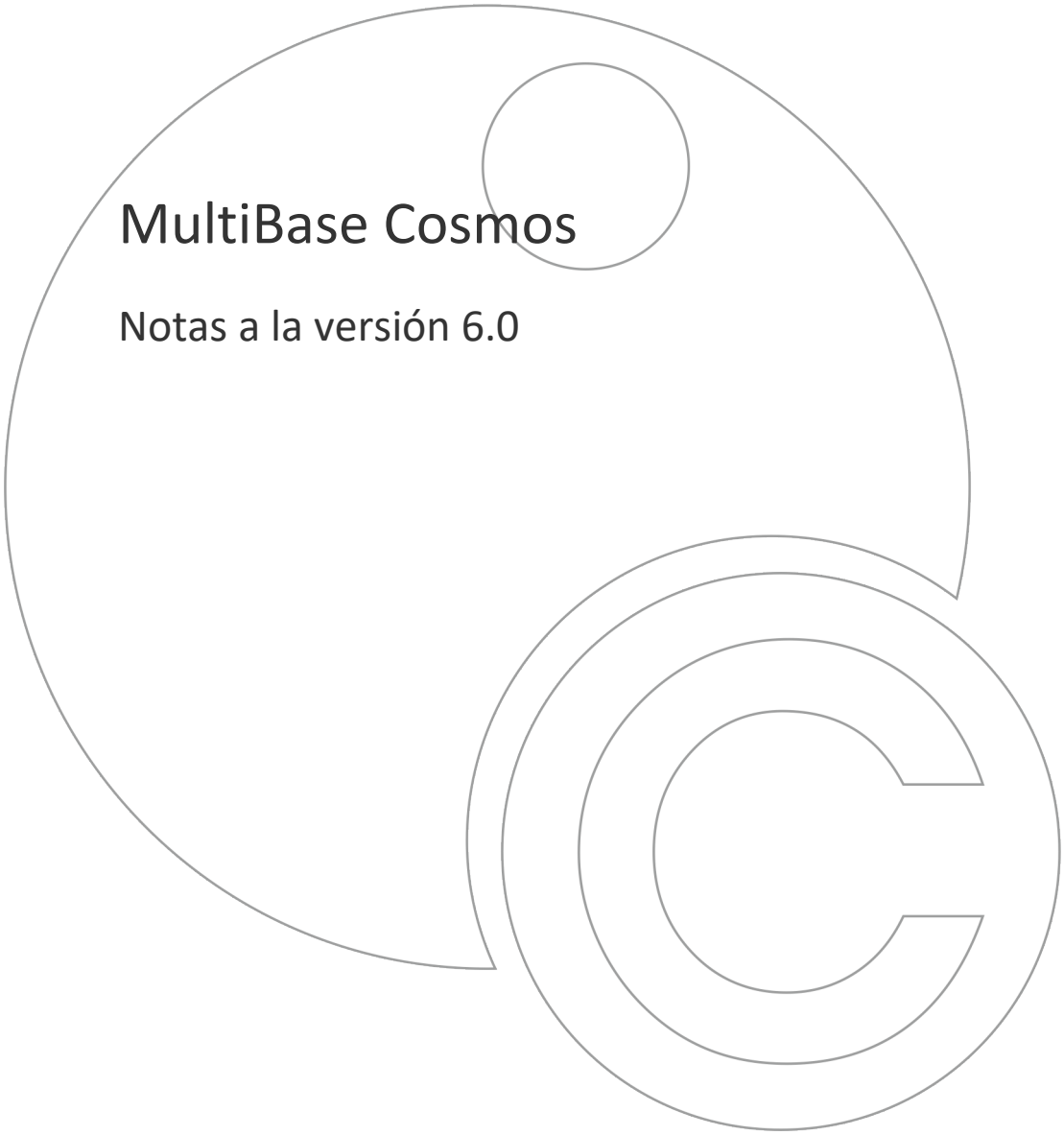
MultiBase Cosmos

Notas a la versión 6.0



BASE100

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

Índice

1. Implementaciones

- Cosmos como servidor de aplicaciones.
  A partir de esta versión se podrá instalar una aplicación de Cosmos que admita peticiones REST desde un cliente Java, Javascript, php, etc. Cosmos permitirá definir servicios REST para ser invocados desde otras aplicaciones vía HTTP.
- Nueva clase JSON.
- CosmosData. En esta versión se distribuye una aplicación desarrollada en Cosmos que permite extraer los datos de una base de datos de manera cómoda y ágil para su posterior estudio y valoración. Las consultas se podrán realizar de una manera gráfica a partir de un repositorio. Estas consultas se podrán guardar para utilizarlas posteriormente y/o modificarlas en caso de ser necesario.
- Nueva librería Cosregexpdll que permite buscar texto en archivos y en cadenas de caracteres empleando expresiones regulares.
- Método de la clase Array que permite ordenar un elemento.
- Casilla de verificación en las listas en árbol.
  Para poder manejar estas listas se han implementado los siguiente métodos y eventos de la clase SimpleControl: método SetListCheckable, método IsNodeChecked, método SetNodeChecked, evento ListRowChecked y evento ListRowUnchecked.
- Eventos. Se ha implementado el evento On Click en controles List Box.
- Posibilidad de que al exportar a Excel una lista no muestre el mensaje en el que se pregunta al usuario si desea abrir el documento generado. Implementación de la variable de entorno ASKWHENEXPORTS.
- Code Insigth. Posibilidad de seleccionar automáticamente el elemento de la lista al pulsar la tecla [Intro]. Solo se permite cuando el elemento sea único en la lista.
- Code Insigth. Se ha añadido una nueva opción que permite elegir el modo de búsqueda del Code Insight. En la ventana “Settings” del entorno de desarrollo se ha añadido una casilla de verificación “Deep search”. Si se marca esta casilla, al pulsar la tecla [.] (punto) buscará el token en los includes y librerías con los que enlaza el módulo (éste será el modo de funcionamiento por defecto). Si no está marcada la casilla buscará solamente en el módulo actual.
- Entorno de desarrollo. En la opción “Find in files”, cuando se seleccione un elemento de la lista el cursor se posicionará al comienzo de la cadena por la que se ha realizado la búsqueda.
- CTSQL. Implementación de tablas derivadas en la cláusula FROM.

  Sintaxis:

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

  Una tabla derivada en la cláusula FROM es una subselect que se sitúa después de la sentencia FROM de la query.

La tabla derivada solo existe en la query en la que es ejecutada, es decir, no forma parte del esquema de la base de datos.

Ejemplo:

```
select fecha_año, count(distinct cliente) from (select year(fecha_albaran)
as fecha_año , cliente  from albaranes) group by fecha_año
```

NOTAS:

a) No están implementadas las tablas derivadas como parte de los campos de la instrucción SELECT.

b) La cantidad de columnas de la tabla derivada siempre debe ser igual o mayor a la cantidad de columnas del SELECT principal.

c) Las tablas derivadas se deben emplear cuando queramos anidar funciones agregadas, hallar el promedio de las cantidades compradas, o filtrar las filas de la tabla principal antes de un JOIN.

- CTSQL. La función **count()** de SQL admite como parámetro un nombre de columna, en cuyo caso retornará el número de registros no nulos de la columna indicada.

- CTSQL. Posibilidad de emplear comillas dobles en la sintaxis SQL para referirse a identificadores (tablas, columnas). Por ejemplo:

```
Select "clientes"."descripcion" from "clientes"
```

Para ello habrá que definir la variable de entorno ALLOWQUOTEDIDENTIFIERS=YES.

- CTSQL. Permitir, dentro de la función **to_char,** que los valores del parámetro nlsparam admitan comillas dobles para las variables NLS_NUMERIC_CHARACTERS, NLS_ISO_CURRENCY y NLS_CURRENCY.

- CSQL. Posibilidad de ejecutar el fichero SQL que recibe como parámetro el comando **csql** sin tener que abrirlo y pulsar la opción ejecutar.

- CSQL. Se ha añadido un acelerador de teclado para poder ejecutar la sentencia marcada: [CTRL] + [ENTER].

2. Variables de entorno

- **ODBC_DISALLOW_FOR_UPDATE_ON_LOCK.** Variable de conexión ODBC. Esta variable será necesario definirla cuando se modifique un registro con el método EditUpdate; la sentencia SQL que envíe Cosmos no llevará cláusula “for update” para bloquear el registro.

- **ASKWHENEXPORTS.** Si se define ASKWHENEXPORTS=FALSE no preguntará al usuario ni abrirá el documento tras exportar los datos con el método ExportToExcel de la clase SimpleControl para listas y grid.

- **ALLOWQUOTEDIDENTIFIERS.** Esta variable debe definirse siempre que se precise utilizar comillas dobles para referirse a los identificadores de columna y/o tabla en las sentencias SQL.

  Debe definirse en el fichero de configuración del motor (ctsql.ini) o en la sección “Environment” del fichero de configuración de proyecto o de Cosmos.

  Los valores posibles son YES y NO.

  Si el valor es YES los identificadores de tablas y columnas podrán indicarse entre comillas dobles.

  Ejemplo:

  ```
  Select “clientes”.”descripcion” from “clientes”
  ```

  NOTA: Esta implementación solo permite utilizar comillas dobles, nunca comillas simples.

- **RUNCMDEXT**. Variable de entorno que indica al comando RunCmdExt la línea de comando que debe ejecutar.

3. Clase JSON

Esta clase permite la creación, carga, modificación y consulta de objetos JSON (JavaScript Object Notation).

JSON es un formato de texto para el intercambio de datos en el cual se pueden almacenar números, cadena de caracteres, valores booleanos, arrays y objetos.

3.1 Métodos

- **LoadFromFile.** Carga un objeto de la clase JSON desde un fichero.

  Sintaxis:

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

  Parámetros:

fileName	Ruta del fichero con formato JSON desde donde se desea cargar el objeto JSON.

  Retorna:

TRUE	Si el objeto de la clase JSON se ha cargado correctamente desde el archivo.
FALSE	Si el objeto de la clase JSON no se ha podido cargar correctamente desde el archivo. Esto puede ocurrir si no existe el archivo, no tiene permisos de lectura o si contiene algún error sintáctico.

- **LoadFromChar.** Carga un objeto de la clase JSON desde un objeto Char o una cadena de caracteres.

  Sintaxis:

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

  Parámetros:

string	Cadena de caracteres con el texto JSON.

  Retorna:

TRUE	Si se ha cargado correctamente el objeto de la clase JSON desde el objeto Char.
FALSE	Si no ha podido cargar correctamente el objeto de la clase JSON desde el Char. Esto puede ocurrir si contiene algún error sintáctico.

- **SaveToFile.** Guarda un objeto de la clase JSON en un fichero.

  Sintaxis:

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

  Parámetros:

fileName	Ruta del fichero en el que se guardará el contenido del objeto JSON.
format	Parámetro booleano que indica si el contenido del fichero se guardará formateado (TRUE) o sin formato (FALSE).

Retorna:

TRUE	Si el contenido del objeto de la clase JSON se guardó correctamente.
FALSE	El contenido del objeto de la clase JSON no se guardó correctamente.

- **Trace.** Devuelve en una ventana el valor que tiene asignado el objeto.

  Sintaxis:

  ```
  Trace()
  ```

- **Set.** Añade una nueva propiedad o modifica una existente de tipo Char, numérica, booleana o un objeto de la clase JSON.

  Sintaxis:

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

  Parámetros:

name	Cadena de caracteres que indica el nombre y/o ruta de la propiedad que se desea añadir o modificar.
value	Valor que se desea asignar al objeto JSON.

  Retorna:

TRUE	Si la asignación se realizó correctamente.
FALSE	No se pudo realizar la asignación. Posibles causas: El valor que se asigna no es de tipo CHAR, NUMERIC, BOOLEAN o JSON. El valor del parámetro name no coincide con el nombre de ninguna propiedad.

- **SetValue.** Asigna un valor a un objeto JSON. Si el objeto de la clase JSON ya tiene asignado un valor, lo modifica.

  Sintaxis:

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

  Parámetros:

value	Valor que se desea asignar al objeto JSON.

  Retorna:

TRUE	Si la asignación se realizó correctamente.
FALSE	No se pudo realizar la asignación. Esto puede ocurrir, por ejemplo, si el valor no es de tipo CHAR, NUMERIC, BOOLEAN o JSON

- **Clear.** Limpia un objeto de la clase JSON eliminando sus hijos.

  Sintaxis:

  ```
  Clear () return Boolean
  ```

Retorna:

TRUE	La operación del objeto de la clase JSON se realizó correctamente.
FALSE	Si el objeto de la clase JSON no es de tipo OBJECT o de tipo ARRAY.

- **Delete.** Elimina una propiedad de un objeto de la clase JSON.

  Sintaxis:

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

  Parámetro:

name	Cadena de caracteres que indica el nombre y/o ruta de la propiedad que se desea eliminar.

  Retorna:

TRUE	La operación del objeto de la clase JSON se realizó correctamente.
FALSE	Si no existe la propiedad a la que se hace referencia.

- **SetAsArray.** Indica que un objeto JSON es de tipo array.

  Sintaxis:

  ```
  SetAsArray () return Boolean
  ```

  Retorna:

TRUE	La asignación de tipo array se realizó correctamente.
FALSE	Si el objeto ya es un array o es tipo object pero ya tiene objetos hijos.

- **AddArrayElement**. Añade un valor o un objeto JSON a un JSON de tipo array.

  Sintaxis:

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

  Parámetros:

element	Objeto CHAR, NUMÉRICO o BOOLEANO u objeto JSON que se desea añadir al array.

  Retorna:

TRUE	Si ha podido añadir el elemento al array.
FALSE	Si no ha podido añadir el elemento al array. Esto puede ocurrir, por ejemplo, si el valor no es de tipo CHAR, NUMERICO, BOOLEAN o JSON.

- **Get.** Esta función retorna un objeto de la clase JSON que es una copia del objeto de la clase JSON cuyo nombre o ruta se ha pasado como parámetro.

  Sintaxis:

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

  Parámetro:

name	Cadena de caracteres que indica el nombre y/o ruta de la propiedad.

Retorna:

Un objeto JSON.

- **GetType.** Esta función retorna un string con el tipo de elemento del objeto o propiedad de la clase JSON.

Sintaxis:

```
GetType () return Char
```

Retorna:

El tipo del elemento.

Los valores posibles que puede retornar son: object, string, array, number, boolean, property y unknown.

- **GetSize.** Retorna el número de elementos hijos de un objeto JSON de tipo Object o Array.

Sintaxis:

```
GetSize () return Integer
```

Retorna:

El número de elementos.

- **GetString.** Retorna un String con la representación del objeto de la clase JSON.

Sintaxis:

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

Parámetros:

Format	Booleano que indica si el valor de retorno irá formateado o no.

Retorna:

String con la representación del objeto JSON.

3.2 Operadores

- public operator = (oJson as JSON)

Permite asignar a un objeto de la clase JSON una copia exacta del objeto JSON asignado.

3.3 Conversores

- public conversor Char

La clase JSON dispone de conversores a la clase Char. Este conversor realiza la misma función que el método GetString, pero siempre retorna el valor string del JSON sin formatear.

4. Métodos

4.1 Clase Array

- **Sort.** Permite ordenar los elementos de un array. Los arrays que permite esta acción son de tipo integer, smallint, decimal, date, time, datetime, char, boolean y struct.

  Sintaxis:

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

  Parámetros:

ascending	Indica el orden en que se efectuará la ordenación (TRUE-ascendente o FALSE-descendente).
fieldName	Identificador del elemento de la estructura por el que se desea ordenar. Este parámetro es opcional y solo debe utilizarse si los elementos del array son estructuras. El campo que se indica no puede ser una estructura, debe ser de tipo dato simple (Smallint, Integer, Decimal, Char, Time, Date o Bolean).

- **BinarySearch.** Permite realizar una búsqueda binaria de un elemento de un array previamente ordenado.

  Sintaxis:

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

  Parámetros:

value	Valor que se desea buscar.
ascending	Indica si la ordenación del array es ascendente o descendente (TRUE: ascendente o FALSE: descendente).
position	Posición en la que se ha encontrado el elemento. Si no existe ningún elemento que coincida con el valor de búsqueda retorna 0.
fieldName	Identificador del elemento de la estructura por el que se desea buscar.

  Retorna: El array.

4.2 Clase Char

- **AnsiToUTF8.** Este método convierte una cadena de caracteres de ANSI a UTF8.

  Sintaxis:

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

  Parámetros:

Codepage	Número entero que indica el código de página en la que está codificada la cadena ANSI origen.

- **UTF8ToAnsi.** Este método convierte una cadena de caracteres de UTF8 a ANSI.

  Sintaxis:

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

  Parámetros:

Codepage	Número entero que indica el código de página en que está codificada la cadena ANSI destino.

4.3 Clase Module

- **SetExecStatus.** Este método permite asignar un valor a una variable interna global de Cosmos.

  Sintaxis:

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

  Parámetros:

status	Número entero cuyo valor se desea asignar a la variable interna de Cosmos.

- **GetExecStatus.** Este método permite consultar el valor asignado a la variable interna global de Cosmos mediante el método SetExecStatus.

  Sintaxis:

  ```
  GetExecStatus() return Integer
  ```

4.4 Clase SimpleControl

- **SetListCheckable.** Este método permite mostrar una casilla de verificación en los elementos de una lista en árbol.

  Sintaxis:

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

  Parámetros:

isCheckable	Los posibles valores son: TRUE y FALSE. Si el valor es TRUE en cada nodo del árbol se mostrará una casilla de verificación.
auto	Dependiendo del valor de este parámetro, cada vez que se marque/desmarque la casilla de verificación de un nodo esta acción afectará o no a sus nodos antecesores y descendientes. Los posibles valores con: TRUE y FALSE. Si el valor es TRUE, cuando se marque/desmarque alguna casilla de un nodo se marcarán/desmarcarán todas las casillas de sus nodos antecesores y descendientes.

- **SetNodeChecked.** Permite modificar el estado de la casilla de verificación de un nodo de una lista en árbol.

  Sintaxis:

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

  Parámetros:

node	Identificador del nodo del que se desea modificar el estado.
setCheck	La casilla se mostrará marcada o desmarcada dependiendo del valor de este parámetro. Los posibles valores son: TRUE y FALSE. Si el valor es TRUE la casilla estará marcada. Si el valor es FALSE la casilla estará desmarcada.

- **IsNodeChecked.** Permite consultar si un nodo tiene marcada o desmarcada la casilla de verificación.

  Sintaxis:

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

  Parámetros:

node	Identificador del nodo sobre el que se quiere consultar su estado.

  Retorna:

  TRUE si la casilla está marcada y FALSE si no lo está.

5. Comandos

- **RunCmdExt.** Comando de la clase Form. Permite invocar, desde un formulario (FormTable), un comando indicado en la variable RUNCMDEXT.

6. Eventos

Los eventos implementados en la versión 6.0 de Cosmos son los siguientes:

- **ListRowChecked.** Este evento se lanzará cuando se marque la casilla de verificación de un nodo de una lista en árbol.
- **ListRowUnchecked.** Este evento se lanzará cuando se desmarque la casilla de verificación de un nodo de una lista en árbol.
- **Click** en los controles List Box.

7. Librerías

7.1 Cosregexpdll

Esta nueva dll permite realizar búsquedas de patrones en cadenas de caracteres o archivos utilizando expresiones regulares.

7.1.1 CosNewRegExpr

Esta nueva dll permite realizar búsquedas de patrones en cadenas de caracteres o archivos utilizando expresiones regulares.

Esta función retornará el identificador de la búsqueda que posteriormente se utilizará en el resto de funciones de esta librería.

Sintaxis:

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

Parámetros:

regularExpression	Expresión regular que se utilizará como patrón de búsqueda.
str	Cadena de caracteres o ruta del fichero donde se realizará la búsqueda.
isFile	Booleano. Si su valor es TRUE, indica que el segundo parámetro (str) es el nombre de un fichero dentro del cual se realizará la búsqueda. Si su valor es FALSE, indica que el segundo parámetro es un texto dentro del cual se realizará la búsqueda.
caseSensitive	Booleano. Indica si la búsqueda distinguirá o no entre mayúsculas y minúsculas.

Retorna: El identificador de la búsqueda:

-1.	En caso de error.
> 0.	En caso de haber encontrado alguna coincidencia de la expresión regular en la cadena de caracteres o en el fichero. Este valor de retorno será el que se utilice como identificador de la búsqueda en el resto de funciones de la librería.

7.1.2 CosGetNumMatchesRegExpr

Esta función retorna el número de coincidencias de la expresión regular halladas en la cadena de caracteres o en el fichero.

Sintaxis:

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

Parámetros:

regExprId	Identificador de la expresión regular. El valor de este parámetro es el valor de retorno de la función CosNewRegExpr.

Retorna:

-1.	Cuando no exista ninguna coincidencia de la expresión regular en la cadena de búsqueda o en caso de error en la función.
>= 0.	Este número se corresponderá con el número de coincidencias de la expresión regular.

7.1.3 CosGetMatchRegExpr

Esta función retorna la enésima coincidencia de la expresión regular en el texto indicado en la búsqueda. Será esta función la que dé la posición inicial en la que se ha encontrado la coincidencia dentro de la cadena.

Sintaxis:

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

Parámetros:

regExprId	Identificador de la expresión regular. Este valor es el retornado por la función CosNewRegExpr.
numMatch	Número de coincidencia de la expresión regular que se desea consultar.
Start	Entero por referencia donde se retorna la posición del texto donde se encuentra la coincidencia.
len	Entero por referencia donde se retorna la longitud de la coincidencia.

Retorna:

-1	Cuando no exista ninguna coincidencia de la expresión regular en la cadena de búsqueda o en caso de error en la función.
0	Siempre que exista una coincidencia.

Si se desea saber cuándo se ha llegado a la última coincidencia, éste será el valor que debe consultarse.

7.1.4 CosFreeRegExpr

Esta función libera de memoria la búsqueda de la expresión regular realizada con la llamada a la función CosNewRegExpr.

Sintaxis:

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

Parámetros:

regExprId	Identificador de la expresión regular retornado por la función CosNewRegExpr.

Retorna:

-1	Cuando regExprId no se corresponde con una búsqueda activa (valor de retorno de CosNewRegExpr).
0	Cuando ha conseguido liberar la expresión regular de memoria.

8. Csql

A partir de la versión 6.0 de Cosmos se podrá ejecutar un fichero SQL desde la línea de comando utilizando el comando **csql.**

Sintaxis:

```
csql [fichero] [-ini inifile] [-runscript –connection <connexion>
-database <base de datos>]
```

Nuevos parámetros:

-runscript	Ejecuta el script sql sin esperar a que el usuario pulse el botón de ejecutar.
-connection <conexion>	Abre la conexión indicada. Esta conexión debe existir en el fichero de configuración que se pasa en el parámetro "–ini". Parámetro obligatorio.
-database <bbdd>	Abre la base de datos indicada. Parámetro obligatorio.

NOTAS:

a) Se podrán borrar y crear bases de datos.

b) En caso de error la ejecución se detendrá, se mostrará por pantalla el error al usuario, se abrirá el Sql-Interactivo y el cursor quedará posicionado en la línea del fichero que ha provocado el error.

c) Abre la base de datos que se indica en el parámetro database independientemente del valor que se indique en la variable de entorno DBNAME.

9. Correcciones

- Code Insight. En algunas ocasiones tardaba mucho en mostrar la lista de métodos, dando la impresión de quedar detenida la ejecución.
- Cosrun. Los controles que estaban en la parte no visible de un Grid o un Box con scroll no tenían el tamaño adecuado.
- Cosrun. Si utilizábamos el driver de SQLSERVER nativo 11.0 fallaba al modificar un registro cuando se empleaba el método EditUpdate (ver el apartado Variables de entorno).
- Cosrun. El método UnloadToXml no descargaba correctamente los datos cuando había tildes o eñes en las etiquetas de las columnas.
- Cosrun. No era correcta la ordenación en lista String con autosort en columnas de tipo Date o numéricas y valores nulos.
- Cosrun. En un control Variable de la clase Page con las propiedades multilínea, auto wrap y vertical Text, al generar el informe las líneas se mostraban superpuestas.
- Cosrun. En las listas String el método ShowListFilterBar no filtraba correctamente cuando el valor de algunos de los campos de la lista era null.
- El método NumRows de la clase FormTable no retornaba el número correcto de registros leídos cuando el runtime se conectaba a una base de datos de Oracle vía ODBC.
- Cosrun. El método ResetListCellStyles de la clase Simplecontrol no funcionaba correctamente en los control List Box de tipo árbol.
- Cosrun. No funcionaba correctamente el método GetData del ActiveX de la Winsock dll, uno de los parámetros de este método no retornaba el valor correcto.
- Cosrun. Problema de refresco en la pantalla de Debug cuando se utilizaban algunos ActiveX de tipo Control de Form.
- Cosrun. Cuando en el evento “on Enter” de un control se llamaba al método SetFocus para cambiar el foco a otro control, se ejecutaba dos veces el evento “on Exit” del control que tenía el foco.
- Prnpag32.Dll. Se producía un error de memoria si al obtener la propiedad Label de control con la la función getPropStr ésta no tenía ningún valor.
- CTSQL. Al ejecutar una select de una view con agregados retornaba valores nulos.
- CTSQL. Error de protección general cuando se ejecutaba la función **to_char** de un número sin decimales con la máscara C99G999G999G999D99pr.