

Using Virtual Mode in Spread COM

FarPoint

This white paper describes using virtual mode in a Spread COM control. This white paper consists of the following contents:

| Contents | Page |
|---|-------------|
| How Does Virtual Mode Work? | 2 |
| Turning On Virtual Mode | 3 |
| How Can I Customize Virtual Mode? | 4 |
| Summary of Virtual Mode Properties | 7 |
| Returning Information About Your Data in Virtual Mode | 8 |
| When to Use Virtual Mode | 8 |
| Using Virtual Mode in a Bound Control | 8 |
| Using Virtual Mode in an Unbound Control | 11 |
| Using Virtual Mode When Importing Files | 16 |

Throughout the white paper, we assume that you are familiar with the Microsoft® Visual Basic® development environment and with Spread features, such as adding the control to your toolbox, putting the control on a form, and data binding. We also assume that you have installed the Spread ActiveX control. If you are not familiar with these things, please review the *Spread User's Guide* (for any Spread COM product version 3.0 or later) and the tutorial, "Adding Data in Spread COM," available on the FarPoint web site (<http://www.fpoint.com/support/whitep/whitep.html>).

This paper was originally written for version 3.0 of the Spread COM product. For later versions, simply use the VirtualRefresh Method in place of the VRefresh action.

This white paper uses a symbol (↪) to indicate code continued from one line to the next in the paper that should be typed all on one line in the code window.

How Does Virtual Mode Work?

Using virtual mode with your Spread control can improve your control's performance considerably. Turning virtual mode on lets the control read into a buffer only the amount of data necessary to display the requested rows. With virtual mode turned off, the control reads in an entire recordset before displaying any records.

For example, if you have a Spread control access a large database with thousands of records and virtual mode is turned off, the control will read in all the records in the database before displaying any records in the control. Reading in so many records may take a long time. Accessing the same database with virtual mode turned on means the control will read in only, for example, a few hundred records before displaying them.

Turning On Virtual Mode

Turn on virtual mode by setting the **VirtualMode** property to True.

Be Aware



Virtual Mode Behavior. There will obviously be some behavioral differences resulting from buffering a subset of records rather than reading in the entire set of records. For example, your application will not be able to sort your database or perform similar tasks that require the use of the entire recordset at once. Since virtual mode increases responsiveness by dealing with just a subset of the recordset at a time, then by definition virtual mode is not capable of performing database-wide tasks. Be aware also, that a Spread control using virtual mode cannot print using the SmartPrint option or freeze rows. Refer to the *Spread COM User's Guide* when you need more details about the Spread control's behavior when using virtual mode.

Specifying the number of rows

While in virtual mode, the control ignores the setting of the **MaxRows** property and uses the number of items specified by the **VirtualMaxRows** property. The default setting of the **VirtualMaxRows** property is -1, which specifies that the control use however many items are in the recordset, but that the control does not know the exact number.

If you prefer, you can tell the control the exact number of items in the recordset, or you can guess the number of items. If you guess the number of items, overestimate, because the control will read in only the number of items specified by the **VirtualMaxRows** property.

For best results, for example, to have the control best reflect where the user is in the recordset, set the **VirtualMaxRows** property to the number of items in the recordset.

Be Aware



Changing the recordset at run time. If you set the **VirtualMaxRows** property to the number of items in your recordset for a bound database, then bind the Spread control to a different recordset at run time, set **VirtualMaxRows** to -1 or to the number of items in the new recordset before you refresh the Data control. If you do not reset the value of the **VirtualMaxRows** property, the Spread control might not read in all the records in the new recordset.

What can I do with virtual mode?

You can customize how many records are buffered, along with many other virtual mode features. “How Can I Customize Virtual Mode?” on page 4 describes many of your options for customizing virtual mode. You will see some of these customization options in use in the examples in this white paper.

When virtual mode is turned on and the control requests additional data, the **QueryData** event occurs, offering you an opportunity to include your own code to add more data or further customize the control.

How does virtual mode handle property settings?

A spreadsheet can have properties that determine the behaviors of individual cells, rows, columns, and the entire sheet. An application using virtual mode will typically set sheet and column properties in a **Form_Load** event. Then, as each subset of rows comes into view, the application will typically set row and cell properties for those rows. The row and cell properties come from an external source (the source of the data, such as a bound database, a file, or an array). The properties may be set automatically, as when bound to a database, or set in the application’s code (such as in the **QueryData** event).

As you work with virtual mode, particularly if you use the **QueryData** event, keep in mind these distinctions about how the Spread control is handling property settings.

How Can I Customize Virtual Mode?

As described earlier, virtual mode reads in a certain amount of data, which is then available for viewing or working with. Additional data is not read into the control until it is requested by the user or the application.

The amount of data read into the control is called the *virtual page*. You can set the number of records to include in the virtual page. The control also can store records that have been read in memory, for faster control access. You can adjust this buffer size to an optimal size to conserve system resources or to increase control responsiveness, depending on your application’s needs.

In virtual mode, the control does not necessarily know how many rows of data there are in the complete set of data. Because the control might not know the number of rows, the scroll bar cannot use the scroll box to accurately represent where the user is in the data. Therefore, you can either tell the control the total number of records or an estimate of the number of records, or you can have the control display a special scroll bar.

The following sections discuss these ways to customize virtual mode.

**Tip**

Details about Customizing Virtual Mode. Refer to the *Spread COM User's Guide* when you need more information about customizing virtual mode than is described in the tables and examples in this section.

Optimizing Application Speed or Access

The properties for customizing virtual mode, the **VirtualRows** and **VirtualOverlap** properties, are set to default values to help optimize your application's speed. You can adjust the virtual mode properties to further optimize your application as needed.

Be Aware

Some applications, such as binding to a very large database on a remote network, can slow the **QueryData** event noticeably. The following properties can help improve response time.

The VirtualRows Property

The default number of rows read into the virtual page, as set by the **VirtualRows** property, is 0. Since the Spread control always requests at least as many rows as can be displayed in the control, having the **VirtualRows** property equal to 0 lets the control load only as many rows as is necessary for viewing. This behavior is intended for spreadsheets that need to be resized at run time to show more or fewer rows.

If the number of rows visible in your control does not vary, then you may want to set the **VirtualRows** property to the number of displayable rows. If you want your user to be able to scroll farther in the control without having to wait for more records to load into the virtual page, you can set the **VirtualRows** property to a larger number than the number of displayable rows.

**Tip**

Deducing the Number of Virtual Rows. In the absence of an adequate **VirtualRows** value (a value at least as great as the number of displayable rows), the Spread control deduces the number of rows to read into the virtual buffer from the *Rows-Loaded* parameter passed in the **QueryData** event. The **QueryData** event occurs whenever the Spread control requests more virtual data.

The VirtualOverlap Property

The default number of rows to hold in memory from the previous virtual page is 0. This behavior frees the Spread control from using any memory for this function unless it is specifically needed.

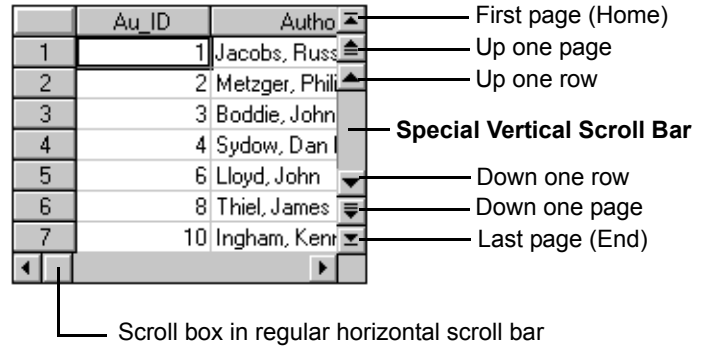
If you want your user to be able to scroll farther in the control without having to wait for more records to load into the virtual buffer, you can set the **VirtualOverlap** property to retain a number of rows from the previous virtual page.

Creating a Special Vertical Scroll Bar for Virtual Mode

You can tell the spreadsheet the total number of rows in your data by setting the **VirtualMaxRows** property. Doing so will allow the vertical scroll bar to accurately represent the user's current location in the data set. If you do not want to or cannot tell the spreadsheet the number of rows in your data, you can tell the vertical scroll bar not to reflect the number of rows of data by setting the **Virtu-**

alScrollBuffer property, or you can choose to create a special vertical scroll bar that might provide better navigation than the default vertical scroll bar.

Instead of a scroll box, the special vertical scroll bar (shown at right) displays scroll arrows for moving to the first or last page, up or down one page, and up or down one row. You can specify which scroll arrows display. Specify to display the special scroll bar by setting the **VScrollSpecial** property to True. Customize the arrows it displays by setting the **VScrollSpecialType** property.



Tip **Displaying No Arrows.** While it is possible to set the special vertical scroll bar to display no arrows (setting 7), it may not be too useful. Your user will see a blank scroll bar, with no buttons or means of scrolling. You may want to just choose to have no scroll bar instead (set the **ScrollBars** property to None or Horizontal).

Summary of Virtual Mode Properties

You can customize virtual mode to act the way you want in specific situations by using the guidelines shown in the following table.

This table of guidelines is for your future reference. You can see these properties in action in the examples in “When to Use Virtual Mode” on page 8.

| Ask yourself... | Use property... | With these settings... |
|---|--|---|
| What is the total number of rows in the recordset? | Virtual- MaxRows (Default: -1) | <p>I know the exact number of rows: Set this property equal to your table’s record count. Doing so lets you scroll in the spreadsheet without losing the record pointers. The vertical scroll bar will reflect a more accurate position of its place within the recordset, and the row headers will be able to accurately number the rows.</p> <p>I have an estimate of the number of rows: Overestimate rather than underestimate the number of rows. The control will not read the records past the value you assign.</p> <p>I don’t know the number of rows: Set this property to -1. Doing so means your spreadsheet does not have a reference to the end of the recordset. You can compensate by displaying the special vertical scroll bar described in “Creating a Special Vertical Scroll Bar for Virtual Mode” on page 5.</p> |
| How many rows do you want read into the virtual buffer at a time? | VirtualRows (Default: 0) | <p>Set this property to the number of rows you want read into the virtual page, the subset of records that Spread deals with at any one time. It is the amount of data that you can act on (search, for example) when using virtual mode.</p> <p>Note: If the spreadsheet can display more rows than the VirtualRows property indicates, the VirtualRows property value is ignored.</p> <p>Tip: For more information about optimizing your application, see “The VirtualRows Property” on page 5.</p> |
| How many retrieved rows do you want to keep in the buffer when more rows are read? | VirtualOverlap (Default: 0) | <p>Set this property to the maximum number of rows that you want Spread to keep in memory outside of the virtual page.</p> <p>Tip: For more information about optimizing your application, see “The VirtualOverlap Property” on page 5.</p> |
| Do you want the vertical scroll box to reflect the number of rows in the virtual buffer or the total number of rows in the recordset? | VirtualScroll- Buffer (Default: False) | <p>Reflect rows in virtual buffer: Set this property to True.</p> <p>Reflect rows in total recordset: Set this property to False. Specify the number of rows with the Virtual-MaxRows property.</p> <p>Tip: You can display a special vertical scroll bar instead, especially if you do not know the total size of the recordset. See “Creating a Special Vertical Scroll Bar for Virtual Mode” on page 5.</p> |
| Do you want to refresh the virtual window? | Action (run-time only) | <p>Set this property to 30 (SS_ACTION_VMODE_REFRESH).</p> <p>Tip: The QueryData event (with the <i>ActionRequested</i> parameter equal to 5) occurs when this action takes place.</p> |

Returning Information About Your Data in Virtual Mode

When using virtual mode, you can return information about your data with the properties listed in the following table.

| <i>To return...</i> | <i>Use property...</i> |
|---|------------------------|
| The number of rows in the virtual buffer | VirtualCurRowCount |
| The current top row of the virtual buffer | VirtualCurTop |



Tip

The Virtual Buffer Range. These properties provide the range of rows in the virtual buffer. The Spread control in virtual mode needs to read in more rows when the user scrolls up past the top row displayed (the value of the **VirtualCurTop** property). Similarly, the control needs more rows when the user scrolls down past the last row displayed (the sum of the values of the **VirtualCurTop** and **VirtualCurRowCount** properties).

Refer to the *Spread COM User's Guide* when you need more information about these properties.

When to Use Virtual Mode

A Spread control with virtual mode turned on often responds faster and conserves more system resources than a control with virtual mode turned off. The larger your recordset, or the more data you have in your data set, the more likely that using virtual mode will be beneficial.

The following sections describe using virtual mode with a bound control, with an unbound control, and when importing files.

Using Virtual Mode in a Bound Control

You can bind to a database using virtual mode to improve the performance of your spreadsheet. The larger the number of records in the database, the more valuable virtual mode is for improving responsiveness and conserving system resources.

How “large” should your database be to use virtual mode? Since there are so many possible factors, you should experiment to see if virtual mode improves your particular application. In general, if your application requires thousands of records and tens of columns, it may be a good candidate for using virtual mode. For example, a Spread control bound to a database of 15 columns and 12,000 records could probably benefit from using virtual mode. Even a database as small as 2,000 records can probably benefit.

Be Aware

Bound Virtual Mode Behavior. You can use virtual mode with bound spreadsheets, and it is often very helpful to do so when dealing with large recordsets. However, be aware of the behavior differences resulting from the control handling the subset of records in the virtual buffer rather than the complete set of records in the database. For example, if you have virtual mode turned on and you query the number of records in the database, the Spread control can only tell you the number of records in the virtual buffer. You can return the range of rows in the virtual buffer (see “Returning Information About Your Data in Virtual Mode” on page 8.) Trying to set or return properties for cells outside of this range while in virtual mode will fail.

**Tip**

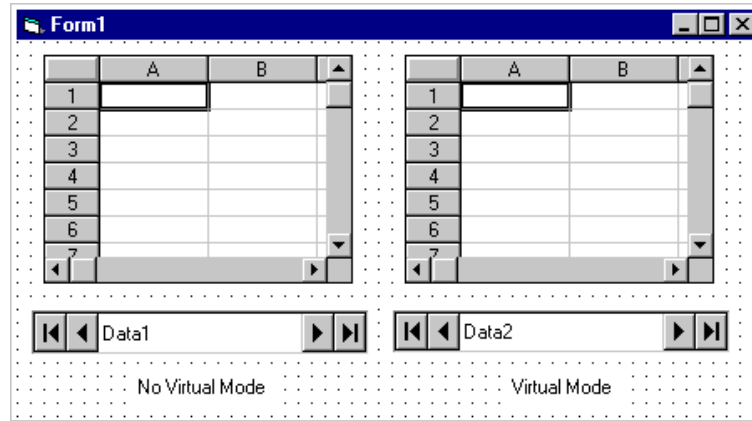
If you search for data with virtual mode turned on, the Spread control searches only the records that are in the virtual buffer, not all the records in the database. Since Spread does not support any search methods, you would also need to provide the code to loop through the cells and check each cell. An easier way to search data while in virtual mode is to bind the spreadsheet and use the **Find** method in the Recordset object.

Your Turn

The following steps bind two Spread controls to a large database (10,000 rows), one control with virtual mode turned off and the other with virtual mode turned on and customized.

1. First, you need a database to bind. For this example, we’re using *biblio.mdb*, a sample database that comes with Visual Basic. Make sure a copy of *biblio.mdb* resides on your hard disk. If it does not, install it from your Visual Basic installation CD.
2. Next, you need a program that will display the records in your database. Start a new Visual Basic project.
3. Now, you need to add the Data control to your project.
 - a. Place the standard Visual Basic Data control on the form.
 - b. Leave the Data control’s name as the default *Data1*.
 - c. In the Data control’s properties, specify the database and table you want to bind to:
 - i. Set the **DatabaseName** property to the path and name of your database (*biblio.mdb*).
 - ii. Set the **RecordSource** property to the name of a table in your database. For this example, use the *Authors* table.
4. You need to add the Spread control and link it to the Data control.
 - a. Add the Spread component to your project.
 - b. Place a Spread control on the form.
 - c. In the Spread control’s properties, specify the Data control to use. Set the **DataSource** property of the Spread control to *Data1*.
 - d. Size the Spread control and the Data control so you can see more cells.

5. Repeat Step 1 through 4 for another Spread control and another data control (vaSpread2 and Data2) on the same form. Label the controls as shown in the following figure:

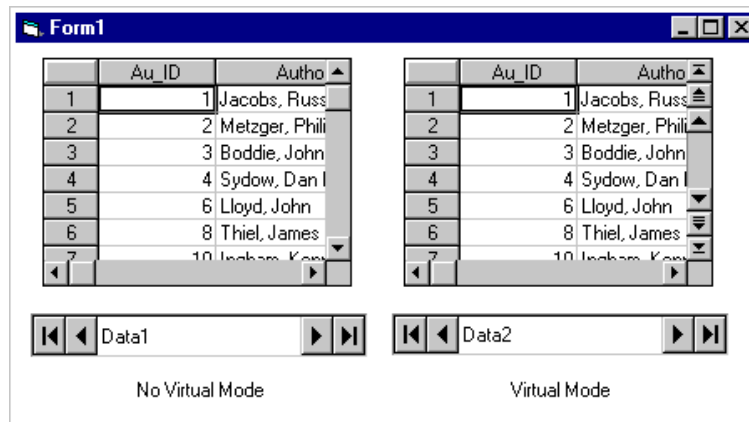


6. In the **Form_Load()** event, add the following code:

```
Private Sub Form_Load()
    ' Turn on virtual mode for Spread2
    vaSpread2.VirtualMode = True
    ' Indicate an unknown recordset size
    vaSpread2.VirtualMaxRows = -1

    ' Set the size of the virtual buffer
    vaSpread2.VirtualRows = 20
    ' Set the size of the overlap
    vaSpread2.VirtualOverlap = 10
    ' Set up the special vertical scroll bar to display all arrows
    vaSpread2.ScrollBars = ScrollBarsBoth
    vaSpread2.VScrollSpecial = True
    vaSpread2.VScrollSpecialType = 0
End Sub
```

7. Run the project. Your project should look similar to the following figure.



8. Move around in the data in both controls to compare the differences in behavior. Besides differences in response speed, you might notice that the numbering of the rows is different between the controls. The control that is in virtual mode does not know the exact number of rows, and therefore the row numbering is incorrect. You might want to hide the row header for spreadsheets that are in virtual mode, rather than having it display incorrect row numbers.

Do More



Customize Virtual Mode. You can customize virtual mode in many ways. Try adjusting some of the virtual mode settings in this example, or adding or deleting different virtual mode properties. See “Summary of Virtual Mode Properties” on page 7 for a listing of the virtual mode properties, their settings, and uses.

Using Virtual Mode in an Unbound Control

If you choose to use virtual mode to increase your unbound spreadsheet’s performance, you need to use the **QueryData** event to populate your cells.

When virtual mode is turned on, the **QueryData** event occurs when the Spread control requests additional data. For example, when the form first loads, the control requests a certain number of rows of data. You can populate these requested cells with your data. Then, as you scroll through the spreadsheet, the **QueryData** event occurs every time the form needs more data to replenish the virtual buffer. On each request, you can populate the new cells with your data.



The following steps create a virtual spreadsheet that handles a small set of 25 records. When you run the finished example, you can see in the Immediate window the sequence of events and their associated parameters that occur for all of the actions for which the **QueryData** event occurs:

- Populating the form during load
 - Scrolling down past the last row in the current virtual page
 - Scrolling up past the first row in the current virtual page
 - Pressing the Page Up, Page Down, Home, or End key in the special virtual scroll bar
 - Pressing a defined key (F5) to refresh the virtual page
1. Start a new Visual Basic project.
 2. Add a Spread control and size the control to see several rows and columns.
 3. In the **Form_Load()** event, add the following code to declare variables, define the data, and customize virtual mode:

```
Private Sub Form_Load()
    'Declare variables.
    Private mNames() As String
    'Define data to input.
    ReDim mNames(25)
    mNames(0) = "Ann-Marie"
    mNames(1) = "Bo"
    mNames(2) = "Bob"
    mNames(3) = "Bobby"
    mNames(4) = "Brian"
    mNames(5) = "Cheri"
    mNames(6) = "Chris"
    mNames(7) = "Don"
    mNames(8) = "Gina"
    mNames(9) = "Greg"
    mNames(10) = "Gregg"
    mNames(11) = "Jim"
    mNames(12) = "John"
    mNames(13) = "Julia"
    mNames(14) = "Kevin"
    mNames(15) = "Leanne"
    mNames(16) = "Lydia"
    mNames(17) = "Mark"
    mNames(18) = "Rick"
    mNames(19) = "Robby"
    mNames(20) = "Ronnie"
    mNames(21) = "Scott"
    mNames(22) = "Sean"
    mNames(23) = "Teri"
```

```

mNames(24) = "Tiwana"
'Display a single column.
vaSpread1.MaxCols = 1
'Turn on virtual mode.
vaSpread1.VirtualMode = True
'Define number of rows in recordset.
vaSpread1.VirtualMaxRows = 25
'Specify the virtual mode special scroll bar.
vaSpread1.VScrollSpecial = True
End Sub

```

4. In the **KeyDown** event, add the following code to your project to define the F5 key as the function to refresh the current virtual page:

```

' Define F5 as key to refresh the current virtual page.
Private Sub vaSpread1_KeyDown(KeyCode As Integer, Shift As
↳Integer)
    If KeyCode = vbKeyF5 Then vaSpread1.Action = ActionVModeRefresh
End Sub

```

5. Add the following code to your project to load the data using the **QueryData** event.

Because your spreadsheet is not bound to a database, you need to indicate the number of rows that are loaded in the **QueryData** event. The following code indicates how many rows you actually put into the Spread control. In most cases, it is the same number the control is asking for. After each data request, this code sets the value of the *RowsLoaded* parameter to the number of rows that were loaded. The *Row* parameter is the first row of the virtual page. The *RowsNeeded* parameter is how many rows need to be loaded into the virtual page. The *Direction* parameter indicates if you are scrolling up or down.

```

'Use the QueryData event to load and display data.
Private Sub vaSpread1_QueryData(ByVal Row As Long, ByVal
↳RowsNeeded As Long, RowsLoaded As Long, ByVal Direction As
↳Integer, AtTop As Boolean, AtBottom As Boolean)

    ' Set up Immediate window to display virtual mode row values.
    Dim sDebug As String
    Dim lRow As Long
    sDebug = "QueryData:" & vbCrLf & vbCrLf
    sDebug = sDebug & vbTab & "VirtualCurTop = " &
↳vaSpread1.VirtualCurTop & ", VirtualCurRowCount = " &
↳vaSpread1.VirtualCurRowCount & vbCrLf & vbCrLf
    sDebug = sDebug & vbTab & "Row = " & Row & ", RowsNeeded = "
↳& RowsNeeded & ", Direction = " & Direction

```

```
Select Case Direction
  Case 1
    sDebug = sDebug & " (Down)"
  Case 2
    sDebug = sDebug & " (Up)"
  Case 3
    sDebug = sDebug & " (Top)"
  Case 4
    sDebug = sDebug & " (Bottom)"
  Case 5
    sDebug = sDebug & " (Refresh)"
End Select

' Load data into virtual page
For lRow = Row To Row + RowsNeeded - 1
  vaSpread1.SetText 1, lRow, mName(lRow - 1)
Next lRow
RowsLoaded = RowsNeeded

' Display data about visible rows in debug window.
sDebug = sDebug & ", RowsLoaded = " & RowsLoaded
Debug.Print "-----"
Debug.Print sDebug
End Sub
```

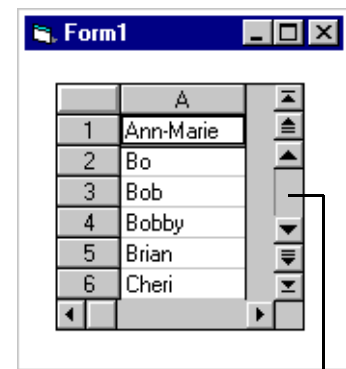
6. Add the following code to your project to display the values that pass when the **TopLeftChange** event occurs:
-

```
'Display previous and current top displayed row number.
Private Sub vaSpread1_TopLeftChange(ByVal OldLeft As Long,
  ↳ByVal OldTop As Long, ByVal NewLeft As Long, ByVal NewTop
  ↳As Long)
  Dim sDebug As String
  sDebug = "TopLeftChange:" & vbCrLf & vbCrLf
  sDebug = sDebug & vbTab & "VirtualCurTop = " &
  ↳vaSpread1.VirtualCurTop & ", VirtualCurRowCount = " &
  ↳vaSpread1.VirtualCurRowCount & vbCrLf & vbCrLf
  sDebug = sDebug & vbTab & "OldTop = " & OldTop & ",
  ↳NewTop = " & NewTop
  Debug.Print "-----"
  Debug.Print sDebug
End Sub
```

7. Add the following code to your project to display the values that pass when the **VirtualClearData** event occurs:

```
'Display the number of rows cleared.
Private Sub vaSpread1_VirtualClearData(ByVal Row As Long,
↳ByVal RowsBeingCleared As Long)
    Dim sDebug As String
    sDebug = "VirtualClearData:" & vbCrLf & vbCrLf
    sDebug = sDebug & vbTab & "VirtualCurTop = " &
↳vaSpread1.VirtualCurTop & ", VirtualCurRowCount = " &
↳vaSpread1.VirtualCurRowCount & vbCrLf & vbCrLf
    sDebug = sDebug & vbTab & "Row = " & Row &
↳", RowsBeingCleared = " & RowsBeingCleared
    Debug.Print "-----"
    Debug.Print sDebug
End Sub
```

8. Run the project. Your project should look similar to the figure at right. If necessary, expand the Immediate window to better see the messages that will display as events occur.
9. Move around in the control's data. Try the following actions using the special virtual scroll bar:
- Go up or down one row.
 - Go up or down one page.
 - Go to the first or last page.
 - Scroll up past the first row in the current page.
 - Scroll down past the last row in the current page.



Special Vertical Scroll Bar

Try the following actions using other keys:

- Press the Page Up or Page Down key.
- Press F5 to refresh the page.

Notice which messages display in the Immediate window for which navigation actions:

- When the control requires additional rows to display, the **VirtualClearData** event occurs to clear out the appropriate number of rows (the specified or calculated **VirtualRows** value minus the **VirtualOverlap** value). Then the **QueryData** event occurs to request the additional rows.
- When the top row changes, the **TopLeftChange** event occurs.

Each event displays the values of its parameters in the Immediate window. For example, moving up or down one row may display only the previous (*OldTop*) and current (*NewTop*) row numbers when the **TopLeftChange** event occurs.

Moving beyond the virtual page also displays the messages generated when the **VirtualClearData** and **QueryData** events occur.

**Tip**

Use All the QueryData Parameters. In step 5, the last two parameters in the **QueryData** event are useful when you have no indicator on your recordset that shows the current record, such as for Btrieve type of databases having the possibility of an outside user adding new records to the front of the recordset. In conjunction with the *Row* and *RowsNeeded* parameters, you can calculate where to have the record pointer start for leading the virtual page. Use the *AtTop* and *AtBottom* parameters to indicate whether you are at the end of your recordset.

Do More

Customize Virtual Mode. You can customize virtual mode in many ways. Try adjusting some of the virtual mode settings in this example, or adding or deleting different virtual mode properties. See “Summary of Virtual Mode Properties” on page 7 for a listing of the virtual mode properties, their settings, and uses.

Using Virtual Mode When Importing Files

In certain cases, you can use virtual mode to speed the operation of importing data from an existing file.

A Spread-formatted file or Excel-formatted file contains a stream of records, each record containing one or more properties. When a Spread control reads such a file, the following process occurs:

1. The Spread control resets its own virtual mode setting to the default state (False, or turned off).
2. The Spread control reads and processes the first record from the file. Any properties present in the first record are read and set.
3. The Spread control reads and processes the remaining records, setting any properties present in the record at the time the record is processed.

If the imported file was saved as a Spread virtual mode spreadsheet, then one of its records contains the virtual mode information (the **VirtualMode** property set to True or False). The Spread control turns virtual mode on when it processes a record with the **VirtualMode** property set to True. Since Excel does not provide a **VirtualMode** property, an Excel-formatted file cannot contain a virtual mode setting.

Be Aware

Importing Can Negate a Virtual Mode Setting. When importing an existing file, the Spread control automatically sets its own **VirtualMode** property to False, and then uses the **VirtualMode** property setting (if any) from the imported file. This process will override whatever value you set for the **VirtualMode** property in the Spread control.

Virtual Mode Behavior When Importing Non-Spread Files

Non-Spread files (such as an Excel-formatted or tab-delimited file) do not store a setting for the **VirtualMode** property. When the Spread control imports a non-Spread file, the control's **VirtualMode** property is automatically set to False. Be aware that this process will override whatever value you set for the **VirtualMode** property in the Spread control.



It is not possible to load a non-Spread file in virtual mode. Attempting to do so will turn virtual mode off in your application.

Virtual Mode Behavior When Importing Spread Files

Only Spread files store a setting for the **VirtualMode** property. When the Spread control imports a Spread file, the control's **VirtualMode** property is automatically set to the **VirtualMode** setting provided by the loaded file. If you import the Spread file through the buffer, the behavior is the same.

The **VirtualMode** setting transfers from the loaded file as a sheet-level property. (For an overview of the levels of properties, see "How does virtual mode handle property settings?" on page 4.) The other read-write virtual properties (such as **VirtualMaxRows** and **VirtualOverlap**) are also stored as sheet-level properties. The read-only virtual properties (**VirtualRowCount** and **VirtualCurTop**) are not stored in the file.

When your application loads a Spread file with virtual mode turned off, the loaded file transfers a copy of all sheet, column, row, and cell properties to your control in the record stream. With virtual mode turned on, the loaded file transfers the sheet and column properties to your control, but it only transfers the properties of the rows and cells that are in the virtual buffer. Thus, the control in virtual mode always holds the sheet and column settings in memory, but it holds the settings only for the buffered rows and cells in memory at any given time.



Create a Spread Template. In most cases, loading a virtual spreadsheet is probably not very useful. You can, however, use such a file as a template. For example, you can use Spread Designer to set sheet and column properties (including the **VirtualMode** property). Then use Spread Designer to save the properties to a file. In the **Form_Load** event, your application can load the file (thus restoring the sheet and column properties set with Spread Designer). Your application would be using the file as a template of sheet and column properties; your application can then set the row and cell properties in the **QueryData** event.

Acronyms Defined

This white paper uses the following acronyms.

| <i>Acronym</i> | <i>Definition</i> |
|-----------------------|---|
| ADO | ActiveX Data Objects |
| BIFF8 | Binary File Format 8 (Excel 97 and Excel 2000 format) |
| CD | Compact Disk |
| DLL | Dynamic Link Library |
| OCX | 32-bit ActiveX control extension |
| VBX | 16-bit Visual Basic control extension |

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.

© 2001-2005 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 COM is a trademark of FarPoint Technologies, Inc.

Other brand and product names are trademarks or registered trademarks of their respective holders.

Notes