New Features

Document Version:	2.2
Document Owner:	Helen Mabey

Contents

1 Introduction

This document describes the new functions available in Equinox 7. For further information on how to implement any of the below features please contact support.

2 Zoom

Equinox 7 now has the ability to resize forms to fit any screen size. This function comes with the following options and control methods:

2.1 ScreenZoomEnabled

Zoom can be enabled and disabled with the client ini setting `ScreenZoomEnabled`. The possible values for this ini key are:

0	Disable the screen zoom system
1	Enable the zoom system
2	Automatically zoom data entry forms to fill the Equinox window
5	Do not change the font size while zooming windows
6	Do not change the font size while automatically zooming the windows

If you select option 1 then the user will be able to use the following keys to zoom and shrink the Equinox form to their desired size:

- Ctrl-Numpad-Plus or Ctrl-Shift-Plus: Enlarge
- Ctrl-Numpad-Minus or Ctrl-Shift-Minus: Shrink
- Ctrl-Numpad-Multiply or Ctrl-Shift-Multiply: Fill the Equinox window with the form
- Ctrl-Numpad-Divide or Ctrl-Shift-Divide: Return the form to its original loaded size

If you select option 2, when you open an Equinox form for the first time it will automatically stretch to fit the current screen size. The Control keys will also work to allow the user to zoom to a size of their choice. This manually zoomed proportion will then be maintained until the user logs out of the application.

Please Note: you may need to adjust your code if you have placed items on forms in specific locations using `objectname.pixelx=` and `objectname.pixely=`.

2.2 SysZoom

Developers can also control the use of the Zoom system with *SysZoom*, which is a read/write workarea. The values and operation of *SysZoom* are the same as that for *ScreenZoomEnabled* – the only difference is that it is controlled programmatically rather than via the ini file.

2.3 Toolbar Zoom Option

The default data entry toolbar now ends with a new binoculars icon that brings up a dialog allowing the form size to be changed. This dialog is identical to the one brought up by option 4 of the ZoomScreen statement. This toolbar option is also available on the User Defined Toolbar.

3 Enhanced Encryption

Equinox 7 features an alternative encryption mechanism - this is based on the existing Feistel encryption method however it has a much longer auto-generated key plus integration with record versioning, such that no two identical records will be the same, and the subsequent identical saved version of a record will be different to its predecessor. This makes the system both more secure than the existing encryption scheme and also faster. This encryption is applied on a database level but can be overridden at table level if required.

4 Install DLL’s with your Application

Equinox 7 features a new Public Procedure which will install DLLs on a client machine at run time. The process requires that the DLL installer is placed in the client install folder on the server - when the client is run on the client machine, Equinox will check if the DLL is already installed and install it if not.

To use this function, copy the contents of the file DLLInstaller.prc (located in the client installation folder) into a Public Procedure library within your application. Then make sure that your added DLL installers are in a folder called “Extras” in the Equinox client install folder and, when the Equinet client is run on any machine, the system will check for extra installers and run them if required.

5 Profiler

The Equinox profiler logs the time spent executing parameter files, form events and server requests, without any changes required to applications. In addition, it implements a system of 500 user trace events that can be configured and used to profile the method language. Events have a start and an end and may be nested but not overlapped. Parameter files that run on the server as remote tasks may also be profiled.

In addition, a monitor DLL may be configured to run separately from the profiler; it is not affected by commands that turn the profiler on and off, and receives all events configured for it. The monitor DLL API may be configured for any suitable profiling or monitoring task, such as supplying information to an external dashboard application.

By using separate user events for the profiler and the monitor, it is possible to run the monitor all the time and still use the profiler to diagnose specific runtime issues.

There are a number of new Method Language Statements to accompany the Profiler, as follows:

5.1 ProfileEvent

Syntax: ProfileEvent EventNumber [, Description, Flags, Action]

Action: [Statement] Configures details of user event *EventNumber*.

Scope: Usable anywhere.

Notes: Supply *EventNumber* as a number between 6100 and 6199. This is the number used in the logs to differentiate it from server requests and form events. Alternatively, use 0 to 99, which will automatically be converted to 61xx.

Supply *Description* as a short text string describing the event.

Flags may be set to *True* to summarise this event individually. If that is done, nested events and client-server requests that happen during this specific event are listed. Note that the configuration option +8 sets this flag for all events.

Action may have the following values:

0	Do nothing
1	Add a start event
2	Add a message

The *Action* parameter is a function provided for convenience as it avoids having to call `ProfileEventStart` in situations where the event is used once, or where it is useful to show a start point with a message.

The system workarea *SysError* is set to a non-zero value if an error occurs, otherwise zero.

Examples:

```
| This example demonstrates a one-off use of an event
ProfileEvent 6100, "Run end of day", true, 1 execute method "EndOfDay" ProfileStop

| This example demonstrates two nested events where the execution time
| of the profiling statements is important
define PPCALCULATE 6110

ProfileEvent PPCALCULATE, "Time in ppCalculate"
ProfileEvent 6100, "Run end of day", true, 1
subtable Customer

       FirstRecord
       while not SysError

              ProfileEventStart PPCALCULATE
              ppCalculate
              ProfileEventStop

       NextRecord
       end while

       ProfileEventStop

End subtable | customer
```

5.2 ProfileEventStart

Syntax: ProfileEventStart EventNumber

Action: [Statement] Adds a trace start event to the profile history.

Scope: Usable anywhere.

Notes: Supply *EventNumber* as a number between 6100 and 6199. This is the number used in the logs, to differentiate it from server requests and form events. Alternatively, use 0 to 99, which will automatically be converted to 61xx.

The statement `ProfileEvent` may be used to add a text description and configure the event.

The system workarea *SysError* is set to a non-zero value if an error occurs, otherwise zero.

Example:

```
ProfileEvent 6100, "Call ppCalculate"

subtable Customer
```

```
     FirstRecord
     while not SysError

      ProfileEventStart 6100
      ppCalculate
      ProfileEventStop

     NextRecord
     end while
     ProfileEventStop
end subtable |      Customer
```

5.3 ProfileEventStop

Syntax: ProfileEventStop

Action: [Statement] Adds a trace stop event to the profile history for the currently running event.

Scope: Usable anywhere.

Notes: The system workarea *SysError* is set to a non-zero value if an error occurs, otherwise zero.

Example:
```
| This example demonstrates a one-off use of an event ProfileEvent 6100, "Run end
| of day", true, 1
execute method "EndOfDay" ProfileStop
```

5.4 ProfileMessage

Syntax: ProfileMessage EventNumber

Action: [Statement] Adds a trace message to the profile history.

Scope: Usable anywhere.

Notes: Supply *EventNumber* as a number between 6100 and 6199. This is the number used in the logs, to differentiate it from server requests and form events. Alternatively, use 0 to 99, which will automatically be converted to 61xx.

The statement `ProfileEvent` may be used to add a text description and configure the event.

The system workarea *SysError* is set to a non-zero value if an error occurs, otherwise zero.

Example:
```
| This example uses ProfileMessage to trace when a route is found
number routeID(8, 0), fare, lowestFare, count

ProfileResume
ProfileEvent 6101, "Fare found"

| Summarise this event and start it now
ProfileEvent 6100, "Search for best route", 1, 1

ppGetFirstRoute routeID, fare
lowestFare = fare + 1
```

```
while routeID

       if lowestFare > fare then

              ProfileMessage 6101
              ppAddRouteToList routeID, fare
              lowestFare = fare

       end if
       ppGetNextRoute routeID, fare

end while

ProfileEventStop
ProfilePause
| Produce a log including the full trace history in order to see the
| individual 'Fare found' messages
ProfileLog 8
```

5.5 ProfileLog

Syntax: ProfileLog [Flags]

Action: [Statement] Outputs profile reports to the log file and clears the trace history.

Scope: Usable anywhere.

Notes: By default, the log contains an event summary, however other reports may be added via the optional Flags parameter.

Flags may contain a combination of the following values:

+4	Include full trace history
+8	Include details for all events (otherwise only include details where requested via `ProfileEvent`)
+16	Add client-server requests to the trace history

A log is produced when Equinox terminates if the profiler is enabled and there is anything in the trace history.

The system workarea *SysError* is set to a non-zero value if an error occurs, otherwise zero.

Example:
```
| This example produces a log with a full trace including client-server request
| details plus an overall summary.
ProfileLog 20
```

5.6 ProfilePause

Syntax: ProfilePause

Action: [Statement] Halts profiling.

Scope: Usable anywhere.

Notes: Profiling is suspended, however the data already captured is retained. From the keyboard, Ctrl-F12 has the same effect.

The system workarea *SysError* is set to a non-zero value if an error occurs, otherwise zero.

Example:
```
| This example avoids profiling during SaveRecord
ProfilePause
SaveRecord
ProfileResume
```

5.7 ProfileResume

Syntax: ProfileResume

Action: [Statement] Starts or re-starts profiling.

Scope: Usable anywhere.

Notes: Profiling is re-started if necessary, and if already in progress does nothing. From the keyboard, Ctrl-F11 has the same effect.

The system workarea *SysError* is set to a non-zero value if an error occurs, otherwise zero.

Example:
```
| This example avoids profiling during SaveRecord
ProfilePause
SaveRecord
ProfileResume
```

5.8 ProfileReset

Syntax: ProfileReset [Flags]

Action: [Statement] Resets the profiling system.

Scope: Usable anywhere.

Notes: The trace history is cleared, and any extra resources used by the profiler's previous run are freed. The log file is renamed using the time stamp of the last log update or deleted.

If the optional *Flags* parameter is supplied as zero, the profiler is disabled. If the parameter is not supplied or is less than zero, the profiler is re-initialised using the `ProfileApplication` ini key setting. Otherwise *Flags* may be a combination of the following values:

+1	Profiling enabled
+2	Turn trace on (otherwise initially off)
+4	Include full trace history in the log
+8	Include details for all events in the log (otherwise only where requested via `ProfileEvent`)
+16	Add client-server requests to the trace history
+32	Delete any existing log file (otherwise it is renamed including a timestamp)

The system workarea *SysError* is set to a non-zero value if an error occurs, otherwise zero.

Example:
```
| This example disables the profiler
ProfileReset 0

| This example restore the ini file settings
ProfileReset -1
```

5.9 ProfileEventSwap

Syntax: ProfileEventSwap OldEventNumber, NewEventNumber

Action: [Statement] Combines `ProfileEventStop` and `ProfileEventStart` into one statement.

Scope: Usable anywhere.

Notes: Supply the *EventNumbers* as a number between 6100 and 6199. This is the number used in the logs, to differentiate it from server requests and form events. Alternatively, use 0 to 99, which will automatically be converted to 61xx.

This statement is a more efficient way to code the following:

```
        ProfileEventStop OldEventNumber
        ProfileEventStart NewEventNumber
```

The system workarea *SysError* is set to a non-zero value if an error occurs, otherwise zero.

Example:
```
ProfileEvent 6100, "Call ppCalculate1"
ProfileEvent 6101, "Call ppCalculate2"

subtable Customer
      FirstRecord

        while not SysError

               ProfileEventStart 6100
               ppCalculate1
               ProfileEventSwap 6100, 6101
               ppCalculate2
               ProfileEventStop

        NextRecord
        end while

        ProfileEventStop
end subtable | customer
```

6 Postgres (PG) Mirror

In addition to the SQL Mirror option, Equinox 7 includes the ability to mirror an Equinox database to a Postgres instance. The Postgres mirror will update in the background.

Postgres Ini settings are as follows (similar to the SQL ini keys):

6.1 NoPGMirror:<appname>= 1

Temporarily disables the mirror in this client for *appname*. The DLL (pgmirror*.dll) is required to be present and is invoked, but no calls are made to Postgres.

6.2 NoPGMirror= 1

Temporarily disables the mirror in this client. The DLL (pgmirror*.dll) is required to be present and is invoked, but no calls are made to Postgres.

6.3 PGServer:<appname>=

PGServer:<appname>=<hostname>[:<portnumber>]

Assigns a Postgres instance and optionally a port-number to *appname*. The default port-number is 5432, e.g. `PGServer:test=localhost:5455`

6.4 PGServer=

Assigns a default Postgres server and optional port-number if there is no app specific setting, e.g. `PGServer=10.0.1.24`

if no relevant PGServer key is found, Equinox is unable to connect to the Postgres instance.

6.5 PGTrace=

This key name is used for debugging errors in the Postgres Mirroring function.

Valid values are:

0	disable statement output (default)
+1	enable statement output to print window - detailed
+2	enable output to log - detailed
+4	enable output to Print Window – errors only
+8	enable output to log file – errors only

6.6 PGDisableTextValidation=

- Text data is not checked by the DLL and is subject to Postgres validation
- Characters < 32 are replaced with spaces
- Embedded character zeros are always replaced with spaces

6.7 PGReportValidationErrors=

This optional key name controls the reporting of Postgres data conversion errors. For example, if there is a date in Equinox which does not conform to the required format in Postgres, when you try to convert that record to Postgres an error would normally be produced. However, if this ini key is present and set to 1, these conversion errors will not be reported, and the data is automatically corrected to a valid value.

6.8 PGInfoCacheSize=

Caches the .HDR files which contain field mapping data for Postgres:

20	default
50	maximum
0	disables caching

6.9 PGBackgroundWrite=

If set to zero, turns off the background update and all Postgres mirroring is executed in the foreground.

6.10 PGUpdateStats=

If set to zero, disables the collection and reporting of foreground and background Postgres times:

- Time spent updating Postgres in the foreground (Postgres only - not including time spent in the DLL preparing statements)
- Foreground time waiting for cache buffer space to become available
- Background time spent updating Postgres

6.11 PGTimeout=

Sets the maximum time in milliseconds that a Postgres command can take before the client assumes it has failed. A value of zero allows an infinite amount of time - this was the previous default.

6.12 PGMirrorIgnoreError=

If set to 1, this ini key makes Equinox into the master database and ignores errors in Postgres. The default of 0 means that, if a write to the Postgres database fails, a write to the Equinox database will also fail.

Note that if background updating is enabled, any errors will be reported after the event, so this mode is only compatible with the `PGMirrorIgnoreError=1` ini key. Operators must review the log files to determine if mirror integrity errors have occurred.

6.13 NoMirrorDLL =

When enabled in the application ini file this key avoids loading the mirror DLLs entirely in the server. This helps with supporting applications which have been converted but where the support staff do not have access to the mirrored data.

7 SQL Mirror Improvements

Equinox 7 provides some new SQL Mirror ini keys as follows:

7.1 SQLMirrorIgnoreError

If set to 1 (default), this ini key makes Equinox into the master database and ignores errors in SQL. This value is also set if mirror record checking is enabled, since SQL errors do not occur in line.

If set to 0, if a write to the SQL database fails, a write to the Equinox database will also fail.

Note that if background updating is enabled, any errors will be reported after the event, so this mode is only compatible with the `SQLMirrorIgnoreError=1` ini key. Operators must review the log files to determine if mirror integrity errors have occurred.

7.2 NoMirrorDLL =

When enabled in the application ini file this key avoids loading the mirror DLLs entirely in the server. This helps with supporting applications which have been converted but where the support staff do not have access to the mirrored data.

8 Eventual Consistency Checking (ECC)

This is a background mirror verification process for both SQL and Postgres mirroring, which by default is not enabled. The functionality makes operating a mirror much more flexible, increases application uptime, speeds up recovery times after operational failures and allows performance issues to be more easily addressed.

The ECC can check a list of tables to ensure that the mirror table matches the Equinox one, and can do this repeatedly if required.

This function can be controlled via the following ini file settings:

8.1 MirrorCheckMode=

This key determines the startup mode of the ECC.

0	indicates that it is disabled
1	indicates that it will process the table list in the `[MirrorCheckTables]` section once (default)
2	indicates that it will do process the table list repeatedly

8.2 MirrorCheckPriority=

This key allows the developer to control the processing speed (and therefore CPU load) used by the ECC. Possible values are between 1 and 100:

100 (default)	indicates as fast as possible
1	makes the process wait several seconds in between short bursts of activity

Likely results are that full speed will mean a few percent of CPU load for the Equinox server, and a more substantial load for the mirror software, depending on configuration.

8.3 MirrorCheckAllTables=

If this ini key is set to one and the ECC is enabled via `MirrorCheckMode`, the server will add every table in the application that is mirrored to the list of tables to check, unless the table is disabled via `tablename=0` in the `[MirrorCheckTables]` section.

`MirrorCheckAllTables=` defaults to zero. The initial state of the ECC is that it is running but does nothing unless otherwise configured, or if a method or operator-initiated verify begins.

The above ini keys can be placed in the `[MirrorCheckTables]` section, in the header part of the ini file, or positioned using the following method language statements:

8.4 MirrorCheckSettings

Syntax: MirrorCheckSettings [priority, mode, logging]

Scope: Usable anywhere

This statement changes the configuration of the ECC. Priority determines the processing speed in the same way as the `MirrorCheckPriority` ini key. Mode is equivalent to `MirrorCheckMode`. Logging may be set to:

0	no logging
1	log errors
2	include beginning and end of table operations
3	include table list changes
4 (default)	include details of record corrections

8.5 MirrorCheckResetList

Syntax: MirrorCheckResetList [remakelist]

This statement empties the list of tables being processed by the ECC and stops the processing of the table currently being checked (if any). If the optional parameter `remakelist` is set to *True*, every mirrored table is added to the list (the default value is *False*).

8.6 MirrorCheckAddTable

Syntax: MirrorCheckAddTable table

The table indicated is added to the ECC list.

8.7 MirrorCheckRemoveTable

Syntax: MirrorCheckRemoveTable table

The table indicated is removed from the ECC list and, if it is currently being processed, the operation is halted.

8.8 MirrorCheckPause

Syntax: MirrorCheckPause [yesorno]

This statement temporarily pauses the ECC in an equivalent way to pressing the *Halt* button during an interactive batch operation. If the parameter `yesorno` is set to true, the ECC is paused, otherwise it is resumed.

8.9 MirrorVerifyRecord

Syntax: MirrorVerifyRecord [updaterecord]

Action: (Statement) Checks an Equinox record against its mirror

Scope: Usable in forms and sub-table blocks.

Notes: This statement checks the current record against the data held in the mirror. `SysReply` contains the result of the comparison. The optional parameter *UpdateRecord* now takes effect as follows:

0 (or missing)	just compare
1	compare and correct if necessary

If an unexpected error occurs during the update, `SysError` is set to a non-zero value, otherwise zero.

8.10 MirrorSetUpdateMode

Syntax: MirrorSetUpdateMode [mode]

Notes:
Setup of the Eventual Consistency Checker (ECC) and the Mirror Record Checker (MRC) is controlled by this ini key, which may take the following values:

0	These functions are disabled
1	The ECC is enabled
2	The MRC is also enabled

Single user Equinox defaults to zero – use mirrors directly. Clients and the server default to 2 – send mirror updates to the MRC process running in the server. However, client and single user optimise, and database modification and upgrade modules also operate on the mirrors.

If the optional parameter mode is set to one (default), this statement prevents the client from issuing insert, update or delete record operations on the mirror, which therefore means that the mirror will become out of date and will subsequently need to be corrected.

If mode is set to 2, mirror operations are not carried out immediately, but are instead stored in server table queues which are later processed by a background mirror record checking server process. Single user Equinox adds to the queues but does not process them; mirror updates will therefore be delayed until the server is run again. If the queues are not empty when the process terminates, they are stored in *Database\<tablename>.mrc* files.

This mode may also be enabled using the ini key/value `NoMirrorDll=2`, as follows:

0	restores normal operation
-1	returns the current setting in the system workarea SysReply

Note that the default state for the Equinox mirror system is that it updates the mirror immediately and in line with the Equinox record update.

8.11 Mirror Record Checker

There is now a facility in the server to check mirror data as records are changed in Equinox, by adding them to a table queue processed by a background server process. There is therefore minimal impact on the Equinox task, at the cost of a small delay to the mirror update plus extra load on the server, which now takes care of the mirror updating.

The mirror record checker is enabled and disabled by using the `MirrorDisable` method language statement with a value of 2 to activate it; see below. It may also be enabled using the optional ini key `NoMirrorDll=2`; if however *NoMirrorDll* is set to 1, the mirror record checker will never run. The default state is disabled (running but inactive).

Single user Equinox adds to the queues but does not process them; mirror updates will therefore be delayed until the server is run again. If the queues are not empty when the process terminates, they are stored in *Database\<tablename>.mrc* files.

Note that this system is separate from and complementary to the table-based mirror eventual consistency checker. If implemented on all clients, it removes the need for the clients to have access to SQL and can thereby increases the data security of the system.

JDisplay stats has an extra item: "Mirror Records Queued" which is the total number of records in the table queues waiting to be processed. The statement *ServerStats* with a data type of 1 (to return ECC stats) now additionally returns information about the record checker table queues.

8.12 New options on the Execute Statement

Syntax: Execute command [, parameter]

Notes: The following new commands are available:

- CorrectMirror
- CorrectAllMirrors
- VerifyMirrorTable
- CorrectMirrorTable

These commands check the Equinox records in the database table or tables specified against the mirror version and, if required, the "Correct" versions correct the mirror data. If a priority of 3 (low priority) is specified in the preceding `ExecuteSetMode` statement and the ECC is available (client-server), the task is added to the ECC table list for processing in the background by the server. For example:

```
ExecuteSetMode -1,,, 3
Execute CorrectMirrorTable Orders
```

Otherwise the task runs in the foreground as specified by the remaining `ExecuteSetMode` parameters, as normal.

9 Record Versioning

If this function is enabled an 8-byte version number is added to the end of each record and a field called `Equinox_rvn` is created in a mirrored database. The record number is incremented each time the record is written, including optimises. Record versioning allows the Eventual Consistency Checker and the foreground mirror checking commands to operate more efficiently, since they can quickly determine if a record is up to date without having to check all of its fields. However all commands can work without it.

The function is enabled via a check-box on the Storage Manager dialog in database definition.

If you use the `IsRecord` function, `SysReply` will contain the record version number of the current record, or zero if there is none.

10 New Timer Functions

Equinox 7 includes improvements to the timer functions, including a 64-bit timer which returns a value in millionths of seconds rather than hundredths. There are also several new functions which allow the timing of running code:

10.1 Timer64

Syntax: Timer64

Action: [Function] Returns a 64bit value containing the number of microseconds (millionths of a second) since the timer was initialised (when Equinox was launched).

Scope: Usable anywhere.

Notes: The value returned by this function is not comparable with that returned by the `Timer` function.

Example:
```
int64 start, taken
int i

start = timer64

for i = 1 to 1000000
next

taken = timer64 - start
print "test took", str(taken,,,4), "microseconds"
```

10.2 TimeStart

Syntax: TimeStart TimerHandle

Action: [Statement] Initialises *TimerHandle* so it may be used later by the `TimeElapsed` and `TimedOut` functions.

Scope: Usable anywhere.

Notes: TimerHandle must be of type *handle*. It receives a 64-bit value which has no meaning outside of the timer statements.

The system workarea *SysError* is set to a non-zero value if an error occurs, otherwise zero.

Example:
```
define TEST_COUNT 1000000

int i
handle t
int64 ms

PrintDevice ""

TimeStart t

for i = 1 to TEST_COUNT
next
```

```
ms = TimeElapsed(t)
print "test result", Str(ms,,,4)
```

10.3 TimedOut

Syntax: logical TimedOut(TimerHandle, Microseconds [, MultiShot])

Action: [Function] Returns true if at least the specified number of microseconds has elapsed since `TimeStart` was called on *TimerHandle*.

Scope: Usable anywhere.

Notes: *TimerHandle* must be of type handle and must previously have been passed to the `TimeStart` statement.

If *Microseconds* is less than zero, the function returns false; if it is zero, the function returns true. Positive values are checked against the timer and the function returns *False* if the amount of time that has passed is less, otherwise true.

If the optional parameter *MultiShot* is passed as true and the interval has been reached, the function also adds the value of *Microseconds* to *TimerHandle* such that, if it is used repeatedly, it acts as a multi-shot timer, returning *True* once each time the specified number of Microseconds elapses. For example, if a 100-microsecond interval is used, the function returns true once after 100 microseconds, then again after 200 microseconds, and so on, like a metronome tick.

Examples:
```
| This example repeatedly calls SomeProcedureCall until one second has elapsed
handle t

TimeStart t
repeat
SomeProcedureCall
until TimedOut(t, 1000000)

| This example uses a multi-shot timer
handle t
int i, n

PrintDevice ""

TimeStart t
repeat

  n += 1
  if TimedOut(t, 1000000, 1) then

        i += 1
        print i, now, Str(n,,,4)
        if i = 10 then exit repeat

  end if
end repeat
```

10.4 TimeElapsed

Syntax: int64 TimedElapsed(TimerHandle)

Action: [Function] Returns the number of microseconds since `TimeStart` was called on *TimerHandle*.

Scope: Usable anywhere.

Notes: *TimerHandle* must be of type *Handle* as previously passed to the `TimeStart` statement. The return value is a 64bit value.

Example:
```
| This example times the Sieve of Eratosthenes
define LIMIT 1000

handle t
int i, j, k, numbers[LIMIT], prime[LIMIT]

TimeStart t

| identify the non-primes
k = LIMIT / 2
for i = 3 to k step 2

        if numbers[i] = 0 then

                for j = i * 2 to LIMIT step i
                numbers[j] = 1
                next

        end if

next

| transfer the primes
j = 1
prime[j] = 2

for i = 3 to LIMIT step 2

        if numbers[i] = 0 then

                j += 1
                prime[j] = i

        end if

next

i = TimeElapsed(t)

ArraySetDimensions 1, prime, j

PrintDevice ""
print "Found", j, "prime numbers in", Str(i/1000,,,4), "milliseconds"
for i = 1 to j step 5
        print Vector(prime, i, 5),;;
next
```

11 Remote Task Improvements

Equinox 7 includes improvements to the *ExecuteSetMode* statement which allows feedback from the running task to be displayed on the calling client. There are now multiple configurations available with this statement, including the ability to run additional clients locally or on the server.

The `ExecuteSetMode` *Where* parameter may now take the following values for Equinox modules:

1	module runs on the remote server in-process Method, Process, Export, Import, Query only
2	module runs locally in a new instance of the current executable (equinet.exe if running on the server)
4	module runs in this client (current module suspended)
8	module runs in this client in a new activity
16	remote server runs module in new instance of current executable
34	module runs locally in a new instance of equinet.exe
48	remote server runs module in new instance of equinet.exe
66	module runs locally in a new instance of equinet64.exe
80	remote server runs module in a new instance of equinet64.exe

Modules that run in a new instance require an additional licence.

Execute *Shell*, *Command*, *Task* and *System* now respond to `ExecuteSetMode`:

The *Where* parameter may be:

0 (default)	run locally
1	run on the remote server
2	run locally (identical to not using `ExecuteSetMode`)

The `ExecuteSetMode` *Wait* parameter may take the following values:

-2	Monitor remote task in a new activity
-1	Do not wait
0	Wait with hourglass
1	Monitor in this activity with no dialog
2	Monitor in this activity with a dialog
3	Monitor in this activity with a dialog (optimise/verify only)

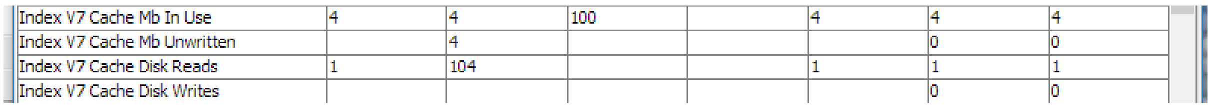
The following system workareas affect batch mode and optimise tasks (as before):

`SysAutoCommit`	Presses the *Commit* button when optimising tables
`SysBeep`	Makes a sound on task completion
`SysDuration`	Displays the time taken on the status line
`SysPause`	Waits while displaying final statistics
`SysPauseTime`	Waits for this number of seconds if *SysPause* is enabled
`SysSuppress`	Displays an Equinox running dialog instead of statistics
`SysSuppressHalt`	Prevents *Halt*, *Continue* and *Quit* buttons being shown

12 Index Caching

Equinox 7 features two new index cache managers, each of which have advantages over the V5/V6 one.

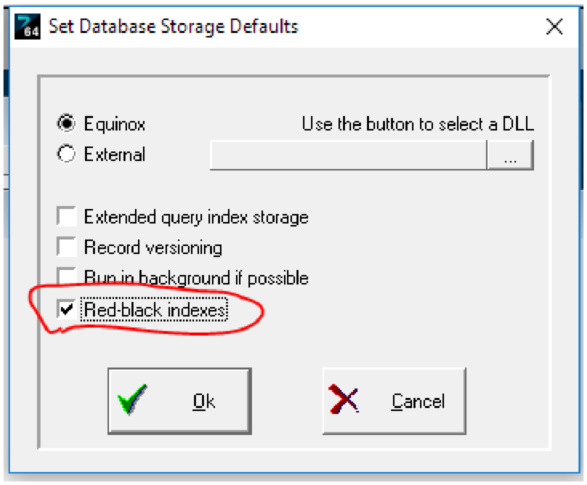
The recommended (and default) index cache manager for 64-bit is a new B-Tree cache which handles each table separately, and keeps all of the index in memory; sufficient RAM is therefore required in the hosting machine. A simple way to determine the amount required is to add up the sizes of all the index files, plus additional to allow for index growth. 32-bit can still use this system, but is limited by the amount of memory available to a 32-bit process - some less than 2GB. The amount of memory available to the process is now shown in the JDisplay statistics pane, as per the following example:



Index V7 Cache Mb In Use	4	4	100		4	4	4
Index V7 Cache Mb Unwritten		4				0	0
Index V7 Cache Disk Reads	1	104			1	1	1
Index V7 Cache Disk Writes						0	0

If there is insufficient memory to use the new system, or if the new system runs out of memory, Equinox automatically reverts to the old scheme.

An alternative to this is to use a completely different type of index called a red-black index. The files created by this system are different from the B-Tree ones, and the database must be converted in the Database Design/Storage Manager dialog by selecting the *Red-black indexes* checkbox and resaving the database.



Set Database Storage Defaults

[● Equinox] ○ External

Use the button to select a DLL

☐ Extended query index storage

☐ Record versioning

☐ Run in background if possible

☑ Red-black indexes

The main property of this index is that it ensures a balanced tree which guarantees that nodes will take a similar amount of time to access, and limits the depth of the index tree more than the B-Tree version. Although there is the potential for this scheme to be much faster than B-Tree indexes for databases with very large numbers of records and lots of updates, for other databases, especially smaller ones which are regularly optimised, the B-Tree system may be a better choice.

Version 7 has been carefully designed to make good default choices of configuration when the application is opened, however it is possible to take control using the following ini keys:

`MaxCacheMemory=<Mb>`	the maximum available memory used for record and index caching
`IndexBufferingDefault=<version>`	the default BTree cache version - see below for the Equinox choice of default
`IndexBufferSize=<Kb>`	the size of the version 4 per-handle cache, default 4,096
`IndexBufferTotalSize=<Mb>`	the size of the version 5 contended cache, default 512 for Win64 and 256 for Win32

Equinox chooses the default index cache manager as follows:

For Win64:

- If there is less than 2GB available, proceed as version 6.

- If the total size of record and index data plus 2GB is less than available memory, the version 7 uncontended cache is set as default and the record cache configures itself as version 6.
- If the total size of index data plus 2GB is less than available memory, the version 7 uncontended cache is set as default and the record cache is disabled.
- Otherwise, proceed as version 6.

For Win32, Equinox uses the same algorithm, with a figure of 1GB rather than 2GB.

The B-Tree index managers for each table may optionally be configured in the ini file `[IndexBuffering]` section. Note that in the single user equinox.ini file the section is called `[<Appname>IndexBuffering]` to allow for multiple application use by a single Equinox.exe; also note that *red-black* indexes are a separate system and are not affected.

Key values are:
`<Tablename>=<version>` Tablename may be e.g. TBL23 or Customer.

Recognised versions are:

- 7 (V7 uncontended cache)
- 6 or 5 (V5 contended cache)
- 4 (V4 caching scheme).

If a particular scheme isn't supported by the software currently running, it downgrades to a version that is supported.

13 GUI Enhancements

13.1 Repeated Buttons

Form buttons now have a “Repeated” property. If set to *True*, the button will repeat in the same way as fields, and clicking the button will change the form to that iteration before running its method. If the iteration zero button has a name, repeated buttons have the same name with the value of `sysiteration` appended.

Example:
The following example will label each repeated button (named btnID) on the form, with the corresponding customer surname.

Before Display:
```
        set text, "btnID"&SysIteration, cuSurname
```

When the button is used, it will update the *Updated Date* and *Time* on the corresponding record.

This Button:
```
        Date vdDate
        Time vtTime

        StartEdit
        ServerDateTime vdDate, vtTime
        cuUpdatedDate = vdDate
        cuUpdatedTime = vtTime
        SaveRecord
```

13.2 Picture Improvements

If a picture item doesn't fill the space assigned to it, the remaining space is painted with the colour of the filled box underneath it or, if none present, the dialog background colour. In previous versions this was always filled with the standard grey.

14 AutoExecHelp

Application developers can now make a bespoke Help system by creating a public procedure called `AutoexecHelp`. The procedure receives the Help type required in `SysTaskControl`; in the procedure, set `SysError` to 0 if you've satisfied the request, to 1 if you want Equinox to load its own Help (or the application Help file if configured), or exit method abort if Equinox should not call the procedure again for this application.

`SysTaskControl` may contain:

- keyword:"<text>"
- search:
- context:<dialog help number>
- contents:<dialog help number>
- help:
- quit:

These correspond to the various ways in which Equinox calls the existing *Help* systems.

Note that developers can also provide Help for their own application code (for example public procedure parameters) using this system, as it is called at any time F1 is active.

Example:
The following example will load an online Beta version of the Equinox Help when F1 is pressed within the method language of your application.

```
public procedure AutoExecHelp
       string vscommand, vshelpcommand

       vscommand = AfterStr (SysTaskControl, "keyword:", 1)

       Left(vscommand,1) = ""
       Right(vscommand,1) = ""

       vshelpcommand = "start
http://support.compsoft.co.uk:8080/hlp/langref.esp?lrCommand=" & vscommand

       execute system vshelpcommand

end procedure |      AutoExecHelp
```

15 AtomicIncrement

This is a new function which provides application-wide unique values without the need for explicit workarea locking.

Syntax: AtomicIncrement workarea, newvalue

This statement is intended to provide a simpler, faster and more reliable mechanism for supplying an application-wide unique numeric value, and replaces code like the following:

```
lock publicworkarea

       publicworkarea += 1
       mynewvalue = publicworkarea

unlock publicworkarea
```

All numeric workareas are compatible with this statement, and it may also be used with shared local workareas in multi-activity tasks.

16 ServerStats

This function provides details of server statistics in JSON format to the method language.

Syntax: ServerStats statisticsdata [, datatype]

This function provides server statistics in JSON format to the method language in the memo item *statisticsdata*. The following values may be supplied for the data type parameter:

0 (default)	supplies the information in the JDisplay Statistics pane
1	supplies statistics for the Eventual Consistency Checker

17 Segment Size

The Segment Size ini setting can now be applied on an application by application basis, so if you run multiple applications on one server, you can apply the segment size option to one or more applications.

New S/U key `<Appname>SegmentSize=<value>`. This is also optionally available in the server for ease of ini file management etc.

18 Delete Database

There is now a new option to delete a database via the method language. Obviously, this is not an option you want to make available to all your users, but you may want certain administrators to be able to do this.

Syntax: Execute Deletedatabase [Database Name}

Note: Optimise permission is required for the user executing this code

Example:
```
Execute DeleteDatabase myDatabase
```

19 Export/Import Improvement

Equinox version 7 features a new Export and Import data type *CSV (RFC 4180)* which handles embedded quotes and carriage return/linefeeds.