# MultiBase Cosmos

## Notas a la versión 4.2

BASE100

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

## Índice





[NT_CO_4.2_v1]

# 1 Nuevos métodos

## 1.1 Clase ActiveX

**GetObject**

Este método se utiliza para obtener un objeto ActiveX, hijo del control ActiveX actual, cuyo nombre se pasa como parámetro. Esto hace que no sea necesario declarar y crear un objeto ActiveX por cada nivel que tenga el objeto ActiveX padre. No funciona para objetos indexados.

Sintaxis:

```
GetObject(activexname as Char) return Activex
```

Parámetros:

```
Activexname: Indica el nombre del Actives.
```

Retorna:

```
Objeto ActiveX.
```

## 1.2 Clase SimpleControl

**IsColumnVisible**

Devuelve TRUE si la columna especificada de una lista de columnas es visible.

Sintaxis:

```
IsColumnVisible(column as Smallint) return Boolean
```

Parámetros:

| | |
|---|---|
| Column | Indica la posición de la columna dentro de la lista. |

Retorna:

TRUE si la columna indicada como parámetro es visible.

FALSE si no lo es.

**SetColumnVisible**

Muestra u oculta la columna especificada de una lista de columnas.

Sintaxis:

```
SetColumnVisible(column as Integer, visible as boolean)
```

Parámetros:

| | |
|---|---|
| Column | Indica la posición de la columna que queremos ocultar o mostrar. |
| Visible | Sus valores son TRUE o FALSE, dependiendo de si lo que se desea es visualizar u ocultar la columna. |

**GetType**
Devuelve un *string* indicando el tipo de un control:

| | | | | |
|---|---|---|---|---|
| WINDOW | BUTTON | RADIO | CHECK | DROPLIST |
| DROPEDIT | EDIT | TEXT | BOX | LINE |
| PORC | TAB | BUTTONLIST | MBLIST | BOXLIST |
| BAR | PANEL | LINES | USER | MENU |
| MBGRID | SPLIT | SPIN | SLIDER | BANDGROUP |
| ACTIVEX | MDICLIENT | CALENDAR | BITMAP | |

Sintaxis:

```
GetType () return Char
```

Retorna:

Nombre del control.

**ShowAllColumns**
Muestra todas las columnas de una lista de columnas.

Sintaxis:

```
ShowAllColumns()
```

**GetCurrentItem**
Devuelve el ítem de la columna en curso. Para controles tipo lista de columnas y listas de botones y cajas.

Sintaxis:

```
GetCurrentItem() return Smallint
```

Retorna:

El elemento en curso.

**SetOrderBy**
Permite ordenar una lista de columnas por varias columnas de forma ascendente o descendente. Se especifica tal y como se haría en la cláusula ORDER BY de una sentencia SQL utilizando el número de las columnas. (Por ejemplo: “1 desc,3,5 desc”).

Sintaxis:

```
SetOrderBy(orderby as char)
```

Parámetros:

Orderby Ordenación que queremos realizar.

**GetOrderBy**
Devuelve una cadena con el orden definido en una lista de columnas. (Por ejemplo: “1 desc,3,5 desc”).

Sintaxis:

```
GetOderBy() return Char
```

Retorna:

La ordenación utilizada en el método SetOrderBy.

**GetColumnLabel**
Devuelve el título de una columna de un control lista de columnas.

Sintaxis:

```
GetColumnLabel(column as Smallint) return Char
```

Parámetros:

| | |
|---|---|
| Column | Índice de la columna de la que se quiere mostrar el título. |

Retorna:

Título de la columna.

**GetColumnName**
Devuelve el nombre de una columna de un control lista de columnas de tipo SQL.

Sintaxis:

```
GetColumnName(column as Smallint) return Char
```

Parámetros:

| | |
|---|---|
| Column | Índice de la columna de la que se quiere mostrar el nombre. |

Retorna:

Nombre de la columna.

**AddColumnFilter**
Añade un filtro a una lista de columnas por la columna, operador y valor especificados. El filtro se añade a los que pudiera haber con anterioridad.

Sintaxis:

```
AddColumnFilter(column as Integer, operador as Char, values as Char, refresh
as Boolean default TRUE)
```

Parámetros:

| | |
|---|---|
| Column | Índice de la columna. |
| Operador | Operador que se va a utilizar en el filtro.<br>Los operadores validos para las listas tipo SQL son: =, <, <=, >, >=, <>, matches, like.<br>Los operadores válidos para las listas tipo *string* son: =, <, <=, >, >=, <>.<br>Hay que tener en cuenta que los operadores serán validados por el gestor de base de datos al que estemos accediendo.<br>Values: Indica los valores por los que queremos filtrar.<br>Refresh: TRUE si queremos que refresque la lista y FALSE si no queremos que lo haga. |

**ResetColumnFilter**
"Resetea" los filtros de la columna especificada de una lista de columnas.

Sintaxis:

```
ResetColumnFilter(column as Smallint)
```

Parámetros:

| | |
|---|---|
| Column | Índice de la columna. |

**ResetFilters**

"Resetea" todos los filtros de columnas de una lista de columnas.

Sintaxis:

```
ResetFilter()
```

**SetColumnWidth**

Permite modificar el ancho de una columna en píxeles de un control lista de columnas.

Sintaxis:

```
SetColumnWidth(column as Smallint, values as Smallint)
```

Parámetros:

| | |
|---|---|
| Column | Índice de la columna. |
| Values | Tamaño en píxeles que se le quiere dar a la columna. |

**GetColumnWidth**

Permite obtener el ancho de una columna de un control lista de columnas.

Sintaxis:

```
GetColumnWidth(column as Smallint) return Smallint
```

Parámetros:

| | |
|---|---|
| Column | Índice de la columna. |

Retorna:

Ancho en píxeles de la columna que recibe como parámetro

**SetColumnPosition**

Permite modificar la posición de una columna de un control lista de columnas.

La posición de display de una columna no modifica el número de la columna, de modo que para acceder a cualquier información de una columna se debe utilizar el número de columna original.

Sintaxis:

```
SetColumnPosition (column as Smallint, values as Smallint)
```

Parámetros:

| | |
|---|---|
| Column | Índice de la columna. |
| Values | Indica el índice de la columna donde se va a mostrar la columna. |

**GetColumnPosition**

Permite obtener la posición de display de una columna de un control lista de columnas.

Sintaxis:

```
GetColumnPosition (column as Smallint) return Smallint
```

Parámetros:

| | |
|---|---|
| Column | Índice de la columna. |

Retorna:

Posición en que se mostrará la columna.

**GetColumnSum**

Permite obtener el sumatorio de los valores de una columna de un control lista de columnas. [En las listas de tipo SQL se ejecuta internamente una Select Sum().] El valor de la suma lo devuelve en el parámetro pasado por referencia.. Sólo para controles lista de columnas.

Sintaxis:

```
GetColumnSum (column as Smallint, VAR sum as Object) return Boolean
```

Parámetros:

| | |
|---|---|
| Column | Índice de la columna. No permite expresiones. |
| Sum | Objeto en el que se obtiene el resultado de la suma. |

Retorna:

TRUE siempre que no se produzca un error.

FALSE si se produce un error.

NULL si al objeto que se aplica es nulo.

**IsNodeOpen**

Recibe como parámetro el índice del nodo y devuelve si el nodo está abierto o no. Sólo para controles listas en árbol.

Sintaxis:

```
IsNodeOpen (index as Integer) return Boolean.
```

Parámetros:

| | |
|---|---|
| Index | Índice del nodo. |

Retorna:

TRUE si el nodo está abierto.

FALSE si está cerrado.

NULL en caso de error.

**CollapseNode**

Cierra el nodo especificado de una lista en árbol. Sólo para controles lista en árbol.

Sintaxis:

```
CollapseNode (index as Integer, recursive as Boolean default FALSE) return
Boolean
```

Parámetros:

| | |
|---|---|
| Index | Índice del nodo que se quiere cerrar. |
| Recursive | TRUE si se quieren cerrar todos sus hijos recursivamente y FALSE en caso contrario. |

Retorna:

TRUE en caso de haber cerrado el nodo.

FALSE si el nodo ya está cerrado.

NULL en caso de error.

**ExpandNode**

Abre el nodo especificado de una lista en árbol. Recibe como parámetro el índice del nodo que se quiere abrir y si se quiere que se abran todos sus hijos recursivamente o no. Sólo para controles lista en árbol.

Sintaxis:

```
ExpandNode (index as Integer, recursive as Boolean default FALSE) return Boolean
```

Parámetros:

| | |
|---|---|
| Index | índice del nodo que se quiere abrir. |
| Recursive | TRUE si se quieren abrir todos sus hijos recursivamente y FALSE en caso contrario. |

Retorna:

TRUE si abre el nodo.

FALSE si el nodo ya estaba abierto.

NULL en caso de error.

**SetTabPageIcon**

Permite cambiar el icono de una página de un *tab control* en tiempo de ejecución. Sólo para controles tab.

Sintaxis:

```
SetTabPageIcon (page as Smallint, icon as Smallint) return Boolean
```

Parámetros:

| | |
|---|---|
| page | Índice de la página sobre la que se quiere cambiar el icono. |
| Icon | Índice del icono. |

Retorna:

TRUE si la operación se ha realizado correctamente, FALSE en caso contrario.

## 1.3 Clase Form

**SetMinTrackSize**

Establece el tamaño mínimo de un Form.

Sintaxis:

```
SetMinTrackSize (xSize as Integer, ySize as Integer) return Boolean
```

Parámetros:

| | |
|---|---|
| xSize | Ancho de la ventana en píxeles. |
| ySize | Alto de la ventana en píxeles. |

Retorna:

TRUE si no hay error y FALSE si ha habido alguno.

**SetMaxTrackSize**

Establece el tamaño máximo de un Form.

Sintaxis:

```
SetMaxTrackSize (xSize as Integer, ySize as Integer) return Boolean
```

Parámetros:

| | |
|---|---|
| xSize | Ancho de la ventana en píxeles. |
| ySize | Alto de la ventana en píxeles. |

Retorna:

TRUE si no hay error y FALSE si ha habido alguno.

## 1.4 Clase SqlCursor

**GetDataToFile**

Guarda en el fichero indicado en “fileName” la columna de tipo BLOB o CLOB en una conexión ODBC contra la base de datos Oracle.

Sintaxis:

```
GetDataToFile (indCol as Smallint, fileName as Char) return Boolean
```

Parámetros:

| | |
|---|---|
| indCol | Posición de la columna dentro de la lista de columnas indicadas en la Select. |
| fileName | Nombre del fichero. |

Retorna:

TRUE si la operación se ha realizado correctamente, FALSE en caso de error.

## 1.5 Clase SqlServer

**LoadFrom**

Permite cargar una tabla desde un fichero “unl” a partir una línea determinada del fichero.

Es igual al método Load, pero con un parámetro más en el que se le puede indicar el número de línea del fichero “unl” desde el que se desea cargar.

Sintaxis:

```
LoadFrom (file as Char, table as Char, columnlist as Char default NULL, delim
as Char default NULL, quoted as Boolean default NULL, oem as Boolean default
FALSE, line as
interger) return integer
```

Parámetros:

| | |
|---|---|
| file | Fichero ASCII desde el que se va realizar la carga de los datos. Ruta completa. |
| tabla | Nombre de la tabla sobre la que se va a realizar la carga. |
| Columnlist | Lista de columnas, separada por comas, de la tabla que se desean cargar. |
| Delim | Carácter utilizado como separador de campos en el fichero ASCII. |
| Quoted | Indica si los campos de tipo Char en el fichero ASCII van entrecomillados o no. Si es NULL el valor que toma es el definido en la variable de entorno DBQUOTED. Si ésta no está definida su valor por defecto será FALSE (sin comillas). |
| Oem | Si es TRUE carga el fichero con el conjunto de Caracteres OEM, y si es FALSE lo carga en ANSI. |
| Line | Línea desde la que se quiere iniciar la carga. |

Retorna:

Número de filas que se han cargado.

## 1.6 Clase Event

**GetControl**

Devuelve el objeto de la *SimpleFormControl.*

Sintaxis:

```
GetControl () return SimpleFormControl
```

Retorna:

Un objeto de la clase SimpleFormControl.

## 1.7 Clase Module

**SetHeaderContextMenu**

Permite definir un menú de contexto para la cabecera de un tipo de control, como por ejemplo una lista de columnas. Se ejecutará al pulsar el botón derecho del ratón. El menú especificado permanecerá asignado durante la ejecución de la aplicación. Para desactivarlo habrá que ejecutar este método con el argumento menú a nulo.

Sintaxis:

```
SetHeaderContextMenu(controlType as Char, VAR aMenu as Menu)
```

Parámetros:

| | |
|---|---|
| controlType | *String* con el tipo de control al que se va asociar el menú. Sus posibles valores son: “LIST” y “GRID”. |

aMenu Objeto menú.

**SetContextMenu**

Permite definir un menú de contexto para un tipo de control que se ejecutará al pulsar el botón derecho del ratón. El menú especificado permanecerá asignado durante la ejecución de la aplicación. Para desactivarlo habrá que ejecutar este método con el argumento menú a nulo.

Sintaxis:

```
SetContextMenu(controlType as Char, VAR aMenu as Menu)
```

Parámetros:

controlType *String* con el tipo de control al que se va asociar el menú. Los posibles valores de este parámetro son:

| | | | | |
|---|---|---|---|---|
| BUTTON | RADIO | CHECK | DROPLIST | DROPEDIT |
| EDIT | TEXT | BOX | LINE | PERCENT |
| TAB | LIST | MENU | GRID | BOXGROUP |
| PANEL | LINES | USER | BAR | BUTTONGROUP |
| SPLIT | SPIN | SLIDER | ACTIVES | MDICLIENT |
| CALENDAR | | | | |

aMenu Objeto menú.

## 1.8 Clase FormTable

**GetColumnSum**

Permite obtener el sumatorio de la columna especificada para las filas de la lista en curso del FormTable. Se ejecuta internamente un Select Sum().

Sintaxis:

```
GetColumnSum(sqlExpresion as Char, VAR sum as Object)
```

Parámetros:

SqlExpresion Columna o expresión SQL.

Sum Variable por referencia donde se almacenará el valor de la suma.

Ejemplo:

```
articulos.GetColumnSum("pr_vent", Suma)
articulos.GetColumnSum("pr_vent - pr_cost", Suma)
```

**GetWhere**

Devuelve una cadena de caracteres con las actuales condiciones de búsqueda del FormTable.

Sintaxis:

```
GetWhere() return Char
```

# 2 Nuevos comandos

## 2.1 Clase Form

Para aplicar al control en curso (el que tiene el foco, por ello, sólo se utilizan estos comandos desde el menú de contexto):

**HideColumn**
Se aplica a un control lista de columnas. La implementación por defecto oculta la columna en curso del control lista de columnas que tiene el foco. Equivale a ejecutar el método SetColumnVisible de la lista con el parámetro a falso.

**ShowAllColumns**
Se aplica a un control lista de columnas. La implementación por defecto muestra todas las columnas de la lista que estuvieran ocultas. Equivale a ejecutar el método *ShowAllColumns* de la lista.

**ResetColumns**
Se aplica a un control lista de columnas. La implementación por defecto "resetea" los atributos de las columnas (alineamiento, ancho, tipo, orden máscara, colores de fondo y texto, filtro y estado visible u oculto) al valor por defecto. Equivale a ejecutar el método *ResetListColumns* de la lista.

**AscendingSort**
Se aplica a un control lista de columnas. La implementación por defecto ordena la lista de forma ascendente por la columna en curso.

**DescendingSort**
Se aplica a un control lista de columnas. La implementación por defecto ordena la lista de forma descendente por la columna en curso.

**FilterByColumn**
Se aplica a un control lista de columnas. La implementación por defecto filtra la lista por el valor de la columna en curso. El filtro se añade a los que pudiera haber con anterioridad.

**ResetFilters**
Se aplica a un control lista de columnas. La implementación por defecto elimina los filtros sobre las columnas.

# 3 Entorno de desarrollo

- Posibilidad de Comentar/Descomentar bloques de código utilizando la combinación de teclas [Ctrl + /].
- Code Insight opcional. Se puede desactivar desde la pestaña “Editor” del menú “Tools-Settings”.

# 4 Implementaciones

- Tooltips. Antes sólo existían en los controles Push Button, Text, Button Group y Box Group, mostrando el texto de Label cuando estaba marcado con el flag de No Label. Ahora se extiende al texto de Comments si lo tiene, y a los controles Radio Button, Check Box, Spin y Slider.

- Iconos desde ficheros ICO y GIF → 1 fichero = 1 icono.
  En la sección Icons del "cosmos.ini" o del fichero "ini" del proyecto, se define la entrada del fichero de iconos, de igual manera que se define un fichero de iconos tradicional de Cosmos. La única diferencia es que el fichero llevará extensión **.lst** en lugar de **.bmp**, y será un fichero de texto. Cada línea del fichero **.lst** tendrá el *path* absoluto del fichero **.ico** o **.gif**. También se puede definir con un *path* relativo al fichero **.lst**.

  Ejemplo:

  La entrada en el fichero "ini" del proyecto, en la sección Icons:

  ```
  [icons]
  lista128=.\iconos.lst, 128, 128, RGB(255,0,255)
  ```

  indica que para el fichero de iconos **lista128** se utilizará el fichero **iconos.lst** que está situado en el mismo directorio que el proyecto.

  Contenido del fichero de iconos "iconos.lst":

  ```
  iconos\ABSTR08A.ICO
  iconos\ACAD0A.ICO
  iconos\ACAD0B.ICO
  iconos\ACAD0C.ICO
  iconos\ACAD0D.ICO
  iconos\ACAD1.ICO
  iconos\ACAD4.ICO
  iconos\ALPHA5.ICO
  iconos\AMI.ICO
  iconos\AMI0A.ICO
  iconos\AMI0B.ICO
  iconos\AMI1.ICO
  iconos\AMI1A.ICO
  iconos\AMI1B.ICO
  iconos\AMI1C.ICO
  iconos\Apple 2.ico
  iconos\Apple Emblem.ico
  iconos\Apple Logo.ico
  iconos\BMW.ICO
  iconos\BeQuickTime.ico
  iconos\Big Apple.ico
  ```

  Estos iconos no pueden guardarse desde el editor de ficheros de iconos de Cosmos.

- Estilo visual. A partir de esta versión, si Cosmos se ejecuta sobre Windows XP o Windows Server 2003 y el modo Tema está activado, Cosmos mostrará los controles con el aspecto de dicho tema. Por el contrario, si se ejecuta sobre un sistema operativo anterior a XP, o bien en XP o Windows Server 2003 pero el modo Tema no está activado, el aspecto visual será el clásico de Windows.

  Para mantener en Cosmos el aspecto clásico de Windows con el modo Tema activado en Windows XP o Windows Server 2003 bastará con definir la variable de entorno COSMOSXPTHEMESTYLE=FALSE en la sección "Environment" del fichero "Cosmos.ini" o del fichero "ini" de la aplicación. Su valor por defecto es TRUE.

  Existe una variable de entorno, NEWLOOKANDFEEL, que podrá tener los valores TRUE y FALSE. Al igual que COSMOSXPTHEMESTYLE, se podrá definir tanto en la sección "Environment" de "Cosmos.ini" como en la

sección “Environment” del fichero “ini” del proyecto. Si su valor es TRUE, mostrará los menús y los botones especiales (hilight border, no border, no label) con aspecto diferente al tema de Windows. El valor por defecto de estas variables de entorno es TRUE.

El resultado de las diferentes combinaciones de estas variables de entorno es el siguiente:

| COSMOSXPTHEMESTYLE | NEWLOOKANDFEEL | W98, W2K o XP/2003 sin temas | XP/2003 con temas |
|---|---|---|---|
| TRUE | TRUE | Botones especiales y menús con nuevo aspecto. Resto de controles, aspecto Windows tradicional. | Botones especiales y menús con nuevo aspecto. Resto de controles con aspecto acorde al tema utilizado. |
| TRUE | FALSE | Todos los controles con aspecto Windows tradicional. | Todos los controles con aspecto acorde al tema utilizado. |
| FALSE | TRUE | Botones especiales y menús con nuevo aspecto. Resto de controles, aspecto Windows tradicional. | Botones especiales y menús con nuevo aspecto. Resto de controles, aspecto Windows tradicional. |
| FALSE | FALSE | Todos los controles con aspecto Windows tradicional. | Todos los controles con aspecto Windows tradicional. |

WINADJUSTHEIGHT. Se define en la sección “Environment” del “Cosmos.ini” o del fichero “ini” del proyecto. Su valor será el tamaño que debe aumentarse en los Forms Cosmos. Si no está definida, su valor es 0. Un valor recomendado es 8 para Forms creados en sistemas Windows 2K o Windows XP sin temas, y que se ejecutan en Windows XP con temas.

- Cosrep: Posibilidad de salvar y leer repositorios en formato “xml”. Al abrir un repositorio se podrá elegir el tipo de repositorio (“.crf” o “.xml”). Se guardará en el formato original, pero con la opción “SaveAs” se podrá salvar en otro formato.

  Se añaden las opciones “Collapse all” y “Expand all” en el menú de edición de las tablas de un repositorio, que permiten respectivamente cerrar y expandir la lista de manera recursiva, es decir, que se cerrarán o abrirán todos lo hijos del nodo que se marque.

- Cosmos. Se pueden importar repositorios en formato “xml” a un proyecto. Se trabaja con ellos de la misma forma que con los “crf”. Se distinguen por un icono diferente.

  Se añaden las opciones “Collapse all” y “Expand all” en los menús de contexto de la vista de un proyecto y de un repositorio.

  Se añade la opción “Find Table” al menú pop-up de la vista del repositorio. Permite buscar una tabla y posicionarse en ella.

- Se ha implementado la posibilidad de poder poner el texto de los controles *check box* y *radio button* a la izquierda.

- Controles Lista. En versiones anteriores de Cosmos, cuando el foco estaba situado en un elemento de una lista y se tecleaba el nombre del elemento, se situaba en el siguiente cuyo nombre comenzaba por el último carácter tecleado. Ahora mantiene un buffer, de tal manera que se sitúa en el siguiente elemento cuyo

nombre se corresponda con todos los caracteres teclados. Tiene un timeout de un segundo, tras el cual limpia el buffer.

- Cosrun. Si la Licencia no se ha registrado en un máquina, el *cosrun* funcionará, pero recordará la necesidad de registrar la Licencia cada 20 minutos con un "banner".
- El método NumRows del FormTable devuelve el número total de filas del FormTable aunque esté activa la variable QUERYBUFFERING y no se hayan leído todas las filas.
- En modo EditQueryLike no se tenían en cuenta los atributos Upper y Lower.
- Los métodos LoadSelect y Count devuelven el número total de filas, incluso si la lista es de tipo SQL con la variable de entorno CTRLISTSQLBUFFERING activada.
- Las listas de tipo árbol permiten multicolumnas.
- Se añaden las variables de entorno:

  DEACTIVATEDTEXTCOLOR: Sirve para redefinir el color de *foreground* con el que aparecen los campos de un FormTable cuando están desactivados.

  Ej.: miForm.SetOption("DEACTIVATEDTEXTCOLOR", RGB(128, 64,64))

  DEACTIVATEDPKCOLOR: Sirve para redefinir el color de *foreground* de los campos claves de un FormTable.

  BLOCKSHIFTCOLUMNS: Sirve para bloquear el movimiento dinámico de las columnas en los GRID o en las listas de columnas de forma global. El valor de la variable debe ser un Boolean (su valor por defecto es FALSE).

  En el caso de los Grid, el valor de esta propiedad para activarlo debe ser FALSE, mientras que allowShiftColumn debe ser TRUE. Esto hay que hacerlo por cada Grid.

  Ej.:  miForm.SetOption("BLOCKSHIFTCOLUMNS", TRUE)

  Se pueden modificar con los métodos SetOption de la clase Form o Module o definir como variables de entorno.

- Code Insight. Nueva implementación. Búsqueda de objetos dentro de *includes*.
- Control Grid. Se añade la posibilidad de ordenar automáticamente las filas de un *grid* asociado a un *formtable*. Para ello, se ha añadido una propiedad nueva en la edición de un control GRID de Cosmos, llamada "Automatic sort". Si esta propiedad está activada, en ejecución se reordenarán las filas del *grid* al pulsar con el botón izquierdo del ratón sobre la cabecera de una columna. **Sólo funcionará con las columnas del *grid* que estén asociadas a objetos del *formtable* de tipo "Is column".**

# 5 Eventos

- LButtonUp. El evento se produce al soltar el botón izquierdo del ratón.
- RButtonUp. El evento se produce al soltar el botón derecho del ratón.
- MouseMove. El evento se produce al mover el cursor del ratón por encima del control. Controles del tipo: BOX, TAB y BITMAP.
- MouseEnter. El evento se produce cuando el ratón entra en el área del control.
- MouseExit. El evento se produce cuando el ratón sale del área del control.
- NodeExpand. Se produce al abrir o cerrar un nodo de una lista en árbol.

LButtonUp, RButtonUp y MouseMove son para controles de tipo: BOX, TAB y BITMAP.

Los eventos MouseEnter y MouseExit son para los controles de tipo: BUTTON, RADIO, CHECK, DROPLIST, DROPEDIT, EDIT, TEXT, BOX, TAB, BUTTONLIST, MBLIST, BOXLIST, BITMAP, MBGRID, SPIN y SLIDER.

# 6 Propiedades

- Se añade la propiedad Selected en los controles de tipo DropEdit.
- Se añade la propiedad “AllowShiftColumn” a los controles lista para permitir mover las columnas o no, individualmente de cada lista.

  Por defecto su valor es True para las listas y False para los Grid.

  Para poder mover las columnas debe estar activa esta propiedad y la opción BLOCKSHIFTCOLUMNS debe ser FALSE.

# 7 Correcciones

- PRNPAG32.DLL
  No liberaba memoria cuando se utilizaban bitmaps.
  **Corregido, libera memoria.**

- Método Replace de la clase Char.
  No liberaba memoria.
  **Corregido, libera memoria.**

- Error al utilizar el opción Find String In Files del entorno de desarrollo.
  **Corregido.**

- Cuando un control estaba deshabilitado y no poseía un evento (DblClick en un Botón), lo recibía el control que estaba detrás de él (Box), en lugar de capturarlo y no procesar nada.
  **Corregido.**

- Mejora del *scroll* con rueda de ratón en listas y *grid*.

- Cajas con *scroll*.
  En ejecución, al cambiar el foco de un control dentro de una caja con *scroll* se hace *scroll* automáticamente para que el control con el foco esté siempre visible en su totalidad o lo máximo posible. Incluso si el control está dentro de otro contenedor.
  Se modifica el tamaño del marcador de la barra de *scroll* para que refleje el porcentaje de la caja que está visible.
  En diseño no admite el doble clic en el área de las barras de *scroll*.
  Se corrige un error de repintado de las cajas con *scroll* al redimensionar la ventana en ejecución.
  **Corregido.**

- Errores en ActiveX
  Al invocar un método de un ActiveX que retornaba un Variant de tipo ActiveX no devolvía bien el objeto.
  **Corregido.**

- Si se intentaba utilizar un control ActiveX que no estaba registrado en el sistema, se producía un error de protección general.
  **Corregido.**

- Code Insight. Si en el *main* se ponía Self seguido de un punto, la lista de métodos aparecía vacía.
  **Corregido.**

- Si a un control *Edit field* cuya variable (time) estaba asociada al campo de una tabla le marcábamos la propiedad Date/Timer Picker, a la hora de hacer una consulta con el comando EditQueryLike no devolvía nada.
  **Corregido.**

- En el *preview* no funcionaban las teclas [Av Pág] ni [Re Pág], ya que el foco lo tenía el visualizador del listado y no uno de los botones de la parte inferior.
  Al pulsar las teclas [Av Pág] y [Re Pág] se muestra la página siguiente o anterior del listado, según corresponda.

- Al ejecutar la instrucción UNFOLD sobre un control *Edit field* de tipo Fecha con la propiedad DateTimePicker, aparece el control calendario con el día en curso marcado, pero no marca la fecha que tiene el control *edit*.
  La fecha indicada en el control *edit* se marca correctamente en el control calendario.

- Los métodos TestFile, FileSize, Delete, Copy, Move y SetFileAttributes de la clase module no encontraban el fichero origen si el nombre contenía dos espacios seguidos.
  **Corregido.**

- No se ordenaba el primer nivel del árbol de un repositorio mediante la opción Sort del menú.
  **Corregido.**

- Error de conversión al utilizar el operador % con números decimales con el valor mayor del máximo Integer.
  **Corregido.**

- Cuando se registra una Licencia debemos validar primero si tenemos permisos de lectura/escritura sobre el registro.
  Se valida si tiene permisos de escritura en el registro de Windows. Para ejecutar un Cosmos licenciado ya no hace falta tener permisos de escritura.

- Si un Form tenía sólo campos clave de una tabla y la propiedad “RELOADONROWCHANGED”, al hacer Query, Next y Previous se producía el error: “Cannot perform methods Update and Delete in table whithout primary key”.
  **Corregido.**

- El método GetOption de las clases Form, FormTable y Module devuelve un Integer en lugar de un Smallint.
  **Modificado.**

- Code Insight. Si se pulsa “.” en el entorno de desarrollo en una línea de más de 256 caracteres devolvía una aserción inválida.
  **Corregido.**

- Code Insight. Búsqueda de objetos dentro de clases derivadas de SqlServer.
  Se mejora la búsqueda de estos objetos.

- Code Insight. Si el punto (.) se encuentra dentro de un área de comentario (// o {}), no actúa el Code Insight.
  **Corregido.**

- Editor. Aceleradores para comentar/descomentar líneas y bloques de código.
  Con CTRL. + / comentamos una línea.

- ActiveX. Visualización de ActiveX gráficos.
  **Se mejora.**