PlugIn Wizard

リファレンスガイド

Windows® and Mac™ OS

ACI

PlugIn Wizard リファレンスガイド
Windows® and Mac™ OS

このマニュアルに記載されている事項は、将来予告なしに変更されることがあり、いかなる変更に関してもACI SAは一切の責任を負いかねます。このマニュアルで説明されるソフトウェアは、本製品に同梱のLicense Agreement（使用許諾契約書）のもとでのみ使用することができます。

注意

このソフトウェアの使用に際し、本製品に同梱のLicense Agreement（使用許諾契約書）に同意する必要があります。ソフトウェアを使用する前に、License Agreementを注意深くお読みください。

目次

はじめに

4th Dimensionは、数多くのコマンドからなる強力なプログラミング言語を有していますが、4D言語には含まれていない機能を追加する必要が出てくることもあります。

4Dプラグインは、ディベロッパーが必要としている機能を4D言語に追加するひとつの手段です。

PlugIn Wizardは、4D用のプラグイン（Mac OS、Windows、もしくは両プラットフォーム用）を簡単に作成できるように手助けをします。

PlugIn Wizardは、Windowsの開発プロジェクト上でMacintosh toolboxを利用するといった異プラットフォームに渡る開発にも使用できます（第3章の「Mac2Win (Altura)を使用する」を参照してください）。

▼PlugIn Wizardは以下の３つのステップで使用します：

1. **プラグインのルーチンの定義、および、その記述（パラメータや戻り値）をグラフィカルなインタフェースを使用して行う。**

   プラグインが作成されれば、定義したルーチンは4Dのコマンドとして利用することができます。
   プラグインエリアを設計することも可能です。

2. **PlugIn Wizardは以下のものを作成する：**

   ■ 4Dが必須とするリソースを含んだファイル

   ■ 定義およびルーチンの宣言を含んだC/C++のソースコードファイル

   ■ CodeWarrior10またはVisual C++ 4.0で使用するコンパイラプロジェクト

   ■ プラグインのドキュメント（RTF形式ファイル）

3. **以上のステップを完了したら、各ルーチンに実際のコードを書き込み、コンパイルして、リンクさせる。**

   プラグインを利用可能になります。

ディスクへインストールする

▼ ご使用の環境にPlugIn Wizardをインストールするには：

1 PlugIn Wizardファイルが入っているフォルダを、ご使用のハードディスクにコピーする。

■ Mac OSでは、以下のファイルがコピーされます：

- **■ PlugIn Wizard**
- **■ PlugIn Wizard.comp**
- **■ PlugIn Wizard.data**

■ Windowsでは、以下のファイルがコピーされます：

- **■ PlugIn Wizard.4DB**
- **■ PlugIn Wizard.4DC**
- **■ PlugIn Wizard.RSR**
- ■ PlugIn Wizard.4DD

PlugIn Wizardファイルがハードディスク上にコピーされれば、インストールは完了です。いつでも利用することができます。

データベースファイルを初めて開いた時点では、"Sample"という名前の省略時プロジェクトが、あらかじめ含まれています。

必要条件と制限事項

PlugIn Wizard はMac OS 上および Windows 上での 4th Dimension バージョン 6 のデータベースです。PlugIn Wizard を使用するには、4th Dimension バージョン 6 以降が必要です。

■ Mac OS 上では、Metrowerks CodeWarrior 10 以降用のプロジェクトが生成されます。

■ Windows 上では、Microsoft Visual C++ 4.0 以降用のプロジェクトが生成されます。

PlugIn Wizard の制限事項

■ PlugIn Wizard データベースに格納できるプラグインプロジェクトの数は、ハードディスクの空き容量に依存します。

■ 作成したプラグインには、何千もの個別のルーチンを入れることが可能です。

■ プラグインルーチンのパラメータは 25 個に制限されます。

■ プラグインルーチンの名前は 15 文字（1 バイト英数字）に制限されます。

この章では、BLOB内の値を検索するルーチン（FindInBlob）を持つBlobPlugInという名前のプラグインを作成します。

このルーチンでは、次の３つのパラメータを使用します。検索対象とするBLOB、BLOBのサイズ、そして検索する値です。戻り値としては、見つかった文字列の位置を返します。

PlugIn Wizardデータベースを使う

1 4Dアプリケーションをダブルクリックし、PlugIn Wizardデータベースを開く：

- Macintoshでは、PlugIn Wizard.compファイルを選択します。
- Windowsでは、PlugIn Wizard.4DCファイルを選択します。

2 要求に応じて新規データファイルを作成する。

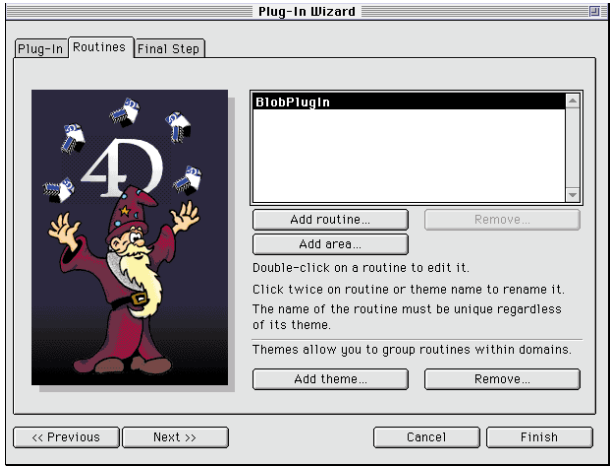
3 PlugIn Wizardを開き、「ファイル」メニューから「New」を選択する。

4 「Plug-In」タブで、このパッケージの名前を入力する。

名前は文字、スペース、および数字のみを使用することができます。コメントを記述することもできます。コメントはプラグインのドキュメントにも使用されるほか、ソースコード内にもコメントとして表示されます。

5 「Next」ボタンをクリックする。

「Routine」タブが表示されます。



6. 「Add routine...」ボタンをクリックする。

ルーチン入力画面では、パラメータ、およびルーチンの返り値を設定します。

7. ルーチン名として"FindInBlob ” と入力する。

8. 「Add」ボタンをクリックして１番目の引数を追加する。

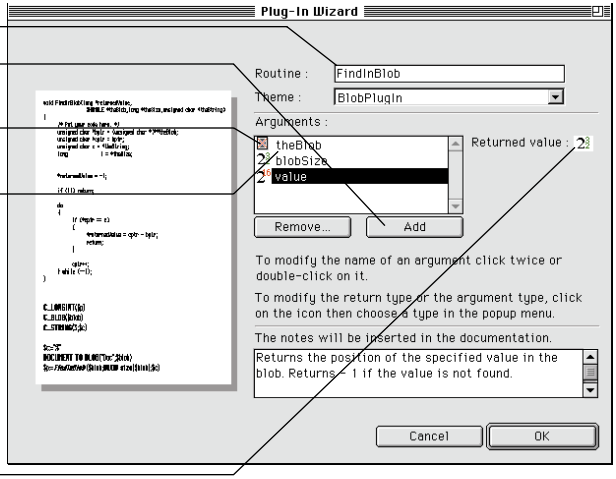
9. 引数名の左にあるアイコンをクリックして、ポップアップから「Blob」タイプを選択する。

10. 引数名をダブルクリックして編集できるようにして、"Arg1"の代わりに"theBlob"と入力する。

11. ２番目の引数を追加する。
名前は"blobSize"、タイプはlong integer（倍長整数）にします。

12. ３番目の引数を追加する。
名前は"value"、タイプはshort integer（整数）にします。

13. "Returned value"の右側にあるアイコンをクリックして、ポップアップから「long integer（倍長整数）」を選択する。



ルーチンのコメントを記述することもできます。コメントはプラグインのドキュメントにも使用されるほか、ソースコード内にもコメントとして表示されます。

注：ルーチン名は15文字（１バイト英数字）を越えてはなりません。名前には文字、スペース、および数字のみが使用できます。

14. 「Final Step」タブを選択して、リソース、ソース、およびドキュメントを生成する。

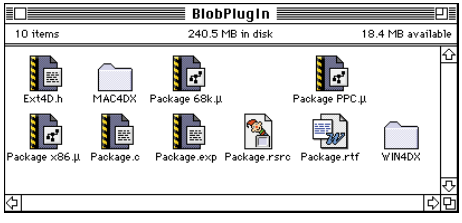
15. 「Make All」ボタンをクリックする。

ここで、Mac OSおよびWindows両プラットフォームの結果を見てみましょう。

結果を見る

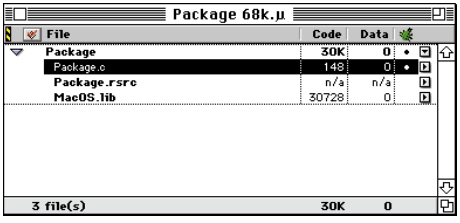
Mac OS

Mac OSでは、「BlobPlugIn」フォルダには以下のものが含まれます：



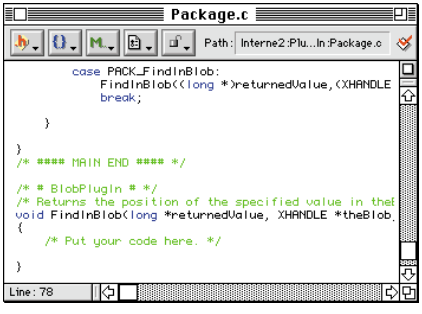
1. 「Package 68k..ｵ」アイコンをダブルクリックする。

生成されたプロジェクトがCodeWarriorで開かれます。



2. 「Package.c」をダブルクリックする。

Package.cは自動的に生成されたCソースコードです。



FindInBlob関数はここにあります。すぐに記述できるようになっています。

3. 関数の中身に以下のコードを入力する：

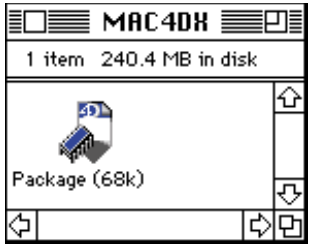
```
void FindInBlob(long *returnedValue, XHANDLE *theBlob, long *blobSize, short* value )
{
	unsigned char *blobptr = (unsigned char*) **theBlob;
	long position = 0;

	*returnedValue = -1;

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

4. **「Project」メニューから「Make」を選択する。**

MAC4DXの中を見てください。作成したプラグイン（"Package(68k)"という名前）が使用可能になっています。

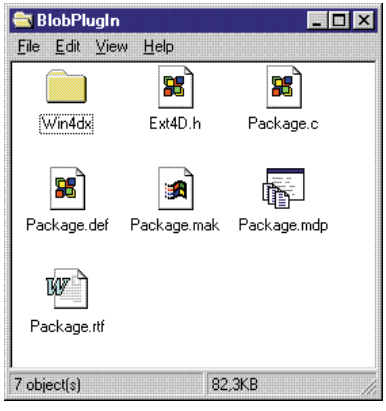


5. **作成したプラグインの入っている「MAC4DX」フォルダを4Dデータベースのフォルダへコピーします。そのデータベースを開き、以下のメソッドを記述して、プラグインのテストをします：**

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

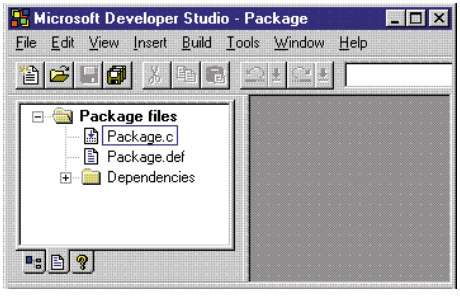
Windows

Windows上での「BlobPlugIn」フォルダには以下のものが含まれます：



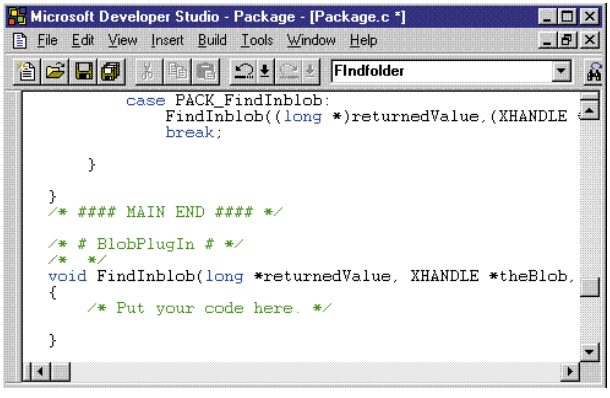
1. **「Package.mdp」アイコンをダブルクリックする。**

生成されたプロジェクトがVisual C++で開かれます。



2. 「Package.c」をダブルクリックする。

Package.cは自動的に生成されたCソースコードです。



FindInBlob関数はここにあります。すぐに記述できるようになっています。

3. 関数の中身に以下のコードをタイプ入力する：

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

	*returnedValue = -1;

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

4. 「Build」メニューから「Build」を選択する。

WIN4DXを見てください。作成したプラグイン("Package.4DX"という名前)が使用できるようになっています（"Package.RSR"ファイルも必要です)。

プロジェクトの追加および修正

▼ データベースにプラグインプロジェクトを追加するには：

1. 「File」メニューから「New」を選択する。

▼既存のプラグインプロジェクトを修正するには：

1. 「File」メニューから「Open」を選択する。

2. プロジェクト名をダブルクリックする。

▼ プラグインプロジェクトを削除するには：

1. プロジェクト名を１回クリックする。

2. 「Remove…」をクリックする。

関数とテーマ

4Dコマンドと同様に、プラグインルーチンはテーマごとにグループ化することができます。省略時のテーマはプラグイン名になります。

▼ 特定のテーマにルーチンを追加するには：

1. **テーマ名をクリックする。**
2. **「Add routine」をクリックする。**

▼ ルーチンのテーマ（またはその他の情報）を変更するには：

1. **ルーチン名の左にあるアイコンをダブルクリックする。**
2. **ルーチン入力フォームで、ルーチン名の下にあるポップアップメニューから新規テーマを選択する。**

警告：テーマを削除する場合、そのテーマにリンクされているルーチンもすべて削除されます。

生成されたファイル

生成されたファイルについての説明です。

両プラットフォーム共通

■ **Ext4D.h** ： Extension Kit定義を含んでいるヘッダファイルです。

■ **Package.c（またはPackage.cpp**）**：プラグインルーチンを含んでいるソースファイルです。

■ **Package.rtf***：パッケージの資料書類です（RTF書式ファイル）。

Mac OS

■ **Package.rsrc***：リソースファイルです。

■ **Package 68k..ｵ（Package 68k++..ｵ*）とPackage PPC》..ｵ（Package PPC++..ｵ**）**：プラグインの生成に使用されるCodeWarriorプロジェクトファイルです。

■ **MAC4DX**：プラグインが生成されるフォルダです（リンク実行後）。

■ **Package x86..ｵ（Package x86++..ｵ**）**：Macintosh上からWindowsコードを生成する場合に使用するCodeWarriorプロジェクトファイルです（WIN4DXに生成されます）。

■ **Package.exp**：書き出された定義ファイルです（Package x86..ｵまたはPackagex86++..ｵから参照されます）。

■ **Package.RSR***：Package.rsrcのWindows版です。

■ **WIN4DX**：プラグインが生成されるフォルダです（リンク実行後）。

Windows

■ **Package.RSR**[1]：リソースファイルです（WIN4DXにあります）。

■ **Package.mpd（PackageCpp.mpd**[2]**）とPackage.mak（PackageCpp.mak**[2]**）**：プラグインの生成に使用されるVisual C++プロジェクトファイルです。

■ **Package.def**：書き出された定義ファイルです（Package.makまたはPackageCpp.makから参照されます）。

■ **WIN4DX**：プラグインが生成されるフォルダです（リンク実行後）。

注：Visual C++またはCodeWarrior x86プロジェクトを使用している場合は、「PPC_H」フォルダをプロジェクトフォルダに入れるのを忘れないようにしてください（第3章の「Mac2Win (Altura)を使用する」を参照してください）。

プラグインエリア

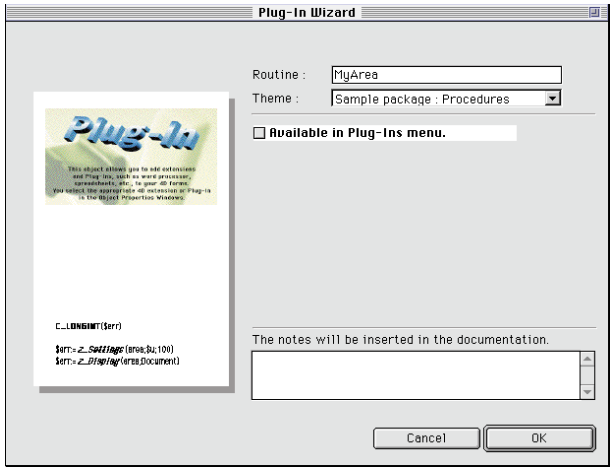
PlugIn Wizard version 1.1ではプラグインエリアを作成することができます。

▼ プラグインエリアを作成するには：

1. 「Routine」タブで、「Add area」ボタンをクリックする。

1. これらのファイルは「Build」、「Generate」、「Write」または「Make All」をクリックすることで上書きされます（「Final Step」タブ）。
2. これらのファイルは、「Final Step」タブで「Write C++ source」を選択した場合に生成されます。コンパイラプロジェクト、ヘッダ、書き出された定義ファイルは上書きされませんので、その他のファイルをプロジェクトに含めることが可能です。

以下の画面が表示されます。



2. 4D の「プラグイン」メニュー（ユーザモード時）からエリアを利用できるようにするには、「Available in Plug-Ins menu」ボックスにチェックを入れる。

PlugIn Wizard は、自動的に"_"や"%"記号を名前の前に追加して、ルーチンがプラグインエリアを参照するように指示します。

3. 「OK」をクリックする。

PlugIn Wizard はエリアを管理する関数の中身のサンプルを生成します。この関数は、エリアに送られる基本的なイベント（initialization、deinitialization、update イベント）を制御します。PlugIn Wizard はエリアのプライベートデータを含んだストラクチャもあらかじめ定義します。
所定のサンプルはすぐに使用できるようになっています（エリアの中には十字が描かれます）。

より詳しい情報は、External Kit に関するドキュメントを参照してください。

Mac2Win (Altura)を使用する

Windows用のクロスプラットフォームな開発をするために、PlugIn Wizard version 1.1はAlturaからMac2Winを参照するソースやプロジェクトを生成します。この開発キットは、Windows開発プロジェクト内からMacintosh Toolboxをコールできるようにするライブラリ（ASINTPPC.Lib）とヘッダファイルを含みます。

Mac2Winを使用すれば、同じソースをWindowsでもMacintoshでも使用することができます。

これらのコールは、グラフィックな操作を使用する場合や、Macintoshメモリ'Handle'を扱わなければならない場合に、非常に有効です。

▼ PlugIn Wizardで生成したプロジェクトでMac2Winを使用するには：

1. **プラグインを設計し、PlugIn Wizardを使用してファイルを生成する。**
2. **Extension Kitに入っている「PPC_H」フォルダを探す（Extension Kitは「ACI Product Line」CD-ROM内にあり）。**
3. **「PPC_H」フォルダをPlugIn Wizardで生成したプロジェクトが入っているフォルダ内にコピーする。**
4. **コンパイルおよびリンクを実行する。**

   プラグインを利用することが可能になりました。

   この機能は、Visual C++およびCodeWarrior x86プロジェクトの両方で利用することができます。

上級ユーザの方へ

■ このバージョンのPlugIn WizardはCソースを上書きするので、ソースを生成する前にパッケージのすべてのルーチンを必ず定義しておいてください。

Cソースを生成した後でルーチンを追加、もしくは削除したい場合は、元のPackage.cを安全な場所に保存し、新しいソースコードを生成して、その後で古いソースコードと新しいコードをマージしてください。

■ Macintosh上で、独自のリソースを使用する場合は、別のファイルにそのリソースを入れて、コンパイラプロジェクトにその特別なリソースを含めてください。

■ プラグインをすべてのMacintosh（68kまたはPowerPC）で動かしたい場合は、FATプラグインを作成してください（68kとPowerPCのコードが1つのファイルに含まれます）。

▼ FATプラグインを作成するには：

1. **Package (68k)という名前の68kプラグインをコンパイルする。**
2. **Package (PPC)という名前のPPCプラグインをコンパイルする。**
3. **Package (PPC)を複製して、Package (FAT)という名前に変更する。**
4. **リソースエディタを使用してPackage (68K)とPackage (FAT)プラグインを開く。**
5. **Package (68k)から'4DPX'リソースをPackage (FAT)へコピーして、保存する。**