equinox6

New Features

Document Version:	1.1
Document Owner:	Mike Massey

Contents

1 Overview

1.1 Introduction

This document describes the new functions available in Equinox 6. The remainder of this section describes the key benefits of the upgrade from Equinox 5, while the remainder of the document provides further detail on each of the specific features.

1.2 Benefits

The key benefits of Equinox 6 are:

64-bit
Equinox 6.0 has had a major update to make it compatible with, and take maximum advantage of, the latest Microsoft operating systems: Windows 7, Windows 8 and Windows Server. The 64-bit version can take advantage of extra memory addressability and faster CPU modes to provide significant performance advances.

Equinox 6 can run seamlessly on either Win32 or Win64 operating systems, and supports mixed operation, with clients and the server running on either platform. It thus retains full compatibility with popular older versions such as XP and Vista. Applications are completely OS-transparent and do not require separate versions for different platforms.

Data Caching
The extra memory available in a 64-bit operating system is used by Equinox to cache records in memory, providing a big leap in performance.

Networking Performance Enhancements
For larger applications, considerable effort has been put into server responsiveness and network traffic optimisation to ensure good performance for larger numbers of concurrent users.

Enhanced Multi-Activity Clients
Equinox clients can now run multiple activities at both design and run time, allowing developers more flexibility and access to application features. This also enables more flexible applications that, for instance, can load and unload multiple forms, run reports and perform other tasks without exiting the current module.

Large Integer Support
The Equinox database manager and the method language have been updated to support 64 bit integers which allow up to 18 significant digits, simplifying and speeding up functionality such as currency calculations and number crunching applications.

Method Language Improvements
In addition to the 64 bit integer implementation, many restrictions on very large methods have been relaxed or removed, array operations have been extended, rationalised and improved, and a large number of small improvements and additions have been made to language functions and statements. The 64 bit executables take advantage of CPU facilities to provide significant speed improvements.

2 Equinox 64

There are now both 32 and 64 bit versions of all the Equinox executables, and they may be used interchangeably, depending on operating system constraints and hardware capability. In particular, both 32 and 64 bit clients may connect to either a 32 or 64 bit server.

Applications created in previous versions of Equinox will run, with minor changes required in order to ensure compatibility with Win32 and Win64. Equinox 6 prompts you to resave the application and indicates where changes need to be made.

Other than performance, and possible connectivity issues with external software (see section 10), there is no difference between 32 and 64 bit versions.

3 Data Caching

Equinox 6 can cache record data in memory. The caching is configurable on a table by table basis, and is most appropriate where the entire table can be held in memory, since for very large data sets the operating system disk caching gives good performance compared to an in-application LRU (least recently used) cache.

There is an important balance to be struck when allocating memory to Equinox record and index management, and leaving sufficient resources for the operating system to work at maximum efficiency. In order to reduce operating system memory management issues, table caches larger than 4MB are held in 4MB pages.

Memory allocated to index caching is assigned first - index caching has not changed from Equinox 5. The record cache is then assigned in the order specified in the configuration; if a table is not cached, a message is written to the server log.

Server cache settings are held in a section called [TableCache] containing the following keys:

TableCacheOnByDefault	Set to 1 to include all tables, or 0 to only include those specified. Defaults to 1 (ON) in Win64 and 0 (OFF) in Win32.
OSReservedMemory	The amount of memory in megabytes to reserve for the OS. Defaults to 384MB.
MaxCacheSize	The maximum amount of memory to allocate to the cache. Defaults to available memory less OSReservedMemory.
CacheImmediateWrite	Set to 1 to turn off write caching, 0 to turn it on. Defaults to 1.
CacheEncrypted	If this value is set to 1 and the table is encrypted, the cache data is also encrypted.
<TableName>=size	Size indicates the table cache size in MB, or alternatively a negative number indicating a percentage of the file. For example, -130 indicates a cache size 1.3 times the size of the record data when the cache is created.

Tables which are included in the cache as a result of TableCacheOnByDefault are added to the cache configuration list after ones specified in the ini file section, sorted in reverse size order (smallest first). The cache includes all records including deleted ones. If the cache becomes smaller than the size of the record data, the first records in the table are cached.

Single user cache settings are held in a section called [<AppName>TableCache], one per application, containing the same keys as above.

4 Other Client/Single User ini Keys

Other client/single user ini keys are:

EnableInformLocking	Suitable for high volume applications with large numbers of users; disables the facility to update non-current records on the screen and thus reduces network traffic. Note that setting SysAutoUpdate to zero also has a similar but lesser effect.
ProcLibCacheSize	Contains the number of public procedure libraries to cache on the client. The default is 64, which is usually sufficient for all except very large applications.

5 Database Changes

The database manager can now store integers of up to 18 significant digits. Numbers with decimal places are unchanged. The smallest possible space in the database record is used, given zero decimal places and the number of significant digits, as follows:

- 1-2: 1 byte
- 3-4: 2 bytes
- 5-8: 4 bytes
- 9-18: 8 bytes

The page size for BLB files is now configurable, which allows for much larger files; each data item is stored in one or more pages. The Properties dialog in the Table menu now contains the following sizes:

Page Size	**Maximum Total Size of Data**
32 bytes	128GB default, as version 5
256 bytes	1TB
2K	8 TB
16K	64TB

The maximum size of the text for logical items has been increased to 40 characters.

6 Equinox Dlls

There are 32 and 64 bit versions of all Equinox Dlls – see below. Use the generic name in Equinox, without adding 32 or 64. If Equinox asks you to choose a Dll file, you may choose either.

CUSTOM	Allows developers to change the Equinox icon for an application specific one. Replace the icon in a copy of Custom32.Dll and rename it <AppName>32.Dll, and similarly for Custom64.Dll. You may also create specific splashes for 32 and 64 bit versions – see *Other Changes*.
DDE	Equinox interface to Microsoft Dynamic Data Exchange.
EQODBC	Equinox interface to Microsoft Open Database Connectivity.
SQLMIRROR	Equinox interface to SQL providing data mirroring.

SQLDBMS	Equinox interface to SQL providing data storage.

7 Method Language Changes

There are 3 new integer types to handle the new 64 bit integer type implemented in the database manager. The following integer types may be declared, including a number of types with similar ranges, present for historic reasons:

Name	**Bits**	**Range**
Byte	8	-128 to +127
Ubyte	8	0 to 255
Short	16	-23768 to 32767
Ushort	16	0 to 65535
Int	32	-2147483648 to 2147483647
Uint	32	0 to 4294967296
Long	32	-2147483648 to 2147483647
Ulong	32	0 to 4294967296
Int32	32	-2147483648 to 2147483647
Uint32	32	0 to 4294967296
Int64	64	-9223372036854775808 to 9223372036854775807
Uint64	64	0 to 1152921504606846975
IntOS	varies	as Int64 in a 64 bit OS, otherwise as Int32.
UintOS	varies	as Uint64 in a 64 bit OS, otherwise as Uint32.

Number(SignificantDigits, 0) has a signed decimal range as in version 5, however SignificantDigits can be 1 to 18.

The types int, uint, long, ulong are no longer allowed in **external declarations**, as they are ambiguous in version 6 and may refer to either 32 bit, 64 bit values, or values which are 32 bit in Win32 or 64 bit in Win64. These types produce compiler errors and must be changed for the correct alternative type ending in 32, 64 or OS.

Data size restrictions are broadly the same as version 5 in order to ensure compatibility with Win32. The maximum data size for arrays and memos may be changed via the ini key ProcAllocSize, which has a default of just less than 2GB.

Array management has been improved. While it has always been possible to pass array vectors as parameters, this area has been thoroughly tested and improved, particularly where the type of the array argument is not the same as that of the procedure parameter. For example:

```
String s[]
s = “Hello”, “09”, “4”, “123”, “world”
SortNum vector(s, 2, 3)
Print s,;;

Procedure SortNum int v[]
      ArraySort v
End Procedure
```

correctly produces the output:

```
Hello 4 09 123 world
```

Other changes are:

- The method language file handling commands can now manage files larger than 4GB.
- There is no longer a limit on the number of nested brackets.
- Internal restrictions on the size of statement blocks (for example IF … END IF, SWITCH … END SWITCH) have been removed.

8 Method Language Functions

8.1 BinaryAnd, BinaryOr, BinaryNot, BinaryXor

When it finds the operators And, Or or Xor, Equinox decides by context whether to apply a logical operation resulting in true or false, or a binary operation which combines the bits in each argument to produce a new number, as shown in the truth tables below:

And	**0**	**1**
0	**0**	**0**
1	**0**	**1**

Or	**0**	**1**
0	**0**	**1**
1	**1**	**1**

Xor	**0**	**1**
0	**0**	**1**
1	**1**	**0**

Not	**0**	**1**
	1	**0**

Inside a conditional expression it applies a logical operator, however it applies a binary operator inside brackets and in all non-conditional expressions.

The new Binary operators work in a similar way to the existing ones but always apply a binary operation; that is to say, the truth tables above are applied to each bit of the operands in turn. Logical operators work in a similar way, except they interpret zero as False and every other value as True (1).

For example:

```
print 0x42 BinaryAnd 0x3 | 2
print 0x42 BinaryOr 0x3 | 67 or 0x43
print 0x42 BinaryXor 0x3 | 65 or 0x41
```

Note that And, Or and Xor could be substituted in the above statements. In conditional expressions the Binary operators must be used if a bitwise operation is required:

```
if 0x42 and 0x5 then print "yes" else print "no" | yes – logical operation
if 0x42 BinaryAnd 0x5 then print "yes" else print "no" | no – binary
operation
if (0x42 and 0x5) then print "yes" else print "no" | no – binary operation
x = 0x42 and 0x3 | 2 – binary operation
x = 0x42 <> 0 and 0x3 <> 0 | 1 (true) – logical operation
```

There is no implementation of LogicalAnd, LogicalOr, LogicalNot, or LogicalXor; in order to achieve the required result, simply use a logical expression. For example:

```
if (a <> 0 or b <> 0) and c then print "yes" else print "no"
print 0x42 <> 0 and 0x3 <> 0 | 1 (true)
```

Logical Xor is slightly trickier (but not a common requirement):

```
if not a = 0 xor not b = 0 then print “true” else print “false”
```

The main two caveats are as follows:

If writing an expression containing brackets always write `if (a <> 0 and b <> 0)` rather than `if (a and b)`, as for example, the expression `(2 and 1)` equates to zero / false.

Avoid the temptation to write `if MyString then` as this looks like if `Mystring <> “” then`, but in fact the compiler interprets it as `if MyString <> 0 then`, as a logical operation is numeric.

Binary operations are usually only used in systems programming, where bits are used to contain individual values; however some Equinox commands use this technique – for example IsNetwork:

```
If IsNetwork BinaryAnd  16 then | secondary activity
```

For historic reasons, the Equinox Not operator is always interpreted as a logical operator, and is not subject to the same rules as And, Or and Xor.

8.2 IsNetwork

This function has been updated and now returns the following values, which are compatible with previous versions of Equinox:

- 0 Single user
- 3 Interactive client
- 5 Server remote task
- +16 indicates a secondary activity (invoked via the Shift key and navigator/ menu option, etc)
- +32 indicates a webserver method
- +64 indicates a 64 bit method.

8.3 LogN(NumberExpression)

This function returns the natural logarithm of NumberExpression.

9 Method Language Statements

9.1 ArrayNeg

This statement applies the expression minus <element> to each element of the array.

9.2 ArrayNot

This statement applies the expression BinaryNot <element> to each element of the array.

9.3 FlushAllDatabases [HowExpression]

HowExpression is a new optional parameter that allows more control over the record cache:

0 or missing:	Close files no longer in use, but still open (as before)
1:	Flush the record write cache
2:	Stop the record cache
3:	Start the record cache

9.4 SetExecuteIndex IndexName [, ListNumber, HowExpression, LowerLimit, UpperLimit]

The last three parameters have been added so that the statement allows similar control over the index to the Group Records menu option:

IndexName is the index to use (applied to the table to which it belongs).

ListNumber is a number between 1 and 16 - you can specify up to 16 separate indexes, one per table used. Equinox always matches the index specified to its table, so the indexes in the SetExecuteIndex list do not have to match the order of the tables used.

HowExpression may contain the following values: 0=every record (default), 1=first, 2=last, 3=range.

If HowExpression=3, the limit parameters are used. If one is missing, no limit is applied to that end.

9.5 ArrayFind

There are two new options for the existing SearchOrder parameter on the ArrayFind function:

ArrayExpression, SearchExpression [, SearchOrderExpression, ColumnExpression, IndexItem1, IndexItem2, IndexItem3, IndexItem4]

4 Ignores the case in string searches

8 ArrayFind returns IndexItems (1,2,3,4) as the location to insert a new value rather than returning zero value indexitems

9.6 New UserDetails statement

This can be used in conjunction with **First/NextUserName** to return the details of the specified user:

UserDetails LoginName [, FullName, UserGroup, Category, UserExpiryDate, PasswordExpiryDate, Flags]

LoginName must be the value returned from the **First/NextUserName** statement or a string set explicitly in the method. All other parameters are optional.

FullName and **UserGroup** are strings which will contain the user's full name and the name of the user group to which they are assigned.

Category is a number as follows:

0	**Normal User**
1	**Application Tester**
2	**Developer**

UserExpiryDate will contain the date (if any) that the user will expire.

PasswordExpiryDate will contain the date (if any) that the user's password will expire.

Flags will contain a number as follows:

0	no options set
1	the user's password is set to never expire

2 the user's password is set to force a change on next login

Example:

The following example returns the user information for the first user in the Developer user group.

```
string vsLogin, vsFullName, vsUserGroup
number vnCategory, vnFlags
date vdUserExpiry, vdPwordExpiry


FirstUserName vsLogin, vsFullName, "Developer"

UserDetails vsLogin, vsFullName, vsUserGroup, vnCategory, vdUserExpiry,
vdPwordExpiry, vnFlags

Print "Login: ", vsLogin
Print "FullName: ", vsFullName
Print "User Group: ", vsUserGroup

switch vnCategory
       case 0 |      Normal User
              Print "Category : Normal User"
       case 1 |      Application Tester
              Print "Category : Application Tester"
       case 2 |      Developer
              Print "Category : Developer"
end Switch

Print "User Expiry Date: ", vdUserExpiry
Print "Password Date: ", vdPwordExpiry

Switch vnFlags
       case 0 |      no options set
              Print "Flags : " & vnFlags & " no options set"
       case 1 |      password never expires
              Print "Flags : " & vnFlags & " password never expires"
       case 2 |      force password change at next login
              Print "Flags : " & vnFlags & " force password change at next
login"
End switch
```

10 External Software Connectivity

Equinox 6 has been designed so that applications will run in Win32 or Win64 without alteration. In order to achieve this with external Dlls, some work is required to configure external declarations so that Equinox understands what parameters to pass.

Firstly, ensure that your Dll provider has supplied both 32 and 64 bit versions of the required software; as the executable format is different, 32 bit Dlls cannot normally be called from Win64, and vice versa. If you encounter problems in this area, Compsoft may be able to help by writing bespoke inter-platform connectivity software – please contact our Support department.

In order to run the example Dll “MyDll” on both platforms, put both Dlls in the same folder, name one MyDll32.Dll and the other MyDll64.Dll, then in the external statement refer to the Dll as “<path>\MyDll”, or simply “MyDll” if they are in the Equinox folder.

As in previous versions, if you are using system API calls, use generic names for the Dlls: “Kernel”, “User”, and “GDI”. There is no need for the file extensions (e.g. .exe, .dll etc.); in fact it is more likely to fail if you include these extensions.

Examine each parameter and determine if its size will change in Win64; also convert all int and long parameters to the relevant size integer type – int32, int64 or intOS. Use IntOS where the item is an operating system value such as a handle or a pointer that is 32 bits in Win32 and 64 bits in Win64.

As Equinox runs in native mode, communication with .NET applications is not straightforward, as .NET runs in its own sand-boxed runtime environment; if you can't use file transfer or operating system inter-application data transfer facilities, please contact Compsoft Support for advice.

11 Multiple Tasks

Equinox 6 supports multiple modules running in the same executable task, which may be started either from the Navigator or from the method language. In order to start a concurrent task from the Navigator or menu, simply hold down the Shift key while selecting the option or pressing the Navigator button.

ExecuteSetMode has been enhanced to allow the task to run locally; use a value of 8 for WhereExpression.

While there is a practical limit on how many modules can run before the display becomes confusing, the feature allows greater modularisation of form elements, running a report or other module without unloading a form, and at design time allows examining public procedures while editing a form and so on.

The following method language commands help to manage multiple activities/tasks:

11.1 ExecuteTaskHandle

ExecuteTaskHandle is an existing function that returns the task ID for the current activity (as returned in SysReply from the Execute statement). It is zero for the root activity.

11.2 ExecuteTaskParent

ExecuteTaskParent is a new function that returns the task ID for the activity that started the current activity, or zero.

11.3 ChangeActivity TaskID

This is a new statement that changes the edit focus to the activity identified by the TaskID parameter. SysError is non-zero if the task is not currently running.

12 Speed Comparisons

The following features have contributed to substantial speed improvements:

- Record cache
- 64 bit port
- Method language review
- Server TCP/IP transport improvements
- Server lock contention tuning

The extent of the improvement depends heavily on the nature of the application, the operating environment and the hardware. As documented here, version 6 allows developers considerable scope to tune the software themselves.

13 Allow Syntax Highlight Colours to be Changed

These colours are not application specific. Add lines to the client or single-user ini file in one of the following two formats:

- SyntaxColour#=r,g,b
- SyntaxColour#=0xBBGGRR

where # is a number between 1 and 11 which indicates what the colour is used for (see below); r, g and b are decimal numbers separated by commas; BB, GG and RR are two digit hexadecimal numbers with no separators (this format is for those familiar with Windows COLORREF numbers).

For example, these two lines create the same colour for different uses:

- `SyntaxColour7=0x9F3FFF`
- `SyntaxColour1=255,63,159`

The colour usage types and defaults are as follows and are set by default to:

1	variable, field, workarea (RGB:0,0,0)
2	reserved word (RGB:0,0,255)
3	function(RGB:255,0,0)
4	DLL reference(RGB:0,0,0)
5	predefined constant (RGB:0,0,0)
6	preprocessor directive (RGB:0,0,255)
7	statement (RGB:255,0,0)
8	property (RGB:0,0,0)
9	procedure (RGB:0,0,0)
10	comment (RGB:0,128,0)
11	quoted string (RGB:128,128,128)

Decimal numbers may be omitted and default to zero, and the hexadecimal digits are parsed as a single number which becomes the colour value:

- `SyntaxColour7=0xFF00`
- `SyntaxColour1=,255`

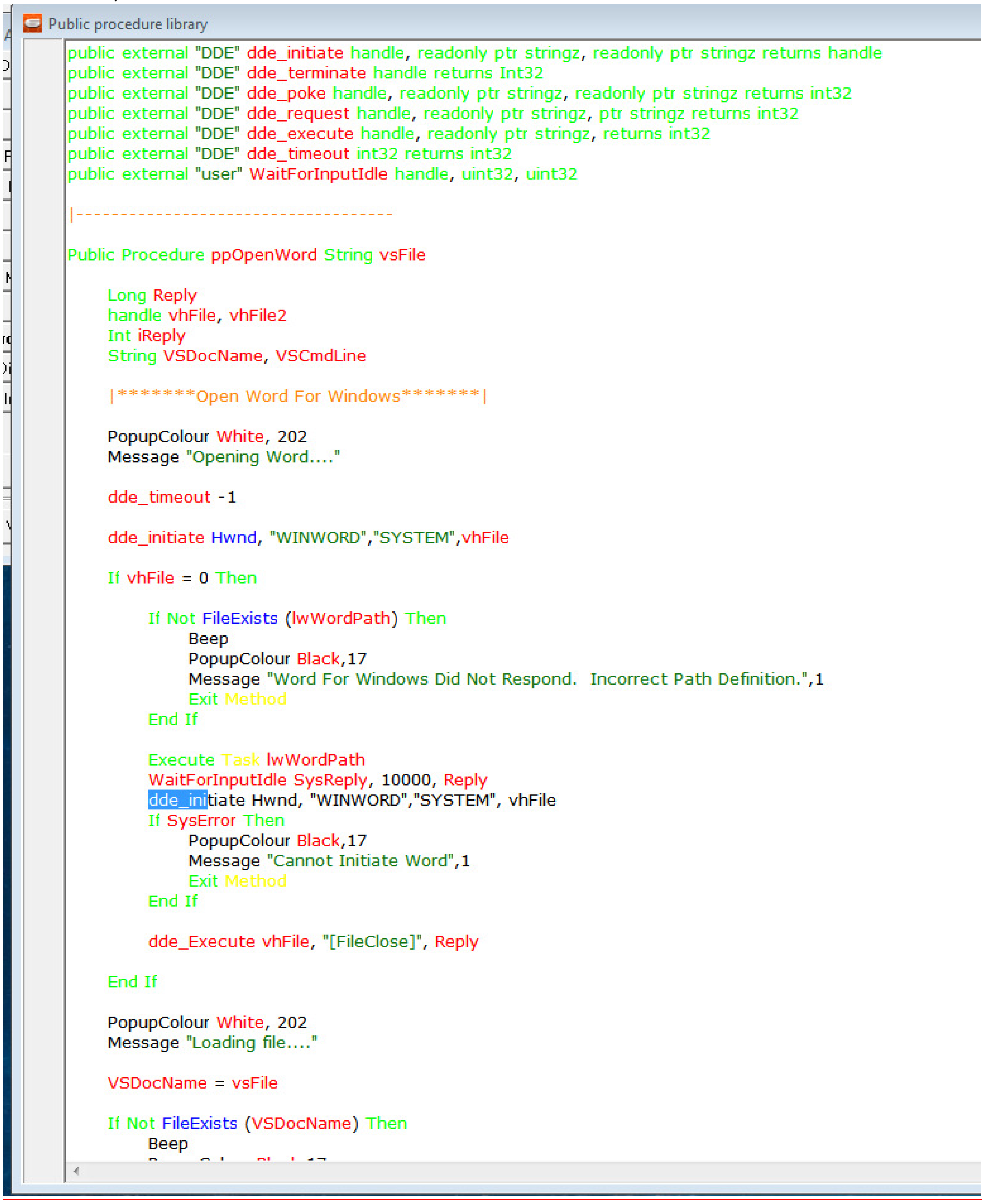
Both options above produce light green.

13.1 Example ini File

```
[Applications]
V6 A to Z=V6AtoZ,172.0.0.1:6029

[Equinox]
IterationXY=1
ScreenPixelHeight=900
ScreenPixelWidth=1440
SyntaxColour1 = 255,0,0
SyntaxColour2 = 0,255,0
SyntaxColour3 = 0,0,255
SyntaxColour4 = 255,255,0
SyntaxColour5 = 0,255,255
SyntaxColour6 = 255,0,255
SyntaxColour7 = 0,0,0
```

```
SyntaxColour8 = 199,199,199
SyntaxColour9 = 100,150,100
SyntaxColour10 = 255,132,9
SyntaxColour11 = 13,113,13
```

This would produce code which looks like this:

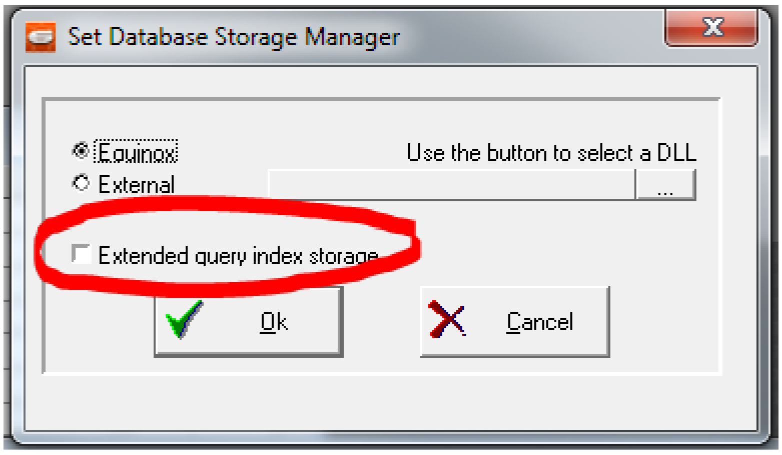


Public procedure library

```
public external "DDE" dde_initiate handle, readonly ptr stringz, readonly ptr stringz returns handle
public external "DDE" dde_terminate handle returns Int32
public external "DDE" dde_poke handle, readonly ptr stringz, readonly ptr stringz returns int32
public external "DDE" dde_request handle, readonly ptr stringz, ptr stringz returns int32
public external "DDE" dde_execute handle, readonly ptr stringz, returns int32
public external "DDE" dde_timeout int32 returns int32
public external "user" WaitForInputIdle handle, uint32, uint32

|------------------------------------

Public Procedure ppOpenWord String vsFile

    Long Reply
    handle vhFile, vhFile2
    Int iReply
    String VSDocName, VSCmdLine

    |*******Open Word For Windows*******|

    PopupColour White, 202
    Message "Opening Word...."

    dde_timeout -1

    dde_initiate Hwnd, "WINWORD","SYSTEM",vhFile

    If vhFile = 0 Then

        If Not FileExists (lwWordPath) Then
            Beep
            PopupColour Black,17
            Message "Word For Windows Did Not Respond.  Incorrect Path Definition.",1
            Exit Method
        End If

        Execute Task lwWordPath
        WaitForInputIdle SysReply, 10000, Reply
        dde_initiate Hwnd, "WINWORD","SYSTEM", vhFile
        If SysError Then
            PopupColour Black,17
            Message "Cannot Initiate Word",1
            Exit Method
        End If

        dde_Execute vhFile, "[FileClose]", Reply

    End If

    PopupColour White, 202
    Message "Loading file...."

    VSDocName = vsFile

    If Not FileExists (VSDocName) Then
        Beep
```

14 Extended Query Index Storage

The space used to store query index instances has been increased and it is now possible to store about 4,000 instances rather than about 1,000 in previous versions. This feature can be enabled and disabled in the Database Definition > Storage Manager dialog; it applies to all tables in the database:


Set Database Storage Manager

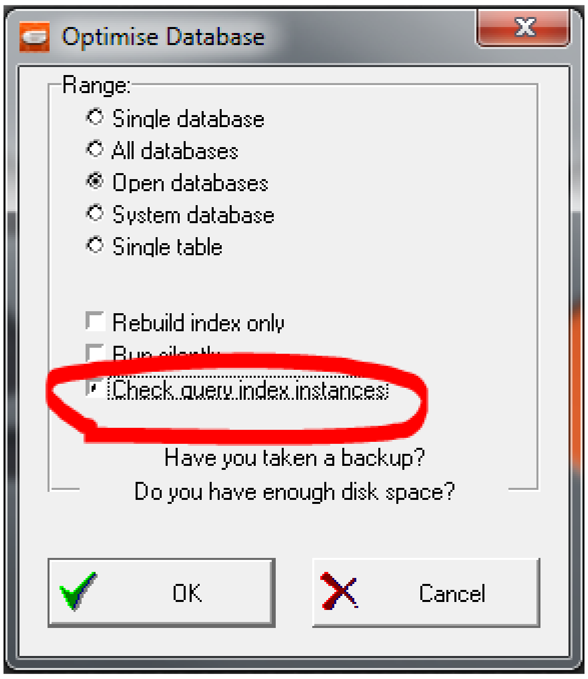
(● Equinox) ○ External

Use the button to select a DLL

☐ Extended query index storage

15 Index Management

The Optimise Database dialog has a new checkbox called "Check query index instances" which is enabled for the option "Optimise Open Databases". If checked, Equinox opens each table in turn and, in addition to the existing optimise if required, removes any orphaned index instances.


Optimise Database

Range:
○ Single database
○ All databases
● Open databases
○ System database
○ Single table

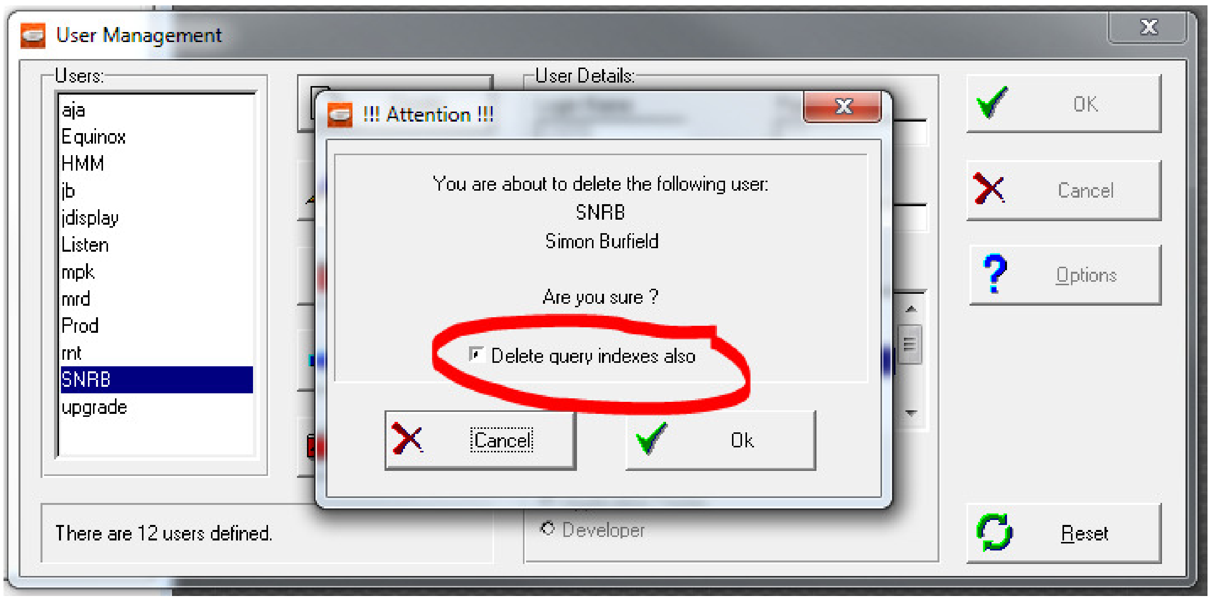
☐ Rebuild index only
☐ Run silently
☑ Check query index instances

Have you taken a backup?
Do you have enough disk space?

The method language Execute VerifyDatabase command automatically checks instance lists in the tables checked, as above.

In Index Management, deleted users are identified as "(Unknown)".

In User Management, there is now a confirmation dialog on deleting a user, which also contains a checkbox (defaulting to true) which when checked will delete the index instances for the user being deleted.



You are about to delete the following user:
SNRB
Simon Burfield

Are you sure ?

☑ Delete query indexes also

There is a new function, IndexInstances (tablename), which returns the number of instances currently used for the named table.

15.1 Method Language Changes for Index Management

15.1.1 BlankIndex

BlankIndex IndexName [, UserName, NoInstance]

Blankindex has an additional optional parameter NoInstance which defaults to false and, if true, deletes the index instance if it exists (rather than creating it if necessary and deleting any index entries).

15.1.2 UserDelete

UserDelete Username [,DeleteIndexInstances]

UserDelete has an additional optional parameter, DeleteIndexInstances, which defaults to false and, if true, deletes any index instances associated with the user.

SetIndex, GroupSetIndex, MarkIndex, UnMarkIndex and UpdateIndex all now return SysError = 1 if the user specified is not found. Other behaviour remains unchanged.

15.1.3 Ini File Changes

The SysDiagnostics = 1024 / 2048 / 3072 facilities in previous builds have been removed. Instead there is a new .ini key, CheckQueryIndexes. This is only required if advised by Compsoft, and can take the following values:

0: no effect

1: make the optimise database query index checkbox default to checked

2: send a certain amount of debug to the device specified by the Debug= ini key

3: 1+2

4: send more verbose debug

5: 4+1

6: 4+2

7: 4+2+1

16 Other Changes

The following are additional changes made in Equinox 6:

- Equinox 6 allows a separate 64 bit EPI file called <Appname>64.EPI which may contain 64 bit specific images – principally to allow a different splash. If not present, the 64 bit executables use the standard EPI file.
- Very large memos requiring more than 32K vertical pixels are now printable in reports, via the expand memos feature.
- The maximum number of Equinox users has been increased to 2048.
- Equinox 6 implements the latest production version of OpenSSL, version 1.0.1c. There are no changes to functionality.
- The developer repository window now remembers whether it is hidden or not between Equinox sessions.