MultiBase Cosmos

Notes to version 7.0

Index

[NT_CO_7 0_v1_en.docx]

1. Implementations

- New automatic edit mode in the Cosmos maintenance's forms in Grid controls. The implementation has been developed for both Edit Mode and Non Edit Mode (environment variable AUTOEDITMODE).

- New visual style in the Form controls.

- The next operations will be allowed in Grid controls:

  o Select a range of rows.
  o Copy the data of the rows to the clipboard.
  o "On Click" event on the Grid's client area.
  o Query and modify the show order of the columns of the Grid.
  o Adjust the width of the columns to the maximum size of the visible data when you click on the column header separator.
  o View the full value of the cell of a Grid control. It is possible to visualize it in the tooltip window that appears when the mouse pointer passes over the cell.

- In the following controls: Edit field, Text and Panel, it is possible to copy the text to the clipboard when they are disabled.

- Edit Field Control. You can use an edit control as a rich text edit control.

- In Tree List controls:

  o It is possible to add filters in the header of the Tree-type List Box controls (AllowColumHeaderFilter method).
  o Whether the first column is visible, when pressing the left arrow key, the focus will place on the parent node of the selected element.

- It is possible to sort by Datetime type columns in List controls.

- It is possible to include constants in the Text controls of the Page class so that when generating a report, the Cosmos runtime changes them with its value.

- SplitButton. This button will have the functionality of the Push Button control and also the option to display a menu or a Form.

- The possibility of signing PDF documents from Cosmos has been implemented. See **CosSignPdf**.

- The GetProperty and SetProperty methods of the Control class will support the "underline" and "Strikeout" values for the Font property.

- TTTOOLS. Two new parameters have been implemented to be able to repair all the tables in the database or only one of them.

- Development environment:

    o It is possible to visualize the structure in tree of the controls children of a control of the Screen or of a page of impression, in such a way that the parent-child structure of the screen can be easily visualized. With the pointer located on the control, press the right mouse button and select the option "Control Info".

    o The "Esc" key allows to exit the TEST option of a Screen when the focus is not in an edit field.

2. Automatic edit mode

An automatic mode has been implemented to modify the records of a table.

This mode can only be used in:

a) Maintenance Forms using the predefined Form commands (editing and non-editing modes).
b) Whether the control used in the maintenance of the table is a Grid.
c) The table is in Browse status.

2.1 Activation

This mode can be activated at the module level, in the Form class or globally.

- Module level:

    Module.SetOption("AUTOEDITMODE", TRUE)

- Form class level:

    objForm.SetOption("AUTOEDITMODE", TRUE)

- Globally:

    Environment variable AUTOEDITMODE=TRUE

2.2 How it works

2.2.1 Cosmos Edit Mode

Whether this mode is activated, launching the On-click event in the Grid control of a Formtable will edit the child control and the table will be edited for modification. Automatically:

a) The **AcceptEdit** and **CancelEdit** commands wil be activated.
b) The querying and delete commands will be deactivated.
c) The EditUpdate command and the On Edit table event of the current table will be sent.
d) The record will be locked.

Whether the field value is not modified, the **CancelEdit** command will be sent automatically**.**

Whether the value is modified, the AcceptEdit command will be sent when the current file is changed.

When the cursor is on the last row of the Grid control and the down arrow key is pressed, the Formtable will be set to status New. To add the record, press the up arrow to change the active row, or execute the AcceptEdit command.

2.2.2 Cosmos Not Edit Mode

Whether this mode is activated, launching the On-Click event in the Grid control of a Formtable will edit the child control. Whether the field value is modified, the command Update will be executed automatically. Whether the field is not modified, no command will be executed.

Whether de cursor is positioned in the last row of the Grid control and the down arrow key is pushed, the Formtable will be set to New state. In order to add the new row, the up arrow key must be pushed, or run the Add command.

2.3   Keyboard keys

- Moving the cursor between rows of the grid:

  In order to moving the cursor from one row to the next or previous row, you can use the next keys:

  [Down arrow]   Go to the next row or add new row.

  [Up arrow]   Go to the previous row.

  NOTE: If the control that is being edited at that moment has these keys captured, the focus will not change row, but the actions assigned to that control will be executed. For example, in the case of a drop list, when you press the up arrow or the down arrow keys, we go through the elements of the list and the current row will not be changed.

- Scrolling between the columns of a row:

  - Cursor movement arrows:

    [Right arrow] Go to the next column.

    [Left arrow]   Go to the previous column.

  - [Tab]. Go to the next field in the same row. When you reach the last column of the row, the focus will be placed on the next control that has been established in the tabulation order of the Form (it will not be placed in the next row of the Grid).

  - [Shift]+[Tab]. The focus will be positioned in the previous column. Upon reaching the first column of the row the focus will be positioned in the corresponding control according to the tabulation order of the Form (it will not be placed in the previous row of the Grid).

  - [Return]. Pressing this key will focus on the next column. When you reach the last column of the row, the focus will be placed on the next control, as established in the tabulation order of the form (it will not be placed on the next row of the grid).

3. Custom style on Form controls

A new display style has been implemented for the controls of a Form.

In this new style the controls will be drawn with rectangular borders and flat colors, without taking into account the visual style of the Windows OS where the application is running.

In addition, you can change the default colors of the background and text of all controls, by type of control, respecting the colors of those who were assigned a color other than this at design time or at runtime.

These changes will only have an effect on execution, they can't be viewed at design time.

To activate this new display mode it will be necessary:

1. Define the **COSMOSVISUALMODE** environment variable in the project configuration file.
    The value of this variable must be 7.

2. Create a new section in this file with the name **[Custom Colors].**
    This section will only have to be defined if you want to change the colors that Cosmos assigns by default.

    Indicate the name of the property and assign it a color. The color can be assigned by:

    a. The expression Rgb (red, green, blue), being red, green and blue the shades of color in base 10.
    b. With the hexadecimal value represented by the color, using an expression in the following format: #RRGGBB, where RR, GG and BB are the hexadecimal value corresponding to red, green and blue respectively.

Colors in BOX controls

Name	Description
Color_Text_Box	Color of the box text.
Color_Back_Box	Background color of the box.

Text colors and header background colors in GRID and LIST controls

Name	Description
Color_Text_Header	Text color of the control header.
Color_Text_Header_Disabled	Text color of the header when the control is disabled.
Color_Text_Header_Hot	Color of the text when the mouse pointer is over the header.
Color_Text_Header_Pressed	Color of the text by pressing the header.
Color_Back_Header	Background color of the control header.
Color_Back_Header_Disabled	Background color of the header when the control is disabled.
Color_Back_Header_Hot	Background color when the mouse pointer is over the header.
Color_Back_Header_Pressed	Color of the text when clicking on the header.

Color de texto, color de fondo y color de bordes en controles Button

Name	Description
Color_Text_Button	Text color.
Color_Text_Button_Disabled	Text color when the button is disabled.
Color_Text_Button_Hot	Text color when the mouse pointer is over the button.
Color_Text_Button_Pressed	Text color when the button is being pressed.
Color_Text_Button_Focused	Text color when the control has the focus.
Color_Back_Button	Background color of the button.
Color_Back_Button_Disabled	Background color when the button is disabled.
Color_Back_Button_Hot	Control background color when the mouse pointer is over the button.
Color_Back_Button_Pressed	Background color of the control when the button is pressed.
Color_Back_Button_Focused	Background color of the control when it receives the focus.
Color_Border_Button_Disabled	Color of the border of the control when it is disabled.
Color_Border_Button	Control border color.
Color_Border_Button_Hot	Color of the control edge when the mouse is over the button.
Color_Border_Button_Pressed	Color of the control edge when the button is pressed.
Color_Border_Button_Focused	Background color of the control when it receives the focus.

Background color, text color and border color of auxiliary buttons for Spin controls, Edit Field with combo, Drop Llist and Drop Edit

Name	Description
Color_Text_AuxButton	Icon color of the auxiliary control button. In the case of the Edit field only if they have assigned the default icon.
Color_Text_AuxButton_Disabled	Icon color of the auxiliary button when it is disabled. In the case of the Edit field only if they have assigned the default icon.
Color_Text_AuxButton_Hot	Icon color of the auxiliary button when the mouse pointer is positioned on the button. In the case of the Edit field only if they have assigned the default icon.
Color_Text_AuxButton_Pressed	Icon color of the auxiliary button when pressed. In the case of the Edit field only if they have assigned the default icon.
Color_Text_AuxButton_Focused	Color of the auxiliary button icon when it receives the focus. In the case of the Edit field only if they have assigned the default icon.
Color_Back_AuxButton	Background color of the auxiliary control button.
Color_Back_AuxButton_Disabled	Background color of the auxiliary button when it is disabled.
Color_Back_AuxButton_Hot	Background color of the auxiliary button when the mouse pointer is positioned over the button.
Color_Back_AuxButton_Pressed	Background color of the auxiliary button when pressed.
Color_Back_AuxButton_Focused	Background color of the auxiliary button when it receives the focus.

Background color and border color in Tab controls

Name	Description
Color_Text_Tab	Control text color.
Color_Back_Tab	Background color of the control.
Color_Border_Tab	Color of control edge.

Text color in Text controls

Name	Description
Color_Text_Text	Control text color.
Color_Text_Text_Disabled	Text color when the control is disabled.

Text color of a Radio Button control

Name	Description
Color_Text_Radio	Control text color.
Color_Text_Radio_Disabled	Text color when the control is disabled.

Text color of a control Check

Name	Description
Color_Text_Check	Control text color.
Color_Text_Check_Disabled	Text color when the control is disabled.

Text color of a Hightlighted control

Name	Description
Color_Text_Highlight	Control text color.
Color_Text_Highlight_Hot	Text color when the mouse pointer is over the button or over the tabs of a Tab control.
Color_Text_Highlight_Pressed	Control text color when pressed.
Color_Text_Highlight_Warm	Text color of the non-active button of a Split Button when the mouse pointer is over the button.
Color_Back_Highlight_Hot	Background color of the control when the mouse is positioned over it.
Color_Back_Highlight_Pressed	Background color when the control is pressed or selected.
Color_Back_Highlight_Warm	Color of the non-active button of a Split button when the mouse pointer is on the button.

4. Rich text in Edit Control

The possibility of using an editing control as an editing control with rich text has been implemented.

To do this, you have to check the Check control "Rich Text" in the properties dialog of the control in the General tab.

4.1 Methods

All the methods listed below belong to the SimpleControl class, but are indicated in this section for clarity.

- RichTextAddText

  Add text to the end of the editor.

  ```
  RichTextAddText (text as Char)
  ```

- RichTextGetProperty

  Checks the value of a property of the selected text or the paragraph where the cursor is located.

  ```
  RichTextGetProperty (name as Char ,VAR value as Object ,VAR indexes as ArgList
  OF Object)
  ```

  Parameters:

Name	Property name.
Value	Property value returned by the function.
Indexes	Parameter reserved for future implementations.

- RichTextSaveToFile

  Saves the content of the rich text control in a file with the indicated format.

  ```
  RichTextSaveToFile (fileName as Char ,fileFormat as Char) return
  Boolean
  ```

  Parameters:

filename	File full path.
fileFormat	"RTF" or "TXT".

- RichTextSetProperty

  Assigns property to the selected text of a rich text or to the paragraph where the cursor is located.

  ```
  RichTextSetProperty(name as Char ,VAR value as Object ,VAR indexes as ArgList
  OF Object)
  ```

  Parameters:

Name	Property name.
Value	Property value.

BASE100

Indexes   Parameter reserved for future implementations.

- RichTextPrint.

Prints the contents of the control to the indicated printer.

```
RichTextPrint (printerName as Char ,showPrintDlg as Boolean default FALSE)
```

Parameters.

PrinterName   Name of the printer for which you want to print.

showPrintDlg   If its value is TRUE, it will show the printer selection box.

PrinterName parameter	showPrintDlg parameter	The content of the control will be printed by:
NULL	FALSE	Default printer.
NULL	TRUE	A dialog box to select the printer will be displayed.
Not Null	FALSE	Selecter printer.
Not Null	TRUE	It will show the dialog box to select the printer and the value indicated in the first parameter will be ignored.

4.2   Properties

Name	Description
ALIGNMENT	Indicates the alignment of the paragraph where the cursor is located: 1: left, 2: right, 3: centered.
BACKCOLOR	Background color.
BOLD	The selected text will be shown in bold. If it is already in bold, the property will be removed.
CARETPOSITION	Cursor position. The first character is 0.
FILE	File assigned to the control.
FONT	Text font.
FORECOLOR	Text color.
ITALIC	The selected text will be displayed in italics. If it is already in italics, the property will be removed.
LEFTINDENT	Indicates the indentation of the paragraph where the cursor is located with respect to the current indentation of the paragraph. A negative indent indicates that it will be suppressed to the extent indicated in the parameter. The unit that is used is the "twip". 1,440 twips correspond to 1 inch, or 57 twips to 1 mm.
LINESPACING	Spacing between lines. Allowed values: 0 Simple spacing. 1 1,5 lines spacing. 2 Double spacing.
NUMBERINGBULLET	Graphical point of enumeration of the paragraph where the cursor is located.
NUMBERINGLETTER	Lowercase letter for the enumeration of the paragraph where the cursor is located.
NUMBERINGNUMBER	Integer number for the enumeration of the paragraph where the cursor is located.
NUMBERINGROMAN	Roman numeral in lowercase letters for the enumeration of the paragraph where the cursor is located.

BASE100

Name	Description
NUMBERINGULETTER	Capital letter for the enumeration of the paragraph where the cursor is located.
NUMBERINGUROMAN	Roman numeral in capital letters for the enumeration of the paragraph where the cursor is located.
UNDERLINE	The selected text will be underlined. If it is already underlined, the property will be removed.

5. SplitButton Control

The Split button is a more complex control than a Push Button control. It is a split button that has a predetermined action followed by a divider line and an arrow.

If you press the secondary button of the control (on the arrow) a menu or a form will be displayed, and it will not execute the predefined action of the button. If you click on the primary button, the default action will be executed (On Click event or the associated button command), that is, it will work as a Push Button control.

These buttons are defined at runtime, not at design time, by invoking the SetSplitButton method of the SimpleControl class.

6. Environment variables

- **ALLOWCOPYCONTROLDATA**. Indicates to the runtime that the user can copy to the clipboard the contents of a control of type Edit, Text or Panel when they are disabled using a combination of keys. The key combination associated with this action is [Control] + [Click].

  This variable can be defined in the project configuration file or in the Cosmos configuration file in the [Environment] section, in the module (SetOption method) or in the Form class of the form (SetOption method).

  The allowed values are: TRUE and FALSE.

- **AUTOEDITMODE.** Activates the automatic editing mode in the Form.

  This variable can be defined in the Cosmos configuration file, in the project configuration file or through the SetOption method in the module or in the Form.

  The allowed values are: TRUE and FALSE.

- **CONNECTIONSTRING.** Environment variable for ODBC connections. It has been implemented to allow connection to an ODBC driver using a connection string.

  Allowed values: TRUE/YES o FALSE/NO.

  If the value indicated is TRUE or YES, the connection string must be defined. This connection string will be indicated in the DBNAME environment variable or in the Connect method of the SqlServer class.

  If the value is FALSE or the connection with the driver is not defined, it will be done using the user or system DSN defined in the system data source.

- **DEBUGFORMTABLE**. It allows showing in a tooltip the name of the table and the column to which the field of the Formtable in which the focus is positioned belongs.

  The variable can be defined in the project configuration file or in the Cosmos configuration file in the [Environment] section.

  Allowed values: TRUE and FALSE.

- **SHOWGRIDTOOLTIP.** It allows to show in a tooltip the complete content of a cell of a Grid control when the mouse passes over that cell. The tooltip will only be shown if the contents of the cell are cut off.

  The variable can be defined in the project configuration file or in the Cosmos configuration file in the [Environment] section, in the module (SetOption method) or in the Form class (SetOption method).

  Allowed values: TRUE and FALSE.

7. SimpleControl class methods

- **AllowGridMultiSelect.** Allows you to select one or more rows in a Grid control.

  Syntax:

  ```
  AllowGridMultiSelect (allow as Boolean)
  ```

  Parameters:

  allow   Indicates whether or not more than one row will be allowed in a grid control.

         Allowed value: TRUE and FALSE.

  To select a range of records in a Grid control, the mouse pointer must be positioned in the first row of the range to be selected and, keeping the Shift key pressed, move the mouse pointer to the last row of the range to be selected.

- **CopyGridSelectedRows.** Allows to copy the contents of the rows of a Grid control to the clipboard, in csv format and separated by a semicolon.

  Syntax:

  ```
  CopyGridSelecdRows ()
  ```

- **GetGridClickedRow.** It allows to obtain the identifier of the row of a Grid control.

  Syntax:

  ```
  GetGridClickedRow () return integer
  ```

  Returns:

  The number of the last row where the Grid control was clicked.

  NOTE: If the On Click event that is launched is one of the child controls, the row number will not be updated.

- **GetGridColumnsOrder.** It allows to obtain the order in which the columns of a Grid are displayed.

  The order can be set by the user (so that the user can move the columns of a Grid the value of the AllowShiftColumns property must be true) or programmer with the SetGridColumnsOrder method.

  Syntax:

  ```
  GetGridColumnsOrder() return Char
  ```

  Returns:

  A string of characters with the names of the controls that are part of the Grid, separated by the character "|".

- **GetGridSelectedRows**. It allows to consult the list of selected rows in the Grid.

  Syntax:

  ```
  GetGridSelectedRows(VAR firstRow as Integer ,VAR firstCol as Integer ,VAR
  lastRow as Integer ,VAR lastCol as Integer)
  ```

  Parameters:

firstRow	Parameter type Integer passed by reference where the number of the first selected row is returned.
firstCol	Parameter type Integer passed by reference where the number of the first selected column is returned.
	The return value will be 0, since the partial selection of the columns is still not allowed.
lastRow	Parameter type Integer passed by reference where the number of the last selected row is returned.
lastCol	Parameter type Integer passed by reference where the number of the last selected column is returned.
	The return value will be 0, since the partial selection of the columns is still not allowed.

- **SetGridColumnsOrder.** It allows to establish an order in the columns of a Grid.

  Syntax:

  ```
  SetGridColumnsOrder(columnName as Char)
  ```

  Parameters:

columnName	String of characters where the ordering sequence to be established is indicated. The syntax will be the following::

  ```
  id_control1|id_control2|id_control3|…
  ```

  Where id_controln is the identifier name of the control.

- **SetSplitButton.** When invoking this method, a Push Button control will be displayed as a Split Button control.

  Syntax:

  ```
  SetSplitButton(VAR menu as Menu ,VAR form as Window ,splitType as Integer)
  ```

  Parameters:

menu	Optional parameter of type Menu that indicates the name of the menu that will be displayed when the arrow of the button is pressed.
form	Optional parameter of type Form that indicates the name of the Form object that will be displayed when the button's arrow is pressed.

BASE100

splitType   Mandatory parameter of Integer type that indicates a sum of properties that the button will have.

The allowed values of these properties can be:

0   The arrow will be shown on the right side of the button.

1   The arrow will be displayed on the bottom of the button.

2   No Split. The separator line is not shown, and when the button is pressed the On Click event is not executed, but the Menu or the Form will be displayed, even if the arrow has not been pressed.

Note: The menu and form parameters are self-excluding, that is, only one of them must have value. The other parameter must have a NULL value.

8. Events

- The events implemented in this version are the following:

    o **Click:** over the client area of the Grid controls.

9. Constant values in Page Class and Prnpag32.dll API

A series of constants has been implemented in the Page class and in the "* .pag" files used by the "prnpag" dll.

These constants are defined at design time in the Label property of the Text controls. It will be the runtime of Cosmos responsible for assigning the corresponding value during its execution.

The constants are:

%PAGE%	Assigns the current page number to the control.
%TOTALPAGES%	It will assign to the control the total number of pages in the list. This value will be correct if the last page has not yet been sent to the printer, that is, all the pages in the list are in the buffer.
%YYYY%	It will assign to the control the year of the current date with four digits.
%MM%	It will assign to the control the month of the current date with two digits.
%DD%	It will assign to the control the month day of the current date with two digits.
%MON%	It will assign to the control the abbreviated name of the month of the current date.
%MONTH%	It will assign to the control the full name of the month of the current date.
%DY%	It will assign to the control the abbreviated name of the day of the week of the current date.
%DAY%	It will assign to the control the full name of the day of the week of the current date.
%USERNAME%	It will assign to the control the name of the user that executes the listing process.

10. TTTools

Two new parameters have been added to the TTTools index check and repair tool.

Sintaxis:

TTTools [-ndl | dl] [-all] [path] [table]

Parameters:

[-ndl | dl]   Optional and self-excluding parameters:

ndl   The indexes are regenerated with no duplicate limit.

dl   The indexes are regenerated with a limit of duplicates (32,767).

If no value is indicated, the indexes are regenerated without limit of duplicates.

[-all]   Optional parameter. All the tables in the database will be repaired. The process does not ask for confirmation.

[path]   Optional parameter. Directory of the database without the extension (dbs).

[table]   Optional parameter. Name of the table to be repaired.

NOTE: The parameters [-all] and [table] are self-excluding.

Example:

Repair all the tables in the database without displaying the graphical interface of the tool:

```
tttools.exe -all c:\cosmos\proyectos\almafac
```

Repair a table in the database without displaying the graphical interface.

```
Tttools.exe c:\cosmos\proyectos\almafac clientes
```

11. Bug fixes

- Cosrun. The data was not ordered correctly in a tree-type List Box control.

- Cosrun. When exporting to Excel the contents of a Grid control in which more than 2 columns were assigned a background color, a cosrun error occurred.

- Cosrun. When exporting to Excel the contents of a Grid that had been scrolled, the columns that were not visible were not exported.

- Cosrun. The scroll bar button in the lists was not shown with the style of the Windows theme.

- Cosrun. Tree type lists. If the environment variable DRAWTREELISTNODELINES was set to FALSE, the Glyph (symbol [+] or [-]) was not drawn in the correct place.

- Cosrun. Pressing the scroll button on the Grid controls and List controls did not scroll until the button was released. Now, as the button is pressed, the rows or columns of the control will be moved.

- Cosrun. The Check controls in the lists were not drawn in the correct position if the COSMOSXPTHEMESTYLE environment variable was set to FALSE.

- Cosrun. Execution error in Cosmos in the Sql.Connect method when making an "attach" to non-existent ODBC connection, after having established a previous ODBC connection and having made the disconnection.

- Cosrun. The ShowFontDialog method did not correctly return the charset indicated in the second parameter.

- In the development environment sometimes the selection of multiple controls with the option "Net Select" did not work.

- CTSQL. The LIMIT clause of the SELECT ... LIMIT .. INTO TEMP. Statement did not work.

- Cosrun. The ToChar method did not work correctly if the value of the first parameter was DAY, DY or D and the date corresponded to a Saturday (it always returned Sunday).

- Prnpag32.dll. The call to the function setGroupExportDataReport passing it as parameter "ALL" generated a Cosrun error..

- CTSQL. The "round" function of Ctsql does not work if you try to round more than 2 digits.

- Cosmos WebServer. If in the configuration file of the Cosmos WebServer application there was a blank line between the RESOURCEPATH variable and the [JAVA] section, the service would not start because the application did not load the file correctly.

- Cosrun. In the Grid control, when scrolling with the horizontal scroll, there came a moment when clicking on a cell the focus was not positioned in that cell, but in the previous column.

- Cosrun. When exporting the contents of a Grid control to Excel, if the cells had the bold font defined, this attribute was not exported in some cases.

- Development environment. When clicking on the code editor of a Form class the combination of keys [Control] + [T] on the name of a control, if the tab of the Tab control where it was defined was different from the current tab, the focus was positioned on the control but did not change the Tab control page.

- Sometimes, when using icons files or graphics in JPG and PNG format, the following error occurred: "Unable to load COSGRA32.DLL". The runtime has been modified to load the graphical DLL the first time it is used, minimizing the possibility of the error occurring.

- The GetProperty method of the Control class did not return any value if the Font property had not previously been assigned a value with the SetProperty method. It should show the default Font used by Cosmos.

- Cosrun. A general protection error occurred when some of the properties of the json file were null or "".

- Cosrun. The buttons with the Highlight border property were always painted with borders. In this version, if the control has this option checked, the border of this will not be painted until the mouse pointer is not positioned over it.

- The NumRows method did not work when the PostgreSQL ODBC was used.