README for App-Link™ Universal Server Edition Version 2.0

This document includes updated information for the documentation provided with the App-Link RADX Control Set 2.0. The information in this document and in the Help system is more up-to-date than that in the manual. The list of known problems outlined in this document will be corrected in subsequent updates.

Contents

Introduction

The Universal Server edition of App-Link provides the application programmer with the ability to easily create server applications that expose services to other programs via the App-Link Universal Client. The Universal Client is a special version of the App-Link control that can only talk to a Universal Server version of App-Link. Once you have developed a server application using the Universal Server, you can publish its interface for developers.

Developers that wish to access the services of your server can do so via the Universal Client control set, which can be freely distributed with your Server. Developers can also download the Universal Client control set from our web site. What this means is that you can now easily develop a programming API to your application. Sure, you can also do this with OLE or DDE, but with App-Link it's much easier, faster and more reliable.

The Universal Server and Client controls provide the same properties, methods and events as our standard App-Link RADX controls. For this reason, we reuse the same manual and on-line help file. Aside from the obvious difference in control names, the only other difference between the two products is the distribution requirements. The Universal Server edition allows you to redistribute a design-time version of our Universal Client control and companion license file. For more information on distribution, please refer to the section entitled ***Distribution of Universal Server Applications*** below.

What's New in Version 2.0

The following items run down the major differences between this version of App-Link and version 1.1:

- New 32-bit and 16-bit OCX controls
- New 32-bit NetBIOS communications server
- New property page dialog for configuration of OCX controls
- New methods
- New events
- Updated manual
- Additional example programs

New Methods

Unlike the original Microsoft VBX control specification, the new OCX technology supports the use of methods. App-Link 2.0 now includes method counterparts for the psuedo-methods previously supported via the *Command* property. Although the *Command* property is still available for your use, we encourage you to use the method implementations instead.

The following methods are supported App-Link 2.0:

METHOD	DESCRIPTION
Broadcast	Broadcast a message
GetComputerName	Retrieve the local computer name
GetNextQueuedMessage	Receive the next message from the queue in-line (ReceiveMessage is not fired)
Send	Send a message
ReceiveNextQueuedMessage	Receive the next message from the queue and fire the ReceiveMessage event

New Events

Events are used by custom controls to inform the application that a particular action related to the control has taken place.

The following events were added to App-Link 2.0:

EVENT	DESCRIPTION
Error	Handles asynchronous App-Link error events
NotifyMessage	Fires when a message is placed in a socket's queue

General Information

Compatibility with Version 1.1

From a run-time perspective, an App-Link 2.0 application is 100% compatible with its version 1.1 counterpart. You'll be able to develop applications that take advantage of the new OCX technology, and integrate them with exisiting applications that use the version 1.1 VBX. In fact, we even include an updated VBX control and run-time libraries as part of the 2.0 package.

Migrating source code from VBX to OCX technology is pretty simple too. Most of the conversions are done under the covers the first time that you load the application under VB4. The VB4 IDE will detect the presence of an App-Link OCX-equivalent control and ask you if you would like to convert the application. Select Yes to proceed with the conversion process.

Note: The manual states that the paramaters for the ReceiveMessage Event handler should be changed to indicate ByVal. This is an error in the manual, and can be ignored. The help file is correct.

User-Type Conversion Functions

App-Link 2.0 continues to provide support for converting a VB user-type to string and vice-versa. However, due to differences in 32-bit versus 16-bit architectures, you will need to pay close attention to the alignment of variables within structures that are ***shared*** between 32-bit and 16-bit applications. VB4 32-bit applications require user-type members to be aligned on natural boundaries. For example, a 4-byte *Long* variable must be aligned on a 4-byte boundary. If you create a type that is shared between 16 and 32-bit applications, you must ensure that the type definition accounts for proper alignment. *The conversion functions read and write memory blocks and do not look at the details of the underlying structure*.

The following is an example of a valid user type definition that can be shared between 16 and 32-bit applications:

```
User Type
    L1 As Long
    L2 As Long
    I1 As Integer
End Type
```

All members fall on natural boundaries. L1 and L2 are aligned on 4-byte boundaries, and I1 is aligned on a 2-byte boundary.

This user type cannot be shared between 16 and 32-bit applications:

```
User Type
    L1 As Long
    S1 As String * 3
    L2 As Long
End Type
```

The alignment of L2 will not be on a 4-byte boundary because of the odd-length specification of S1. To fix the alignment problem, you'll need to define S2 as String * 4, or introduce another filler variable of length 1 between S1 and L2 as follows:

```
User Type
    L1 As Long
    S1 As String * 3
    Filler1 As String * 1
    L2 As Long
End Type
```

Now the alignment is correct.

Distribution of Universal Server Applications

Okay, you've debugged your App-Link Universal Server application and are ready to distribute it. In order to run the application though, you'll need to include the App-Link run-time files, but which ones? This section guides you through the process of selecting the correct set of App-Link Universal Server run-time files to distribute.

Select the description that best describes the environment your application will run in from the list below, then follow the instructions beneath your selection to copy the required App-Link Universal Server run-time files. It's that simple.

Windows 3.x, WFW, or Windows95 / Windows NT (running 16-bit App-Link applications exclusively, with no plans for running 32-bit App-Link applications in these Win32 environments):

Copy all of the files under the \ALUNS20\REDIST\WIN16 sub-directory.

If your application uses the 16-bit OCX version of the controls, you need to distribute the Universal Server control - ALUS16.OCX and Universal Client - ALUC16.OCX. Otherwise, if you are using the VBX controls you need to include the Universal Server run-time library - ALUS.DLL and Universal Client VBX - ALUC.VBX. Please be sure to rename ALUS.DLL to ALUS.VBX prior to distributing your application. Failure to do so will cause the application to fail when you try to load it.

To use the Universal Client in design mode you also need to distribute the license file for the Universal Client. This file is named ALUC.LIC and can be found in your WINDOWS\SYSTEM or SYSTEM32 sub-directory.

The following files are necessary if you are using App-Link Universal Server for communications over a NetBIOS network:

APLKNB.EXE
APLKNBP.DLL

Windows95 or Windows NT running 16-bit or 32-bit applications or both:

Copy all of the files under the \ALUNS20\REDIST\WIN32 subdirectory.

If your application uses the 16-bit OCX version of the controls, you need to distribute the Universal Server control - ALUS16.OCX and Universal Client - ALUC16.OCX. If your application uses the 32-bit OCX controls, you need to distribute ALUS32.OCX and ALUC32.OCX. To use the Universal Client in design mode you also need to distribute the license file for the Universal Client. This file is named ALUC.LIC and can be found in your WINDOWS\SYSTEM or SYSTEM32 sub-directory.

The following file is necessary if you are using App-Link Universal Server for communications over a NetBIOS network:

ALNB32.EXE

Under no circumstances should you distribute the design-time version of the Universal Server VBX (ALUS.VBX) or the Universal Server license file (ALUS.LIC), as that is a violation of the license agreement. These design-time components are licensed to a single user, and should not be included in the distribution of your application.

List of Known Windows Problems Affecting App-Link RADX

The following is a list of Windows problems that we are aware of that affect the operation of the App-Link RADX controls.

GPF Under Windows 95 Running Multiple 16-bit Applications

When running multiple 16-bit applications concurrently, shutting down the applications in a certain order may cause Windows 95 to crash. This problem is caused by a bug in the Microsoft OLE layer and can be resolved by applying their OLE32 Update. The file can be downloaded at: **http://www.microsoft.com/windows/software/oleupd.htm**. If you have not done so already, we also recommend that you download and install the Microsoft Windows 95 Service Pack 1.

Windows NT Hangs Unloading 16-bit Applications

When running 16-bit applications concurrently, unloading 16-bit applications may cause Windows NT to hang. This problem is caused by a bug in the Windows NT WOW layer. To resolve this problem you must install the Windows NT 3.51 Service Pack 4. The file can be downloaded at:

http://www.microsoft.com/bussys/winnt/winnt-public/fixes/usa/nt351/ussp4/i386

In addition, you must configure any 16-bit applications to run in a separate WOW address space. This option is available when adding program items to a group window in the Program Manager. See the Windows NT 3.51 user documentation for more information on running 16-bit WOW applications in a separate address space.