PX0410

Developing Custom Solutions for PerfectOffice

An Overview of the PerfectOffice Development Environment

Table of Contents

NOV-B00656846
CONFIDENTIAL

PX0410

Overview

One of the biggest trends in software suites is the development of custom solutions for routine complex tasks. ~~For example, you have various common tasks that you perform everyday. Maybe it's~~ For example, you may have a daily assignment to gather ~~gathering~~ sales data from your regional offices, ~~making~~ make a chart of the data, and e-mailing it to all of the sales managers.PerfectOffice provides the tools to accomplish these complex business tasks. Many of these tasks are already automated through QuickTasks, templates, and other methods. Most existing automation applies to generic business processes, such as writing letters, performing mail merges, etc. However, out-of-the-box tasks are too specific to completely meet your business needs.

PerfectOffice is capable of far more. The applications have the potential to automate any business process. Using the built-in capabilities of the applications in PerfectOffice you can create custom tasks — solutions to your problem that are customized to your exact needs, right down to the specific file name of your database, or the e-mail addresses of your sales managers. The capabilities exist in your applications. All you need is a way to tell the applications involved the specific steps and the sequence of those steps. Once you define that "map" you can tell the applications to repeat your steps at any time.

The key to applying the power of PerfectOffice is its custom development environment. Using this environment you can create scripts and/or full-fledged applications which leverage the power of PerfectOffice. The applications serve as large "components" whose functionality is accessible through PerfectOffice's development tools. Combining these components with high level tools greatly decreases development time and increases flexibility. Rather than make an entirely new program to crunch your numbers, make a chart, or fax a document, you can drive (or automate) the PerfectOffice applications to perform these actions for you.

This papers details PerfectOffice's development environment and the tools available for creating custom solutions. A set of criteria is explained that help developers to select the best tool for specific types of custom solutions.

Note: This document describes development options for PerfectOffice 3.0, and the application programs that come with it — WordPerfect 6.1, Presentations 3.0, GroupWise 4.1, and Quattro Pro 6.0.

NOV-B00656847
CONFIDENTIAL

PX0410

Custom Solutions Terminology

To understand the area of custom development and how it applies to PerfectOffice, you need to understand the terminology of custom development. Here are a few key terms:

- **Automation** — controlling an application without direct user intervention and without the need to display the application's user interface. Automation can occur from either an external source, such as another program, or from an embedded language.

- **Script** — a series of statements that describe a process. When you execute the script the process is reproduced.

- **Scripting Language** — an external language designed to ease application automation. In contrast, macro languages usually embedded in applications. PerfectScript serves as both the applications' macro and scripting language.

- **Scripting** — the process of designing, creating, and using a script to reproduce a sequence of steps in an application. In contrast to traditional programming, in which a programmer creates an application with its own services and features, scripting focuses on accessing features of existing applications through a script.

- **Solution** — something that solves a problem. In the context of PerfectOffice, a solution is usually some form of automation that controls the applications in PerfectOffice to accomplish a specific result. QuickTasks are examples of pre-defined solutions.

- **Custom Development** — creating your own solutions. Custom solutions development usually occurs at a higher level, i.e. at a level closer to the user,

NOV-B00656848
CONFIDENTIAL

than traditional programming. Instead of the application developer doing custom development, it can be done by the end user, or by an intermediate source, such as a corporate IS department, VAR (value added reseller), or integrator. The advantages of custom development are that a solution can be tailored to the actual end user's specific situation and custom solutions take less time to develop.

- **Development Environment** — the combination of a development (scripting) language and the tools which help to create a script. Tools vary, but usually include a dialog box editor, debugger, text editor, recorder, and other tools.

The PerfectOffice Development Environment

By far the most common approach to creating custom solutions with PerfectOffice is to use PerfectScript, the native scripting (macro) language. PerfectScript is shared by a number of the major PerfectOffice components: WordPerfect, Presentations, GroupWise, and limited support in Quattro Pro. This sharing not only extends to using the same scripting language, but also a single script can control any or all of these applications at the same time.

As powerful as PerfectScript is, some developers prefer to use Microsoft Visual Basic to create a "front end" for their custom solution. Visual Basic, for example, could provide an interface to drive Quattro Pro to crunch numbers or access a database, then WordPerfect used to format the final result. As a front end, Visual Basic remotely controls Quattro Pro and WordPerfect, commanding them to open documents, chart data, write text, compute formulas, save and print documents, and so forth. PerfectOffice supports Visual Basic automation using a set of proprietary VBX controls that tap directly into the power of PerfectScript.

For network-based custom solutions, PerfectOffice supports AppWare (formally called Visual AppBuilder), a custom development tool that uses visual objects to create stand-alone programs. An AppWare program is created by "dragging" objects onto an electronic canvas, then connecting the objects to define the flow of the program. For example, an AppWare program might display the standard PerfectOffice File Open dialog box, allowing the user to specify a file. From there, the AppWare program might query an Oracle database on the server, pump the data into Quattro Pro, link the chart into a WordPerfect document, print it, and

NOV-B00656849
CONFIDENTIAL

close it. The advantage of AppWare is that knowledge of programming language syntax is not required to build applications that integrate with PerfectOffice.

A fourth method of developing custom solutions for PerfectOffice is using the C/C++-based WOAPI (WordPerfect Open Applications Programming Interface) built into several of the PerfectOffice applications, including WordPerfect, GroupWise, Desktop Application Director (DAD), and Presentations. WOAPI is also supported in Informs 4.1 or later; however, InForms is not part of PerfectOffice.

a custom solution that uses WOAPI — also called the third-party handler (TPH) — provides the most flexibility and power, but it must be written as a Windows DLL, following a very strict set of integration rules. WOAPI applications are typically designed to integrate with a single PerfectOffice application, though it is possible to develop a system that ties into several. Examples of shipping PerfectOffice functions that are (or were at one time) implemented through WOAPI are GroupWise enabling (accessing GroupWise functionality from the applications' menus), OBEX support (Borland Office 2.0), and QuickCorrect (WPWin 6.0). Many shrink-wrapped add-on products for WordPerfect and GroupWise are developed with through WOAPI.

Why does PerfectOffice need more than one development tool? There are special functional and market advantages of AppWare, Visual Basic, and WOAPI that are unique to each. Through support for these tools, PerfectOffice opens up their advantages to custom developers who can combine the power of PerfectScript and the PerfectOffice applications with the unique advantages of AppWare, Visual Basic or WOAPI.

When creating a custom solution, there a several criteria to guide your choice of development tools. These criteria include the desired degree of control over the applications, programmer experience required, depth of functionality to be implemented, the type of application (EXE or script), the development time, the amount of user interface, distribution concerns, size of the project, and the specific applications involved in the solution. The sections that follow describe these four development tools and their advantages in the context of these criteria.

How-to and hands-on examples are not given in this overview document. Specifics on actually implementing each of the four development approaches is provided in separate documents. See *For More Information* for a list of these publications and resources.

NOV-B00656850
CONFIDENTIAL

Developing Custom Solutions Using PerfectScript

PerfectScript was designed from the beginning as a cross-application language, but it was not until PerfectOffice 3.0 that this capability was realized. Consequently, many users think of PerfectScript as the WordPerfect macro language, since is was first (and most visibly) incorporated into WordPerfect for Windows.

PerfectScript today is shared by several of the PerfectOffice applications: specifically, WordPerfect, GroupWise, and Presentations. QuattroPro supports limited commands through PerfectScript allowing retrieving and setting values or formulas in notebook cells and launching of Quattro Pro macros directly from a PerfectScript command. InfoCentral, and Paradox do not directly support PerfectScript.

About PerfectScript

The function of PerfectScript combines the capabilities of macro languages (such as WordBasic) and scripting languages (such as Visual Basic) used in other Windows programs into a single language. It is used both when recording steps taken in a program, and as a programming environment for creating more sophisticated applications.

The proprietary syntax of the PerfectScript languages borrows from the best of programming languages, including C, Pascal, and Basic. This hybrid approach allowed the original designers of PerfectScript to create a flexible, easy to use, and consistent environment. Developers conversant in any of these languages usually find no trouble in adapting to the PerfectScript syntax.

Because PerfectScript is shared among many of the PerfectOffice applications, it uses "categories" of commands to denote which commands belong to which application. For example, when using the Macro Command Inserter dialog box (you can view this box by choosing **T**ools, **M**acro, **M**acro Bar in WordPerfect for Windows, then clicking the **C**ommands button in the Macros Feature Bar). The categories are viewed by clicking the Type pop-up button.

- **Program** — Programming commands such as If, While, and For. These commands can be used in any application that supports PerfectScript.
- **WPMacroFacility** —

NOV-B00656851
CONFIDENTIAL

Additional programming commands for enhanced features, including dialog boxes. These commands can be used in any application that supports PerfectScript.

- **WPMacroInterpreter** — Additional programming commands for enhanced features, including file I/O. These commands can be used in any application that supports PerfectScript.
- **Presentations** — Product commands for Presentations, such as opening a new slide show, or switching to outline mode. These commands can only be used in Presentations.
- **WordPerfect** — Product commands used to control some aspect of WordPerfect, such as type text or set margins.
- **GroupWise** — Examples are making a new appointment, attaching a document and e-mailing it, or printing your calendar.
- **Quattro Pro** — a limited set of commands (7) for Quattro Pro. These commands allow getting and setting values and properties of cells, launching Quattro Pro macro, and opening and saving files.

These categories can be assigned to two broad groups: "programming commands" and "product function" commands. Programming commands (Program, WPMacroFacility, WPMacroInterpreter) can be used in any PerfectOffice application that supports PerfectScript, whereas product function commands can only be used in a specific application.

Pros and Cons of Using PerfectScript

PerfectScript is often the preferred choice for developing custom solutions because it neatly and easily exploits the features of PerfectOffice applications. Scripts can be created within WordPerfect, and tested and debugged with nothing but the PerfectOffice applications you wish to use. PerfectScript lets you automate the functionality of the PerfectOffice applications, and provides a powerful base of programming commands for decision making and flow control.

PerfectScript's cross-application recording capability gives solutions developers a head start. Much of the code can be recorded in the applications, rather than written from scratch. After recording, the resulting code provides a base which usually requires only minor editing. In addition, PerfectScript offers several methods for creating dialog boxes, including using custom controls that are

NOV-B00656852
CONFIDENTIAL

exclusive to PerfectOffice applications (pop-up buttons, view windows, date/calendar, etc.).

Windows-oriented functionality not found in PerfectScript can be obtained using DLL function calls. PerfectScript supports most of the standard data types used by Windows (the exception is data structures), and can access most any DLL function that does not require a callback. PerfectScript allows for casting return values from DLL calls as a variety of types, including boolean, long integer, short integer, and even long pointers to strings.

PerfectScript supports user-defined functions and procedures, allowing developers to write code in easily transportable segments. Procedures and functions are similar to one another, except that procedures return no value, and functions do. Both procedures and functions can accept arguments, and values can be passed by value (the default) or by reference. In addition, PerfectScript supports arrays with up to 10 dimensions, and 32,768 elements per dimension, providing a great deal of flexibility for the developer.

PerfectScript is extensible. Libraries of scripts (WPL files) make it possible to add new functions to PerfectScript. These additions can be new functions written in the scripting language or function calls to existing DLLs. C and C++ developers can make full-blown language extensions through PerfectScript's PID resource system.

On the down-side, PerfectScript requires that developers learn the scripting language, which can add to the production requirement. Because PerfectScript is a scripting language and not a dedicated language development platform, certain functions — such as string manipulation — can be slower than with other methods.

Reminder: At this time, PerfectScript is not supported by all PerfectOffice applications. The PerfectOffice applications that support PerfectScript are WordPerfect, GroupWise, and Presentations, and limited support for QuattroPro. InfoCentral, and Paradox do not currently support PerfectScript.

Learning More About PerfectScript

PerfectScript is documented in the on-line help that comes with WordPerfect, Presentations, and GroupWise. To obtain more information about PerfectScript and how to create scripts (macros) choose **H**elp, **M**acros from the application's menus. The Macros on-line help in the applications contain a reference of all programming commands, as well as all product function commands specific to that

NOV-B00656853
CONFIDENTIAL

application. The WPUSERS forum on CompuServe has a Macros section where PerfectScript topics are throughly discussed.

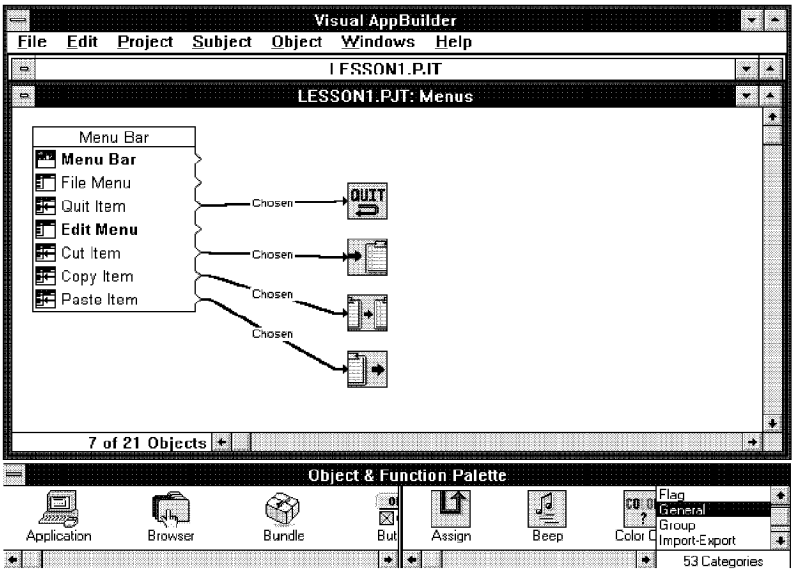
Developing Custom Solutions Using AppWare

AppWare (formerly known as Visual AppBuilder) is a graphical and object-oriented application development tool that allows you to create stand-alone EXE programs. To create an application with AppWare you need only create a new project, and drag objects into the project. By linking the objects with lines, you define the flow you want your program to take. Though AppWare can build completely stand-alone programs that are completely independent of PerfectOffice, it is discussed in this paper in relation to creating applications that integrate with PerfectOffice.

About AppWare

AppWare is a general purpose development tool, and is available from Novell as a stand-alone application, or bundled with the Professional version of PerfectOffice. AppWare combines a unique visual building environment with the advantages of self-contained components. If you can connect the dots, you can probably "program" with AppWare.

AppWare comes with over 50 categories of pre-defined objects, for database access, DDE client and server, OBDC, OLE, Oracle, TUXEDO transaction processing, NetWare, and even text-editing. Using the objects in these categories you can create finished, ready-to-go Windows applications, and in a surprisingly short period of time. For example, using the text objects you can create a simple text editor in about five



NOV-B00656854
CONFIDENTIAL

minutes time.

AppWare also functions as a front-end for creating custom solutions using the PerfectOffice applications. For example, object categories are included for nearly all PerfectScript functions, allowing you to create applications that interact directly with the applications to create documents, notebooks, charts, send e-mail, schedule meetings, manage document files, print, save, and more.

As an example, one possible use of AppWare is developing a simple database front end, and using the data to create WordPerfect merged documents. You could use the database object in AppWare to create a flat-file or relational (single or multi-user) database application, or to access an Oracle database on the server. Then, you'd pass the data from a QuattroPro notebook to a WordPerfect table.

Pros and Cons of Using AppWare

AppWare provides an enormous amount of flexibility in application design (you can even add your own objects for use in an AppWare application, but this requires knowledge of writing Windows DLLs). You can create complete and professional-looking stand-alone Windows EXE programs with AppWare, and thereby have more control over the interface and function of your application.

AppWare speeds development time by allowing you to combine predefined and tested components without actual programming. As with PerfectScript, using AppWare requires a finite learning curve. However, once learned, it is possible to create complex applications in a very short period of time.

Since you are using self-contained objects you do not need to understand how they do they work. Instead you hook together a series of building blocks to produce just the results you need. This is a strong advantage of AppWare, but can be a disadvantage if object do not exist for the functionality you need. AppWare also has strong ties to NetWare and NetWare-based functionality, such as NDS (NetWare Directory Services) and TUXEDO (transaction processing)..

On the plus side, AppWare does not require you to learn the syntax of a programming language, such as colons, parentheses, and periods; rather you draw lines between pictures that represent specific functions or objects. For those not experienced with programming languages, AppWare is an excellent alternative.

One weakness of AppWare is the fact that AppWare executables require that all ALMs (AppWare Loadable Modules) — the objects that give AppWare its

NOV-B00656855
CONFIDENTIAL

functions — must exist on the users system to run the executable. You'll need to make sure to copy all of the ALMs with your EXE file when you distribute it.

Learning More About AppWare

AppWare is documented in a separate white paper titled *Integrating AppWare With PerfectOffice* and in the PerfectOffice Developer's Reference Guide.

**** Warren: more sources? Online information on AppWare? Other sources? ****

Developing Custom Solutions Using Visual Basic

Visual Basic is one of the leading development environments in use today, and is frequently a corporate standard in IS departments. The most common usage of Visual Basic is for custom solution development and database front ends. Novell allows Visual Basic developers to be able to automate PerfectOffice applications in similar fashion to the current OLE 2.0 Automation standard. Novell provides three VBX custom controls with PerfectOffice for this purpose.

Easy creation of user interface is especially simple in Visual Basic. One of the most time consuming parts of creating any Windows program is the user interface — the dialog boxes, buttons, and edit controls users see and interact with. Visual Basic's form design tools are the best tools yet to cut the time to create good user interface.

About Visual Basic

Integrating with PerfectScript from Visual Basic can be divided into three categories. First, the execution of PerfectScript scripts from the application; second, sending individual commands to the PerfectScript enabled applications; third, getting and setting variables in the PerfectScript system.

The WPCIWIN VBX allows the programmer to use existing PerfectScript scripts as part of the Visual Basic solution by playing scripts from the application. This allows you to record much of your solution in PerfectScript, then play back from Visual Basic. This approach further reduces development time. PerfectScript scripts can be stored in two places, as WordPerfect files on disk, or in the prefix of a WordPerfect document (which is commonly known as a template macro.)

NOV-B00656856
CONFIDENTIAL

To set these properties at run-time and play a PerfectScript script that is stored as a WordPerfect document, use the following example:

```
PScript1.ScriptName = "c:\office\macros\test.wcm"        'Path and file name to the script
PScript.ScriptType = PS_FILE_SCRIPT                          'Defined in wpciwin.txt
PScript.Action =PS_ACTION_PLAYSCRIPT                     'Defined in wpciwin.txt
```

To play a PerfectScript script that is stored in the prefix of a WordPerfect document template, set the following properties at run-time:

```
PScript1.ScriptName = "Template Script Name"          'The name of the script
PScript1.ScriptName = "c:\office\macros\test.wcm"   'Path and file name to the template
PScript.ScriptType = PS_TEMPLATE_SCRIPT               'Defined in wpciwin.txt
PScript.Action =PS_ACTION_PLAYSCRIPT                  'Defined in wpciwin.txt
```

The WPCIWIN VBX also allows the Visual Basic programmer to send PerfectScript commands directly to most PerfectOffice applications.

One other feature of the WPCIWIN VBX is access to the PerfectScript Command Inserter. The Command Inserter lists each product's commands and their parameters, and allows the developer to construct the command using the Command Inserter interface, then copy the command to the clipboard. The command can then be pasted to the Visual Basic code window.

The WPDLG VBX custom control gives the Visual Basic developer access to the PerfectFit Technology file dialogs that are found in the PerfectOffice 3.0 applications. This dialog offers unparalleled functionality in the areas of file management, viewer technology, customization and network integration. Using the dialogs in a Visual Basic application can give the solution the common look and feel of the PerfectOffice 3.0 suite.

Using the WPDLG VBX, the developer can access the Open, Save As and Select Directory versions of the PerfectFit file system dialogs. This includes all of the setup, history list and file management capabilities of the dialogs that exist in the PerfectOffice products. The Save As dialog also checks for valid filenames and overwrite requests as do the product file system dialogs.

PerfectOffice 3.0 has brought Quattro Pro and the other applications in the suite closer than ever with interface integration and common look and feel. One of the last remaining pieces of Quattro Pro that has yet to be integrated with the rest of PerfectOffice 3.0 is the scripting language. Quattro Pro still makes use of its own macro language. This necessitates a separate VBX file to integrate Quattro Pro

NOV-B00656857
CONFIDENTIAL

with Visual Basic. Even though it is not as comprehensive as the WPCIWIN VBX, the integration is quite powerful when considering the abilities of the Quattro Pro macro language.

Pros and Cons of Visual Basic

The main advantage of Visual Basic is the integrated development environment. The Visual Basic interface allows the developer to create the application based on the interface objects, and define events based on what the user does with that interface. The event-based programming model is easy to learn and use, and the Visual Basic language is powerful enough to take advantage of lower level functions found in the Windows API and other production DLL's

Visual Basic also takes advantage of existing knowledge and experience. Visual Basic is widely used, especially in corporate MIS departments and independent solutions developers. By opening PerfectOffice to Visual Basic, these professionals can apply their Visual Basic knowledge with the powerful advantages of PerfectOffice's applications.

By using Visual Basic the developer builds on the thousands of Visual Basic add-on utilities, greatly extending Visual Basic's capabilities. Microsoft made Visual Basic extensible through special controls called VBXs, and the third party community has taken full advantage of that extensibility, creating literally thousands of VBX tools. PerfectOffice also integrates with Visual Basic through a special VBX.

For the organization that has a substantial investment in PerfectScript or in WordPerfect macros, using Visual Basic will allow them to leverage that investment by using the functionality described above. The PerfectScript scripts can be used as components of the solution. The developer also has the ability to record PerfectScript code modules and integrate them into a Visual Basic solution. Because the PerfectScript system takes care of the execution of the script, the execution is the same as running the script from PerfectOffice.

However, the PerfectScript code modules that are integrated into the solution must be distributed with the solution. This can make distribution difficult because of the multiples of files that may be needed. Another down-side is that playing PerfectScript commands from Visual Basic offers less control over the execution of PerfectScript commands than playing them from a script gives. Also, playing individual commands from Visual Basic involves more overhead than playing a script directly from PerfectScript, and could be a performance problem in larger

NOV-B00656858
CONFIDENTIAL

applications.

Learning more About Visual Basic

The VBX files provided in PerfectOffice are documented in a separate white paper titled *Integrating PerfectOffice With Visual Basic* and in the PerfectOffice Developer's Reference Guide.

Developing Add-ins Using WOAPI

For the greatest degree of control and integration, a solution or add-in program can exploit the WordPerfect Open API (Application Programming Interface) of the PerfectOffice applications (as of this writing, the PerfectOffice applications that support the WOAPI are WordPerfect, GroupWise, Presentations, and DAD). WOAPI provides a means to "tap" into the command flow of the program, and literally take it over. In most cases, solutions that use the WOAPI deal with only select portions of a PerfectOffice application. For example, most document management solutions are designed to take control of the file saving and opening tasks. Whenever the user choose the open or save commands, the add-in program is activated.

About WOAPI

WOAPI (also called the third-party handler, or TPH) uses a Windows DLL as the add-in mechanism. This DLL is bound with the application when the application is first launched, so that whenever the user chooses a command or invokes an action, it is "passed through" the DLL before it is ever acted upon.

For example, when the user chooses the New command from the File menu, the application passes this information to the DLL, which can determine if it wants to get involved. The DLL can decide to let the action go through, or it can modify the action in any way desired. It can also "trap" or inhibit the action. This is one method that can be employed in add-in applications to remove or restrict features in a PerfectOffice application.

Solutions that use the WOAPI must be written in C or C++, and compiled as a Windows DLL. The basic construction of this DLL is not unlike other Windows DLLs, except that additional entry, exit, and version functions are required to connect with a PerfectOffice application. Additionally, C programming knowledge is required to receive and send messages between the DLL and the

NOV-B00656859
CONFIDENTIAL

PerfectOffice application.

Pros and Cons of Using the WOAPI

The WOAPI allows the C programmer the best of both worlds: the flexibility and functionality of the Windows environment, with a direct connection to PerfectOffice applications. WOAPI is perhaps the best choice if total control and speed are important issues in a specific solution.

However, add-in applications that use the WOAPI must be written in C as a Windows DLL, and requires advanced programming techniques. C/C++ development tools must be used for debugging WOAPI DLLs, and development time of such an add-in is usually much higher than with the other methods described in this paper, and usually require an experienced programmer.

It is also necessary to configure the PerfectOffice application you wish to integrate with to accept the DLL. This configuration involves adding a reference to the DLL in the user's Binary Initialization File (this is a settings file that is shared by all PerfectOffice applications). Novell provides sample code of how to configure a WOAPI DLL with the user's Binary Initialization File in a software developer's kit, available from the Developer Relations Group at Novell.

Learning More About WOAPI

The WOAPI of PerfectOffice is documented in the PerfectOffice Software Developer's Kit, available from Novell. This kit, available in printed and CD-ROM form, provides useful technical information for developing add-in applications using the WOAPI, and come with samples you can use as springboards for your own projects.

Summary

When should you choose one of these development environment over another? This table helps to clarify the types of tasks each custom development environment is most suited for, and the users who can get the greatest benefit from them.

	Development Tool			

NOV-B00656860
CONFIDENTIAL

PX0410

Selection Criteria	PerfectScript	AppWare	Visual Basic	WOAPI
Degree of Control	Medium	Medium	Medium	High
Experience Required	Medium	Low	Medium	High
Depth of Functionality	Medium	Medium	High	High
Development Time	Medium	Low	Low	High
Ease of Creating User Interface	Medium	Medium	Easy	Hard
Ease of Distribution	Easy	Medium	Easy	Easy
Type of Apps to integrate	PerfectOffice apps	NetWare	Other Apps	Create add-ins
Solution Size	Small to Medium	Small to Medium	Small to Large	Medium to Large
Type of Application	Script	EXE	EXE	DLL
Language	PerfectScript	none (visual)	Basic	C/C++/Pascal

For More Information

Refer to the following files for more information on developing custom solutions with the PerfectOffice development environment.

**** Warren: list WWW, home pages, other SDKs, sources ****

File name Location What it discusses

NOV-B00656861
CONFIDENTIAL

PX0410

Disclaimer

(Legal department: please provide desired copyright notice and disclaimer here.)

NOV-B00656862
CONFIDENTIAL