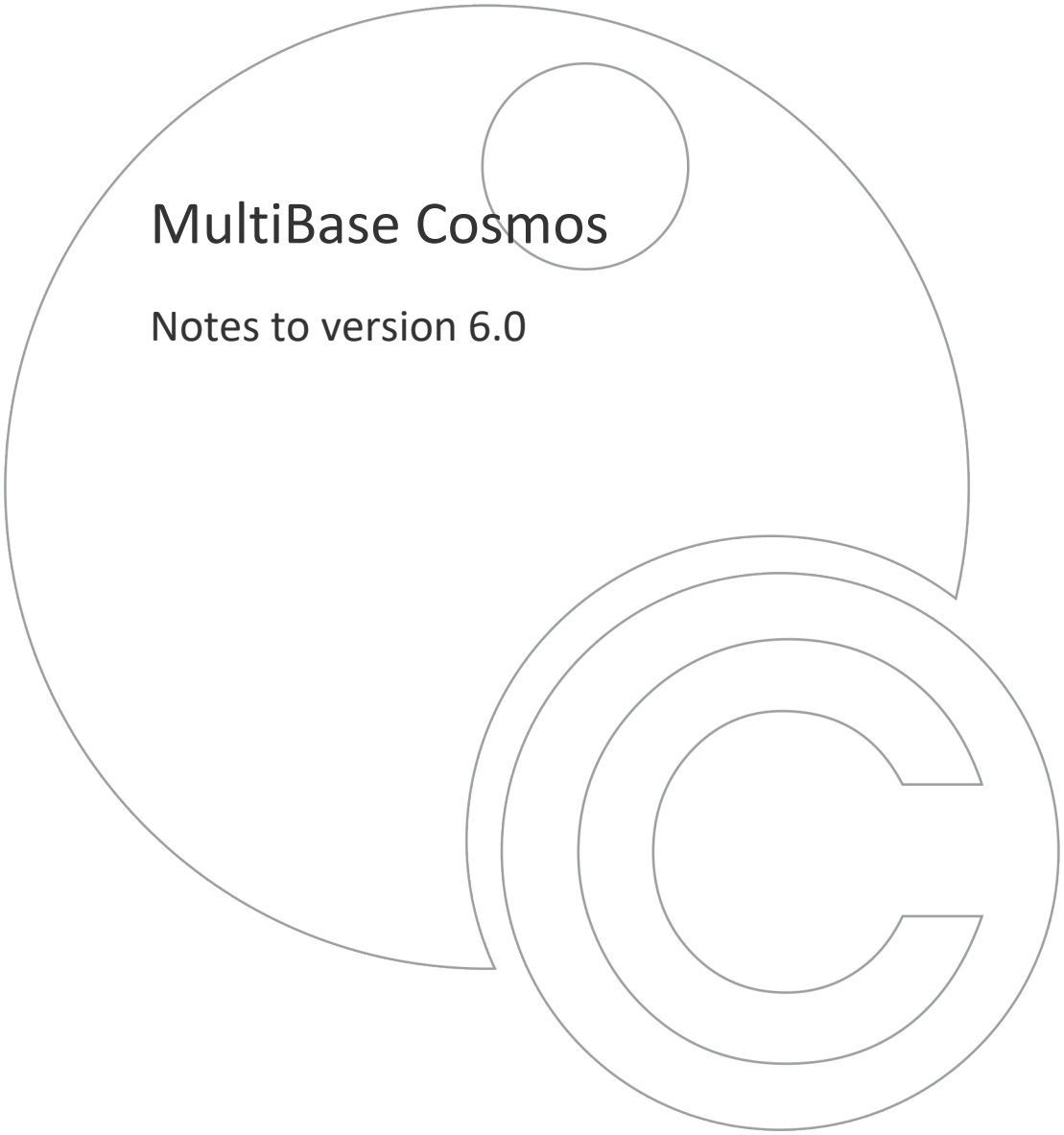
MultiBase Cosmos

Notes to version 6.0



BASE100

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

Index

[NTCO60v1.2en]

1. Implementations

- Cosmos as application server.
  From this version you can install a Cosmos application that supports REST requests from a Java client, Javascript, php, etc. Cosmos allows to define REST services in order to be executed by others applications via HTTP.
- New JSON class.
- CosmosData. In this version is distributed an application developed in Cosmos that allows to extract the data of a database in a comfortable and agile way for its later study and valuation. The queries can be done graphically from a repository. These queries can be saved for later use and/or modified if necessary.
- New library Cosregexpdll that allows you to search for text in files and in strings of chars using regular expressions.
- New method of Array class that allows to sort the array.
- Check box in tree lists.
  In order to use the check box in tree lists the next methods and events has been implemented in the SimpleControl class: SetListCheckable method, IsNodeChecked method, SetNodeChecked method , ListRowChecked event and ListRowUnchecked event.
- Events. New event On Click in List Box controls.
- The ability to export a list to Excel does not display the message asking the user to open the generated document. Implementation of the ASKWHENEXPORTS environment variable..
- Code Insight. Possibility to select the item in the list automatically by pressing the [Enter] key. Only allowed when the item is unique in the list..
- Code Insigth. Added a new option that allows you to choose the Code Insight search mode. In the "Settings" window of the development environment a "Deep search" checkbox has been added. If this box is checked, pressing the [.] (Dot) key will search for the token in the includes and libraries with which the module links (this will be the default operation mode). If it is not checked the box will look only in the current module.
- Development environment. In the "Find in files" option, when an item is selected from the list, the cursor will be positioned at the beginning of the search string..
- CTSQL. Implementation of derived tables in the FROM clause.

  Syntax:

  ```
  SELECT expression, ... FROM   t1,  (SELECT ...)
  ```

  A derived table in the FROM clause is a subselect that is placed after the FROM statement of the query.

  The derived table only exists in the query in which it is executed, that is, it is not part of the schema of the database.

Example:

```
select date_year, count(distinct customer) from (select year(date_invoice)
as date_year , customer  from invoices) group by date_year
```

NOTES:

a) Derived tables are not implemented as part of the fields of the SELECT statement.

b) The number of columns in the derived table must always be equal to or greater than the number of columns in the main SELECT.

c) Derived tables should be used when we want to nest aggregate functions, find the average of the quantities purchased, or filter the rows of the main table before a JOIN.

- CTSQL. The SQL function **count()** supports as a parameter a column name, in which case it returns the number of non-null records in the indicated column.

- CTSQL. Ability to use double quotes in SQL syntax to refer to identifiers (tables, columns). For example:

```
Select "customers"."description" from "customers"
```

To do this, you must define the environment variable ALLOWQUOTEDIDENTIFIERS = YES.

- CTSQL. Allow, within the **to_char** function, that the nlsparam parameter values support double quotes for the NLS_NUMERIC_CHARACTERS, NLS_ISO_CURRENCY, and NLS_CURRENCY variables.

- CSQL. Ability to execute the SQL file that receives as parameter the csql command without having to open it and press the execute option.

- CSQL. A keyboard accelerator has been added to execute the marked statement: [CTRL] + [ENTER].

2. Environment variables

- **ODBC_DISALLOW_FOR_UPDATE_ON_LOCK.** ODBC connection variable. This variable will need to be defined when modifying a record with the EditUpdate method; The SQL statement sent by Cosmos will not carry a "for update" clause to block the record.

- **ASKWHENEXPORTS.** If the ASKWHENEXPORTS variable is defined with FALSE value, the runtime will not ask for the user the and will not open the document after export the data with the ExportToExcel method.

- **ALLOWQUOTEDIDENTIFIERS.** This variable must be defined whenever double quotes are required to refer to column and/or table identifiers in SQL statements.

  It must be defined in the engine configuration file (ctsql.ini) or in the "Environment" section of the project configuration file or Cosmos configuration file.

  Los allowed values are YES and NO.

  If the value is YES, the identifiers of tables and columns can be indicated in double quotation marks.

  Example:

  ```
  Select "customers"."description" from "customers"
  ```

  NOTE: This implementation only allows you to use double quotes, never single quotes.

- **RUNCMDEXT**. Environment variable that tells the RunCmdExt command the command line to run.

3. JSON Class

This class allows the creation, loading, modification and query of JSON objects (JavaScript Object Notation).

JSON is a text format for data exchange in which you can store numbers, strings, boolean values, arrays and objects.

3.1 Methods

- **LoadFromFile.** Loads a JSON class object from a file.

  Suntax:

  ```
  LoadFromFile (fileName as Char) return Boolean
  ```

  Parameters:

fileName	Path of the JSON format file from where you want to load the JSON object.

  Returns:

TRUE	If the object of the JSON class has been successfully loaded from the file.
FALSE	If the object of the JSON class could not be successfully loaded from the file. This can happen if the file does not exist, has no read permissions, or contains any syntactic errors.

- **LoadFromChar.** Loads a JSON class object from a Char object or string.

  Syntax:

  ```
  LoadFromChar (string as Char) return Boolean
  ```

  Parameters:

string	String of characters with JSON text.

  Returns:

TRUE	If the JSON class object has been properly loaded from the Char object.
FALSE	If the method could not successfully load the JSON class object from the Char object. This can happen if it contains some syntactic error.

- **SaveToFile.** Saves an object of the JSON class to a file.

  Syntax:

  ```
  SaveToFile (fileName as Char ,format as Boolean default TRUE) return Boolean
  ```

  Parameters:

fileName	Path of the file in which the contents of the JSON object will be saved.

format	Boolean parameter that indicates whether the contents of the file will be saved formatted (TRUE) or unformatted (FALSE).

Returns:

TRUE	Whether the content of the JSON class object was saved successfully.
FALSE	The content of the JSON class object was not saved correctly.

- **Trace.** Displays a message box with the JSON object value.

  Syntax:

  ```
  Trace()
  ```

- **Set.** Adds a new property or modifies an existing Char, Numeric, Boolean or JSON class object.

  Syntax:

  ```
  Set(name as Char ,value as Object) return Boolean
  ```

  Parameters:

name	String of characters indicating the name and / or path of the property to be added or modified.
value	Value to be assigned to the JSON object.

  Returns:

TRUE	If the assignment was successful.
FALSE	The assignment could not be made. Possible causes: The value that is assigned is not of type CHAR, NUMERIC, BOOLEAN or JSON. The value of the name parameter does not match the name of any property.

- **SetValue.** Assigns a value to a JSON object. If the object of the JSON class already has a value assigned, it modifies it.

  Syntax:

  ```
  SetValue (value as Object) return Boolean
  ```

  Parameters:

value	Value to be assigned to the JSON object.

  Returns:

TRUE	If the assignment was successful.
FALSE	The assignment could not be made. This can occur, for example, if the value is not of type CHAR, NUMERIC, BOOLEAN or JSON.

- **Clear.** Cleans a JSON class object by deleting its children.

  Syntax:

  ```
  Clear () return Boolean
  ```

Returns:

TRUE	The execution of the method was successful.
FALSE	If the object of the JSON class is not of type OBJECT or type ARRAY.

- **Delete.** Removes a property from an object of the JSON class.

  Syntax:

  ```
  Delete (name as char) return Boolean
  ```

  Parameter:

name	String of characters indicating the name and / or path of the property to be deleted.

  Returns:

TRUE	The execution of the method was successful.
FALSE	If the property referred to does not exist.

- **SetAsArray.** Indicates that a JSON object is array type.

  Syntax:

  ```
  SetAsArray () return Boolean
  ```

  Returns:

TRUE	The execution of the method was successful.
FALSE	If the object is already an array or is an object type but already has child objects..

- **AddArrayElement**. Add a value or a JSON object to an object of the JSON class of array type.

  Syntax:

  ```
  AddArrayElement (element as Object) return Boolean
  ```

  Parameters:

element	CHAR, NUMERIC, BOOLEAN or JSON object to be added to the array.

  Returns:

TRUE	If the method has been able to add the element to the array.
FALSE	If the method could not add the element to the array. This can occur, for example, if the value is not of type CHAR, NUMERIC, BOOLEAN or JSON.

- **Get.** This method returns an object of the JSON class which is a copy of the object of the JSON class whose name or path has been passed as a parameter.

  Syntax:

  ```
  Get (name as Char) return JSON
  ```

Parameter:

name String of characters indicating the name and / or path of the property.

Returns:

A JSON object.

- **GetType.** This function returns a string with the name of the element type of the object or property of the JSON class.

  Syntax:

  ```
  GetType () return Char
  ```

  Returns:

  The name of the type of the item.

  The possible values that the method can return are:: object, string, array, number, boolean, property and unknown.

- **GetSize.** Returns the number of child elements of a JSON object of type Object or Array.

  Syntax:

  ```
  GetSize () return Integer
  ```

  Returns:

  The number of child elements.

- **GetString.** Returns a string with the object's representation of the JSON class.

  Syntax:

  ```
  GetString (format as Boolean) return Char
  ```

  Parameters:

  Format Boolean that indicates whether the return value will be formatted or not.

  Returns:

  String with the representation of the JSON object.

3.2 Operators

- public operator = (oJson as JSON)

This method allows you to assign an exact copy of the indicated JSON object to an object of the JSON class.

3.3 Conversors

- public conversor Char

The JSON class has converters to the Char class. This converter performs the same function as the GetString method, but always returns the string value of the JSON without formatting.

4. Methods

4.1 Array Class

- **Sort.** This method allows you to sort the elements of an array. The arrays that allow this action are integer, smallint, decimal, date, time, datetime, char, boolean and struct.

  Syntax:

  ```
  Sort (ascending as Boolean, fieldName as char default NULL)
  ```

  Parameters:

ascending	Indicates the type of order in which the sorting will take place. (TRUE-ascending or FALSE-descending).
fieldName	Identifier of the element of the structure to be ordered. This parameter is optional and should only be used if the array elements are structures. The field that is indicated can not be a structure, it must be of simple data type (Smallint, Integer, Decimal, Char, Time, Date or Bolean).

- **BinarySearch.** This method allows to perform a binary search of a previously ordered array element.

  Syntax:

  ```
  BinarySearch (value as Object ,ascending as Boolean ,VAR position as Integer
  ,fieldName as Char default NULL) return Object
  ```

  Parameters:

value	Value to search.
ascending	Indicates whether the array ordering is ascending or descending (TRUE-ascending or FALSE-descending).
position	Position in which the item was found. If there is no element that matches the search value, it returns 0.
fieldName	Identifier of the element of the structure you want to search for.

  Returns: The array.

4.2 Char Class

- **AnsiToUTF8.** This method converts a character string from ANSI to UTF8.

  Syntax:

  ```
  AnsiToUTF8 (codepage as Integer)
  ```

  Parameters:

Codepage	Integer indicating the page code in wich the source ANSI string is encoded.

- **UTF8ToAnsi.** This method converts a character string from UTF8 to ANSI.

  Syntax:

  ```
  UTF8ToAnsi (codepage as Integer)
  ```

  Parameters:

Codepage	Integer that indicates the code page in which the destination ANSI string is encoded.

4.3 Module Class

- **SetExecStatus.** This method allows assigning a value to a global internal variable of Cosmos.

  Syntax:

  ```
  SetExecStatus (status as Integer)
  ```

  Parameters:

status	Integer whose value you want to assign to the internal Cosmos variable.

- **GetExecStatus.** This method allows you to query the value assigned to the global internal variable of Cosmos using the SetExecStatus.

  Syntax:

  ```
  GetExecStatus() return Integer
  ```

4.4 SimpleControl Class

- **SetListCheckable.** This method allows you to display a check box on items in a tree list.

  Syntax:

  ```
  SetListCheckable(isCheckable as Boolean ,auto as Boolean default TRUE)
  ```

  Parameters:

isCheckable	Allowed values: TRUE and FALSE. If the value is TRUE a check box will be displayed on each node of the tree.
auto	Depending on the value of this parameter, every time a node is checked / unchecked this action will affect its predecessor and descendant nodes. Allowed values: TRUE y FALSE. If the value is TRUE, when you check/uncheck a box of a node will be checked/unchecked all the boxes of its predecessor and descendant nodes.

- **SetNodeChecked.** This method allows you to modify the check box status of a node in a tree list.

  Syntax:

  ```
  SetNodeChecked(node as Integer ,setCheck as Boolean)
  ```

Parameters:

node	Identifier of the node from which to modify the state.
setCheck	The checkbox will be marked or unmarked depending on the value of this parameter. Allowed values: TRUE y FALSE. If the value is TRUE the box will be checked. If the value is FALSE the box will be unchecked.

- **IsNodeChecked.** Allows you to check the status of the node's check box.

Syntax:

```
IsNodeChecked (node as Integer) return Boolean
```

Parameters:

node	Identifier of the node from which you want to check its status.

Returns:

TRUE if the box is checked and FALSE if it is not checked.

5. Commands

- **RunCmdExt.** Command of Form Class. It allows invoking, from a form (FormTable), a command indicated in the variable RUNCMDEXT.

6. Events

The events implemented in version 6.0 of Cosmos are as follows:

- **ListRowChecked.** This event will be thrown when the check box for a node in a tree list is checked.
- **ListRowUnchecked.** This event will be thrown when you uncheck the check box for a node in a tree list.
- **Click** in List Box controls.

7. Libraries

7.1 Cosregexpdll

This new dll allows to search text patterns in strings or in files using regular expressions.

7.1.1 CosNewRegExpr

This function allows to search text patterns in strings or in files using regular expressions.

This function will return the identifier of the search that will later be used in the rest of functions of this library.

Syntax:

```
public dll "cosregexpdll.dll" CosNewRegExpr(regularExpression as char, str as
char, isFile as Boolean, caseSensitive as boolean) return integer
```

Parameters:

regularExpression	Regular expression to be used as a search pattern.
str	String of characters or path of the file where the search will be made.
isFile	Boolean. If its value is TRUE, it indicates that the second parameter (str) is the name of a file within which the search will be performed. If its value is FALSE, it indicates that the second parameter is a text within which the search will be performed.
caseSensitive	Boolean. Indicates whether or not the search will be case-sensitive.

Returns: The search id:

-1.	If an error occurs in the search..
> 0.	If the function finds a match of the regular expression in the character string or in the file.. This value will be used as the identifier of the search in the rest of the functions of the library.

7.1.2 CosGetNumMatchesRegExpr

This function returns the number of matches of the regular expression found in the character string or in the file.

Syntax:

```
public dll "cosregexpdll.dll" CosGetNumMatchesRegExpr(regExprId as integer)
return integer
```

Parameters:

regExprId	Regular expression identifier. The value of this parameter is the return value of the CosNewRegExpr function.

Returns:

-1.	When there is no match of the regular expression in the search string or in case of error in the function.
>= 0.	Number of matches of the regular expression..

7.1.3 CosGetMatchRegExpr

This function returns the nth match of the regular expression in the text indicated in the search. This function gives the initial position in which the match within the chain has been found.

Syntax:

```
public dll "cosregexpdll.dll" CosGetMatchRegExpr (regExprId as integer,
numMatch as integer, VAR start as integer, VAR len as integer) return integer
```

Parameters:

regExprId	Regular expression identifier. The value of this parameter is the return value of the CosNewRegExpr function.
numMatch	Match number of the regular expression to be queried.
Start	Integer by reference where the position of the text where the match is found is returned.
len	Integer by reference where the length of the match is returned.

Returns:

-1	If there is no match of the regular expression in the search string or in case of error in the function.
0	Whenever there is a match.

If you want to know when the last match was reached, this will be the value to be queried.

7.1.4 CosFreeRegExpr

This function frees memory from the search of the regular expression made with the call to the function CosNewRegExpr.

Syntax:

```
public dll "cosregexpdll.dll" CosFreeRegExpr(regExprId as integer) return in-
teger
```

Parameters:

regExprId	Regular expression identifier. The value of this parameter is the return value of the CosNewRegExpr function.

Returns:

-1	When regExprId does not match an active search (return value of CosNewRegExpr).
0	When you have successfully released the regular expression from memory.

8. Csql

Cosmos version 6.0 you can execute an SQL file from the command line using the **csql** command.

Syntax:

```
csql [fichero] [-ini inifile] [-runscript –connection <connection name> -
database <database name>]
```

New parameters:

-runscript	Run the sql script without waiting for the user to press the execute button.
-connection <connection>	Required parameter that opens the indicated connection. This connection must exist in the configuration file that is passed in the "-ini" parameter.
-database <database name>	Required parameter that indicates the name of the database to be used.

NOTES:

a) You can erase and create databases.

b) In case of an error, the execution will stop, the error will be displayed on the screen to the user, the CSQL program will be opened and the cursor will be positioned on the line of the file that caused the error.

c) Opens the database that is indicated in the "-database" parameter regardless of the value indicated in the DBNAME environment variable.

9. Bug Fixes

- Code Insight. Sometimes it took a long time to show the list of methods, giving the impression that the execution was stopped.
- Cosrun. The controls that were in the non-visible part of a Grid or a Box with scroll were not the right size.
- Cosrun. If the native SQL SERVER driver 11.0 was used, it failed to modify a record when using the EditUpdate method (see section Environment Variables).
- Cosrun. The UnloadToXml method did not properly download the data when there were tildes or eñes (ñ) in the labels of the columns.
- Cosrun. Sorting in String list with autosort property in Date or numeric columns and null values was not correct.
- Cosrun. In a Variable control of the Page class with the multi line, auto wrap and vertical Text properties, the lines were overlaid when the report was generated.
- Cosrun. In the String lists, the ShowListFilterBar method did not filter correctly when the value of some of the fields in the list was null.
- The NumRows method of the Form Table class does not return the correct number of records read when the runtime was connected to an Oracle database via ODBC.
- Cosrun. The ResetListCellStyles method of the Simplecontrol class did not work correctly on the Tree Type List controls.
- Cosrun. The GetData method of the Winsock dll ActiveX did not work correctly, one of the parameters of this method did not return the correct value.
- Cosrun. Screen refresh problem on the Debug screen when using some form control ActiveX.
- Cosrun. When in the "On Enter" event of a control the SetFocus method was called to change the focus to another control, the "on Exit" event of the control that had the focus was executed twice.
- Prnpag32.Dll. A memory error occurred if the getPropStr function was executed and the control had not label value.
- CTSQL. A select from a view with aggregates returned null values.
- CTSQL. General protection error when executing the to_char function of a number without decimals with mask C99G999G999G999D99pr.