MultiBase Cosmos

Notes to version 5.0

BASE100

BASE 100, S.A.
www.base100.com

Index

1. Implementations

1.1 Runtime

In this version have been implemented methods and events in the SimpleControl class that allows to customize a List Box control of type string, tree and sql, so you can create your own styles for cells and columns, customize control components using a file to indicate where are the graphic elements that you want to use, and add features such as the ability to search for strings in all the cells in a list or in a column show the result of an arithmetic and logic. Likewise, you can also add filters on header columns, group columns under the same title and create groups and aggregates.

In addition, the user can select the columns that will form groups and aggregate's functions, select the columns that want to sort and modify their order once elected. The appearance of the list after creating groups is a tree list on which the user can open and close the nodes.

2. Improvements

2.1 Runtime

- Modified the runtime to support version 3.4 of OpenOffice and allow export to PDF, HTML and ODS.
- Ability to show / hide the horizontal lines separating rows in tree list controls.
- Ability to dynamically assign the Font, Foreground and Background to a control of the print pages. The method SetProperty of the SimpleControl class should be used to change these properties. This value may be consulted with the GetProperty method of the same class.
- It has been expanded the maximum size of the text that the method MsgText returns.
- There have been modifications needed to run applications developed in Cosmos on Windows 8.
- The runtime has been modified to allow to export the masks used in the reports design when exporting to Excel from the Cosmos Preview window.

3. Environment Variables

COSMOSLISTSKIN

This environment variable tells the runtime the path of the file that defines the graphic elements to customize the list type controls in the application. If this variable is set all list-type controls are shown with the appearance.

DRAWTREELISTNODELINES

Instructs the runtime if drawn or not the lines connecting the nodes of the tree lists.

The possible values are :

TRUE or YES	Lines are drawn. This is the default value.
FALSE or NO	No lines are drawn.

If there is not selected a Windows XP theme or the value of the environment variable COSMOSXPTHE MESTYLE is FALSE, no lines are drawn, regardless of the value that is assigned to variable.

This environment variable is defined in the Environment section of the project configuration file or in the file "cosmos.ini". You may not modify the values of the variable at runtime.

DRAWTREELISTLINES

Indicates whether or not to draw the lines between the rows in the tree lists.

The possible values are:

TRUE or YES	Lines are drawn.
FALSE or NO	No lines are drawn. This is the default value.

This environment variable is defined in the Environment section of the project configuration file or in the file "cosmos.ini". You may not modify variable values at runtime.

ENABLEMENUOPTIONONENABLECOMMAND

Environment variable that indicates whether to enable or not the menu item associated with a command when it is enabled. Must be defined in the project configuration file or the configuration Cosmos file.

The possible values are:

YES or TRUE	When a command is enabled, also enables the menu option associated therewith.
NO or FALSE	When enabled not enable a command menu option associated. It will be enabled manually. This is the default.

ZOOMFORM

Indicates the zoom ratio to be applied on the screens, controls and application fonts.

SHOWCOSMOSTREEWALKDIALOG

This environment variable tells the runtime of Cosmos, in implementing the method TreeWalk, instead of displaying the standard dialog file selection in the Windows operating system version that is running the application (and implemented by the API of that system), display a dialog box with similar functionality and look identical on all versions of Windows.

The possible values are:

YES	The new dialog is launched.
NO	The method will work as in previous versions.

This environment variable is defined in the Environment section of the project configuration file or in the file "cosmos.ini". The values of this environment variable can't be modified in runtime.

SHOWCOSMOSTREEWALKNETWORKDRIVES

Indicates whether or not to display the shared folders on the network when the method call TreeWalk uses the new dialog box instead of the function of the Windows API.

The possible values are: YES and NO, the latter being the default.

This variable must be defined in Section Environment Cosmos.ini file INI file or project.

SHOWCOSMOSTREEWALKLOCALDRIVES

This environment variable indicates whether or not to show local drives when the method TreeWalk use the dialog implemented in version 5.0 of Cosmos instead of the Windows API for choosing a file.

The possible values are: YES and NO. The default is YES.

This variable must be defined in Section Environment Cosmos.ini file INI file or project.

4. Methods

4.1 Module Class Methods

FtpPutFileEx

This mehod allow to send a file to a FTP server. Unlike thel FtpPutFile method, this method don't use the Windows API methods.

Syntax:

```
FtpPutFileEx (remoteHost as Char ,remoteHostPort as Integer, remote-
HostUser as Char,remoteHostPassword as Char, remote-HostDirectory as
Char, remoteHostFile as Char,localFile as Char, showDialog as Boolean,
allowCancelUpload as Boolean) return Integer
```

Parameters:

remoteHost	Remote IP or host name .
remoteHostPort	Remote host port number. The default port is 21.
remoteHostUser	User.
remoteHostPassword	Password.
remoteHostDirectory	Remote host full path to send the file.
RemoteHostFile	Remote host file name.
LocalFile	Local full path of the file to send to the remote server.
ShowDialog	Indicates whether to display the progress window.
AllowCancelUpload	Indicates whether the user can cancel the upload operation.

Returns one of the next codes:

0	If the file is send successfully.
-1	The source file don't exists or the user has no access permission.
-2	Cannot connect to the network.
-3	Remote server not found or remote port not opened.
-4	Cannot be access the remote directory.
-5	Error sending the file.
-6	Process aborted by the user.

-7	Timeout.
-8	Incorrect user/password.
-9	Parameters host, remotehostfile or localfile are null.

GetUrlFileEx

This method allows to download a file from the URL in the first parameter. Unlike the method GetUrlFile, this method don't use the Windows API functions.

Syntax:

```
GetUrlFileEx(remoteFile as Char ,localFile as Char ,showDialog as Boo-
lean ,allowCancelDownload as Boolean) return Boolean
```

Parameters:

remoteFile	Remote file URL.
LocalFile	Full path, including the file name, of the local file where the remote file will be downloaded. The local directory must exists before the download.
ShowDialog	Indicates whether to display the progress window.
AllowCancelDownload	Indicates whether the user can cancel the download operation.

Returns:

TRUE if the download process was successfully. FALSE otherwise.

ShowColorDialog

This methodw shows the standard Windows dialog to select a color.

Syntax:

```
ShowColorDialog(parentWindow as Integer ,defaultColor as Integer ,
VAR retColor as Integer) return boolean
```

Parameters:

parentWindows	Parent window HANDLE. A null value can be accepted.
defaultColor	Indicates the default color that will be selected in the dialog box.
retColor	Selected color.

Returns:

TRUE	Whether the OK button is pressed.
FALSE	Whether the Cancel button is pressed.

ShowFontDialog

Displays the standard Windows dialog for font selection.

Syntax:

```
ShowFontDialog (parentWindow as Integer ,font as Char default "")
return char
```

Parameters:

parentWindow	Handle of the parent window. Accepts null.
font	Font selected by default.

Returns:

A string with the selected source.

An example of the return value is:

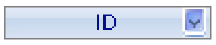
```
Courier New;italic;Size=24
```

4.2 SimpleControl Class Methods

In this section, provided there is a reference to a control list or a list, applies only to the List Box control columns list type (string, sql and / or tree). If a method can be applied to other types of lists will be declared ready.

AllowColumnHeaderFilter

This method tells the runtime of Cosmos that in the header of the specified column displays an arrow that, when pressed, will display a list to select the value for which you want to filter out. When applying this method, in the column header arrow appears, as shown in the following image:



Clicking on the date will be enabled a drop edit control. In the drop-down list shows the filters listed below:

- No vacías. Show all rows with non-zero values.
- Vacías. Show all rows with null values for columns.
- Todas. Show all rows.
- List all the unique values for that column.

In the edit field may indicate the following filters:

- For numeric columns can apply the following operators: “<”, “>”, “>=” and “!=”.

- For alphanumeric columns may indicate the metacharacter '%' to filter with clause like.

Syntax:

```
AllowColumnHeaderFilter(column as Smallint ,show as Boolean
,maxHeaderRows as Integer default -1 caseSensitive as Boolean default
FALSE).
```

Parameters:

Column	Column on which the actions apply.
Show	Indicates whether to allow or not to apply a filter on the column that is passed as a parameter.
maxHeaderRows	Indicates the maximum number of unique values that contain filtering drop edit control. By default, it displays all rows.
caseSensitive	Select whether or not the search is case sensitive.

***IMPORTANT:** If the value shown in parameter casesensitive is FALSE and list type is SQL, client-server installations engine version must be at least 3.4 0.2 in the Windows version. For UNIX / Linux releases of engine versions will be equal to or greater than the 3.2 0.2, 3.4 0.2 y 3.6 0.2.*

AllowExportListCellStyles
This method allows to specify whether the export of lists of type Sql, String and Tree with ExportToExcel methods, ExportToHTML, ExportToPDF and ExportToODS will include include or not the cells and columns styles created with the method CreateListCellStyle. By default, the export does include the styles created with CreateListCellStyle.

Syntax:

```
AllowExportListCellStyles(allow as Boolean)
```

Parameters:

allow	Indicates if the styles are exported or not.

AlternateBackColor
Apply color bands to the rows in a list alternating two colors.

Syntax:

```
AlternateBackColor(BackColor1 as Integer ,BackColor2 as Integer
,nAlternate as Integer)
```

Parameters:

BackColor1	RGB value of the color to be painted the first band.

BackColor2	RGB value of the color to be painted the second band.
nAlternate	Frequency that the second color will be shown in the list.

BackupListRow
This method makes a copy of the data of the row that is given as parameter. This value is stored in memory and can be accessed with the GetBackupListRow method and restored with the RestoreBackupListRow method. This method is applicable to a String type and Tree type List Box.

Syntax:

```
BackupListRow(row as Integer)
```

Parameters:

row	Row identifier.

ComputeListColumnTotals
This method calculates the totals on the columns that are indicated by the method TotalizeColumn.

Syntax:

```
ComputeListColumnTotals()
```

CreateListCellStyle
Create a display style subsequently applied to individual cells or columns in the list.

Syntax:

```
CreateListCellStyle(font as Char ,foreColor as Integer ,backColor as
Integer ,align as Smallint, icon as Smallint) return integer
```

Parameters:

Font	Indicates the formatting attributes for text. The attributes that can be set to define a source are: name, underscore, striped, bold and size. Attributes must be separated by semicolons.
ForeColor	Text Color. If the value for this parameter is -1, when applying the style to the cell or column will not change the color of text.
BackColor	Background color. If the value for this parameter is -1, when applying the style to the cell or column will not change the background color.
Align	Indicates the type of alignment that will give the text of the cell. Its possible values are: 0. Left-aligned text. 1. Centered text. 2. Right-aligned text.

Icon	Optional parameter indicating the icon that will displayed in the cells that has assigned the style.

Returns:

An identifier of the created to subsequently apply to individual cells or columns in the list.

FindColIntoString

This method returns a string with the row identifiers (separated by the character "|"), which coincides with the literal value specified in parameter "text". The search begins at the index item "indexStart '. This makes it possible to obtain all occurrences of text in a column of the list in a single function call.

Syntax:

```
FindColIntoString(column as Smallint ,text as Char ,indexStart as In-
teger default 0 ,exact as Boolean default TRUE) return Char
```

Parameters:

column	ID of the column in which you want to search.
text	Literal to look for.
indexStart	Index of the item in the list from which the search starts. The default value is zero, indicating that the search is performed from the first element.
exact	If TRUE finds rows whose value exactly matches the specified character set. If FALSE looking rows whose value starts with the specified string.

Returns:

A string with the identifiers of all rows that meet the condition. Failure has found returns null no value.

GetBakupListRow

This method allows to query data from the row stored in memory with the BackupListRow method.

Syntax:

```
GetBakupListRow(row as integer) return Char
```

Parameters:

row	Row identifier

Returns:

A string with the row value stored in memory.

GetListCellSyles

This method provides information about the number of styles, the ID of the styles and conditions used to apply styles to a cell.

Syntax:

```
GetListCellStyles(row as Integer ,col as Smallint ,VAR idxArray as
Array OF Numeric ,VAR conditionsArray as Array OF Char) return integer
```

Parameters:

row	Row identifier.
col	Column ID.
idxArray	This parameter will return the IDs of the styles assigned to the cells.
ConditionArray	This parameter will return the conditions of the styles assigned to the cell.

Returns:

The number of styles of the cell. This value indicates the dimension which must to be defined the arrays that are passed as parameters to the third and fourth method.

GetListColumnStyles

This method provides information about the number of styles, the ID of the styles and conditions used to apply styles to a column.

Syntax:

```
GetListColumnStyles (col as Smallint ,VAR idxArray as Array OF Numeric
,VAR conditionsArray as Array OF Char) return integer
```

Parameters:

col	Column ID.
idxArray	This parameter will return the IDs of the styles assigned to the column.
ConditionArray	This parameter will return the condition of the styles assigned to the column.

Returns:

The number of styles of the column. This value indicates the dimension which must to be defined the arrays that are passed as parameters to the third and fourth method.

GetStrListFilterBar

This method allows to query the string assigned to the filters created with the SetStrListFilterBar method or that the user has typed in the filter bar shown with the ShowListFilterBar method.

Syntax:

```
GetStrListFilterBar() return char
```

Returns:

A string with text.

GroupListColumns

This method allows to create a new header under which grouped several columns. The new header will have a title that will be passed as a parameter to the method along with the columns that make up the group. The columns forming part of one of these groups cannot move beyond it.

The maximum number of columns that can be grouped is 10.

Syntax:

```
GroupListColumns(description as Char ,VAR argLst as ArgList OF
Smallint) return integer.
```

Parameters:

Description	Descriptive text in the column group.
Arglist	Indicates the list of columns you want to group.

Returns:

Returns an integer whose possible values are:

-1	Some (s) column (s) indicated more than once. Error Code.
-2	Some (s) column (s) passed as parameter (s) associated with another group. Error Code.
-3	Indicates that at least one of the columns that is passed as a parameter is not adjacent to other columns Error Code.
-4	Indicates that you have exceeded the maximum number of columns allowed in a group. Error Code.
-5	Indicates that the number of columns in the parameter is greater than the number of columns in the list. Error Code.

Positive integer that identifies the group created.

IsCheckedListCell

This method allows to query the value of a cell in a column list check of check type. It applies to type List Box controls columns list (string and sql).

Syntax:

```
IsCheckedListCell(row as Integer ,col as Smallint ,VAR ischecked as
Boolean)
```

Parameters:

row	Row identifier.
col	Column ID.
ischecked	Check control value. TRUE if the value is 1. FALSE otherwise.

ListSetSkin

This method tells the runtime file containing the description of the graphic elements to be used to change the appearance of the List Box control columns list type (string and sql).

Returns:

```
ListSetSkin(skinFile as Char) return boolean
```

Parameters:

SkinFile	File path.

Returns:

TRUE if loaded the list of items and FALSE otherwise.

The graphic elements that can be set in this file are:

SKIN_LIST_ITEM_HEADER_PART_BITMAP_FOCUSED
Indicates the bitmap of the column header when the mouse cursor is over it.

SKIN_LIST_ITEM_HEADER_PART_BITMAP_NORMAL
Indicates the bitmap of the column header when the mouse cursor is not over it.

SKIN_LIST_ITEM_HEADER_PART_BITMAP_DOWN
Indicates the bitmap header of the column when it is held down on the mouse button.

SKIN_LIST_ITEM_HEADER_PART_BORDER_FOCUSED
Specifies the border color of the column header when the mouse cursor is over it.

SKIN_LIST_ITEM_HEADER_PART_BORDER_NORMAL
Specifies the border color of the column header when the mouse cursor is not over it.

SKIN_LIST_ITEM_HEADER_PART_BORDER_DOWN
Specifies the border color of the head of the column when it is held down on the mouse button.

SKIN_LIST_ITEM_HEADER_PART_TEXT_FOCUSED
Indicates the color of the text of the column header when the mouse cursor is over it.

SKIN_LIST_ITEM_HEADER_PART_TEXT_NORMAL
Indicates the color of the text of the column header when the mouse cursor is not over it.

SKIN_LIST_ITEM_HEADER_PART_TEXT_DOWN
Indicates the color of the header text of the column when it is held down on the mouse button.

SKIN_LIST_ITEM_SELECTED_ROW_PART_BITMAP_FOCUSED
Indicates the bitmap of the selected row in the list.

SKIN_LIST_ITEM_SELECTED_ROW_PART_BORDER_FOCUSED
Specifies the border color of the selected row in the list.

SKIN_LIST_ITEM_SELECTED_ROW_PART_TEXT_FOCUSED
Indicates the text color of the selected row in the list.

SKIN_LIST_ITEM_SCROLLBAR_LEFT_ARROW_NORMAL
Specifies the bitmap to be applied on the arrow left shift in the horizontal scroll bar of the list.

SKIN_LIST_ITEM_SCROLLBAR_RIGHT_ARROW_NORMAL
Specifies the bitmap to be applied on the scroll arrow right shift on the horizontal scroll bar of the list.

SKIN_LIST_ITEM_SCROLLBAR_UP_ARROW_NORMAL
Specifies the bitmap to be applied on the scroll arrow upwards in the vertical scroll bar of the list.

SKIN_LIST_ITEM_SCROLLBAR_DOWN_ARROW_NORMAL
Specifies the bitmap to be applied on the down scroll arrow in the vertical scroll bar of the list.

SKIN_LIST_ITEM_HORIZONTAL_SCROLLBAR_THUMB_NORMAL
Specifies the bitmap to be applied on the button drag horizontal scrollbar (thumb) from the list.

SKIN_LIST_ITEM_VERTICAL_SCROLLBAR_THUMB_NORMAL
Specifies the bitmap to be applied on the drag button in the vertical scroll bar (thumb) from the list.

SKIN_LIST_ITEM_HORIZONTAL_BITMAP
Specifies the bitmap to be applied on the horizontal scroll bar.

SKIN_LIST_ITEM_VERTICAL_BITMAP
Specifies the bitmap to be applied on the vertical scroll bar.

SKIN_LIST_ITEM_SORTBAR_BACKGROUND_PART_BITMAP
Specifies the bitmap to be applied on the column sort bar.

SKIN_LIST_ITEM_HEADER_FILTER_DROPDOWN_ARROW_BITMAP
Specifies the bitmap to be displayed as an arrow in list type controls at the top of the columns on which to apply filters.

The bitmaps referenced in the file that is used to define the skin must be in the same directory.

RemoveGroupListColumns

This method deletes the column grouping under a single header when the group was created with the method GroupListColumns.

Syntax:

```
RemoveGroupListColumns(groupId as Integer).
```

Parameters:

GroupId	Group identifier returned by the method GroupListColumns.

Returns:

TRUE if the column has been removed and false if it was an error and failed to eliminate the column.

RemoveListGroupTreeView

This method indicates to the control list that must show the list entries in List Box controls columns list type (string and sql) in classic view and not in a tree after the method call ShowMultiColumnGroupDlg.

Syntax:

```
RemoveListGroupTreeView()
```

ResetListCellStyles

This method resets the styles assigned to a cell in a List Box Columns list type (Strings, Tree and SQL) with the methods and SetListCellConditionalStyle SetListCellStyle. This method don’t restarts the styles assigned to columns.

Syntax:

```
ResetListCellStyles(row as integer, col as Smallint) return boolean
```

Parameters:

row	Row identifier.
col	Column ID.

Returns:

TRUE if the action has been performed and FALSE otherwise.

ResetListColumnStyles

This method resets the styles assigned to the control column of type List Box Columns list (Strings, Tree and SQL) with the methods and SetListColumnConditionalStyle and SetListColumnStyle.

Syntax:

```
ResetListColumnStyles(column as Smallint) return boolean
```

Parameters:

column	Column ID.

Returns:

TRUE if the action was performed, FALSE otherwise.

ResetListFilterBar

This method restarts the value used in the SetStrListFilterBar method to filter the data shown in the list.

Syntax:

```
ResetListFilterBar()
```

RestoreBackupListRow

This method restores to the list row the value stored in memory with the BackupListRow method.

Syntax:

```
RestoreBackupListRow(row as integer)
```

Parameters:

row	Row identifier.

SetCheckListCell

This method allows to modify the control value in a cell check style column in the list control (string and sql).

Syntax:

```
SetCheckListCell(row as Integer ,col as Smallint ,check as Boolean)
```

Parameters:

row	Row identifier.
col	Column identifier.

check	Value to be assigned to the control. Possible values are: TRUE and FALSE. If you pass TRUE, the cell value is 1, if you pass FALSE, the value is 0.

SetComputedColumn

This method indicates that the values of a numeric column of a List Box control (string and tree) are the result of a series of logical / arithmetical operations that will be done when adding or modifying rows in the list . The operands can be columns in the same row, but you cannot use the result of this column as an operand of another calculated column.

Syntax:

```
SetComputedColumn(column as Smallint ,formula as Char) retun booelan
```

Parameters:

Column	Identifier assigned column on which the result of the operation.
Formula	Operation to be performed.

Returns:

Boolean indicating whether the column has been calculated.

NOTES:

If the operands are columns notation is to be indicated:

```
$<number of column >$
```

Calling this method must be done before loading the list.

SetListCellStyle

This method allows to apply a style to a cell. The style must have been previously defined with the method CreateListCellStyle. The style that is defined for a cell overrides defined for a column.

The style of the cell should be assigned after the cell has acquired its value. That is, you cannot assign a style to a cell in a row that don’t exists.

Syntax:

```
SetListCellStyle(row as Integer ,col as Smallint ,style as Integer)
return boolean.
```

Parameters:

Row	Row index.
Col	Column ID.
Style	Identifier format to be applied to row. This identifier is the result of a method call CreateListCellStyle.

Returns:

A boolean value indicating whether or not there was an error in the method call.

SetListCellConditionalStyle

This method allows to apply a style to a cell if it meets the conditions to be specified as a parameter. The style must have been previously defined with the method CreateListCellStyle. The style that is defined for a cell overrides defined for a column.

The style of the cell should be assigned after the cell has acquired its value. That is, you cannot assign a style to a cell in a row that don't exists.

In the case of type List Box controls Sql, if you change the order in which the data were assigned when the style, it will not be respected, ie the styles in these lists are assigned to a particular fi-le/column , and if the data in the cell changes the style does not change.

Syntax:

```
SetListCellConditionalStyle(row as Integer ,col as Smallint ,style as
Integer, condition as Char).
```

Parameters:

Row	Row index.
Col	Column ID.
Style	Identifier format to be applied to the row. This identifier is the result of a method call CreateListCellStyle.
condition	Condition when the cells assigned to the style (see Annex I).

Returns:

A boolean indicating whether or not there was an error in the method call.

SetListColumnStyle

This method allows to assign a style to all cells in a column of a list.

Syntax:

```
SetListColumnStyle(col as Smallint ,style as Integer)
```

Parameters:

Col	Column ID.
Style	Format identifier to be applied to the column. This identifier is the result of a method call CreateListCellStyle.

SetListColumnConditionalStyle

This method assigns a visual style to all cells in a column of a list that satisfy the condition to be specified as a parameter.

Syntax:

```
SetListColumnConditionalStyle(col as Smallint ,style as Integer
,condition as Char)
```

Parameters:

Col	Column ID.
Style	Identifier format to be applied to the column. This identifier is the result of a method call CreateListCellStyle.
condition	Condition that the cells in the column to which is assigned the style must complain (see Annex I).

SetListColumnEditType

This method allows to specify the type of control that will display and edit a cell. By default this control will be text and edit field type.

Syntax:

```
SetListColumnEditType(col as Smallint ,Editstyle as Smallint ,Values
as Char)
```

Parameters:

Col	Column ID.
Editstyle	ID style. The possible values are:

1	Text.
2	Check.
3	Date.
4	Droplist.
5	Numeric.

Values — If the selected control is a drop list in this parameter will indicate the list of values to be displayed in this list. Its value is a string where the elements of the list are separated by the "|" character.

If the selected control is a control check, this parameter is indicated on a string, separated by the "|" character, what the value TRUE and what is FALSE. If not indicated any, will take 1 as TRUE and 0 as FALSE.

SetListTotalsCellStyle

This method allows to assign a style to the cells totals rows where totals will be shown when created with the TotalizeColumn method an computed with the ComputeListColumnsTotals method.

The styles are created with the method CreateListCellStyle.

Syntax:

```
SetListTotalsCellStyle(style as Integer)
```

Parameters:

style	style identifier created by the method CreateListCellStyle.

SetLockColumns

This method allow to lock the horizontal scrolling of a specified number of columns in a list-type control. The order in which the freezing of the columns will be from left to right.

Syntax:

```
SetLockColumns(column as Smallint).
```

Parameters:

Column	Indicates the number of columns to block.

SetStrListFilterBar

This method indicates the text you perform the automatic filter in the current list. As a result of the filter displays all rows that contain the specified text in either column that compose it. The result of this operation is the same as the method call ShowListFilterBar, with the difference that no user interaction.

Syntax:

```
SetStrListFilterBar(text as Char)
```

Parameters:

Text	Literal by which to filter.

ShowListFilterBar

This method shows an edit control which may indicate a string to filter the rows in the list.

Pressing the [Esc] will cancel the action and will launch the event ListCancelEditFilterBar. Pressing [Enter] will be filtered by the input data and launch the event ListAcceptEditFilterBar.

The search is performed as follows: The indicated are literal searches all cells in the list. If the edit control indicated a value display rows that meet all conditions.

Syntax:

```
ShowListFilterBar(show as Boolean, backColor integer, textColor
integer, ,casesensitive as Boolean default FALSE)
```

Parameters:

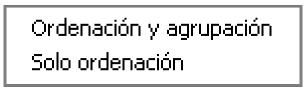
Show	Indicates whether the edit control display. The possible values are:
	TRUE: be painted over.
	FALSE: not show the edit control, and if the control is visible when you call this method, it will be hidden.
backColor	RGB indicating the background color of the edit control.
textColor	Indicating RGB text color.
casesensitive	Select whether or not the search is case sensitive.

IMPORTANT: *If the value shown in parameter casesensitive is FALSE and list type is SQL, client-server installations engine version must be at least 3.4 0.2 in the Windows version. For UNIX / Linux releases of engine versions will be equal to or higher than the 3.2 0.2, 3.4 0.2 y 3.6 0.2.*

ShowMultiColumnGroupDlg

This method allows to display a list of records from a List Box control type as a view of grouped records indicating the columns that are grouped and columns that aggregate values are calculated.

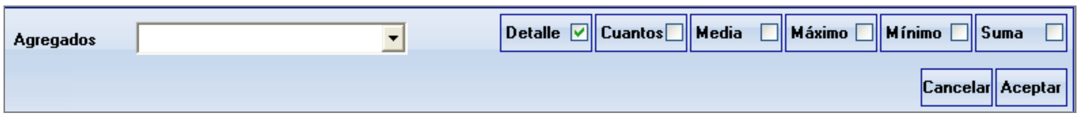
This displays a status bar at the bottom of the list on which may drag columns. The columns can be added to perform two actions: group and order. If on some of them just want to sort the records will have to display the sub-menu shown below and choose "Solo ordenación ".



Besides the two mentioned actions may apply the following features for added columns:

Detalle	Displays records. If you uncheck the groups and shows only the aggregate values.
Cuántos	This function displays the number of records grouped under one group.
Media	Gets the average of the column in the group's records.
Máximo	Gets the maximum value of the column in the group's records.
Mínimo	Gets the minimum value of the column in the group's records.
Suma	Obtains the sum of all column values in the records of the group.

To indicate the added column should be selected in this drop list control bar shown in the following figure.



The appearance of the list after applying this method is similar to a tree list displayed. Each node indicates a breakpoint in a group. The records will be sorted in the order in which the columns were added in the bar, and they will move from the place they originally occupied the top positions.

Syntax:

```
ShowMultiColumnGroupDlg(show as Boolean ,iconOpen as Smallint default
0 ,iconClose as Smallint default 0 ,lockGroupColumns as Boolean
default TRUE ,countTitleTemplate as Char default "Total:"
,maxTitleTemplate as Char default "Máximo:" ,minTitleTemplate as Char
default "Minim:" ,sumTitleTemplate as Char default "Sum:"
,avgTitleTemplate as Char default "Average:")
```

Parameters:

Show	Indicates whether or not to display the toolbar. Possible values are: TRUE and FALSE.
iconOpen	Identifier of the icon to be displayed in the list view to indicate that the group is deployed.
iconClose	Identifier of the icon to be displayed in the list view to indicate that the group is not deployed.
lockGroupColumns	Allows setting the columns that make up the group after moving so they do not move when you scroll horizontally. The possible values are: TRUE and FALSE.
countTitleTemplate	Literal label that will appear as the count () function.
maxTitleTemplate	Literal label that will appear as the function max ().
minTitleTemplate	Literal label that will appear as the min () function.
sumTitleTemplate	Literal label to be displayed as the sum () function.
avgTitleTemplate	Literal label to be displayed as the avg () function.

NOTES:

The maximum is 8 columns for group.

ShowMultiColumnSortDlg

This method shows a sort bar in the bottom of the list. To this was drag the columns that you want to sort.

Once the columns added to the bar you can change the order as well as sorting ascending / descending. To apply sorting must press the [Aceptar] button shown on the right of the bar.

The selected columns will be moved from the place originally occupied to the left positions.

The columns selected by this method may be the queried with the method GetOrderBy.

Syntax:

```
ShowMultiColumnSortDlg(show as Boolean, changeView as Boolean).
```

Parameters:

Show	Indicates whether to display the toolbar or not. Possible values are: TRUE and FALSE.
changeView	This boolean parameter indicates if the sort columns will change the position to the left side of the list.

TotalizeColumns

This method lets you to add total columns in a List Box control of String type or SQL type.

Syntax:

```
TotalizeColumn(column as Smallint ,aggType as Smallint ,condition as
Char default NULL ,title as Char default NULL)
```

Parameters:

column	Column index.
aggType	Total type. The possible values are: 1 (sum), 2 (minimum), 3 (maximum), 4 (middle), 5 (total records).
condition	Condition for which you want to restrict the totalizing function (see Annex I).
title	Label to be entered in the aggregate.

NOTE:

Totals are not calculated automatically. To carry out the calculation will need to call the method ComputeListColumnTotals.

UpgradeBackupListRow

This method allows to change the value stored in memory with BackupListRow method.

Syntax:

```
UpdateBackupListRow(row as Integer, Text as Char)
```

Parameters:

row	Row identifier.
text	Text that is assigned to the row.

5. Events

ListAcceptEditColumnFilter
This event will be lauched when the [Enter] key is pressed after executing the AllowColumn-HeaderFilter method.

ListAcceptEditFilterBar
This event will be launched when the [Enter] key is pressed after displaying the edit field to filter the data.

ListCancelEditColumnFilter
This event will be launched when the [Cancel] key is pressed after running the AllowColumnHeaderFilter method or the focus is lost by the dropt list control.

ListCancelEditFilterBar
This event will be launched when the [Cancel] key is pressed after displaying the edit field to filter the data entered.

6. Bug fixes

6.1 Runtime

- In a control string list type, if the column was numeric and either value was a negative number, clicking on the header to sort the list itself was not taken into account the sign.
  Fixed.
- If a source Cosmos had more than 32,768 lines, and from CEDIT.EXE was pressed the key combination [Ctrl-End], the cursor is not positioned on the last line of the file.
  Fixed.
- If the tabs of a TabControl type control were associated with an icon and the screen showed SetTabVerticalPages method, instead of vertically stacked painted mostra-ban is in Classic View.
  Fixed.
- No cell correctly repainted a list that was published after modifying its value.
  Fixed.
- The method did not account UnloadTo escape characters.
  Fixed.
- When you search using the Query method from Formtable class with condition, if the data on the order contain the word ‘order’, and it was a sort by that column, Cosmos Case not built well.
  Fixed.
- Method AddColumnFilter. It did not work correctly when the value to filter into the condition was "is not null".
  Fixed.
- If displayed a drop list in a PC with two monitors connected, if the drop list was on the right monitor, the window is displayed on the monitor on the left.
  Fixed.
- SQL list with filters. If the method loadSelect was executed again, did not apply the previously existing filters until the sort column button pressed.
  Fixed.
- General Protection Fault when the size of return method headerSend CallWebService parameter exceeded 1000 characters.
  Fixed.
- Method FindCol. If the search text had more than one occurrence in the list, when it reached the bottom of the list this method did not return 0, but the first item in the list that meets the search conditions.
  Fixed.

- Export to Excel / CSV variable smallint or integer mask #, # # # or #, # # 0 from the Preview of Cosmos was not correct.
  Fixed.

- When enabling a command, the menu option associated with it was not enabled (see ENABLEMENUOPTIONONENABLECOMMAND environment variable).
  Fixed.

- Random General Protection Errors when executing the method SetErrorStr.
  Fixed.

- Export to Excel, PDF, HTML and ODT lists. Modified the export lists to export tree each node only if the parent is not collapsed. So far all rows exported, was contracted or not the parent node.
  Fixed.

- The Selected property returned a smallint when it should return an integer.
  Fixed.

IMPORTANT: *This error will persist if run programs compiled with previous versions. That is, when the element of the list exceeds 32,767, the returned data is not correct.*

For successful implementation correctness is necessary that any object that is used to assign or collect property value must be of type integer. It will also be necessary to recompile the application with this version.

If the objects defined as local or global variables, or parameters defined function to assign or collect the value of the selected property and are objects of type integer only need to recompile the application.

6.2 CTSQL

- The sorting of decimal type fields was incorrect when the precision was 3 or higher.
  Fixed.

Annex I

Simple arithmetic

sum
subtraction
multiplication
division

Complex arithmetic

Powers of e (exp)
Powers (^)
Logarithm in base 2 (log2)
Base 10 logarithm (log)
Natural logarithm (ln)
Square root (sqrt)
Sign (sign). If <0 returns -1, if >0 returns 1
Rounding to the nearest integer (rint)
Absolute value (abs)
The minimum value of a list of values (min)
Maximum value of a list of values (max)
Addition of a series of values (sum)
Arithmetic average (avg)

Trigonometric operations

Sinus (sin)
Cosine (cos)
Tangent (tan)
Arcsine (asin)
Arccosine (acos)
Arctangent (atan)
Hyperbolic sine (sinh)
Hyperbolic cosine (cosh)
Hyperbolic tangent (tanh)
Hyperbolic arcsine (asinh)
Hyperbolic arccosine (acosh)
Hyperbolic arctangent (atanh)

Logic

And (&&)
Or (||)
Less or equal than (<=)
Greater or equal than (>=)
Less than (<)
Greater than (>)
Equal than (==)
Different than (!=)
Ternary operator -if then else- (?:)

String Comparison

equalstring
equalstringnocase

Consult null

isnull

Priority table

Operator	Meaning	Priority
=	Asignation	-1
&&	Logical AND	1
||	Logical OR	2
<=	Less or equal than	4
>=	Greater or equal than	4
!=	Different tan	4
==	Equal	4
>	Greater than	4
<	Less than	4
+	Sum	5
-	Subtraction	5
*	Multiplication	6
/	Division	6
^	Power	7

Annex II. Notes about cell-level styles and column-level styles in Cosmos list control

In Cosmos 5.0 version was added the ability of assign visual styles to the list controls at cell-level or at column-level.

Creating and defining styles

An style allows to define a set of visual attributes with a common identifier that will can be assigned to cells and columns in a list control.

The allowed atributes are:

- Typography (Font name, size, bold, italic, underlined, labeled, charset).
- Text color.
- Background color.
- Alignment (left, centered, right).
- Icon.

When a list style is created, it is not mandatory to detail all the style attributes. For example, the field that defines the typography can be null. In this case, the item font will not be changed when the style is assigned but the list font will be used.

If you don't want to change the text color or the background color the value that will be assigned is -1 (minus one) in the style declaration in the foreground and/or background parameters. In this case, the list item will be drawn with text color or background color indicated in the SetRowBackground, SetColumnBackground, SetRowForeground, SetColumnForeground methos but, if none of these methods are called, the list default color will be used.

If you don't want to define a style icon, indicate the value -1 (minus one) in the icon parameter.

An style can be assigned at more than one cell and more than one column in the same list. An style defined for a list control cannot be assigned to a different list control. That is, the styles can be re-used for the items in the list that have been created, but if you want to assign these styles to another list control items, you must redefine these styles for the new control.

Cell-level styles and column-level styles

The main difference between cell-level styles and column level styles is that a cell level style is as-signed to a list item, identified by index and column number, and a column-level style is assigned to the cells in a column for all the rows in the list.

Priority in styles

The cell-level styles are priority on the column-level styles.

That is, if an A style is assigned to a cell in the row 10, column 4 (cell-level style), and a B style is assigned to the column number 4 (column-level style), the 10-4 cell will be drawn with the A style (cell-level style).

The cells in the column number 4 of the other rows will be drawn with the B style (column-level style).

Lists of styles

When an style is assigned to a cell or a column that previously was assigned another style, the new assigned style don't replace the previous style, but the new style is added to the available styles for the cell/column.

This allows that a column cell is drawn with one style or another depending on the cell value.

May be indicated, for example, that the text in a numeric column cell is drawn in red color when the value is a negative number, that the text in the same column cell is drawn in black color when the value is 0, and the text is drawn in blue color when the value is greater than 0.

For example:

The STYLE_1, STYLE_2 and STYLE_3 are created for the LST_1 list control.

These styles define a font, text color, background color, alignment and icon.

The STYLE_1 is assigned to the column number 1. This is a conditional assignment style that will be applied to the column when the cell value is less than 0.

The STYLE_2 is assigned to the same column. This is a conditional assignment style that will be applied to the column when the cell value is 0.

The STYLE_3 is assigned to the same column. This is a conditional assignment style that will be applied to the column when the cell value is greater than 0.

A cell on this column with a value equal to 0 will be drawn with the attributes defined in STYLE_2.

A cell on this column with a value equal to -1 will be drawn with the attributes defined in STYLE_1 because satisfies the condition (less than 0).

A cell on this column with a value equal to 123 will be drawn with the attributes defined in STYLE_3 because satisfies the condition (greater than 0).

The order in which Cosmos will verify the style that will assign to a cell/column is the same than the order in which the cell/column styles were assigned.

This point is important, mainly because can be given the circumstances that more than one conditional cell-level style or column-level style has been assigned to a cell/column and the cell value satisfies both conditions.

For example:

The STYLE_1, STYLE_2 and STYLE_3 are created for LST_1 list control.

Firstly, the STYLE_1 is assigned to the column number 1. The condition for applying this style is that the cell value is less than 0.

After, the STYLE_2 is assigned to the column number 1. The condition for applying this style is that the cell value is equal or greater than 0 and less than 100.

Thirdly, the STYLE_3 is assigned to the column number 1. The condition for applying this style is that the cell value is greater than 80.

In a cell in the column number 1 the value is -3. This cell will be drawn with the STYLE_1 attributes, because the value is less than 0, (STYLE_1), the value is not equal or greater than 0 and less than 100 (STYLE_3) and the value is not greater than 80 (STYLE_3).

In another cell in the column 1 the value is 40. This cell will be drawn with the STYLE_2 attributes, because the value is between 0 and 100 , and the value is not less than 0 and greater than 80.

In another cell in the column 1 the value is 120. The cell will be drawn with the STYLE_3 attributes, because the value is not less than 0 (STYLE_1), the value is not equal or greater and less than 100 (STYLE_2), and the value is greater than 80 (STYLE_3).

In another cell in the column 1 the value is 90. Here we find that the cell don’t satisfies the assign condition of the STYLE_1 (less than 0), but satisfies the assign condition of the STYLE_2 (between 0 and 100) and the STYLE_3 condition (greater than 80).

In this case, the drawing style will be STYLE_2, because this style was assigned before STYLE_3.

In this case, and to avoid problems, it would have been strictly correct that the STYLE_3 condition is that the cell value were equal or greater than 100.

A style applied without condition or applied with a null value in the condition field will not mean that the style is always applied to the cells / columns to which is assigned the style, but his condition is always TRUE. The order in which they check the style in which to draw a cell / column will always be the order of allocation of styles.

If you want to change the styles check order it will be mandatory that the styles are reassigned in the desired order after reset the styles with the ResetListCellStyles or ResetListColumnStyles.

Priority in text and background color

It may be the case that the text or background color at row-level and/or column-level was assigned to a list items with the SetRowBackground, SetColumnBackground, SetRowForeground or SetColumnForeground methos.

In this case, the text or background color check order will be:

1. Text/background color defined in a cell-level style (SetListCellConditionalStyle y SetListCellStyle).
2. Text/background color defined in a column-level style (SetListColumnConditionalStyle y SetListColumnStyle).
3. Text/background color defined at row level (SetRowBackground y SetRowForeground).
4. Text/background color defined at column level (SetColumnBackground y SetColumnForeground).
5. Text/background color at control-level.

Creation, assignment, removing and styles querying methods at cell-level and column-level

- Creation of styles
  CreateListCellStyle

- Assignment of styles at cell-level

SetListCellStyle	Without condition.
SetListCellConditionalStyle	With condition.

- Assignment of styles at column-level

SetListColumnStyle	Without condition.
SetListColumnConditionalStyle	With condition.

- Removing of styles at cell-level

ResetListCellStyles	For styles assigned at cell-level or column-level with and without condition.

- Removing of styles at column-level

ResetListColumnStyles	For styles assigned at column-level with and without condition.

- Querying of styles at cell-level

GetListCellStyles	For styles assigned at cell-level with and without condition.

- Querying of styles at column-level

GetListColumnStyles	For styles assigned at column-level with and without condition.