Data-On-Demand ListBox

Created by Kem Tekinay, MacTechnologies Consulting.

Table of Contents

General Description

The Data-On-Demand ListBox is a subclass of the native REALbasic ListBox and, as far as the end-user is concerned, will behave in the same way. The key difference for the programmer is that the Data-On-Demand ListBox does not store any data. Rather, it is a "virtual" ListBox that relies on you, the programmer, to provide data, row-by-row, as requested. As a result, the Data-On-Demand ListBox is much faster at populating thousands of rows and, because you maintain the data, does not require you to coordinate changes from the data in your database to the ListBox.

About This Manual

This manual is written for the experienced REALbasic developer and assumes that you are familiar with the standard ListBox control.

You will find that I am repeatedly repetitive in a repetitious way. This is done intentionally because I find that nothing wastes more time than trying to find that description of widgets that I knew I read in a manual, only to eventually find it the section entitled "Houseplants." By repeating information in all the relevant sections, I hope to make finding it easier later on.

System Requirements

The Data-On-Demand ListBox requires REALbasic 5.5 or later and should work on any platform. It was developed and tested extensively on the Mac using OS X 10.3.7 and later, and tested for compatibility on Windows using XP Service Pack 1. The Linux version was unavailable for testing so feedback would be appreciated.

Memory Usage

The Data-On-Demand ListBox was designed with speed, not memory usage, in mind. If your RAM requirements are tight, you might want to look elsewhere.

The Data-On-Demand ListBox uses 13 bytes per row of data plus approximately 1 kilobyte. If you let the Data-On-Demand ListBox handle sorts, the RAM requirement during the sort will increase by the combined number of characters in your sort strings plus some overhead.

For example, if your data has 1,000 rows, the basic memory requirement will be 14 K (1,000 rows X 13 bytes each + 1 K). If the Data-On-Demand ListBox handles sorting, and each row uses a 10 byte sort string, the memory usage will increase by 10 K (1,000 rows X 10 bytes each) + overhead, bringing the total memory usage to 24K + overhead during the sort.

Note that these values reflect the arrays and variables used by the subclass. REALbasic may use additional RAM in its internal implementation of the ListBox so these figures should be considered estimates only.

The Concept

The standard REALbasic ListBox works by storing the values of each cell internally. It is up to you to coordinate the display with your database and this is usually done by storing hidden data in each row to identify the proper record in your database. Each row is accessed by its zero-based index.

The Data-On-Demand ListBox does not store any data; rather, it relies on you to provide the data for each row as requested. This request only comes when the data is refreshed, either as a result of a **Reset**

(see "Methods" below), a scroll, or **RefreshData** (see "Methods" below), and takes place in the **RequestRowData** event (see "Events" below).

In order to make this interaction as seamless as possible, the Data-On-Demand ListBox implements three ways of addressing each row:

VisibleRow: This is the index of a row that is actually visible to the end user at the moment. For example, your data may have 1,000 rows, but only 10 are visible at any given time due to the physical size of the ListBox. These VisibleRows are numbered 0 through 9. Setting information in a VisibleRow directly is discouraged; rather, data should be set in the **RequestRowData** event (see "Events" below). *ListCount* will return the number of currently visible rows, although its use is discouraged. Use *ListCountDOD* instead to get the number of VirtualRows currently being handled (see "Properties" below). As the Data-On-Demand ListBox is resized, the number of VisibleRows will change.

VirtualRow: This is the index of each row of data that the Data-On-Demand ListBox is currently handling and corresponds to the number of items in your database. For example, if your data has 1,000 rows, the first VirtualRow will have the index 0 and the last will have the index 999. The index of the every VirtualRow and DataRow (see below) will be the same until the Data-On-Demand ListBox sorts the data. The VirtualRow is only used directly by the *ScrollPosition* property (see "Properties" below). Setting data in a VirtualRow directly when the VirtualRow is not currently visible will not result in an error, but is ultimately meaningless. Retrieving information from a VirtualRow that is not visible will result in a runtime error.

DataRow: This is the index of each row that corresponds to an index in your database. For example, suppose your database has 1,000 records. If the Data-On-Demand ListBox sorts the data, the data from your first record may appear on the tenth row. This tenth row will have the VirtualRow index of 9, but a DataRow index of 0 (keeping in mind that all indexes are zero-based). Almost all interaction with the Data-On-Demand ListBox will take place using the DataRow so you do not have to keep track of where your data ends up after a sort. Setting data in a DataRow directly when the DataRow is not currently visible will not result in an error, but is ultimately meaningless. Retrieving information from a DataRow that is not visible will result in a runtime error.

The Data-On-Demand ListBox provides the tools to convert between the different types of indexes (see "Methods" below). Only *ScrollPosition* (and, by extension, the optional parameter in **Reset**) takes and returns the VirtualRow while all other interaction uses the DataRow. For example, the **RequestRowData** event asks for the information in a DataRow and you can use properties like *Cell*, *CellCheck*, *CellBold*, etc., to set the information in that DataRow without regard to which VirtualRow that DataRow actually appears.

Installation and Usage

This section provides a basic description of how to install and use the Data-On-Demand ListBox. For a more options and complete descriptions of the methods and events mentioned, see the sections entitled "Methods," "Events" and "Properties."

To install the Data-On-Demand ListBox, drag the file named "LB_DataOnDemand" into your project. Make sure that its Super class is to set to "ListBox".

Create a new ListBox in a window and set it's Super class to "LB_DataOnDemand". You should also assign a name. For the purposes of this section, we'll use "lbDOD."

Unregistered copies can be evaluated in the REALbasic IDE and during debugging, but will not allow you to compile a standalone application. If you have a registration code, supply it in the **RequestRegistrationInfo** event by setting the *regName* and *regNumber* variables.

If you want a vertical scrollbar, add that to the window and line it up with the ListBox. Because of the nature of the Data-On-Demand ListBox, you cannot use the ListBox's own vertical scrollbar. Attempting to do so will result in an error.

In the scrollbar's **ValueChanged** event, insert the code "lbDOD.ScrollPosition = me.value".

In the ListBox's **Open** event, set the variable *myScrollBar* to the scrollbar. The Data-On-Demand ListBox will maintain the scrollbar for you.

In the ListBox's **RequestRowData** events, insert code to fill in or clear the cells of a single row. You can use all the traditional methods and properties (*Cell*, *CellBold*, *CellCheck*, etc.) to set this information. **Note**: Because the Data-On-Demand ListBox reuses rows for speed, it is up to you to clear cells that are supposed to be blank. The Data-On-Demand ListBox will not do it for you.

If you would like the Data-On-Demand ListBox to handle sorting for you, place code in the **RequestSortData** event to return the appropriate sort string for each row. **Note**: The string comparisons are case-sensitive. If the sort should be case-insensitive, return all strings in upper or lowercase. Also, sorting will slow down dramatically when the sort string is the same for many rows. Try to differentiate where possible (see the section on "Sorting" below).

Use the **Reset** method to tell the Data-On-Demand ListBox how many records it will be displaying. You can optionally provide the *scrollToRow* value which will set the initial *ScrollPosition* after the **Reset**. The call to **Reset** can occur in the **Open** event, or anywhere else after the **Open** event fires.

If the number of records in your database changes, for example, because of a query by the user, call **Reset** again with the new number of rows to display.

If you make a change to the data in your database, but the number of records has not changed, call **RefreshData.**

If You Read Nothing Else…

The Data-On-Demand ListBox is designed to behave like the standard REALbasic ListBox and will even use the same properties and methods where possible. In this section, I will outline the basic differences between the two and the basic usage of the Data-On-Demand ListBox. I have tried to order these points in a sensible way, but you should read this whole section.

- The Data-On-Demand ListBox does not store any data; rather, it relies on you, the programmer, to maintain a database and supply the data to each row as requested. The request comes in the **RequestRowData** event. If required, you can use the **BeforeRefresh** event to set up data before the data is requested.

- It is up to you to clear blank cells. The Data-On-Demand ListBox reuses rows to enhance speed so it will not do this for you. Failure to clear blank cells will result in cells that appear to repeat data in an almost random fashion.

- Data is requesting in alternating directions. That is, the Data-On-Demand ListBox will request row data from lowest DataRow to highest DataRow on the first pass, from highest to lowest on the second pass, lowest to highest on the third pass, etc. You can override this behavior by setting the *ForwardRequestsOnly* property to "true," forcing the Data-On-Demand ListBox to only request data from lowest DataRow to highest DataRow.

- Using a scrollbar is optional, but you cannot use the ListBox's own scrollbar. Rather, create an independent scrollbar and tell the Data-On-Demand ListBox about it in the **Open** event by setting the *myScrollBar* variable. Using the ListBox's horizontal scrollbar is not prohibited.

- When editing a cell, the scrollbar is disabled and, in fact, all scrolling is disabled. The user must click out of the cell to be able to scroll again.

- The Data-On-Demand ListBox uses the row height for a number of internal calculations. Therefore, the *DefaultRowHeight* property must be set to a positive number. Resetting it to a negative number will generate an error, or cause unpredictable results.

- Do not allow the Data-On-Demand ListBox to get resize to smaller than what is needed to display one row. If you do, your program will crash. Set the window's *MinHeight* and *MinWidth* properties accordingly.

- The native properties of *ListCount*, *SelCount* and *ListIndex* will return values that pertain to the visible rows only, and are therefore meaningless in this environment. Use *ListCountDOD*, *SelCountDOD* and *ListIndexDOD* instead.

- The Data-On-Demand ListBox will handle sorting for you (see the section on "Sorting" below), but will slow down dramatically if much of your sort data is uniform. Try to differentiate where you can.

- If you don't want the Data-On-Demand ListBox to handle sorting for you, return "true" in the **SortColumn** event, just like in the standard ListBox.

- The *column* parameter in the **SortColumn** event might be -1, indicating "unsort." Be prepared for that.

- You can use the *InitialValue* property to set the header information, but don't try to include any additional row data. It will be wiped when the ListBox is reset.

Sorting

The Data-On-Demand ListBox will handle sorting for you when a header is clicked. By its nature, the Data-On-Demand ListBox does not store any information internally so it must ask you for data about each row before the sort begins. The request comes in the **RequestSortData** event where you should set the *result* variable with the string that should be used for comparison.

This method differs from the standard ListBox behavior in that there is no **CompareRows** event. The reasoning is that, while **CompareRows** may ask information about a single row repeatedly, **RequestRowData** will only ask about each row once. However, this does not mean that you cannot control the sort order. For example, if you are sorting on a column of checkboxes, you can set the *result* variable to "a" where the box is checked and "z" where the box is unchecked. This will ensure that the checked boxes will come to the top during an ascending sort of that column.

The Data-On-Demand ListBox uses a case-sensitive sort. To ignore case, convert the sort string to upper- or lowercase before placing it into the *result* variable.

The Data-On-Demand ListBox uses an optimized QuickSort algorithm for very fast sorts. However, like all QuickSort algorithms, it will slow down dramatically when the data is mostly uniform. (Note that blank cells do not count.) To ensure this does not happen, you should consider combining the data from multiple columns to provide unique strings. Even appending a number like the DataRow index will do the trick.

Since you maintain the database that the Data-On-Demand ListBox uses to display rows, the sort is only as fast as the access to your database. If your database has its own sort algorithm, it might be faster to use that than to rely on the Data-On-Demand ListBox. In that case, sort your data in the **SortColumn** event, then return "true" to keep the Data-On-Demand ListBox from sorting your data. Remember that the *column* parameter of the **SortColumn** event could be -1. This will happen if the shift

key is held down while clicking a heading and indicates “unsort.” Your code should be prepared for that.

Also, it might be faster to cache your data at the outset. If so, you can do that in **SortColumn** event too. If you need to perform clean up after the sort, use the **AfterSortColumn** event.

Details

The following tables review the individual properties, events and methods of the Data-On-Demand ListBox. Some are native and unchanged from the standard ListBox and are listed here as a convenience; these are listed in black. Some are overridden to take new or different parameters; these are listed in blue. Some are new to the Data-On-Demand ListBox and provide additional functionality; these are listed in green. And some are native to the ListBox but should not be used because they will result in an error, are unsupported, or are just meaningless within the Data-On-Demand ListBox; these are listed in red.

Where an item exists in the standard ListBox, you should refer to the REALbasic documentation or online help for a description and additional information.

Properties

The following are a list of properties of the Data-On-Demand ListBox. Note that almost all properties use the DataRow index, which is the index of a record within your database (see "The Concept" above). Setting the property of a cell that is not currently visible is meaningless, but will not result in an error. Getting the property of a cell that is not currently visible will result in a runtime error.

Some standard ListBox properties are listed in black and are not modified in any way. In those cases, you are directed to consult the REALbasic documentation and online help for a description.

As in the REALbasic documentation, properties that are in **bold** type are read-only.

Name	Type	Description
ActiveCell		Returns the EditField of the currently active cell. See the REALbasic documentation.
Bold	Boolean	Sets the style of the entire ListBox to bold. See the REALbasic documentation.
CellAlignment	Integer	Parameters are dataRow, column (integers). Used to set or get the alignment of a cell. Best used in the **RequestRowData** event. Otherwise, setting the *CellAlignment* of a cell that is not currently visible will not result in an error, but is ultimately meaningless. Getting the *CellAlignment* of a cell that is not currently visible will result in a runtime error. See the REALbasic documentation for a list of the acceptable values.
CellAlignmentOffset	Integer	Parameters are dataRow, column (integers). Used to set or get the alignment offset in pixels of a cell. Best used in the **RequestRowData** event. Otherwise, setting the *CellAlignmentOffset* of a cell that is not currently visible will not result in an error, but is ultimately meaningless. Getting the *CellAlignmentOffset* of a cell that is not currently visible will result in a runtime error.
CellBorderBottom CellBorderLeft CellBorderRight CellBorderTop	Integer	Parameters are dataRow, column (integers). Sets the border of a cell. See the REALbasic documentation for more details and a list of acceptable values.

Name	Type	Description
CellCheck	Boolean	Parameters are dataRow, column (integers). Sets the checked state of an individual cell. See the REALbasic documentation for a list of acceptable values.
CellHasFocus	Boolean	Returns whether the user is currently editing a cell.
CellTag	**Unsupported**	Because the Data-On-Demand ListBox does not store any data, it cannot store tags for individual rows. Any necessary data should be stored within your database or in an array that uses the DataRow as an index. Using this property will result in a runtime error.
CellType	Integer	Parameters are dataRow, column (integers). Gets or sets the type of a cell. See the REALbasic documentation for a list of acceptable values.
Column	ListColumn	Parameter is columnNumber (integer). Returns the specified visible column. Will not result in an error, but this is of limited value with this type of ListBox. Usage of this property is discouraged.
ColumnAlignment	Integer	Parameter is columnNumber (integer). See the REALbasic documentation.
ColumnAlignmentOffset	Integer	Parameter is columnNumber (integer). See the REALbasic documentation.
ColumnCount	Integer	Returns the number of columns the ListBox contains. See the REALbasic documentation.
ColumnSortDirection	Integer	Parameter is columnNumber (integer). Sets the sort direction for each column. See the REALbasic documentation.
ColumnType	Integer	Parameter is columnNumber (integer). Sets the type of each column. See the REALbasic documentation.
ColumnWidths	String	Gets or sets the widths of each column as a comma-separated list of values. See the REALbasic documentation.
DataField	String	Will not generate a runtime error, but is not useful with this type of ListBox.
DataSource	DataControl	Will not generate a runtime error, but is not useful with this type of ListBox.
DefaultRowHeight	Integer	Determines the height of every row. This must be set to a positive number. Using a negative number will result in an error, or will cause unpredictable results. The Data-On-Demand ListBox will set this to a positive number initially based on the font size.
EnableDrag	Boolean	Set to "true" to allow rows to be dragged. See the REALbasic documentation.
EnableDragReorder	Boolean	Will not generate a runtime error, but is not useful with this type of ListBox.
Expanded	Integer	Will not generate a runtime error, but is not useful with this type of ListBox. Hierarchical data has not been tested.

Data-On-Demand ListBox

Name	Type	Description
ForwardRequestsOnly	Boolean	When "false" (default), the Data-On-Demand ListBox will request DataRows from lowest to highest, then highest to lowest, then lowest to highest, etc., during the **RequestRowData** event. When "true," the Data-On-Demand ListBox will only request DataRows from lowest to highest during the **RequestRowData** event..
GridLinesHorizontal	Integer	See the REALbasic documentation.
GridLinesVertical	Integer	See the REALbasic documentation.
HasFocus	Boolean	Returns whether the Data-On-Demand ListBox currently has focus.
HasHeading	Boolean	Gets or sets whether the headers are visible. See the REALbasic documentation.
Heading	String array	Gets or sets the strings for the headings. See the REALbasic documentation.
HeadingIndex	Integer	Gets or sets the current sort column. Using the *ColumnSortDirection* property is preferred, although you may have to set this property too. See the REALbasic documentation.
Hierarchical	Boolean	Will not generate a runtime error, but is not useful with this type of ListBox. Hierarchical data has not been tested.
InitialValue	String	Sets the initial value of the ListBox. Should only be used to set the initial headers. All other data should be set in the **RequestRowData** event.
Italic	Boolean	Sets the style of the entire ListBox to italic. See the REALbasic documentation.
LastIndex	**Unsupported**	Because the Data-On-Demand ListBox requests data as it needs it, this property is meaningless in this environment. Using this property will result in a runtime error. Use the **RequestRowData** event to provide data and the **RefreshData** method to update the display using your database.
List	String	Parameter is dataRow (integer). Gets or sets the cell content for column 0 of the DataRow. Setting a cell that is not visible will not generate an error but is ultimately meaningless. Getting a cell that is not visible will result in a runtime error. This is the same as using **Cell(dataRow, 0)**.
ListCount	Integer	Returns the number of visible rows. Using this property will not result in an error, but is not useful here. Use *ListCountDOD* instead to get the number of VirtualRows. If you really must know the number of VisibleRows rather than VirtualRows, use the *VisibleRowCount* property instead.
ListCountDOD	Integer	Returns the number of rows that are currently being handled by the Data-On-Demand ListBox. Equivalent to using *ListCount* in a standard ListBox.

Name	Type	Description
ListIndex	Integer	Gets or sets the first selected visible row. Using this property will not result in an error, but is not useful here. Use *ListIndexDOD* instead.
ListIndexDOD	Integer	Gets or sets the first selected DataRow. Equivalent to using *ListIndex* in a standard ListBox except the result is the index in your database. If the Data-On-Demand ListBox has not sorted your data, the DataRow and VirtualRow will be identical. Otherwise, you can convert this index to the VirtualRow index using the **DataRowToVirtualRow** method.
ScrollBarHorizontal	Boolean	Set to "true" to add a horizontal scrollbar to the ListBox. See the REALbasic documentation.
ScrollBarVertical	Boolean	Must be set to "false." The Data-On-Demand ListBox cannot use the standard scrollbar. You should add a separate scrollbar and link it to the Data-On-Demand ListBox in the **Open** event.
ScrollPosition	Integer	Gets or sets the first visible VirtualRow. If the Data-On-Demand ListBox has not sorted your data, the VirtualRow and DataRow will be identical.
ScrollPositionDataRow	Integer	Gets or sets the first visible DataRow. If the Data-On-Demand ListBox has not sorted your data, the DataRow and VirtualRow will be identical.
ScrollPositionX	Integer	Gets or sets the horizontal scroll position. See the REALbasic documentation.
SelCount	Integer	Gets the number of visible selected rows. Using this property will not result in an error, but is not useful here. Use *SelCountDOD* instead.
SelCountDOD	Integer	Gets the number of currently selected rows. Equivalent to using the *SelCount* property in a standard ListBox.
Selected	Boolean	Parameter is dataRow (integer). Gets or sets the selected state of a given DataRow. If the dataRow is a negative number or outside the range of all DateRows, returns "false."
Selected	Integer array	Gets all the currently selected DataRows as an integer array, or sets the current selection to the DataRows in the integer array. Setting the selection will change the *ScrollPosition* to the first selected VirtualRow. When setting the selection, if one of the indexes in the array is invalid, it will be ignored. Also see the **SelectedRange** method.
SelectionType	Integer	Indicates the type of selection allowed. Use 0 for single selection, 1 for multiple selection. See the REALbasic documentation.

Data-On-Demand ListBox

Name	Type	Description
SortedColumn	Integer	Gets or sets the current sort column, but does not initiate a sort. See the REALbasic documentation.
Text	String	See the REALbasic documentation.
TextFont	String	See the REALbasic documentation.
TextSize	Integer	See the REALbasic documentation.
Underline	Boolean	Sets the style of the entire ListBox to underline. See the REALbasic documentation.
UseBlankCellBkgdPaint	Boolean	When "false" (default), the **CellBackgroundPaint** event will only fire for visible rows that contain data. When "true", the **CellBackgroundPaint** event will also fire for rows that contain no data. This is useful, for example, for creating alternate row shading throughout the ListBox. If a row has no data, the *dataRow* parameter of the **CellBackgroundPaint** event will be the blank row and its *isBlankRow* parameter will be "true." [Available only in REALbasic 2005 or later; no effect in previous versions.]
UseFocusRing	Boolean	See the REALbasic documentation.
Version	Double	Returns the version of the Data-On-Demand ListBox.
VisibleRowCount	Integer	Returns the number of rows that are currently visible. Use this instead of the *ListCount* property if you really must know the number of visible rows. Use the *ListCountDOD* property to get the number of virtual rows.

Events

The following are a list of events of the Data-On-Demand ListBox. Note that many events use the DataRow index, which is the index of a record within your database (see "The Concept" above). Setting the property of a cell that is not currently visible is meaningless, but will not result in an error. Getting the property of a cell that is not currently visible will result in a runtime error.

Some standard ListBox properties are listed in black and are not modified in any way. In those cases, you are directed to consult the REALbasic documentation and online help for a description.

As in the standard REALbasic events, some events will return a Boolean. In those cases, return "true" to override the default Data-On-Demand ListBox behavior.

Name	Parameters	Description
AfterSortColumn	column as Integer	Triggered after an internal sort has taken place. Only occurs if the **SortColumn** event returns "false" and should be used to perform any post-sort cleanup. If the **SortColumn** event returns "true" (meaning you've sorted your own data rather than letting Data-On-Demand ListBox do it), cleanup can be done in the **SortColumn** event itself.
BeforeRefresh		The data is about to refresh and the series of **RequestRowData** events are about to occur. Use this event to set up any data you need for the upcoming data refresh.
CellAction	dataRow as Integer, column as Integer	Triggered after an editable cell is edited whether the contents have changed or not. See the REALbasic documentation for more details.
CellBackgroundPaint	g as Graphics, dataRow as Integer, column as Integer, isBlankRow as Boolean	Returns Boolean. Allows you to handle the background drawing of a cell. For example you can use this event to create an alternating display pattern when the ListBox is displayed. You can convert the DataRow to VirtualRow with the **DataRowToVirtualRow** method to see where this DataRow falls within the ListBox. Return "true" if you don't want REALbasic to help paint the background. In REALbasic 2005 or later, you can set the *UseBlankBkgdPaint* property to tell the Data-On-Demand ListBox to fire this event even for rows that contain no data. In that case, the *dataRow* parameter will actually contain the blank row and the *isBlankRow* parameter will be "true." The *UseBlankCellBkgdPaint* property is initially "false" for the sake of compatibility. See the REALbasic documentation for more details.
CellClick	dataRow as Integer, column as Integer, x as Integer, y as Integer	Returns Boolean. Triggered when the end user clicks on a cell. Return "true" to override the default REALbasic behavior. See the REALbasic documentation for more details.

Data-On-Demand ListBox

Name	Parameters	Description
CellGotFocus	dataRow as Integer, column as Integer	Triggered when the user has clicked on an editable cell. You can use the *CellHasFocus* property to see if a cell currently has focus. See the REALbasic documentation for more details.
CellKeyDown	dataRow as Integer, column as Integer, key as String	Returns Boolean. Triggered after the user has pressed a key in an editable cell. Return "true" to keep REALbasic from processing the key. See the REALbasic documentation for more details.
CellLostFocus	dataRow as Integer, column as Integer	Triggered after the user has left an editable cell, either by clicking or pressing tab or return. You can use the *CellHasFocus* property to see if a cell currently has focus. See the REALbasic documentation for more details.
CellTextChange	dataRow as Integer, column as Integer	Triggered after the text of an editable cell has changed as long as the preceding **CellKeyDown** event has returned "false." See the REALbasic documentation for more details.
CellTextPaint	g as Graphics, x as Integer, y as Integer, dataRow as Integer, column as Integer	Returns Boolean. Allows you to handle the text painting of a cell. See the REALbasic documentation for more details.
Change		The selected item has changed. You can use the **Selected** methods and *SelCountDOD* and *ListIndexDOD* properties to get the currently selected DataRows. See the REALbasic documentation for more details.
CollapseRow	dataRow as Integer	The user has clicked on the disclosure triangle of an expanded DataRow. Note that hierarchical items have not been tested and are unsupported.
CompareRows	**Unsupported**	This event does not occur in the Data-On-Demand ListBox. Use the **RequestSortData** event instead.
DoubleClick		The user has double-clicked on an item.
DragReorderRows	**Unsupported**	
DragRow	drag as DragItem, dataRow as Integer	Returns Boolean. The user is dragging a DataRow. Return "true" to allow the drag to occur. See the REALbasic documentation for more details.
DropObject	obj as DragItem	Some object has been dropped onto the ListBox. See the REALbasic documentation for more details.
EnableMenuItemsToo		This is the same as the **EnableMenuItems** event. See the REALbasic documentation on the standard ListBox's **EnableMenuItems** event for more details.
ExpandRow	dataRow as Integer	The user has clicked on the disclosure triangle of a collapsed DataRow. Note that hierarchical items have not been tested and are unsupported.

Name	Parameters	Description
GotFocus		The Data-On-Demand ListBox has gotten the focus. You can also use the *HasFocus* property to determine whether the ListBox currently has focus. See the REALbasic documentation for more details.
HeaderPressed	column as Integer	Returns Boolean. Triggered after the user has clicked on a header. Return "true" to keep the Data-On-Demand ListBox from handling the click. See the REALbasic documentation for more details.
KeyDown	key as String	Returns Boolean. Triggered after the user has pressed a key while the Data-On-Demand ListBox has focus. Return "true" to keep REALbasic from handling the key. See the REALbasic documentation for more details.
LostFocus		The Data-On-Demand ListBox has lost the focus. You can also use the *HasFocus* property to determine whether the ListBox currently has focus. See the REALbasic documentation for more details.
MouseDown	x as Integer, y as Integer	Returns Boolean. Triggered when the user has clicked within the borders of the Data-On-Demand ListBox. Return "true" to allow the **MouseUp** and **MouseDrag** events to fire and to stop the standard processing of the click. You can use the **CoordinatesToDataRow**, **CoordinatesToVirtualRow**, **CoordinatesoVisibleRow**, **CoordinatesToColumn** and **CoordinatesToHeader** methods to determine where the coordinates fall within the ListBox. See the REALbasic documentation for more details.
MouseDrag	x as Integer, y as Integer	Triggered when the user initiates the drag of a row or rows. This event will not fire unless "true" was returned by the preceding **MouseDown** event. You can use the **CoordinatesToDataRow**, **CoordinatesToVirtualRow**, **CoordinatesoVisibleRow**, **CoordinatesToColumn** and **CoordinatesToHeader** methods to determine where the coordinates fall within the ListBox. See the REALbasic documentation for more details.
MouseEnter		The mouse has entered the borders of the Data-On-Demand ListBox. See the REALbasic documentation for more details.
MouseExit		The mouse has left the borders of the Data-On-Demand ListBox. See the REALbasic documentation for more details.

Data-On-Demand ListBox

Name	Parameters	Description
MouseMove	x as Integer, y as Integer	The mouse has moved within the borders of the Data-On-Demand ListBox. You can use the **CoordinatesToDataRow**, **CoordinatesToVirtualRow**, **CoordinatesoVisibleRow**, **CoordinatesToColumn** and **CoordinatesToHeader** methods to determine where the coordinates fall within the ListBox. See the REALbasic documentation for more details.
MouseUp	x as Integer, y as Integer	The mouse button was released within the borders of the Data-On-Demand ListBox. This event will not fire unless "true" was returned by the **MouseDown** event. You can use the **CoordinatesToDataRow**, **CoordinatesToVirtualRow**, **CoordinatesoVisibleRow**, **CoordinatesToColumn** and **CoordinatesToHeader** methods to determine where the coordinates fall within the ListBox. See the REALbasic documentation for more details.
Open	ByRef myScrollBar as Scrollbar	Triggered when the window that contains Data-On-Demand ListBox is opened. Set the *myScrollBar* parameter to allow the Data-On-Demand ListBox to maintain the vertical scrollbar for you. The ListBox will initially be initialized to display zero rows.
RequestRegistrationInfo	ByRef regName as String, ByRef regNumber as String	Triggered when the window that contains the Data-On-Demand ListBox is opened. Set the *regName* and *regNumber* properties with your registration information. Only registered copies of the Data-On-Demand ListBox can be included in compiled applications.
RequestRowData	dataRow as Integer	Triggered when the display is being refreshed. Data can be prepared beforehand in the **BeforeRefresh** event. Data will be requested alternately in ascending and descending order unless the *ForwardRequestsOnly* property is set to "true," in which case it will only be requested in ascending order.
RequestSortData	dataRow as Integer, column as Integer, ByRef result as String	Triggered during a sort by the Data-On-Demand ListBox. Sort data is requested once per row in ascending order. This event will only fire if "false" was returned by the **SortColumn** event. After a sort, the **AfterSortColumn** event will fire.

Name	Parameters	Description
SortColumn	column as Integer	Returns Boolean. Triggered when a sort is about to occur. Return "true" here to keep the Data-On-Demand ListBox from sorting the column. That will also prevent the **AfterSortColumn** event from firing. If the Data-On-Demand ListBox does not sort your data, the DataRow and VirtualRow indexes will always be the same. Note that the *column* could be -1 for an unsort. Your code should be prepared to deal with that. See the REALbasic documentation for more details.

Methods

The following are a list of methods of the Data-On-Demand ListBox. Note that many methods use the DataRow index, which is the index of a record within your database (see "The Concept" above). Setting the property of a cell that is not currently visible is meaningless, but will not result in an error. Getting the property of a cell that is not currently visible will result in a runtime error.

Some standard ListBox methods are listed in black and are not modified in any way. In those cases, you are directed to consult the REALbasic documentation and online help for a description. Others are modified to use the DataRow index. You should consult the REALbasic documentation in those cases too.

Name	Parameters	Description
About		Shows the Data-On-Demand ListBox About box.
AddFolder	**Unsupported**	
AddRow	**Unsupported**	Use the **RequestRowData** event instead.
Cell	dataRow as Integer, column as Integer	Gets or sets the text of a cell associated with a DataRow of a column. Best if used within the **RequestRowData** event. Setting the text of a cell that is not visible will not generate an error, but is ultimately meaningless. Getting the text of a cell that is not visible will result in an error.
CellBold	dataRow as Integer, column as Integer	Gets or sets the bold style of a cell associated with a DataRow of a column. Best if used within the **RequestRowData** event. Setting the bold style of a cell that is not visible will not generate an error, but is ultimately meaningless. Getting the bold style of a cell that is not visible will result in an error. Assign "true" to bold the cell, "false" to make it not bold.
CellItalic	dataRow as Integer, column as Integer	Gets or sets the italic style of a cell associated with a DataRow of a column. Best if used within the **RequestRowData** event. Setting the italic style of a cell that is not visible will not generate an error, but is ultimately meaningless. Getting the italic style of a cell that is not visible will result in an error. Assign "true" to italicize the cell, "false" to make it not italic.
CellUnderline	dataRow as Integer, column as Integer	Gets or sets the underline style of a cell associated with a DataRow of a column. Best if used within the **RequestRowData** event. Setting the underline style of a cell that is not visible will not generate an error, but is ultimately meaningless. Getting the underline style of a cell that is not visible will result in an error. Assign "true" to underline the cell, "false" to make it not underlined.
ColumnValueProvider	column as Integer	See the REALbasic documentation for more details.

Name	Parameters	Description
CoordinatesToColumn	x as Integer, y as Integer	Converts the coordinates given by x and y to the corresponding column. If the coordinates do not fall within the list, it returns -1. If the coordinates are within the list but fall within the white space to the right of the last column, it returns the index of the last column + 1, i.e., the *ColumnCount*. It is up to you to make sure the index is valid before using it. Also see the **CoordinatesToHeader** method.
CoordinatesToDataRow	x as Integer, y as Integer	Converts the coordinates given by x and y to the corresponding DataRow. If the coordinates do not fall within the list, it returns -1. If the Data-On-Demand ListBox has not sorted your data, the DataRow and VirtualRow will be identical. Otherwise, you can use the **DataRowToVirtualRow** method to find the VirtualRow, or simply use the **CoordinatesToVirtualRow** method in the first place.
CoordinatesToHeader	x as Integer, y as Integer	Converts the coordinates given by x and y to the corresponding header. If the coordinates do not fall within the headers, or the headers are not shown, it returns -1. If the coordinates are within the headers but fall in the white space to the right of the last column, it returns the index of the last column + 1, i.e., the *ColumnCount*. It is up to you to make sure the index is valid before using it. Also see the **CoordinatesToColumn** method.
CoordinatesToVirtualRow	x as Integer, y as Integer	Converts the coordinates given by x and y to the corresponding VirtualRow. If the coordinates do not fall within the list, it returns -1. If the Data-On-Demand ListBox has not sorted your data, the VirtualRow and DataRow will be identical. Otherwise, you can use the **VirtualRowToDataRow** method to find the DataRow, or simply use the **CoordinatesToDataRow** method in the first place..
CoordinatesToVisibleRow	x as Integer, y as Integer	Converts the coordinates given by x and y to the corresponding VisibleRow. If the coordinates do not fall within the list, it returns -1.
DataRowToVirtualRow	dataRow as Integer	Converts the given DataRow to the corresponding VirtualRow. If the Data-On-Demand ListBox has not sorted your data, these numbers will identical.
DataRowToVisibleRow	dataRow as Integer	Converts the given DataRow to the corresponding VisibleRow. If the DataRow is not currently visible, it returns -1.
DeleteAllRows	**Unsupported**	Use the **Reset** method instead.

Data-On-Demand ListBox

Name	Parameters	Description
EditCell	dataRow as Integer, column as Integer	Scrolls the cell associated with the DataRow and column into view, if necessary, and temporarily makes the cell editable. Sets the focus to that cell.
InsertFolder	**Unsupported**	
InsertRow	**Unsupported**	Use the **Reset** method instead.
InvalidateCell	dataRow as Integer, column as Integer	Redraws the cell associated with a DataRow and column from scratch. See the REALbasic documentation for more details. While this method will work much like the native implementation, you should consider using the **RefreshData** method instead.
PressHeader	column as Integer	Presses the header of the specified column. See the REALbasic documentation for details.
RefreshData	[recalcNumOfVisibleRows as Boolean = false]	Causes all of the visible rows to be refreshed. Will trigger the **BeforeRefresh** and **RequestRowData** events. The optional *recalcNumOfVisibleRows* parameter will force a recalculation the number of visible rows before refreshing them. This is usually not needed except in cases where resizing the Data-On-Demand ListBox fails to update the display properly.
RemoveRow	**Unsupported**	Use the **Reset** method instead.
Reset	rows as Integer[, scrollToRow as Integer = 0]	Sets up the Data-On-Demand ListBox to display a certain number of rows. Use this method every time the number of rows to display changes. The optional *scrollToRow* parameter allows you to set the initial scroll position of the ListBox. Resetting the Data-On-Demand ListBox will update the number of visible and virtual rows and cause all the data to be refreshed by triggering the **BeforeRefresh** and **RequestRowData** events. There is no need to call the **RefreshData** method immediately after using **Reset**. If a header had been pressed before the reset, the **SortColumn** event will be triggered during the reset. To prevent this, set the *SortedColumn* property to -1 before using **Reset**.
RowPicture	dataRow as Integer	Gets or sets the picture associated with a DataRow. Best used from within the **RequestRowData** event.
SelectAll		Selects every VirtualRow. Put another way, it sets the *Selected* property of every row to "true." To deselect every row, set the *ListIndexDOD* property to -1.

Name	Parameters	Description
SelectedRange	startDataRow as Integer, endDataRow as Integer	Selects or deselects all the VirtualRows between startDataRow and endDataRow inclusive. For example, "**SelectedRange**(1, 10) = true" would select all the VirtualRows starting at DataRow 1 and ending with DataRow 10. If the Data-On-Demand ListBox has sorted your data, those rows may be more or less than 10 rows apart. To force a fixed number of VirtualRows to be affected, use syntax like this: "**SelectedRange**(**VirtualRowToDataRow**(1), **VirtualRowToDataRow**(10)) = true." This would cause all VirtualRows between VirtualRow 1 and VirtualRow 10 to be selected. Note that if rows outside of the given range were selected before using this method, they will still be selected after using this method. You can set the *ListIndexDOD* property to -1 to deselect every row.
VirtualRowToDataRow	virtualRow as Integer	Converts the given VirtualRow to the corresponding DataRow. If the Data-On-Demand ListBox has not sorted your data, these numbers will identical.
VirtualRowToVisibleRow	virtualRow as Integer	Converts the given VirtualRow to the corresponding VisibleRow. If the VirtualRow is not visible, it returns -1.
VisibleRowToDataRow	visibleRow as Integer	Converts the given VisibleRow to the corresponding DataRow. If the VisibleRow is outside the range of currently visible rows, it returns -1.
VisibleRowToVirtualRow	visibleRow as Integer	Converts the given VisibleRow to the corresponding VirtualRow. If the VisibleRow is outside the range of currently visible rows, it returns -1.

Version History

1.23 Added code to force an update of the number of visible rows during a resize. This corrects a problem where pressing the zoom button would resize the ListBox, but not the number of visible rows.

Added code to check whether the user is clicking on a horizontal scrollbar rather than just white space beneath the listed items. Clicking white space deselects all items which is a bad thing if you are just trying to scroll.

Changed behavior so that a click near the last visible row is the same as a click on the last visible row.

Changed behavior so that a click in the white space beneath the last visible row is handled by REALbasic rather than internally.

Added the *recalcNumOfVisibleRows* parameter to the **RefreshData** method.

1.22 Fixed order that **Change** event was called while selecting a single row. This bug would manifest while using the up and down arrow keys.

1.21 Removed **Super.DeleteAllRows** from the **Reset** method to prevent flicker during updates from a thread.

1.2 Added the *UseBlankCellBkgdPaint* property to fire the **CellBackgroundPaint** event even for blank rows. [REALbasic 2005 or later; no effect in previous versions.]

Added *isBlankRow* parameter to **CellBackgroundPaint** event.

Added *ForwardRequestsOnly* and *UseBlankCellBkgdPaint* properties to Data-On-Demand ListBox Property list in IDE.

Changed behavior of *Selected* property so that invalid indexes will not generate an error. For example, "*Selected*(-1) = true" will have no effect and "*Selected*(-1)" will return "false."

When using the integer array variation of the *Selected* property to set the selection, including an invalid index will have no effect and will not generate an error.

Added *CellHasFocus* property.

Changed all examples to reflect changes to **CellBackgroundPaint** event.

1.11 Made double-click detection more reliable.

Shift-clicking a column heading will unsort the ListBox if Data-On-Demand ListBox is handling the sorting.

Corrected a bug in examples that could occasionally lead to a blank, non-responsive ListBox.

Correct some typos in manual.

1.1: Added **CoordinatesToColumn** method.

Changed behavior of **CoordintatesToHeader** method. If the click is in the white space to the right of the last column, now returns the *ColumnCount*.

Fixed bug that required *EnableDrag* to be "true" to enable proper events.

When cell is being editing, now prevents scrolling and disables the scrollbar.

Implemented drag selections when set to multiple selections and *EnableDrag* is "false." Also fixed other drag and selection behavior when *EnableDrag* is "false."

Made checkbox click behavior even closer to that of standard ListBox. Now, a click in the checkbox does not change the selection, but a click in the label next to the checkbox will.

Added SQLite example and made fixes and changes to other examples.

1.0: First release.

Contact

The Data-On-Demand ListBox was created by Kem Tekinay of MacTechnologies Consulting. The latest version is available at *http://www.mactechnologies.com*. All technical support is handled via e-mail at *dod@mactechnologies.com*. Anyone who misspells "Kem" will hear about it.

Legal Stuff