

Supporting ADO with FarPoint Controls

a solution for ActiveX developers

The logo for FarPoint, featuring the word "FarPoint" in a serif font. The letter "o" in "Point" is replaced by a blue, 3D-rendered sphere with a white highlight and a soft blue glow around it.

Supporting ADO with FarPoint Controls

This paper provides an overview of ActiveX® Data Objects (ADO), its data-binding features, and how those features are implemented in FarPoint Technologies™ products.

You might want to read this paper if you own FarPoint Technologies' products and use their data binding features. Throughout the paper, we assume that you are familiar with FarPoint controls, the Microsoft® Visual Basic® development environment, and the basic principles of data binding. A glossary is available in "Acronyms Defined" on page 12.

INTRODUCTION TO ADO?

The following sections provide a brief description of ADO.

Background

ADO provides a new and powerful means of accessing data stored in databases. As a single, high-level technology for accessing all types of databases, ADO is simpler and more efficient to use than the two older data access technologies, DAO (Data Access Objects) and RDO (Remote Data Objects). ADO combines the functionality of DAO and RDO into one interface and provides even more features.

Rather than having to choose DAO or RDO depending on the type of database you need to access, you can use ADO to access any type of database or file. For example, while DAO can connect to Microsoft Access .MDB databases, ADO can access HTML, plain text, Internet, spreadsheet, and e-mail files, as well as .MDB and other types of databases.

The ADO Data Model

ADO is a language-neutral object model that lets you manipulate data accessed by an underlying OLE DB provider. (An OLE DB provider is a data manager that interfaces directly with a database. For example, Microsoft Jet is the data manager on which Microsoft Access is built.)

ADO's object model is simpler than DAO's, containing fewer objects and more properties, methods, and events, and combining much of DAO's functionality into single objects.

Many of the objects in the ADO data model are similar to RDO objects, but with enhanced functionality. For example, ADO's Recordset object contains records returned from a database plus the cursor for those records. (A cursor helps manage and manipulate a set of data.) The Recordset object is similar to RDO's rdoResultset object. With ADO, however, you can open a Recordset object without explicitly connecting to the database source. And if you do connect to the database, you can open multiple Recordset objects on the same connection.

You can learn more about the ADO data model from the references listed in “Where can I find out more about ADO?” on page 5.

Do I have to use ADO?

You do not have to use ADO to bind FarPoint controls to databases right now, but you might want to do so. ADO, DAO, and RDO are different systems for providing data binding in your applications. You will need to decide which system best suits your needs, and use FarPoint Technologies’ products according to which data-binding system you choose.

For you, this will mean that you have a choice of how to bind controls to databases:

- The “old” method that uses DAO or RDO
- The new method that uses ADO

How did ADO come about?

DAO, coupled with the Jet data manager, is currently the most widespread method of desktop data access. However, DAO is restricted to working with .MDB files (created with Access) and with ISAM files (such as those created with Paradox and FoxPro® databases). RDO is currently used for accessing data over a network, but is restricted to working with relational ODBC sources.

ADO is the successor technology to DAO and RDO. ADO goes a step further than either DAO or RDO by being able to gather data from legacy sources and from non-relational data (such as e-mail and HTML pages).

ADO is Part of the UDA Strategy

ADO is part of Microsoft’s Universal Data Access (UDA) paradigm, a new strategy to access data in its current location and format using a single, high-level data access model.

Using UDA tools prevents the expensive and often impractical task of gathering data from various types of databases into a single data store. Also, since UDA is based on open industry specifications, you are not required to commit to a single vendor’s products.

The following UDA tools are available as Microsoft Data Access Components (MDAC):

UDA Tool	Description
ADO	ActiveX Data Objects The interface you use to access data (ADO then uses the other UDA tools)
RDS	Remote Data Service Application that allows data access across a network
ODBC	Open DataBase Connectivity Protocol that allows access to a variety of database types
OLE DB	Object Linking and Embedding DataBase Application that allows access to data stored in both relational and non-relational formats

Together, these tools enable you to access just about any data source, all through the ADO interface.

What will happen to DAO?

DAO technology will be around for many more years. Several new DAO tools are included with the Visual Basic 6.0 development system, as well as a new Jet version. Microsoft Access itself uses Jet and DAO. Ultimately, though, all applications will move to ADO, as

Microsoft has stated that, as of the Visual Studio® 6 development system, ADO is the standard data access object model for Microsoft tools.

Data Binding Differences

This section describes the differences between the “old” data binding technologies (DAO and RDO) and the “new” data binding technology (ADO) for Visual Basic users. Each technology uses a different Visual Basic data control to connect to a database, retrieve data, and link the data to a bound control on a form.

- DAO** When you bind a control to a database using DAO, the bound control uses the Data control to access the database. The Data control manages how your control accesses the database, and you are limited by the Data control’s limitations. For example, you can only bind your control to a database at design time, not at run time.
- RDO** Similarly as for DAO, when you bind a control to a database using RDO, the bound control uses the Remote Data control to access the database.
- ADO** When you bind a control to a database using ADO, the bound control uses the ActiveX Data Objects control (the “ADO data control,” or ADODC) to access the database. The ADO data control, in turn, uses the OLE DB provider to gain universal access to the data source. This model provides you the opportunity to bind your control to databases dynamically, at run time. For example, you can set up database access using the OLE DB provider and naming the database source. Then, when needed, you can bind your control to the database using the new Data properties in Visual Basic 6 and the data control.

ADO also lets you bind an object directly to another object, without using the ADO data control. (For an example, see “Binding at Run Time without Using the ADO Data Control” on page 10.)

Advantages of ADO

ADO can work very similarly to DAO data binding, if you prefer and you are used to that data access method. ADO also provides remote data access. Therefore, ADO can take the place of DAO and RDO data binding, while providing additional features and stability.

Other advantages of using ADO include:

- You can create or change data binding settings at run time as well as design time.
- You can construct and manipulate a Recordset object without being connected to a database, freeing the server for other tasks.
- Your control receives a single message when multiple records are changed in the database, as opposed to a separate message for each record.
- You can use SQL statements to perform tasks such as filtering data from the database.
- You can use database-stored procedures.
- You can modify any record (not just the current record), and delete multiple records.
- You can expose several sets of data from the same data source at the same time.
- Any ActiveX control that supports ADO can be the data source for another ADO control.
- You can access many types of data sources, including legacy and non-relational data.

- You have only one interface to support instead of possibly several APIs to handle the different types of databases.

Why do I use one or the other?

This section provides some guidelines for deciding whether to use ADO for new projects and for existing projects. Remember that, starting with Visual Studio 6.0, ADO is the standard data access object model for Microsoft tools.

For New Projects

You will probably want to use ADO whenever possible to take advantage of ADO's features now and in future updates. We recommend that you consider using ADO in any new projects you create. If you will be using a data source other than Access or working with data over the Internet or an Intranet, you will definitely want to consider using ADO.

Microsoft recommends staying with the combination of Jet and DAO if you are developing a desktop database solution using only .MDB or ISAM files, but realize that you will eventually need to upgrade the project to ADO.

For Existing Projects

Whether or when you convert existing projects to use ADO is up to you. We recommend upgrading, not only to take advantage of ADO's features and stability, but also because you will then be able to take advantage of future updates.

You cannot simply upgrade your existing projects that use DAO or RDO data binding to use ADO; however, you can complete a one-time conversion process for each of your existing projects. The conversion process for each FarPoint product is provided in the Read Me help file that accompanies the product.

What do I do differently?

Setting up data access using ADO can be similar to using DAO. Most of the methods are the same. However, there are additional steps necessary, and some steps that are optional, depending on whether and how you want to customize the access you provide. Once you are connected to your data source, you can access the data almost exactly like DAO.

This paper provides instructions for setting up data binding in Visual Basic 6 using FarPoint ADO controls (with and without the ADO data control) and different database access methods. You can use much more flexible methods of data binding than the ones described here, because of ADO's features and capabilities. These instructions are provided to introduce you to setting up some simple examples of data binding if you have not done so using ADO.

Are there any caveats?

Because ADO's data model is simpler than DAO's, some DAO programmers might initially find it difficult to find the appropriate ADO object, collection, property, method, or event.

ADO 2.0 does not support all of DAO's functionality, such as users, groups, and so on. Using the Access Jet 3.51 database engine with ADO 2.0 can reveal several problems, including an inability to create database objects such as tables and fields. If these items or database security are important to your project, you need to investigate ADO 2.1, which includes a new library called ADOX (ActiveX Data Objects Extensions for DDL and Security). ADOX includes tools for accessing the structure, security model, and procedures stored in a database. If you develop with Access/Jet, investigate the new Jet 4.0 database engine for improvements to that OLE DB provider.

ADO does represent an added layer for OLE DB access. While most projects will benefit from the productivity gain and ease of use despite the extra overhead, not all projects will. ADO does allow you to talk to OLE DB directly if necessary.

ADO's capabilities are growing with each release. You will have to decide whether it is the best time for you to convert your existing project, and weigh the conversion time versus the benefits of using ADO. You will also have to use a development environment that supports ADO controls, such as the Visual Basic 6 development environment.

Where can I get ADO?

ADO is a component of the MDAC (Microsoft Data Access Components) toolgroup, available from the Microsoft website at <http://www.microsoft.com/data/ado/>.

ADO Version 2.0 is part of Microsoft's Visual Studio 6 development system.

ADO Version 2.1 (and Jet 4.0) ships with Microsoft's SQL Server 7.0 and Access 2000.

Where can I find out more about ADO?

The following documentation is available from Microsoft:

- ADO comes with documentation in HTML format. When you install the MDAC components, enable the documentation for ADO. After you have installed ADO, in C:\Program Files\Common Files\System\ADO\Docs, open the file DEFAULT.HTM. The Program Files menu will also have an entry for the MDAC components.
- In the *MSDN™ Library Visual Studio 6.0*, see the topic "Getting Started with ADO 2.0." You can also search online for "ActiveX Data Objects in Visual Basic."

The following websites provide more information about ADO:

Site	Address
Microsoft's ADO Page	http://www.microsoft.com/data/ado/
The Development Exchange	http://www.devx.com
Visual Basic Programmer's Journal	http://www.vbpj.com
Visual Basic World	http://www.vb-world.net
Microsoft's Knowledge Base articles for ADO	http://support.microsoft.com/support
(A good article to start with is: ARTICLE-ID:Q168337 TITLE: HOWTO: Search for ADO Articles by Article Topic/Keyword)	

The following books provide more information about ADO:

Title	Author	ISBN
ADO 2.6 Programmer's Reference	David Sussman	1-861004-63-x
Serious ADO: Universal Data Access with Visual Basic	Rober MacDonald	1-893115-19-4
Beginning Visual Basic 6 Database Programming	John Connell	1-861001-06-1
ADO Examples and Best Procedures	William R. Vaughn	1-893115-16-x

ADO VERSIONS OF FARPOINT CONTROLS

FarPoint provides ADO versions of our ActiveX controls for each product as appropriate. Some FarPoint products do not provide ADO controls because those products do not provide data binding. Other products do not provide ADO versions of controls because they are not necessary.

For a summary list of which FarPoint products provide ADO versions of the ActiveX controls, see “Summary of FarPoint Products” on page 7.

Are there separate controls?

There are separate ADO and DAO versions of ActiveX controls provided for all products that are rowset bound. Products that are simple bound, such as Input Pro™, do not provide separate ADO versions. For more information about rowset-bound vs. simple-bound controls, see “What do simple bound and rowset bound have to do with it?” on page 6.

You must use the correct version of the control in your project according to the type of databinding you want to use. If you want to use ADO, you must use the ADO version of the ActiveX control.

What do simple bound and rowset bound have to do with it?

Simple bound controls have a setting that indicates to the container that the control is simple bound. The container then handles the data flow of the bound properties. The interaction between the bound control and the database is the same whether the control is bound to a DAO, RDO, or ADO data control. Therefore, simple bound controls, such as the controls in Input Pro, do not have separate ADO controls. You can bind the ActiveX controls in Input Pro (and other simple bound controls) to a DAO, RDO, or ADO data control.

On the other hand, rowset bound (or “cursor bound”) controls involve handling a group of records through the data control. The rowset bound control itself, rather than the container, handles the data flow. The interaction between the bound control and the database is different depending on whether the control is bound to a DAO, RDO, or ADO data control. Rowset bound controls that can use DAO and RDO cannot use the newer ADO. Therefore, rowset bound controls, such as the Spread™ control, have separate ADO versions of the ActiveX controls.

Recognizing the ADO Version of FarPoint Controls

For FarPoint products, the ADO version of the ActiveX controls have different naming conventions and icons to distinguish them, as follows:

- In the list of registered controls, the control name is followed by “(OLEDB)”.
For example, for Spread version 3.5, the control name in the list of registered controls is “FarPoint Spread 3.5 (OLEDB).”
- In your project toolbox, the icons for the controls each have a small yellow cylinder with the letters “FP” in them, as shown in the following picture of the spreadsheet ADO control icon for Spread version 3.5.



- The prefix on the name of the control is different: for ADO controls, the prefix is “fp” and for DAO controls, the prefix is “va”.
For example, for Spread version 3.5, the default name of the spreadsheet ADO ActiveX control in the project is “fpSpread1”. The default name of the DAO ActiveX control, for comparison, is “vaSpread1”.
- In the About box for the control, the icon with the yellow cylinder is displayed and the full control name is listed, which includes the acronym ADO.
For example, for Spread version 3.5, the full control name listed for the spreadsheet ADO ActiveX control in the About box is “Spread ADO 3.n.nn (OLEDB)”.

Summary of FarPoint Products

The following table summarizes FarPoint products’ support for ADO.

Product	Simple Bound	Rowset Bound	ADO Support Provided (Release Number)
Button Objx	Not Applicable	Not Applicable	Not Applicable
Calendar ObjX	fpClock, fpCalendar	fpPoster	3.0 and later (forthcoming)
Daily PlanIt	Not Applicable	fpDPlanIt	1.0.05 and later (forthcoming)
Input Pro	All controls	Not Applicable	2.1 and later
List Pro	fpCombo (edit field)	fpList and fpCombo (list box)	2.1 and later
Spread	Not Applicable	fpSpread	3.0 and later
Tab Pro	Imprint control	Tab control	3.0 and later (forthcoming)

BINDING FARPOINT CONTROLS USING ADO

The following sections give general instructions for setting up ADO data binding for FarPoint controls.

What are the basic steps?

The basic steps in a simple example of ADO data binding are:

1. Put your controls on the form.
2. Set up the database (designate the database to which you want to bind and specify which database table you want to access).
3. Tell your control which data control to access, or provide the code for binding directly to another object.

Beyond the Basics

There are many ways to bind your control using ADO. You can bind at design time or at run time. You can use the ADO data control or not. It is not necessary to use the ADO data control because you can bind a data-aware ADO control directly to a Recordset object that is created at run time with code.

The following sections contain detailed examples for binding your control to a database using these four methods:

- “Binding at Design Time Using the ADO Data Control” on page 8
- “Binding at Design Time without Using the ADO Data Control” on page 10
- “Binding at Run Time Using the ADO Data Control” on page 10
- “Binding at Run Time without Using the ADO Data Control” on page 10

*Further
Customizing for
Simple-bound
Controls*

You can further customize your simple-bound data binding using the DataFormat and DataMember properties provided by the Visual Basic 6 development environment. The DataFormat property lets you specify a format to be applied to data as it is read in from the database. The DataMember property lets you specify separate portions of the database to work with, which you would configure in your ODBC settings. For more information about these properties, refer to the Visual Basic 6 documentation.

*Further
Customizing for
Rowset-bound
Controls*

The DataFormat and DataMember properties in the Visual Basic 6 development environment are standard extender properties, but they are only added to simple-bound controls. Data-aware ADO controls that use rowset binding must provide custom properties with functionality similar to the DataFormat and DataMember properties.

For example, FarPoint’s Spread 3.5 ADO control does not support the DataFormat property, but it does support the DataMember property. The DataMember property in the Spread 3.5 ADO control is a custom property, not an extender property.

*Customizing by
Exposing
Multiple
Recordsets*

One example of further customizing is using ADO to expose several sets of data at the same time from the same data source.

Before ADO, the data source could expose only one set of data at a time. When you wanted to look at a different set of data, you had to set properties on the data source to change the set of data, and then call the Refresh method to update the bound controls.

With ADO, a data source can expose several sets of data at the same time. Setting the DataMember property on a bound control indicates which set of data (“data member”) in the data source to bind the control to. Several bound controls set to different data members of the same data source can exist in the same project at the same time.

An example of exposing multiple recordsets is shown in “Exposing Multiple Recordsets” on page 11.

*Binding at
Design Time
Using the
ADO Data
Control*

The following example uses Microsoft Jet 3.51 as the OLE DB provider. Your choice of providers will depend on the location and type of data you are accessing.

You must use the ADO version of your ActiveX control in Visual Basic 6.0 or later. (You can bind controls using other Visual Studio products; consult the documentation for that product for more information.)

Complete the following steps to bind your control to a database using the Jet 3.51 OLE DB provider:

1. Add the Microsoft ADO Data Control 6.0 (OLEDB) to your project, and then create a Microsoft ADO Data Control 6.0 (OLEDB) on the form.
2. Right-click on the data control, and then select ADODC Properties from the pop-up menu. The Property Pages display.
3. In the Property Pages, on the General property page, under Source of Connection,
 - a. Click the Use Connection String option button.
 - b. Click the Build button. The Data Link Properties dialog displays.
4. In the Data Link Properties dialog, on the Provider tab,
 - a. Select Microsoft Jet 3.51 OLE DB Provider (or your preferred provider) from the list of available providers.
 - b. Click the Next button. The Connection tab displays.
5. On the Connection tab,
 - a. Under item 1 on the tab, provide the database name to connect to either by typing it in the field or by clicking the ellipses button and selecting the database using the Select Access Database dialog.
 - b. Under item 2 on the tab, if necessary, enter the user information required to log on to the database.
 - c. Click the Test Connection button.
 - d. If the connection succeeded, select OK in the message box. If the connection did not succeed, repeat step 5.
6. In the Data Link Properties dialog, click OK to continue. The Data Link Properties dialog closes, redisplaying the Property Pages.

In the Property Pages, your connection string has been created in the Use Connection String text box. You can use your connection string at run time to access this database connection.
7. In the Property Pages, click the RecordSource property page.
 - a. On the RecordSource property page, under RecordSource, from the CommandType drop-down list box, select 2 - adCmdTable. (Selecting adCmdTable is one example; you can select other options. The rest of step 7 might differ depending on your selection.)
 - b. From the Table or Stored Procedure Name drop-down list box, select the database table to which you want to connect.
 - c. Click the OK button.
8. Add a FarPoint ADO ActiveX control to your project, and then create a FarPoint ADO ActiveX control on the form.
9. For the control, set the DataSource property to the name of the Microsoft ADO Data Control 6.0 (OLEDB).

Binding at Design Time without Using the ADO Data Control

The following example of binding at design time without using the ADO data control is similar to the steps for binding at design time using the ADO data control, minus the steps for setting up the ADO data control.

The bound control is not aware of what it is actually bound to; the control only knows the name of its data source and the name of the data member it is accessing in that data source.

1. Add a FarPoint ADO ActiveX control to your project, and then create a FarPoint ADO ActiveX control on the form.
2. For the control, set the DataSource property to the name of the data source.

Binding at Run Time Using the ADO Data Control

The following example of binding at run time using the ADO data control is essentially the code equivalent of setting the properties during design time, as described in “Binding at Design Time Using the ADO Data Control” on page 8.

```
Adodc1.CommandType = adCmdTable
Adodc1.ConnectionString = "Provider=Microsoft.Jet.OLEDB.3.51;
Persist SecurityInfo=False;Data Source=<path and filename of
database>"
Adodc1.DataSource = "<name of table or stored procedure>"
Set fpSpread1.DataSource = Adodc1
```

Binding at Run Time without Using the ADO Data Control

The following example of binding at run time without using the ADO data control creates a Recordset object with code, and then binds the Recordset object directly to an ADO fpSpread control named fpSpread1.

```
Dim rs as ADO.Recordset
Sub Form_Load
Set rs = New ADO.Recordset
With rs
.Fields.Append "StringField", adBSTR
.Fields.Append "CurrencyField", adCurrency
.Open
.AddNew
.Fields("StringField") = "Record 1"
.Fields("CurrencyField") = 5.89#
.Update
.AddNew
.Fields("StringField") = "Record 2"
.Fields("CurrencyField") = -50.50#
.Update
.AddNew
.Fields("StringField") = "Record 3"
.Fields("CurrencyField") = 999.99#
.Update
End With
Set fpSpread1.DataSource = rs
End Sub
```

Exposing Multiple Recordsets The preceding example creates one Recordset object and exposes its data for access. You can also create objects that act as data source objects (like a data control does) and that expose several recordsets, each recordset with a different DataMember name. Use the DataSourceBehavior property, which applies to the UserControl and Class objects in Visual Basic 6. (The DataSourceBehavior property is a new data binding feature available in Visual Basic 6.)

For example, if you place the preceding code in a UserControl object's code in the UserControl_Initialize event and instead of there being just one recordset there are three named rs1, rs2, and rs3, you would need to add the following code to the Initialize event:

```
DataMembers.Add "rs1"  
DataMembers.Add "rs2"  
DataMembers.Add "rs3"
```

You would also need to add the following code to the GetDataMember method for the UserControl (the GetDataMember method appears in the list of event procedures for the UserControl object after you set its DataSourceBehavior property to vbDataSource):

```
If DataMember = "rs1" Then  
    Set Data = rs1  
ElseIf DataMember = "rs2" Then  
    Set Data = rs2  
ElseIf DataMember = "rs3" Then  
    Set Data = rs3  
End If
```

Then, when you draw this object on a form in a Visual Basic 6 project and put an ADO data-aware control on the form, the name of the UserControl object appears in the drop-down list for available data sources for the bound control. If you select the UserControl as the data source, the data members rs1, rs2, and rs3 appear in the drop-down list of available data members for the DataMember property of the bound control.

You do not need to specify the DataMembers at design time, in either the bound control or the data source control. The ADO data control does not know what data members it will expose until you set other properties to tell it which OLE DB provider to use and which database to access; only then does it set up the DataMembers collection with the names of the tables and stored procedures available in the database.

ACRONYMS DEFINED

The following acronyms appear in this paper:

Acronym	Definition
ADO	ActiveX Data Objects
ADODC	ActiveX Data Objects Data Control
ADOX	ActiveX Data Objects Extensions (for Data Definition Language and Security)
API	Application Programming Interface
DAO	Data Access Objects
HTML	HyperText Markup Language
ISAM	Indexed Sequential Access Method
MDAC	Microsoft Data Access Components
MDB	Microsoft DataBase
MSDN	MicroSoft Developers Network
ODBC	Open DataBase Connectivity
OLE DB	Object Linking and Embedding DataBase
RDO	Remote Data Objects
RDS	Remote Data Service
SQL	Structured Query Language
UDA	Universal Data Access

About FarPoint

FarPoint Technologies, Inc., a privately held company with corporate headquarters located in Morrisville, North Carolina, USA, is a leading developer and publisher of professional components for Windows development. Our award-winning tools benefit leading corporations, software companies, and independent consultants around the world as a cost-effective solution for building distributed enterprise-wide applications for commercial or in-house use. From the CEO to our newest team member, our goals are the same: to provide you with the innovative tools and support you need to effectively compete in today's competitive development market.

FarPoint's foundation began in 1991 (then Prescription Software) by re-introducing Drivers Professional Toolbox for Windows, what is believed to be the first development package available for the professional Windows developer, with thirteen new DLLs, including a fully featured spreadsheet control. Developers using languages such as C or C++ quickly realized the benefits of using our pre-built DLLs instead of spending their valuable resources to develop these controls internally.

The advent of Visual Basic 1.0 presented FarPoint with a new opportunity in the professional development market. After Microsoft asked that our spreadsheet be made available for their users, we introduced the first spreadsheet control ever available for Visual Basic in our product Visual Architect. Along with the spreadsheet control, Visual Architect offered formatted-edit controls in this new VBX format. The rising popularity of Visual Basic and our controls allowed us to introduce several new products to the market in the coming months: Spread/VBX, Aware/VBX, Grid/VBX, Tab/VBX, and Tab Pro, while continuing support for Professional Toolbox for Windows.

Today, FarPoint is one of the oldest and most respected vendors in the industry. We continue to cater our products to professional Visual Basic and Visual C++ developers by offering our components as a 32-bit ActiveX control, 16- and 32-bit DLLs, and a VBX control.

Our award-winning tools benefit leading corporations, software companies, and independent consultants around the world as a cost-effective solution for building distributed enterprise-wide applications for commercial or in-house use. For more information, check out our web site or contact us directly at the North American office or European office of FarPoint:

North America Contact

FarPoint Technologies, Inc.
808 Aviation Parkway
Suite 1300
Morrisville, NC 27560 USA

Phone: 919-460-4551
email: fpsales@fpoint.com
Web: www.fpoint.com

Europe Contact

FarPoint Europe Ltd.
Whiteleaf, Roundabout Lane
West Chiltonton
Pulborough, West Sussex RH20 2RL
England

Tele: +44 (0) 1798 812 372
Fax: +44 (0) 1798 813 049
email: salesEurope@fpoint.com

Information in the white paper is subject to change without notice and does not represent a commitment on the part of FarPoint Technologies, Inc.

© 2003-2006 FarPoint Technologies, Inc. All rights reserved.

Unless otherwise noted, all names of companies, products, street addresses, and persons contained herein are part of a completely fictitious scenario or scenarios and are designed solely to document the use of a FarPoint Technologies, Inc., product.

Spread for Web Forms and Spread for Windows Forms are trademarks of FarPoint Technologies, Inc. Other brand and product names are trademarks or registered trademarks of their respective holders.