use your own resources put them in another resource file and include this specific resource file in the compiler project.
- If you want that your plug-in works on any Macintosh (68k or PowerPC), make a FAT plug-in (it will contain 68k and PowerPC code in a single file).

To make a FAT plug-in:

- Compile the 68k plug-in - named Package (68k) .
- Compile the PPC plug-in - named Package (PPC) .
- Duplicate Package (PPC) and rename the copy Package (FAT) .
- Open with a resource editor Package (68k) and Package (FAT) plug-ins
- Copy from Package (68k) the '4DPX' resource to Package (FAT) and save it.

