# exgarde EXFUSION

# User Manual

UM0048.GB Issue 4 09/07/2015

www.tdsi.co.uk

TDSi
Unit 10 Concept Park
Innovation Close
Poole
Dorset
BH12 4QT, UK

Tel: +44 (0) 1202 723535
Fax: +44 (0) 1202 724975

Sales Enquiries: sales@tdsi.co.uk
Marketing Support: marketing@tdsi.co.uk
Technical Support: support@tdsi.co.uk

## Foreword



Time and Data Systems International Ltd operate a policy of continuous improvement and reserves the right to change specifications, colours or prices of any of its products without prior notice.

## Guarantee

For terms of guarantee, please contact your supplier.

## Trademarks



Microsoft and Windows are registered trademarks of Microsoft Corporation.

All other brands and product names are trademarks or registered trademarks of 8their respective owners.

## Cautions and Notes

The following symbols are used in this guide:

**CAUTION! This indicates an important operating instruction that should be followed to avoid any potential damage to hardware or property, loss of data, or personal injury.**

**NOTE.** This indicates important information to help you make the best use of this product.

# Contents

# 1. Overview

EXfusion is a mechanism for allowing the EXgarde access control system to respond automatically to changes in an external SQL database (EDB) system. The goal is that whenever changes are made to source tables in the EDB, then EXgarde will respond to those changes as if an EXgarde operator made them.

For example, as new people are added into a Personnel database, those same people could be added into EXgarde without operator action. This includes automatically updating Access Control Units ACUs where necessary.

Fusion is a scripting tool, taking script commands from the EDB and executing the relevant commands against the EXgarde system. This involves data transformation, i.e. the translation of data fields in the EDB into the corresponding EXgarde data fields. Each EDB must therefore contain a new table, which is in effect a stream of commands to EXgarde. Fusion will then process each entry in the table securely without operator intervention. The customer will be responsible for maintaining the content of the command table and error handling.

It is important that both the customer's data and EXgarde are adequately protected from accidental or deliberate damage:

- Access to the EDB will remain under the customer's control, in that the name and password granting access only to the Command Table will be entered by the customer, during installation of EXgarde Fusion, and thus need never be known to TDSi. Access to other tables can be completely prevented by the customer.

- Access to EXgarde will be only be possible via EXfusion. This prevents the EDB user from accessing other parts of the EXgarde database. Fusion is logged into in the same manner as EXgarde so Fusion will only have access to data belonging to the specified Tenant.

# 2. Supported commands

The following EXgarde Commands are currently supported:

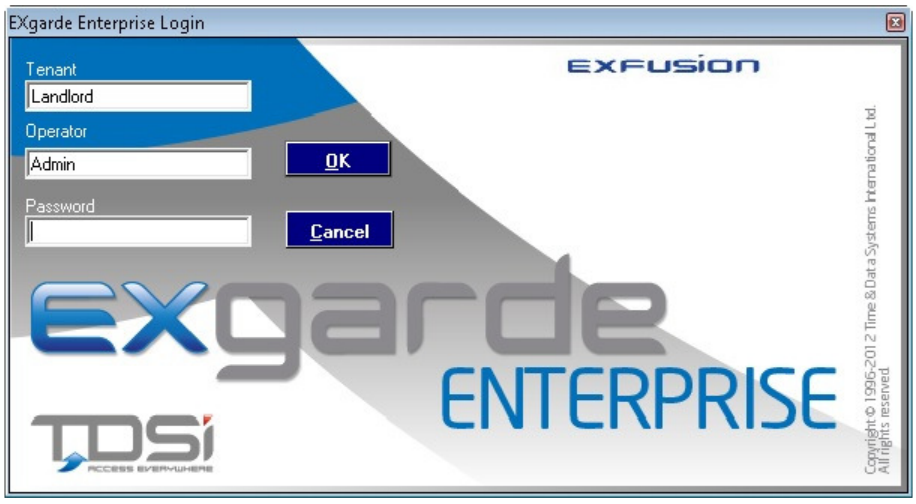
| | |
|---|---|
| Key Holder | New Keyholder |
| | Modify Keyholder |
| | Delete Keyholder |
| | Add to Group |
| | Remove from Group |
| | Add to Community |
| | Remove from Community |
| | Set Community Info |
| Key | New Key |
| | Modify Key |
| | Delete Key |
| | Issue Key to Keyholder |
| | Set Key Status |
| | Move Key to Box |
| Key Box | New Box |
| | Modify Box |
| | Delete Box |

# 3. Quick Installation and setup

- Install EXgarde as required
- Install EXfusion on the same PC as the EXgarde database server or any EXgarde client
- In the EDB, create a table which corresponds to the format described at the back of this document
- Start EXfusion and select New from the Client menu or toolbar button. Enter the details for the EDB. Press Test button to confirm connection to EDB.
- Create a Fusion operator on Exgarde
- Select Properties from Server menu and enter the required details for the EXgarde database
- If not started already by virtue of its set-up, start the service

# 4. Login Screen

The first time you start EXfusion, you will see a login screen:




This requires that you login to EXgarde and sets the context for all Fusion transactions. By logging in it means that:

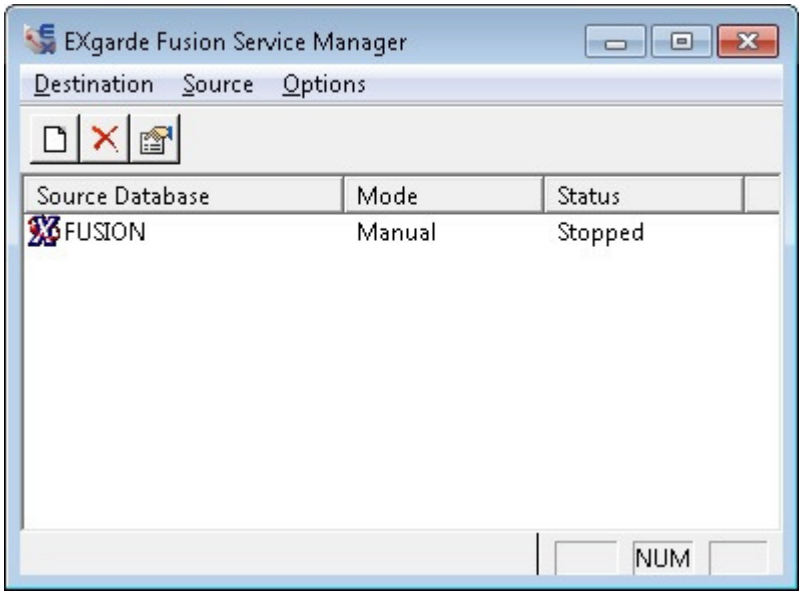
- Any commands executed by Fusion are taken in relationship to objects belonging to the Tenant login as. I.e. when a Key Holder is added then that Key Holder is added to this Tenant.
- The EXgarde event log will show Fusion transactions as Operator events for this particular Operator.
- Fusion will not be able to perform transactions unless this particular Operator has the necessary rights

If you wish, once you have logged in you can set automatic log-in to save you having to log in each time Fusion is launched (see Server Database Properties screen).

# 5. Script Service Manager

## 5.1 Main Screen

When EXfusion is started, you will see the main screen:




This contains three menus: Destination, Source & Options.
The Source menu contains operations related to the Source databases, the EDBs.

- The Destination menu contains operations related to the single destination database, EXgarde. The New, Delete and Properties buttons on the Toolbar are duplicates of the Source menu options.
- The Options menu is used for configuring OPC

## 5.2 Destination Menu

- **Suspend**
  - This will halt all services without exiting Fusion.
- **Hide**
  - This will hide the screen, but Fusion will continue to run. To re-display the screen, re-launch Fusion (i.e. run eXfusion.exe)
- **Properties**
  - This opens the Destination Database screen.
- **Exit**
  - This will close down EXfusion altogether.

## 5.3 Source Menu

- **New**

This opens a blank Source Database screen, which allows you to define a new link to an EDB.

- **Delete**

This will delete the currently selected database service in the list.

- **Properties**

This opens the selected Script Service screen. This allows you to view and control the status of the service as well as altering the Source database configuration.

- **Start & Stop**

This will start or stop polling for the selected service.

## 5.4 Options Menu

- **OPC**

This is used to enable OPC and specify the OPC keyholder group

## 5.5 Source Database list

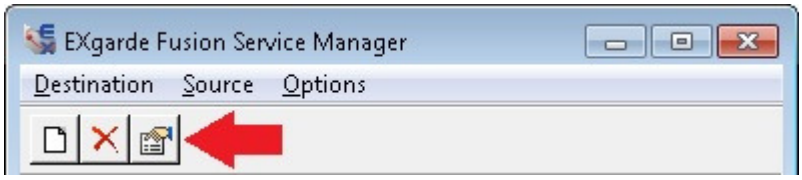
When first started, the list will be empty. Once source database are entered the relevant service is displayed with the run mode and present status. The Delete, Properties and Start/Stop options from the source menu require that the user has already highlighted an entry in this list. Double clicking on an entry automatically displays its Script Service screen.

.

# 6. Script Service

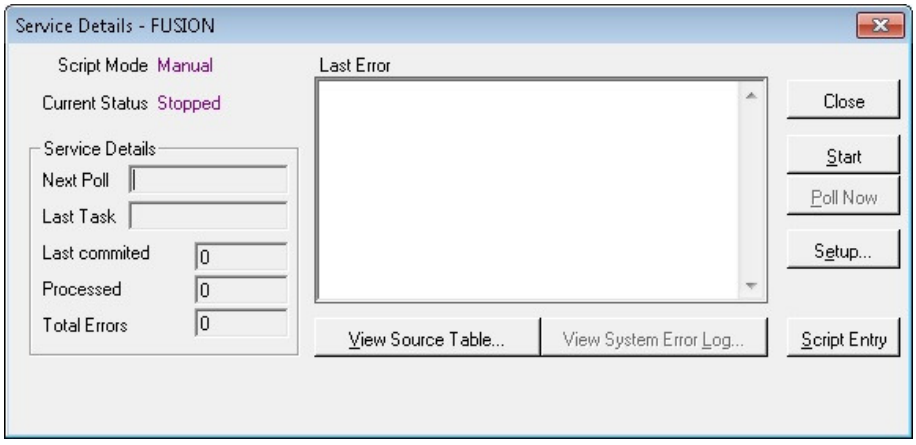
## 6.1 Service Details

The service details screen is obtained by double clicking the service or clicking the toolbar icon.




The screen displays details about the operation of an individual service. The top left hand side shows the script mode and the current status of the service.

This section show details of the commands and polling operation of the service




- **Next Poll**

This is the time that the source command table will be polled for any new commands.

- **Last Task**

The time that the last script command was executed by Fusion.

- **Last committed**

The sequence number of the last successfully executed command.

- **Processed**

The total number of commands processed by the service. This includes successful and unsuccessful commands.

- **Total Errors**

The total number of commands that where unsuccessfully executed.

**Close button**
This closes the window and returns to the Fusion main screen

**Start button**
If the service was set to Manual or was Halted, then this will start the service polling

**Poll Now button**
This will force EXfusion to poll the table and execute any commands, overriding the service poll timer.

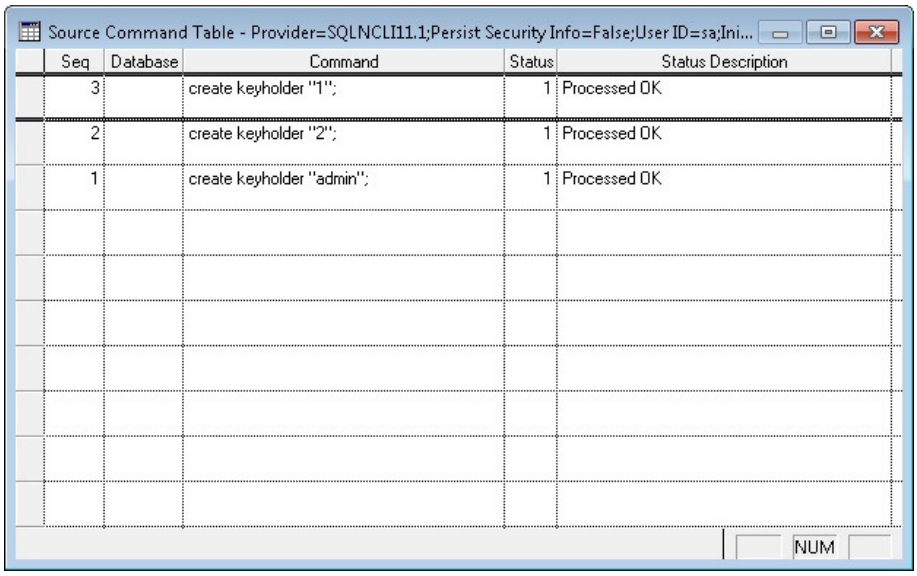
**Setup button**
This will show the Source Database screen to allow alteration of the service setup information.

**View Source Table button**
This will show data currently in the command table on the EDB for this service. If you have chosen the option “Delete Successfully Processed Entries” and a Polling operation has taken place since data was last placed in the table, then the table will only contain error messages.

**View System Error Log**
This will display the error log.

**Fill Source**
This will open a screen, which allows script commands to be directly entered into the command table on the source database.

# 7. Source Command Table




## 7.1 Right-Click Menu

The right-click mouse button allows you to carry out certain functions against the UG_COMMAND table allowing a user to carry out simple administration task from Fusion

- **Refresh**

Updates the table contents with the latest information from the source's UG_COMMAND table.

- **Save Amendments**

The contents of the columns 'Database' and 'Command' can be changed by selecting the cell. Once changed, the new data can be save to UG_COMMAND by this option. This automatically resets the status flag to 0 so the amendments will be executed by Fusion on the next poll.

- **Delete Selected Rows**

Any highlighted rows will be removed from UG_COMMAND table.

- **Clear Status for Selected Rows**

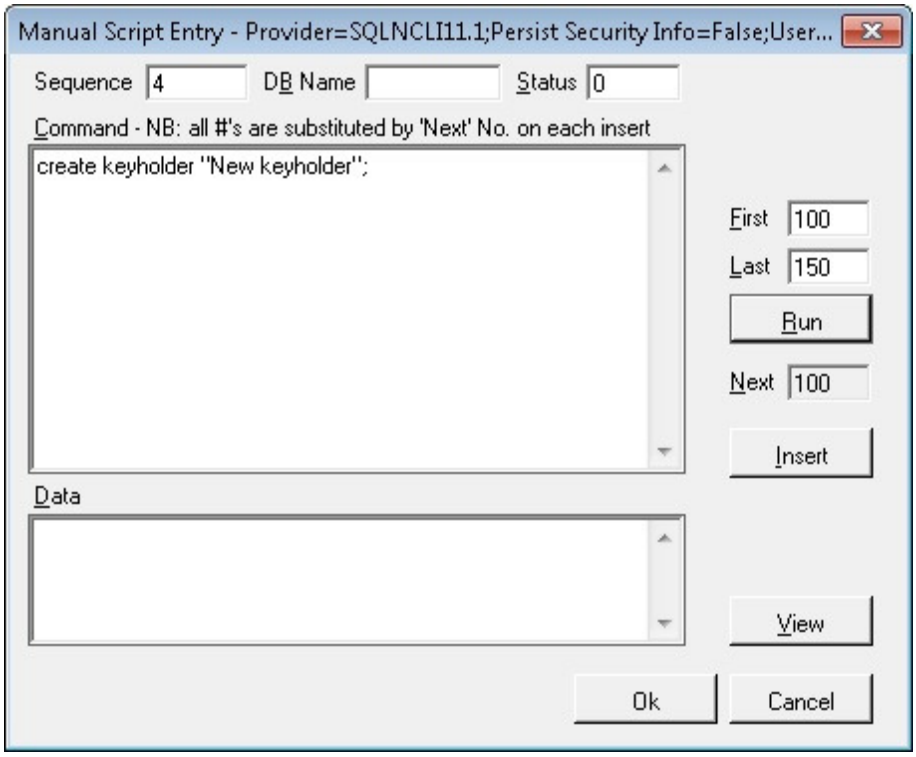
The status of any selected row is reset to 0 and so will be executed by Fusion on the next polling of the table. This is useful if the data in a command was correct but EXgarde did not have the matching data, once EXgarde had been corrected, the originally failed command can be re-run by clearing its status.

- **Select All**

Selects all the rows in the table for use by some of the above commands.

# 8. Manual Script Entry

This screen allows a user to manually enter a command or sequence of commands into the Command table for execution by EXfusion. This a simple method of testing scripts without using an external application. The commands are entered in the format described in Section **XXX** and must be terminated with a semicolon.




A sequence of repeated commands is possible by the use of the wildcard "#" in the script text which is then substituted by a value taken from the range specified in the First and Last number. For example the above command would cause 50 Key Holders called Test100, Test 101 etc., to Test150, to be added to the Group "Everyone" when the Run button is pressed.

- Sequence

This is the command sequence number inserted in the Sequence column in the Command table. This is an automatically incrementing number but the user can override this at any time and the sequence will continue from the user entered number on each insert of a command.

- Database Name

This is the name of the destination database. This is usually blank as default.

- Status

This is the initial status value the command is inserted with. It is usually set to 0 so the command is executed next poll cycle by the service but can be set to Complete (1) or any of the error codes if required.

- First

For automatic sequence runs this is first number in the sequence to be substituted for # in the command string.

- Last

For automatic sequence runs this is last number in the sequence to be substituted for # in the command string.

- Run button

This inserts multiple entries of the command string substituting # with the next number in the sequence range specified by the First and Last fields.

- Next

The show the current value while a sequence is running. It starts at the value of First and runs up to Last after the Run button has been pressed, otherwise it shows the value of First.

- Insert button

This will insert a single entry based on the command string. If the command contains a # then it is substituted by the value of First.

- View button

This displays the Source Table screen, as described in Section **XXXX**, to allow the user to inspect the result of any inserts they have carried out.

- Command Area

This is the area for a user to manually enter a script command. It can be any of the available commands and repeated sequences of commands can be carried out by inserting the character #. The # is then replaced by the next sequence number between the range specified by First and Last.

- Data Area

This allows data to be entered into the Data column of a command. This feature is not used at present.

- OK button

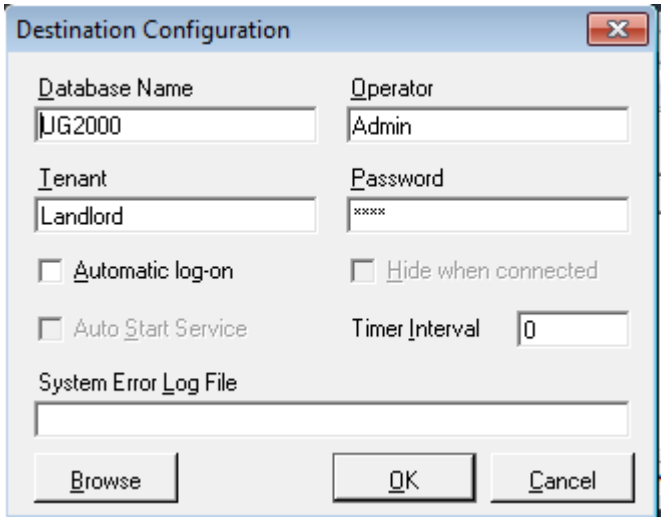
Indicates that the user has completed all data entry and will return to the Service screen. If data has been entered but not committed, then the changes are automatically committed.

- Cancel button

Indicates that the user wish to stop entering data and return to the Service screen. If data has been entered but not committed the user will be asked to confirm whether to commit or not.

# 9. Destination Setup Screen

As well as defining the connection properties to the destination database, this screen can be used to set up unattended log-in when Fusion is launched, and to cause Fusion to be launched when the PC is booted and to hide itself once connected. It also defines the interval between executing commands against the EXgarde system.




- Database

This is the name of the EXgarde database

- Operator

A valid EXgarde Operator within the context of the Tenant

- Tenant

A valid EXgarde Tenant

- Password

The password for the Operator

- Automatic Log-on

If checked, the log-in screen is bypasses when Fusion is launched

- Hide when connected

If checked, the Fusion user interface is hidden after a successful automatic login

- Auto-start service

If checked, Fusion starts when the PC is booted (this does not require a shortcut to Fusion in the Startup group)

- Timer interval

Interval, in $^{1}/_{10}$th of a second, between the execution of each command in UG_COMMAND. I.e. a value of 10 executes a command every second. This is used for all Host databases. This should always be less than the Polling Interval for any Host database.

- System Error Log File

The name (including path) of a log file

- Browse

This allows a user to use the file browser to specify the location of the log file

- OK

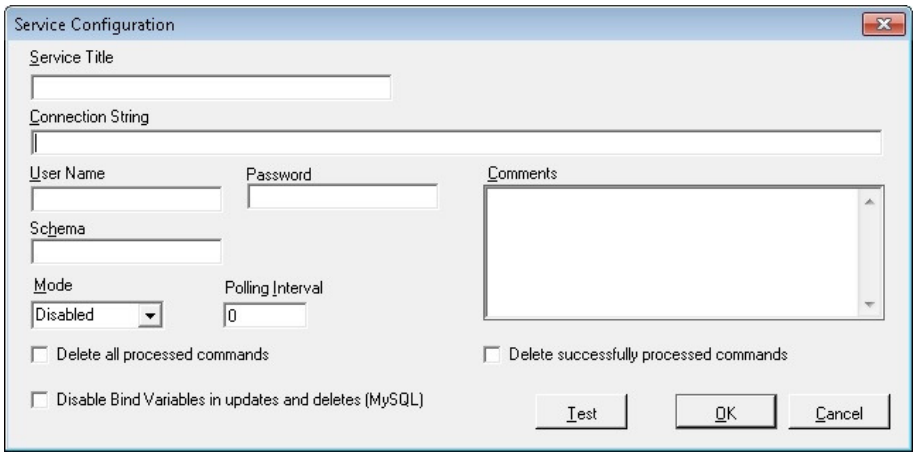
This stores the data entered by the user and returns to the previous screen

- Cancel

This abandons any changes by the user and returns to the previous screen

# 10. Script Service Configuration Screen

The screen defines the connection to the EDB and the operation of the service once connected.




- Service title

Free text

- Connection String

The connection details required should be in the form of an OLE DB initstring.

- Comments

Free text

- User Name

User name (if required) for accessing the host database

- Password

Password name (if required) for accessing the host database

- Schema

Some databases require the specification of particular schema to log on to. If this is required it is entered here otherwise leave this field empty.

- Mode

Is the execution of script commands disabled (DISABLED), started manually (MANUAL) by pressing the start button or started automatically (AUTOMATIC) when Fusion is started.

- Polling Interval

The interval between polling queries to the UG_COMMAND table to check for new entries. Specified in seconds.

- Delete all processed commands

If checked, successfully processed entries will be deleted from the table in the host database

- Delete successfully processes commands

If checked, successfully processed entries will be deleted from the table in the host database

- Test

To confirm that the connection details entered are correct pressing this button will attempt to connect to the database and report whether it succeeded or failed. This command has a long timeout so be patient if it does not respond immediately.

- OK

This stores the data entered by the user and returns to the previous screen

- Cancel

This abandons any changes by the user and returns to the previous screen

# 11. Command Table Definition

## 11.1 Table Definition

Fusion works off a table called UG_COMMAND created on the EDB by the customer. UG_COMMAND is to be defined with the five columns as specified below.

| Column | Name | Data Type | Width | Nulls |
|---|---|---|---|---|
| 1 | CMD_SEQ | INTEGER | 4 | Not Null |
| 2 | DBNAME | CHAR | 8 | Not Null |
| 3 | COMMAND | LONG VARCHAR | 0 | Not Null |
| 4 | STATUS | INTEGER | 4 | Not Null w/Default 0 |
| 5 | DATA | LONG VARCHAR | 0 | OK |

CMD_SEQ : A unique incremental identification number but not necessarily sequential. A specified sequence number method is used to identify a particular row so as to make the system less database specific, as some do not automatically generate a ROWID. The method for generating this number is dependent on the database and will be left to the user to implement all that is required is that Fusion can use the number to order and uniquely identify a single row.

DBNAME : Name of database to execute commands against.

COMMAND : The script command string.

STATUS : Indicates the status of the command. It has a default value of 0 to indicate a non-processed command. On successful execution it is set to 1. If execution was not successful against the EXgarde system then a negative error code is entered.

DATA : Additional data for the command such as a BLOb containing an image. For future expansion only.

The following is an example SQL definition of the Command table. The data types may vary between databases.

```
Create Table UG_COMMAND  ( CMD_SEQ INTEGER NOT NULL,
                           DBNAME CHAR (8)  NOT NULL,
                           COMMAND LONG NOT NULL,
                           STATUS INTEGER NOT NULL WITH
                           DEFAULT,
                           DATA LONG  )
```

## 11.2 Script Command Format

Commands are of the format:

[Command] [Object type] [Object name] [Parameter Name] = [Parameter value] [Parameter Name ] = [Parameter Value]…[Parameter Name ] = [Parameter Value];

## 11.3 Rules

- all objects must be uniquely named; a Create or Modify command that would result in a duplicate will be rejected
- There is an “equals” sign between each [Parameter Name] and [Parameter Value]
- Parameters are optional in the case of a Create command and are ignored in the case of a Delete command.
- Multiple parameters can be concatenated in one command
- Commands, objects and constant values are all non-case sensitive
- All commands must be terminate by a semi-colon
- Any data over the specified length of a data field will be truncated to that length.
- All strings (names and parameter values) should be enclosed in single- or double-quotes
- All key operations MUST include the “***technology***” parameter
- For Keyholder photograph’s UG_COMMAND contains a filename of the image file instead of the binary image. The filename must contain the full path to file from the location of the Fusion application. The image can be in BMP, JPEG, GIF or TIFF.
- To store a photograph of a keyholder, the keyholder must first be created and then modified with the image path specified.
- Keys cannot enter a PIN or validity period for a key until it has been issued, the key can then be modified with the required details.
- A maximum validity duration of 700 days exist. The command will not be executed if the time between ***the from*** and ***to date*** is greater than this or if only a to date is specified that then time between the date of execution and the specified date is not greater than this.
- To set a validity date to unrestricted use a blank value e.g. ***todate =’’***

- Although validity dates can be specified down to the minute this is ignored and the nearest whole day is used e.g. ***fromdate='02/03/03 12:30' todate='05/03/03 13:30'*** the card will be valid from 02/03/03 00:00 to 06/03/03 00:00

- To keep image storage to a minimum all images are reduced to a maximum width of 300 pixels, and will keep their aspect ratio. If the original image is not in the aspect ratio100/110 (width/height) then the image will need to be cropped within EXgarde to be printed correctly.

- The parameters can be in any order except Technology must follow key number.

- AddToCommunity, RemovefromCommunity and SetCommunityInfo all refer to Key Holder groups

- AddToCommunity & RemovefromCommunity affect the Keyholder Group membership of a key holder. If the parameter 'DefaultGroups=YES' is included then the keyholder is automatically added to or removed from the Default access groups of that keyholder group.

- Previous Fusion scripts for keyholders will still work with info1 to info8 placed in tenant info and info9 to info 16 placed in 'All Key Holders' keyholder group.

# 11.4 Commands

| Command | Object type | Object name | Parameter Name | Parameter value |
|---|---|---|---|---|
| **create** | **keyholder** | *(20 characters)* | **longname** | *(50 characters)* |
| | | | **comment** | *(50 characters)* |
| | | | **info1** | *(30 characters)* |
| | | | **info2** | *(30 characters)* |
| | | | **…** | *…* |
| | | | **info15** | *(30 characters)* |
| | | | **info16** | *(30 characters)* |
| **modify** | **keyholder** | *(name)* | **name** | *(20 characters)* |
| | | | **longname** | *(50 characters)* |
| | | | **comment** | *(50 characters)* |
| | | | **info1** | *(30 characters)* |
| | | | **info2** | *(30 characters)* |
| | | | **…** | *…* |
| | | | **info16** | *(30 characters)* |
| | | | **info16** | *(30 characters)* |
| | | | **picture** | *(filename)** |
| **delete** | **keyholder** | *(name)* | | |
| **create** | **box** | *(name)* | | |
| **modify** | **box** | *(name)* | **name** | *(20 characters)* |
| **delete** | **box** | *(name)* | | |
| **create** | **key** | *(8-digit number)* | **keyholder** | *(20 characters)* |
| | | | • b ox | *(20 characters)* |
| | | | **technology** | *(see list below)* |
| **modify** | **key** | *(8-digit number)* | **box** | *(20 characters)* |
| | | | **technology** | *(see list below)* |
| | | | **Status** | *(see list below)* |
| | | | **pin** | *(0000-9999)* |
| | | | **fromdate** | *(see note below)* |
| | | | **todate** | *(see note below)* |
| **delete** | **key** | *(number)* | **technology** | *(see list below)* |
| **issue** | **key** | *(number)* | **technology** | *(see list below)* |
| | | | **keyholder** | *(20 characters)* |
| **addtogroup** | **keyholder** | *(name)* | **group** | *(20 characters)* |
| **removefromgroup** | **keyholder** | *(name)* | **group** | *(20 characters)* |
| **addtocommunity** | **keyholder** | *(name)* | **community** | *(20 characters)* |
| | | | **defaultgroups** | *Yes/No* |
| **removefromcommunity** | **keyholder** | *(name)* | **community** | *(20 characters)* |
| | | | **defaultgroups** | *Yes/No* |
| **setcommunityinfo** | **keyholder** | *(name)* | **community** | *(20 characters)* |
| | | | **info1** | *(30 characters)* |
| | | | **…** | *…* |
| | | | **info8** | *(30 characters)* |

**Only .jpg, .bmp and .gif image types are recognised by EXgarde.*

# 11.5 System Specific Data

The following are a list of constants or the Technology and the Status parameter in Key commands. It should also be noted that the format of dates used within Key commands should specified in the same format as the System Locale settings of the PC on which Fusion is installed on.

**Key Technologies**

‘Microcard’
’ASR Prox’
’Magnetic’
’TDSi Wiegand’
’26-bit Wiegand’
’Keypad’
’37-bit Wiegand’
’34-bit Wiegand’
‘Octopus 44-bit Wiegand’
‘Me Lucky 34-bit Wiegand’
‘Universal Decode’

**Key status**

‘Available’
’In use’
’Lost’
’Damaged’
’Suspended’

# 11.6 Status Codes

| | |
|---|---|
| 0 | (Unexecuted) |
| 1 | Processed OK |
| -1 | End of file |
| -2 | Error |
| -3 | Unknown command |
| -4 | Unknown object type |
| -5 | Unknown parameter name |
| -6 | Key Holder already exists |
| -7 | Key Holder does not exist |
| -8 | Cannot delete Key Holder |
| -9 | Cannot modify Key Holder |
| -10 | Box already exists |
| -11 | Box does not exist |
| -12 | Cannot delete Box |
| -13 | Cannot modify Box |
| -14 | Key already exists |
| -15 | Key does not exist |
| -16 | Cannot delete Key |
| -17 | Cannot modify Key |
| -18 | Cannot set Key status |
| -19 | Key technology missing |
| -20 | Key number missing |
| -21 | Box name missing |
| -22 | Key Holder name missing |
| -23 | Invalid Key Technology |
| -24 | Invalid Key |
| -25 | Invalid PIN Number |
| -26 | Invalid From Date |
| -27 | Invalid Until Date |
| -28 | Invalid Key State |
| -29 | Access Group does not exist |
| -30 | Missing Group name |
| -31 | Key Holder is Already in Group |
| -32 | Unable to add Keyholder to Group |
| -33 | Unable to remove Keyholder from Group |
| -34 | Insufficient Operator rights |
| -35 | Unable to open image |
| -36 | Image conversion failed |
| -37 | Unsupported image format |
| -38 | Key is un-issued |
| -39 | Validity duration exceeds maximum |
| -40 | Key Holder Group does not exist |
| -41 | Missing Key Holder Group name |
| -42 | Key Holder is Already a member of Key Holder Group |
| -43 | Unable to add Key Holder to Key Holder Group |
| -44 | Unable to remove Key Holder from Key Holder Group |
| -45 | Key Holder is Not a member of Key Holder Group |

## 11.7 Command Examples

*create keyholder “bob” longname = “bob smith” comment = “Test person” info1 = “Student” …. info16 = “Aardvarks”;*

*modify keyholder "bob" picture = "C:\bobsmith.bmp”*

*delete keyholder “bob”;*

*create box “Visitor”;*

*modify box “Visitor” “Visitors”;*

*delete box “Visitors”;*

*create key 1111 technology = “Proximity”;*

*modify key 1111 technology = “Proximity” keyholder = “fred” box = “Vistitors” pin = “1234” fromdate = “26/05/2001” todate =”28/05/2001”;*

*delete key 1111 technology = “Proximity”;*

*issue key 1111 technology = “Proximity” keyholder = “fred”;*

*addtogroup keyholder = “bob” group = “Students”;*

*removefromgroup keyholder = “bob” group = “Students”*

## 11.8 ODBC Connections

The most common method of connecting Fusion to an EDB is via ODBC. As well as declaring the Data Source within Window’s ODBC Data Source Administrator the SQL.INI file within the EXgarde directory will require updating. The following changes enable the use of an ODBC connection and specify the details of the EDB.
In the section [win32client] add the following line where EDBalias is alias associated in the [odbcrtr] section. This is the Database name to be specified in the Service Configuration screen.
**dbname = *EDBalias*,sqlodb32**

In the section [win32client.dll] add the following line to enable odbc connections for the client. Also check that the file sqlodb32.dll is installed in the EXgarde directory.
**comdll=sqlodb32**

Create a new section at the bottom of the file by typing
**[win32client.odb32]**

No other data goes in this section but under that another new section is required with the following entry. This establishes the alias for Fusion to use

when referring to the Data Source where *DataSourceName* is the name specified with the ODBC Data Source Administrator.

**[odbcrtr]**
**remotedbname=*EDBalis*, DSN=*DataSourceName***
The following is an example of a SQL.IBI file from a stand-alone installation of EXgarde that has been update to include a Data Source called ‘My Thingy Database’ which will be referred to as ‘Fred’.

# 11.9 Implementation

The customer's application can be altered to enter the required commands into the UG_COMMAND table but a more seamless method is to use database triggers to create the appropriate commands when particular changes occur to the EDB via normal operation. This allows the customer's application to remain unchanged but still to be able to pass information to the EXgarde system.

For example if the EDB is a personnel system, on the insertion of a record in the person table a trigger would insert a "create keyholder" command in UG_COMMAND with data extract from the EDB.
The following are example triggers taken from TDSi's Visitor Management System which is based on a SQLBase database called Visitors.

The triggers use two main database procedures *GetUniqueID* and *UGCOMMAND*. *GetUniqueID* retrieves a unique sequence number, using SQLBase's, *sysdbsequence* functionality, to be inserted into CMD_SEQ and *UGCOMMAND* inserts the final row into the UG_COMMAND table. This procedural method was used as these steps are repeated for each trigger but could be coded in each trigger if procedures are not available.

```
PROCEDURE GetUniqueID static
parameters
 Receive Number: nUniqueID
local variables
 Sql Handle : hSql
 Number: nSeq
 Number: nFetch
actions
 On Procedure Startup
  Call SqlConnect( hSql )
 On Procedure Execute
  call SqlPrepareAndExecute(hSql, 'select sysdbsequence.nextval from
syscolumns into :nSeq'  )
  call  SqlFetchNext (hSql, nFetch )
  set nUniqueID= nSeq
  call SqlCommit(hSql)
  return 0
 On Procedure Close
  Call SqlDisconnect( hSql )


PROCEDURE UGCommand static
parameters
 Long String: sCommand
local variables
 Sql Handle : hSqlSeq
 Sql Handle : hSqlCmd
 Number: nSeq
 Number: nFetch
 Long String sData
 String: sDBName
actions
 On Procedure Startup
!
!        connect cursors and retrieve stored procedures
```

```
!
  Call SqlConnect( hSqlCmd )
  set sDBName='UG2000'
  Call SqlPrepare( hSqlCmd , 'insert into UG_COMMAND
(CMD_SEQ,DBNAME,COMMAND,STATUS) values (:nSeq, :sDBName,
:sCommand, 0 ) ' )
  Call SqlConnect( hSqlSeq )
  Call SqlRetrieve(hSqlSeq,  'GetUniqueID', ':nSeq', ':nSeq' )
 On Procedure Execute
  Trace sCommand
  if sCommand = STRING_Null
   return 0
!
!         Get unique ID for new command
!
  Call SqlExecute( hSqlSeq )
  Call SqlFetchNext( hSqlSeq, nFetch )
!
!         Insert new command
!
  Call SqlExecute( hSqlCmd )
  return 0
 On Procedure Close
  Call SqlDisconnect( hSqlSeq )
  Call SqlDisconnect( hSqlCmd )
```

The following is a simple trigger called key_insert placed on the Visitors table key which on the insertion of a new record inserts a “create key” command in UG_COMMAND. The command uses the number and technology fields from that record and has the keybox hard-coded to ‘Visitors’ and the status to ‘Available’

```
create trigger key_insert after insert on key
 ( execute UGCOMMAND ( 'create key ' || number || ' box="Visitors" technology = '
|| technology || ' status = Available;')
 )
for each row
```

The following creates a keyholder when a new visitor is created. Very similar to the above key creation but to simplify the viewing of the UG_COMMAND the keyholder creation is split into two commands. The first creates the keyholder including the first two info fields and the next modifies the same keyholder and enters info fields 3 & 4. It is not necessary to split the command like this.

```
create trigger visitor_insert after insert on visitor
 ( execute inline ( name, company, vrn, phone, unique_id)
procedure CreateVisitor static
parameters
 String: sName
 String: sCompany
 String: sVrn
 String: sPhone
 String: sUniqueID
local variables
 Sql Handle : hSqlCmd
 String: sVisitorGroup
 Long String: sCommand
actions
 On Procedure Startup
!
```

```
!         connect cursor and retrieve stored procedure
!
  Call SqlConnect( hSqlCmd )
  Call SqlRetrieve( hSqlCmd ,  'UGCOMMAND', ':sCommand', '' )
 On Procedure Execute
  set sCommand = 'create keyholder "' || sName || '" info1 = "' || sCompany || '" info2
= "' || sVrn || '";'
  Call SqlExecute( hSqlCmd )
  set sCommand = 'modify keyholder "' || sName || '" info3 = "' || sPhone || '" info4 =
"' || sUniqueID || '";'
  Call SqlExecute( hSqlCmd )
  set sVisitorGroup = 'Visitors'
  set sCommand = 'addtogroup keyholder "' || sName || '" Group= "' || sVisitorGroup
|| '";'
  Call SqlExecute( hSqlCmd )
  return 0
 On Procedure Close
  Call SqlDisconnect( hSqlCmd )
)
for each row
```

Non-directly related tables could use multiple calls. For example for an entry to the Visit table, the trigger modifies the key status and adds the keyholder to a group. The flexibility and amount of automation that can be obtained with Fusion is only reliant on the strength of the trigger language of the EDB and the amount of relevant information it contains.

Time and Data Systems International Ltd
Unit 10 Concept Park
Innovation Close
Poole
Dorset
BH12 4QT
UK

+44 (0)1202 723535

+44 (0)1202 724975
http://www.tdsi.co.uk/

sales@tdsi.co.uk
marketing@tdsi.co.uk
support@tdsi.co.uk