mv.NET Core Objects
Getting Started

A product from BlueFinity

Copyright Notices

Document ref: mvNET_GS
Revision 4.6.0

Contacting Us

We are always very happy to be able to discuss all aspects of our products with our customers - prospective and current alike. You can contact us via the following means:

Website:	www.bluefinity.com
Email:	support@bluefinity.com
Address:	10260 SW Greenburg Road, Suite 700, Portland, OR 97223, USA
Address:	575-599 Maxted Road, Hemel Hempstead, Herts, HP2 7DX, UK

Trademark Acknowledgements

The mv.NET product and logo are trademarks of BlueFinity International.

All other trademarks and trade names are the property of their respective owners and are used in this documentation for identification purposes only

Contents

Welcome to mv.NET

Firstly, thank you for either purchasing one or more of the mv.NET products, or for taking the time to explore the great functionality that they can provide to you and your fellow developers.

This chapter outlines the members of the mv.NET family of products and also summarizes the contents of this guide.

The mv.NET Family of Products

Core Objects is one of the members of the mv.NET family of products authored by BlueFinity. mv.NET is ***the*** essential tool for any Multi-value database developer wishing to create .NET based application interfaces to their current or new multivalue database file system.

Core Objects not only provides the underlying framework upon which all other mv.NET products are based - it also provides a wealth of end-user capabilities to allow the developer to rapidly create feature-rich, high performance applications using the powerful tools provided by Microsoft's .NET environment.

The design goal of mv.NET is to enable the Multi-value developer to combine the power and flexibility of proven Multi-value technology with the state-of-the art, feature rich .NET environment. Its design also enables and encourages the developer to leverage, wherever possible, previously acquired multivalue skills.

BlueFinity's team of software engineers has huge knowledge and experience of using both multivalue systems and the .NET environment. We proudly regard ourselves as being one of the foremost companies in providing this technology

bridge and look forward to working with you to enable you to meet your software development goals.

Feature Overview

The Core Objects product provides a 100% native .NET interface to all Multi-value database platforms, allowing .NET developers to access all aspects of Multi-value systems – both data and program code – from within their .NET application.

The Core Objects architecture has been designed with both performance and flexibility in mind. This, combined with an implementation that provides seamless integration with the .NET environment, provides a powerful tool for enabling Multi-value developers to harness the full power of both their Multi-value system and the .NET platform.

Core Objects also has strong integration with Microsoft's Visual Studio.NET product, allowing the Multi-value developer to carry out virtually all aspects of application creation from within the VS.NET environment.

The product's key features are as follows:

- Feature-rich, Multi-value data structure aware data objects authored in 100% managed .NET code.
- High performance connections from client to database server using a variety of transport technologies dependent on flavor of Multi-value database platform.
- Sophisticated session pooling and session sharing functionality.
- Advanced fetch-on-demand and background data retrieval technology, ensuring maximum application database performance
- Support for all major multivalue platforms.
- Support for stateless applications, e.g. Web Services, featuring optimistic locking, automated state retention/reconnection and database connection pooling.

- A fully compliant ADO.NET managed data provider implementation allowing Multi-value data access from non-mv aware 3rd party products.

The mv.NET Suite

Core Objects is one of three products within the mv.NET suite; the suite comprising of:

- **Core Objects** - object oriented native .NET access to Multi-value databases.
- **Solution Objects** - Strongly-typed class-based access to your MultiValue database.
- **Adapter Objects** - complete implementation of an ADO.NET managed data provider for multivalue databases, offering a standardized interface to database access.

Getting Started Guide Contents

The contents of this guide are designed to allow a developer to easily install and perform the initial configuration activities of the Core Objects product. A summary of each chapter follows:

Client-side Installation

This chapter takes you through the process of installing Core Objects on the developer's workstation.

Defining Server Profiles

This chapter covers the process of creating a server profile to define the type and location of your Multi-value database server(s).

The Server Console Window

This chapter explains the process of installing the Core Objects server-resident components on the Multi-value database server using the Data Manager's Server Console Window.

Defining Account Profiles

This chapter describes the process of creating an account profile to define the accounts that you wish to connect into on your Multi-value server(s).

Testing Connections

This chapter explains how to test the connection to your Multi-value server prior to establishing a full-blown session with a data/application account.

Defining Login Profiles

Login profiles allow you assign a logical name to a server and account profile pairing. This chapter explains why this capability has been provided and explains how to create and maintain login profiles.

Product Licensing and Activation

In order to access the full functionality of mv.NET you will need to enter licensing details and request product activation codes. This chapter explains the principles of mv.NET licensing and takes you through the screens associated with this activity.

Installation Procedure Summary

The installation of Core Objects involves several discrete steps. These may be summarized as follows:

1. Run the CIDSetup.exe routine on your development workstation/system.
2. Use the Data Manager utility program (which will have been installed as part of the above step) to create one or more server profiles to define the type and location of your Multi-value database server(s).
3. Use the Data Manager's Server Console Window to establish a terminal emulation session to the database server and use its server components download option to install the product's server-side routines.
4. The Server Console window should then be used to 'enable' each of the application accounts that you wish to access via mv.NET.
5. If you wish, the Server Console window can also be used to download mv.NET's SOP demo account which is used by some of the sample applications supplied with the product

6. After downloading the server routines and enabling all your data accounts, you should then, using the Data Manager, create an Account Profile for each of the accounts that you have enabled.

7. After creating an Account Profile, you should then use the Data Manager's Test Connection option to make sure that you can successfully connect into the account. Once this connection test completes successfully, you are then ready to start using Core Objects.

8. Create login profile entries to pair together server and account profiles under a logical name.

9. After you have defined server and account profiles and successfully connected into your database account, you will be able to license and activate the client and server components of mv.NET.

Client-side Installation

The installation of Core Objects begins with the execution of its main setup routine on your development workstation. This chapter takes you through this process and details the components installed.

The Client Interface Developer Setup Routine

The routine required to install Core Objects onto your development workstation is CIDSetup.exe. Your supplier of mv.NET will have supplied this routine to you or will have provided you with details of the location from which to download the latest version.

On executing CIDSetup.exe, a series of on-screen prompts are displayed. Finally, an installation password is required after which the suite of Core Objects components is installed.

Installed Components List

When the setup process has completed, the following components will have been installed onto your system:

- The Core Objects class library (and supporting executables and assemblies)
- 2 Windows services used to control session pooling and license management
- Example .NET projects illustrating the use of Core Objects
- Visual Studio integration components
- The Data Manager utility
- Sample Server and Account profile definitions (accessible via the Data Manager)

High Level Initial Overview

In order to get up and running with mv.NET, you will need to perform a small amount of initial setup work. This chapter presents a summary of this work in order to allow the reader to see the 'big picture' before starting out.

All of the topics covered in this chapter are described in detail within subsequent chapters in this document.

The Data Manager

mv.NET provides a utility called the "Data Manager". This is one of the most important utilities that comes with the product as it allows you to not only define database server connectivity information but also allows you to connect into servers and access the data and programs contained within accounts on that system. A short-cut to the Data Manager will have been created by the CIDSetup routine within your mv.NET Start menu list.

Database Connectivity and Profiles

To allow you to define how mv.NET connects into the various accounts contained within your database, mv.NET uses the concept of "profiles". There are 3 different types of profiles:

Server profiles - these allow the basic connection details to a database server to be defined. The details in this profile are used when accessing any account on that server.

Account profiles - the information within this type of profile is divided into 2 main parts. The first part defines the missing bits of the connection definition that allows mv.NET to know how to connect into a specific account on the database server. The second part allows control information relating to how mv.NET manages connections to that account to be defined.

Account profiles are defined within the context of a parent server profile. There may be many account profiles for each server profile and, in fact, this is the reason why mv.NET implements this definition model. That is, by having the concept of a parent server profile, the information within it only needs to be defined once, irrespective of which account is being connected to on that server.

Login profiles - these profiles simply provide a means of associating a logical name to a server/account profile pairing. They are provided as a means of abstracting the names of actual servers and accounts so that these physical names are not accidentally hard coded into an application.

The Data Manager allows you to create and maintain all these profiles. This activity is described in detail in dedicated chapters within this document.

Server-side Routines

As well as installing software within the Windows (.NET) environment, to connect into a database, mv.NET requires that some BASIC routines are pushed onto the database server. These routines act as the database-side footprint of the product and are the components which handle the physical database actions being triggered by .NET code.

To allow these BASIC routines to be pushed on a database server, mv.NET provides a terminal emulation window which allows you to login and navigate to command

level within the account which you wish to host these server-side routines. Once at this position, the terminal emulation window can be used to automatically download all of the relevant routines. The name given to this terminal emulation window is the "Server Console Window".

Testing Connections

Once you have defined a server and account profile and have downloaded the server-side routines you are able to test the connection. The Data manager provides a quick an easy to use option to allow to perform this task.

Product Activation and Licensing

To start accessing the contents of your accounts using the Data Manager, you will have to either start your evaluation period or enter an activation code supplied by your product supplier. The chapter at the end of this document covers this process in detail.

Defining Server Profiles

The first task to perform after installing the client-side components is to create a definition describing the type and location of your Multi-value database server. This set of information is collectively known as a 'Server Profile'. A server profile holds most of the information needed by Core Objects to successfully connect into a Multi-value database server. This chapter takes you through the creation of a new server profile and explains the different sets of definitions that comprise its content.

Running the Data Manager

Creating a server profile will be the first of many tasks that you perform using mv.NET's Data Manager utility. This program is an important part of mv.NET, as it not only provides the means to perform a wide range of administration tasks but can be actively used as an integral part of your application development process.

A standalone version of the Data Manager will have been installed as part of the CIDSetup.exe routine and a shortcut to it will have been placed in your Start\Programs\BlueFinity\mv.NET menu.

Upon running the Data Manager, the following form will be displayed:

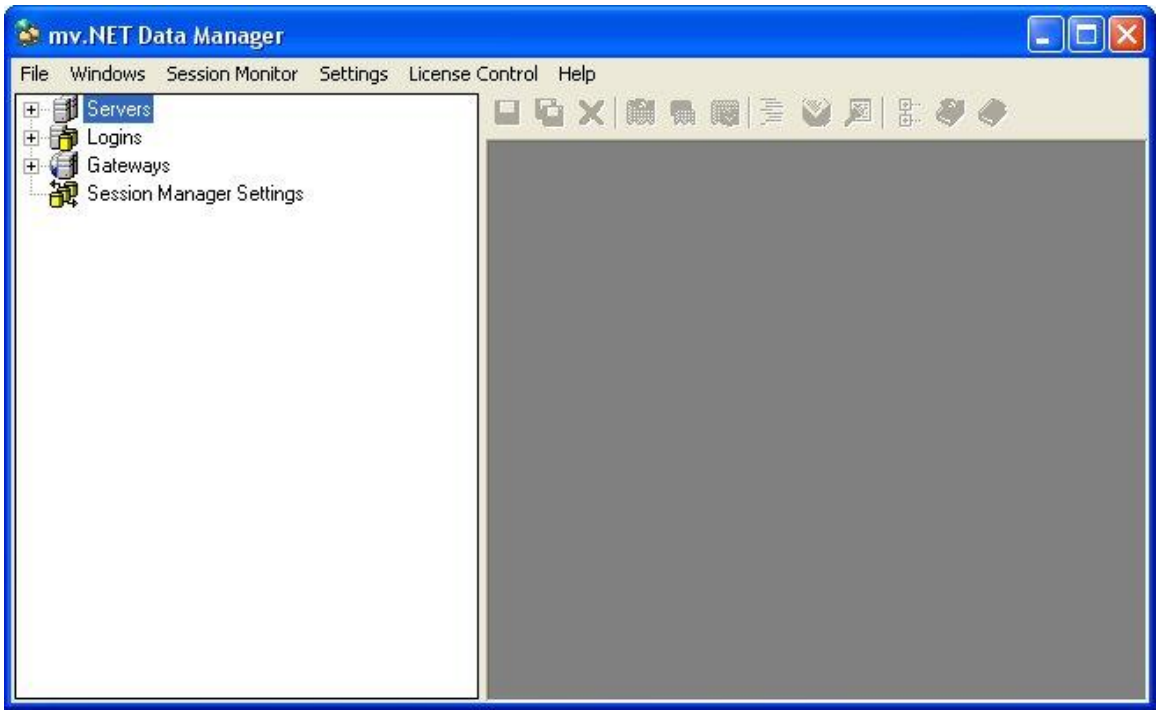

Diagram 1: The Data Manager Explorer

The left-hand section of the Data Manager window contains a treeview comprising of several types of nodes. The node of interest to us at this stage is the Servers node. At the top right of the Data Manager is a toolbar that contains a range of buttons that become enabled at various points during the use of the Data Manager.

On expanding the Servers node, you will see a list of sample server profiles that have been installed as part of the setup procedure. At this stage you have a choice – you may use and adapt one of the sample profiles, or, create a new one from scratch. To edit one of the existing profiles, right-click the relevant node and select 'Edit Server Profile'. To create a new profile, right-click the Servers node and select 'Add Server Profile'. You will then be prompted for the name of your new profile.

The Server Profile Maintenance Window

Irrespective of whether you create a new server profile or whether you edit an existing one, you will be displayed the Server Profile Maintenance window.

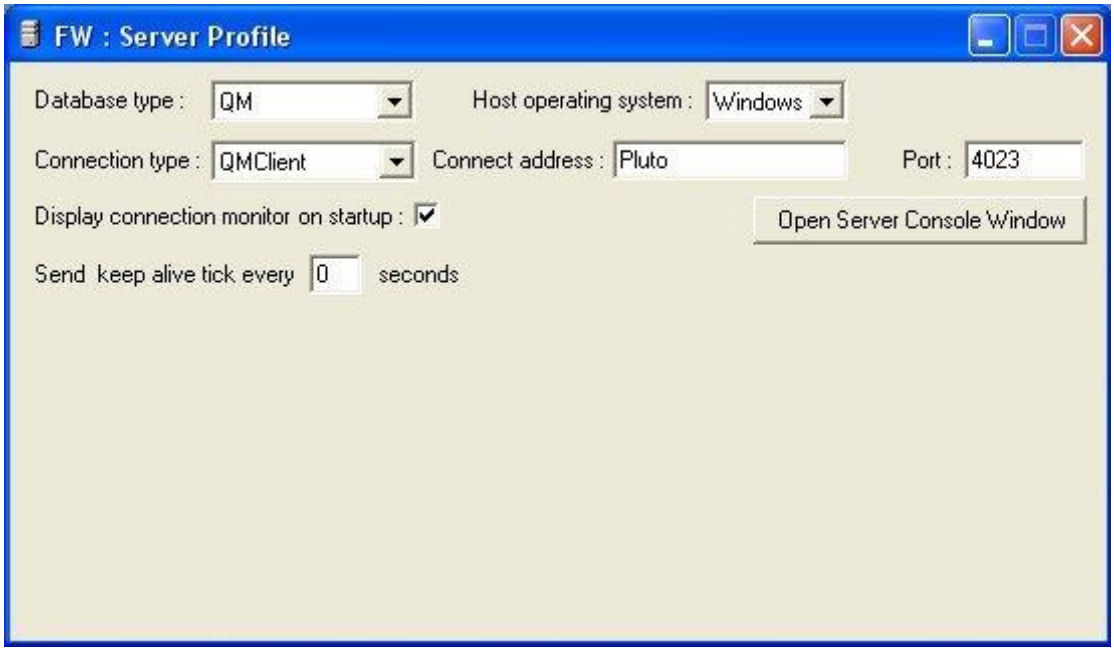

Diagram 2: The Server Profile Maintenance Window

The top section of this form allows you to define the general aspects of your database server. The various fields in this section are explained below:

Database type: This allows you to specify the 'flavor' of Multi-value platform that this database installation represents.

Host operating system: This allows you to specify the operating system running on the server.

Connection type: This allows you to specify the type of connection that needs to be established with the server. The options available here will vary depending on the *Database type* selected.

Connection address: This should be set to the address of the server. This can be either an IP address or a resolvable system name.

Port: This allows you to (optionally) specify the port of the listening service for the specified connection type. See next section for more details. For some connection types this field is not required and will be hidden.

Display connection monitor on startup: If this field is ticked, a connection monitor window will be automatically displayed when a new connection to this server is established.

***** Note, the Session Monitor application must be running for this to occur *****

Send keep alive tick: This input field allows you to indicate that traffic is to be automatically generated between client and server in order to avoid session inactivity timeout from occurring. By specifying a non-zero number in this field, a small number of characters will be transmitted from client to host every so often to prevent the communications link from timing out.

The **Open Server Console Window** button in the top section of this window will open a terminal emulation window onto the server. Please refer to the next chapter for more details.

Connection Specific Details

Below is a table detailing the way in which various fields within the server profile should be used depending on the selected connection type.

Connection Type	Field	Description
IP	Port	The port number of the Telnet listener. Defaults to 23 if left blank **Or** The port number of the SSH listener, prefixed by the letters "ssh", e.g., "ssh22" indicates an SSH-based connection on port 22.
UniObjects	Service	The name of the UniObjects listening service. Defaults to 'udcs' for Unidata and 'uvcs' for UniVerse if left blank. Note, the 'Port' prompt changes to a 'Service' prompt for UniObjects-based definitions.
SAC	Port	The name of the UniVision service. Defaults to 'UniV' if left blank.

Defining Connection Negotiations

Within the server profile definition form, if you select a connection type (for example 'IP') that requires a negotiation sequence to be followed in order to gain access to the database account, an extra region of input fields will be presented in the first tab of the form:

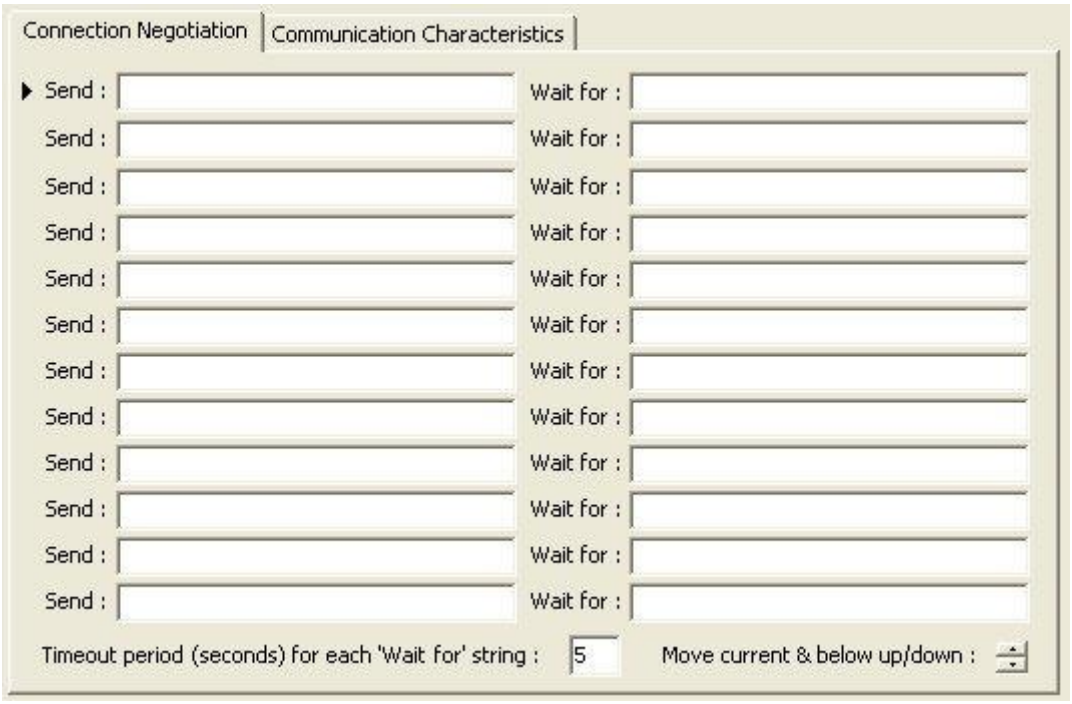

Diagram 3: Connection Negotiation Fields

The various fields within this section of the form are explained below:

Send/Wait for: These input fields allow you to specify the character strings to be sent to the server to gain access to the required account. After each send string you may specify a character sequence that must received within the stream of characters coming back from the server before sending the next send string. In such a way you can correctly synchronize the transmission of send strings to the host.

Within the Send and Wait for input fields you may include various special 'marker' strings:

{account} – will insert the *Account* field from the relevant account profile. When you logon to a server, you will always need to specify both a server and an account

profile name. The account profile definition contains an *account* field – it is the value of this field which is inserted in place of the {account} marker at run time.

{user} – will insert the *User* field from the relevant account profile.

{password} – will insert the *Password* field from the relevant account profile.

{password#2} – will insert the *Password#2* field from the relevant account profile.

{prompt#1} – will insert the *Prompt#1* field from the relevant account profile.

{prompt#2} – will insert the *Prompt#2* field from the relevant account profile.

{sleep} – will cause the connection negotiation process to sleep for 1 second. The word sleep may be followed by a space and then an integer number to sleep for more than one second; e.g. {sleep 3} will cause a sleep for 3 seconds.

~n – will insert character n; e.g. ~13 will insert a carriage return character.

Note that the Wait for strings ***are*** case sensitive.

The send/wait for pairings need to navigate the logon process down to command level within the target account, at which point the (final) send string of 'MVNET.START~13' should be sent with no wait for after it. This command starts the mv.NET IP listener process, after which, mv.NET will take over the negotiation process and eventually place the listener into a 'ready' state.

Timeout period: Allows you to specify the maximum number of seconds that a Wait for string will be waited for. If a Wait for string does not appear in the received characters buffer within this number of seconds after its associated send string has been transmitted, the login process will be abandoned.

It is, obviously, important to get the send and wait for strings absolutely correct, otherwise Core Objects will get lost part way through the connection process and will fail to establish a link to the server. To that end, there is a button on the server profile maintenance window named 'Open Server Console Window'. Clicking this button will launch a simple terminal emulation window which, amongst other things, will allow you to observe the exact sequence and character content of the series of prompts and messages that the Multi-value system displays during the connection process.

The spin up/down button in the bottom right of this window allows you to insert and remove lines mid-way within the send/wait pairs. The arrow-head in the left-hand edge of the tab indicates the "current" row.

Connection Control

At the foot of the connection negotiation tab is an area which allows you to specify settings that control the creation of new connections to this server:

Diagram 5d: Connection Control Settings

The first 3 input fields allow you control the suspension of new connections to this server based on the rate of new connection attempt failures.

The last 2 input fields allow you to restrict the rate of new connection attempts based on a fixed allowance for a specified period of time.

The purpose of these input fields is to allow you to control the behavior of the Session Manager in situations where a database server is encountering network connection problems or where it is under a high degree of stress. In such situations it is common for connections to the database server to become unstable and so the above settings allow you to restrict the impact that mv.NET has on the database server. That is, in such situations, usually the last thing that the database server needs is extra workload associated with trying to establish new connections continually.

Communication Characteristics

In addition to specifying connection negotiation details, the second tab allows the definition of communication characteristics:

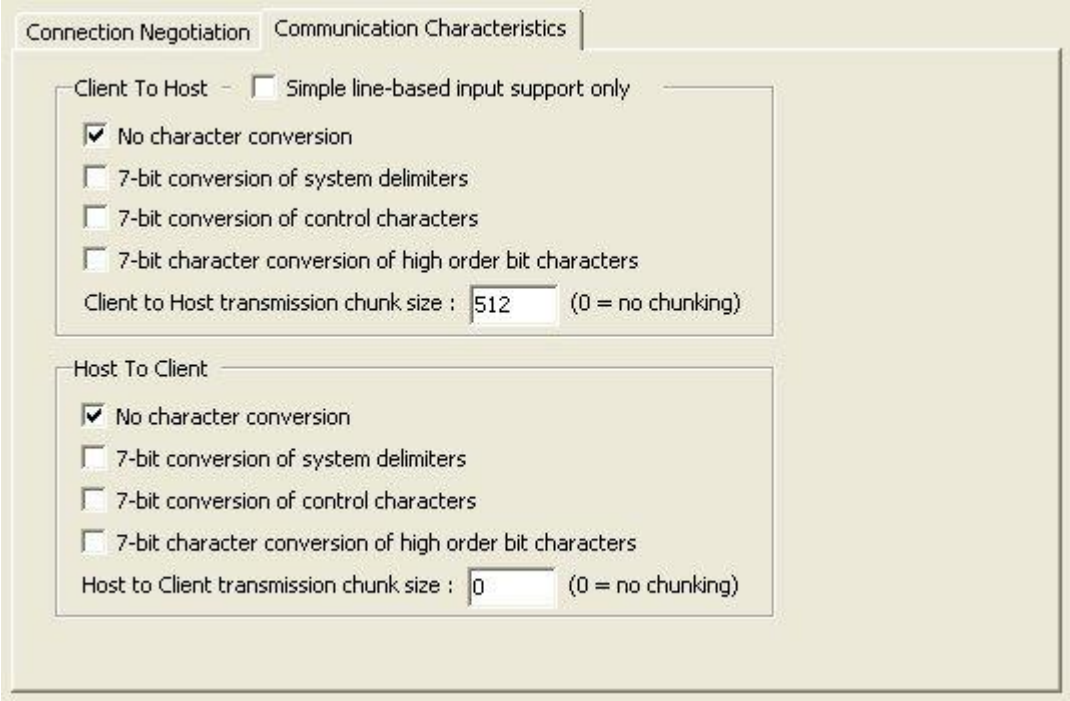

Diagram 4 : Communication Characteristics

The various fields on this form are explained below:

Client to Host: These input fields allow you to specify the nature of the communications link to the server in terms of data flow from client to server (host). The check boxes allow you to define 7/8 bit characteristics. The **chunk size** input field allows you to specify the maximum number of bytes that may be transmitted in a single uninterrupted burst to the host.

Host to Client: These input fields allow you to specify the nature of the communications link to the server in terms of data flow from the server (host) to the client system. The check boxes allow you to define 7/8 bit characteristics. The **chunk size** input field allows you to specify the maximum number of bytes that may be transmitted in a single uninterrupted burst from the host.

The following table describes the effect of ticking each checkbox

Checkbox Name	Effect
No character conversion	All characters are transmitted without any form of conversion being applied.
7-bit conversion of system delimiters	All characters in the ASCII range 250-255 are converted into 7-bit forms.
7-bit conversion of control	All characters in the ASCII range 1-31 are

characters	converted into 7-bit forms.
7-bit conversion of high-order bit characters	All characters in the ASCII range 128-255 are converted into 7-bit forms.

Note, the following scenarios may require the use of character conversion:

- situations where IP connection types are being affected by stty and other terminal control settings on the database host
- where the Windows system hosting the mv.NET Session Manager service is configured to use an Eastern European or Slavic (Cyrillic alphabet) based language setting
- NLS support is being used on a U2 database

Finally, the "Simple line-based input support only" checkbox allows you to indicate whether the database platform only supports old-technology line-based input capabilities. This should usually be left unticked unless you have specific reasons to believe that it applies to your particular database server.

Preparing Your Database Server

The series of actions that you will need to perform before loading mv.NET's server-side components onto your database server will vary according to the type of platform you are using. Basically, there are up to 4 steps, some of which will only be relevant to some platforms:

1. Create an MV.NET account
2. Create an SOP account (if you want to install the demo account)
3. Create an MV.NET Windows/UNIX user to be used to login to the MV.NET account
4. Create an SOP Windows/UNIX user to be used to login to the SOP account

The following sections contain directions for each supported database platform. The following chapter covers the topic of downloading the server-side components.

NOTE: The MV.NET account that you create is only used as a holding place for the mv.NET server routines. You do not login to this account directly, thus there is no need to create an account profile for this account.

D3

Logged in as the sysprog user, create 2 accounts called MV.NET and SOP. Use the update-md command to create login ids for both accounts.

jBASE

For Telnet-based Connectivity

Create 2 Windows/UNIX folders to house the MV.NET and SOP accounts. Create 2 Windows/UNIX users called MV.NET and SOP. Set the 'Local Path' for each of these users to the relevant folder created above. Create the appropriate remote.cmd file within each of these accounts, making note of the following guidelines:

1. PATH must contain \MV.NET\bin
2. JBCOBJECTLIST must contain \MV.NET\lib
3. Must have a JEDIFILENAME_MD
4. Must have a JBCLISTFILE
5. JEDIFILEPATH must contain \MV.NET or must Q-Point to MVNET.READIMAGES, MVNET.CONTROL and MVNET.CONNECTS

For the MV.NET account

1. JBCDEV_BIN to point to \MVNET\bin
2. JBCDEV_LIB to point to \MVNET\lib

For jRemote/jAgent Connectivity

In order to be able to use jRemote as a means of connecting to jBASE 5, a number of preparatory steps are necessary on the jBASE server.

Windows

As Administrator:

1. Install and start jbase_agent as a service with Account authentication:
   a. jbase_agent install -A account
   b. jbase_agent start
2. Create a JAGENT_USER file in the root of the Administrator folder, i.e. "C:\Users\Administrator" on Windows 7 or Windows 2008:

a. CREATE-FILE JAGENT_USER 3 11

3. Add a User ID and Password for each user account:

   a. jbase_agent adduser SOP

   b. jbase_agent passwd SOP N3wP9ssw0rd

4. Modify the system environment variables to add settings for the following entries:

   a. JBCRELEASEDIR

      i. Set to the location of the jBASE 5 release directory, i.e. "C:\jBASE5\5.2"

   b. JBCGLOBALDIR

      i. Set to the location of the jBASE 5 release directory, i.e. "C:\jBASE5\5.2"

   c. JEDIFILENAME_SYSTEM

      i. Set to the location of the jBASE SYSTEM file which will contain extended records for each of the user accounts, i.e. "D:\jBAccounts\SYSTEM"

   d. JEDIFILEPATH

      i. Set to the current location together with the location of the folder under which the jBASE accounts reside, i.e. ".;D:\jBAccounts"

5. For each account to be used, an extended SYSTEM file entry should be created. As there will possibly be a need to compile programs, extended "PATH", "INCLUDE" and "LIB" information should be provided, for example:

```
SOP

01 D
02 D:\jBAccounts\SOP
03
04
05
06
07
08
09
```

```
10
11
12
13
14
15
16
17
18
19
20 ESYSTEM_START
21
22 D:\jBAccounts\SOP
23 D:\jBAccounts\SOP\bin
24 D:\jBAccounts\SOP\lib
25
D:\jBAccounts\SOP\bin;D:\jBAccounts\mv.net\bin;D:\jBASE5\5.2\bin;C
:\WINDOWS\System32;C:\Program Files\Microsoft Visual Studio
8\VC\bin;C:\Program Files\Microsoft Visual Studio
8\Common7\IDE;C:\Program Files (x86)\Microsoft SDKs\Wi
ndows\v6.0A\bin;C:\Program Files (x86)\Microsoft Visual Studio
8\SDK\v2.0\Bin
26
D:\jBAccounts\SOP\lib;D:\jBAccounts\mv.net\lib;D:\jBASE5\5.2\lib
27 D:\jBAccounts\SOP;D:\jBAccounts\mv.net
28 D:\jBAccounts\SOP\MD
29
30
31
32
33
34
35
36 MVNET=D:\jBAccounts\mv.net]TERM=vt220]INCLUDE=C:\Program
Files\Microsoft Visual Studio 8\VC\include;C:\Program Files
(x86)\Microsoft Visual Studio 8\VC\include;D:\Program Files
(x86)\Microsoft Visual Studio\VC\include]LIB=D:\jBASE5\5.
2\lib;C:\Program Files\Microsoft Visual Studio
8\VC\LIB\amd64;C:\Program Files\Microsoft Visual Studio 8\VC\lib
37 ESYSTEM_END
------------------------ End Of Record ------------------------
```

6. A password should be set for each SYSTEM account using the jBASE "PASSWORD" command, i.e.:

```
jsh Administrator ~ -->PASSWORD
Enter account name to modify password : SOP
Enter old password for account SOP :
Enter new password :
Re-enter new password :
jsh Administrator ~ -->
```

   a. Note that the password should match the one set via jbase_agent

UNIX/Linux

If you use the generally accepted method of having a “jbase” or “jbaseadm” user to own and administer the jBASE installation, then this account can also be used to start “jbase_agent” and also be the location for the JAGENT_USER and SYSTEM files.

Otherwise, 'jbase_agent' can be started by any user with the appropriate environment variables 'JBCRELEASEDIR', 'JBCGLOBALDIR', ('LIBPATH, SHLIBPATH, LD_LIBRARY_PATH' as appropriate for the UNIX/Linux version) and 'PATH' set.

Assuming a “jbase” user:

1. Modify the user’s .profile to make sure that all of the required environment variables are set. “JEDIFILENAME_SYSTEM” should also be set to point to a local “SYSTEM” file rather than using the default jBASE SYSTEM file at “/opt/jbase5/5.2/src/SYSTEM”.

2. At the jshell, create a JAGENT_USER and SYSTEM files:

   a. `jsh jbase ~ -->CREATE-FILE JAGENT_USER 3 11`

   b. `jsh jbase ~ -->CREATE-FILE DICT SYSTEM 7`

3. Add a User ID and Password for each user account:

   a. `jsh jbase ~ -->jbase_agent adduser SOP`

   b. `jsh jbase ~ -->jbase_agent passwd SOP N3wP9ssw0rd`

4. For each account to be used, an extended SYSTEM file entry should be created, for example:

```
     sop

 001 D
 002 /home/sop
 003
 004
 005
 006
 007
 008
 009
 010
 011
 012
 013
 014
 015
 016
```

```
017
018
019
020 ESYSTEM_START
021
022 /home/sop
023 /home/sop/bin
024 /home/sop/lib
025 /home/sop/bin:/home/mvdotnet/bin
026 /home/sop/lib:/home/mvdotnet/lib
027 /home/sop:/home/mvdotnet
028 /home/sop/MD
029
030
031
032
033
034
035
036 CC=gcc
037 ESYSTEM_END
```

5. A password should be set for each SYSTEM account using the jBASE "PASSWORD" command, i.e.:

```
jsh jbase ~ -->PASSWORD
Enter account name to modify password : SOP
Enter old password for account SOP :
Enter new password :
Re-enter new password :
jsh jbase ~ -->
```

   a. Note that the password should match the one set via jbase_agent

6. Once all of this is in place, check that the settings are correct by issuing a "LOGTO", i.e.

   a. jsh jbase ~ -->LOGTO sop

7. After the "LOGTO", a "jdiag" can be run in the "logged to" account to check that the environment settings are correct.

8. Back in the "jbase" account, at the shell prompt, start the "jbase_agent" daemon:

```
Linux-/home/jbase: jbase_agent -A account
(4832|47753408801968) NOTICE starting up jAgent, Process Per
Connection mode, listening on port 20002, SocketAcceptor.h +63
```

9. With the default “jbase_agent” settings, logging messages will occur after the daemon is started, so you may need to adjust the logging level as required. A logging level of “0” will suppress logging, i.e.:

```
Linux-/home/jbase: jbase_agent -A account –L 0
```

10. It should be noted that, with a "jbase/jbaseadm" *NIX user starting the "jbase_agent" daemon, group read/write permissions are required on all of the accounts to be accessed in order for jRemote connections to succeed.

Client

1. Copy an existing jBASE Server Profile definition and paste it into the Data Manager tree. Change the “Connection type” to “jRemote” and set the port to the value used by jbase_agent, probably “20002”.

2. Copy the “jremote.dll” found at (for example) “.:\jBASE5\CurrentVersion\clients\jremote” to the following locations:

   a. “C:\Program Files (x86)\BlueFinity\mv.NET\Version4.0\bin” or “C:\Program Files\BlueFinity\mv.NET\Version4.0\bin”, depending on platform.

   b. C:\ProgramData\BlueFinity\mv.NET\Version4.0\bin\ClusterExe” or “C:\Documents and Settings\All Users\Application Data\BlueFinity\mv.NET\Version4.0\bin\ClusterExe”, depending on platform.

3. Try a “Test Connection” to see if your set-up works. You should see output such as the following:

```
Attempting to establish connection ...

Connection successfully established.
Connected as port number: 12
```

4. You should now be ready to use mv.NET to connect to jBASE via jRemote.

UniVerse

1. Using UniAdmin, create a new **Pick flavor** mv.NET account which represents a new folder at o/s level. Do not point it to any existing accounts.
2. Using UniAdmin, create a new **Pick flavor** SOP account which represents a new folder at o/s level. Do not point it to any existing accounts.
3. Create an Windows/UNIX user called mv.NET with admin rights (just to be sure). Amend the profile of this user to have the 'Local path' point to the mv.NET folder.
4. Create an Windows/UNIX user called SOP with admin rights (just to be sure). Amend the profile of this user to have the 'Local path' point to the SOP folder.
5. Use the UniAdmin\Network Services\UDTelnet Server option to create 2 new Telnet users called MV.NET and SOP. Set the Startup directory for each of these users to point to the relevant newly created folder.

UniData

1. Create 2 Windows/UNIX folders to house the MV.NET and SOP accounts. Create 2 Windows/UNIX users called MV.NET and SOP.
2. Use the UniAdmin utility to create 2 new accounts pointing to the 2 newly created folders.
3. Use the UniAdmin\Network Services\UDTelnet Server option to create 2 new Telnet users called MV.NET and SOP. Set the Startup directory for each of these users to point to the relevant newly created folder.

mvBASE (and other generic mv implementations)

Logged in as SYSPROG, create 2 accounts called MV.NET and SOP with a privilege level setting of SYS2 or higher.

The Server Console Window

After a server profile definition has been created, sufficient information to allow the Data Manager to establish a terminal emulation session to the server will have been entered. This chapter covers the Server Console window that can be invoked for a server profile and describes the various series of actions that must be performed using the session window to complete the installation and setup of Core Objects, the most important of which is the downloading of Core Objects' server-side components.

Functionality Overview

The Server Console window provides the following capabilities:

- Terminal emulation window
- Server component download
- Demo SOP account download
- Account enabling

The server component download and account enabling actions must be performed to complete the setup of Core Objects. The download of the demo SOP account is optional.

Opening a Server Console Window

Under each server profile is a node called 'Server Console Window'. Double-clicking this node will result in a window allowing the address and port of the telnet listener for that server to be entered. Upon clicking OK in this window, a Server Console window will be displayed. There is also a button within the server profile maintenance window called 'Open Server Console Window' – clicking this button will also result in a Server Console window being displayed (using the address and port entered within the server profile definition).

Terminal Emulation

The black area within the Server Console window represents a terminal emulation region. The only emulation mode supported by the Server Console is VT220.

Using the emulation window, you may logon to the server and perform any of the command level activities supported by the Multi-value platform.

Server Component Download

Before a client application can perform any form of communication with the Multi-value server, you must download the mv.NET server components onto your database system. This, basically, involves the download, compile and catalog of 80 or so DataBASIC routines onto the server, along with the creation of a handful of control files. To do this, you will need to follow the following series of steps (note, please ensure that you have prepared your database server as outlined in the previous chapter):

1. Using either the Server Console window or another terminal connection, create a new account on the system called 'MV.NET' or similar. This account will be referred to hereon as the 'mv.NET account'.
2. Within the Server Console window, logon to the mv.NET account and make sure that you are at the command prompt.
3. Select the console window's menu option: Action\Download Server Components. A extra region will be displayed at the foot of the window containing the following check boxes:

- Force use of $INCLUDEs – this indicates that within the source code downloaded to the server, a '$' symbol will be prefixed to any INCLUDE statements. This is only needed for a small number of legacy Multi-value platforms.
- Suppress use of named common – this prevents the mv.NET common area from being 'named'. You should only use this option after consulting BlueFinity.
- Use D3 flash compile option – this option is only available if your Multi-value database type is set to D3. If this option is ticked, the COMPILE command is used with the (o option.
- Use D3 core-locking compile option – this option is only available if your Multi-value database type is set to D3. If this option is ticked, the COMPILE command is used with the (κ option.

After ticking the required options, click the Start button to commence the download process. You will be asked to confirm the database type.

At the end of the download process Windows Notepad will be invoked to display a trace of the download action. This trace will have been saved to disk, so you may close this window.

Account Enabling

Once you have downloaded the server components into the MV.NET account, you need to 'enable' each of the data/application accounts that you wish to access via mv.NET. This 'enabling' action simply refers to the creation of a handful of file (or 'Q') pointers within the data account back to the mv.NET account, and the cataloging of the downloaded DataBASIC routines (if necessary).

Note, there is no need to enable the MV.NET account (i.e. the account into which the server-side routines were loaded).

To 'enable' an account, navigate your way to command level within the account and then select the Server Console's menu option: Enable Application Account. This option will ask you confirm the database type and that you are at command level within the account and will then perform a series of actions resulting in the execution of a program called 'MVNET.ENABLE' which checks to make sure that the account is ready for mv.NET usage. A message saying 'Account structure OK' should be displayed at the end of its execution.

SOP Demo Account Download

To run some of the sample applications provided with Core Objects, you will need to download a demo data account onto your Multi-value server. Before doing this, you will need to create a new (empty) account on your Multi-value server called 'SOP', or similar. This will be referred to hereon as the 'SOP account'.

Within the Server Console window, logon to the SOP account and make sure that you are at the command prompt. Select the console window's menu option: Action\Download Demo SOP Account. Click the Start button to begin the download.

At the end of the download, a data generator program will be invoked on the Multi-value server within the SOP account. You need to complete the on-screen prompts to generate data within the files that the download process will have created.

Do not forget to enable the SOP account after you have run the data generation (see previous section).

Defining Account Profiles

Each server profile holds the definition of the main aspects controlling how Core Objects clients locate and connect into a Multi-value database server. However, this definition is not complete, as it needs augmenting with the details of an account on the server so that an account specific connection can be established. The Account Profile is the entity which holds this extra information and there may be several account profile definitions within a single server profile - each representing a different account or application on the server.

An account profile provides the means to define 2 main areas of information. Firstly, account specific logon settings and secondly, account specific session pooling and housekeeping settings.

This chapter takes you through the creation of a new account profile and explains the different sets of definitions that comprise its content.

Accessing Account Profiles

Under each server profile node within the Data Manager's explorer tree is a node called 'Accounts'. Expanding this node lists the account profiles that have been created within the server profile. To modify an existing account profile, right-click the account profile name and select 'Edit Account Profile'.

To create a new account profile, right-click the 'Accounts' node and select the Add Account Profile option. This option will prompt you for a profile name. After entering a name (or after selecting to edit an existing account profile) the following window will be displayed allowing you to enter the details of your new profile:

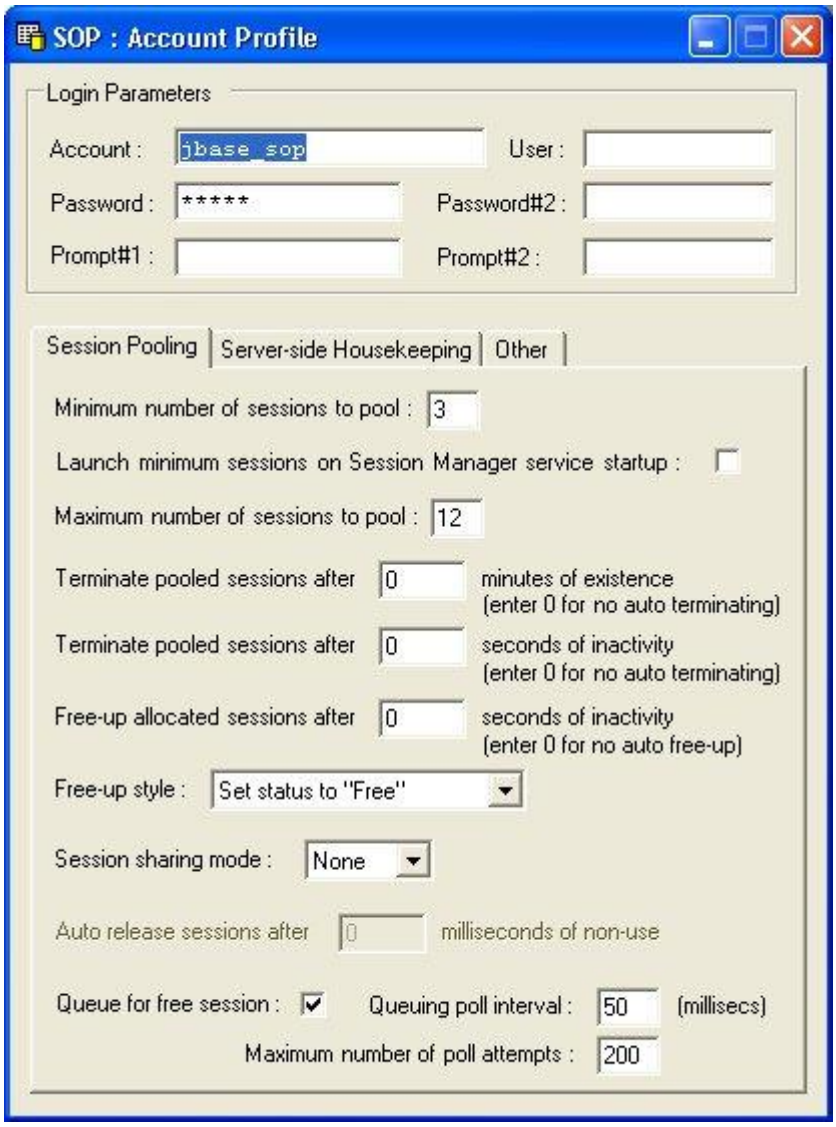

Diagram 5: The Account Profile Maintenance Window

Account Profiles Settings

The various fields on the account profile maintenance form are explained below:

Login Parameters: This group of input fields allows you to specify data that can be used to control Core Objects' login process to correctly connect to the required account. The values of these 6 fields can be inserted into connection negotiation send and wait for strings at run-time using special markers fields – see previous section Defining Connection Negotiations.

These fields, combined with the server profile definition, thus provide a complete definition to mv.NET of how to connect into a specific account on a specific server.

The table below lists how these 6 fields can be utilized for the variety of connection types that may be used within a server profile:

Connection Type	Description
IP	The 6 Login Parameters fields may be inserted into the send and wait for strings of a server profile's connection negotiation definition by the use of the special markers {account}, {user}, {password}, {password#2}, {prompt#1 and {prompt#2}.
UniObjects	(UniVerse and UniData only) The *Account* field should be set to the path of the directory holding the required account. The *User* field should be set to the Windows/Unix user that is to be used for the login process. *Password* should be set to the user's password.
QMClient	(QM only) The *Account* field should be set to the required QM account. The *User* field should be set to the Windows/Unix user that is to be used for the login process. *Password* should be set to the user's password.
SAC	(UniVision only) The Account field should be set to the name of the database account that is to be connected into.

Session Pooling: This group of input fields allows you to specify session pooling settings for this particular account. The paragraphs below describe the purpose of each of these fields.

Minimum number of sessions to pool: Allows you to indicate the minimum number of sessions that may be contained within the pool. The **Connect on SM startup** checkbox allows you to indicate whether this minimum number of sessions should be automatically established upon Session Manager service startup, or whether it should be reached through the natural increase in demand for connections. However, once this minimum number of sessions is reached, the Session Manager will not allow them to be released.

Maximum number of sessions to pool: Allows you to specify the maximum number of sessions that may be held open (ready for allocation) within the session pool at any one time. The minimum allowable value in this field is 1.

Terminate pooled sessions after n minutes of existence: Allows you to indicate the maximum life span of a session within the pool. When a session has been in existence for greater than the amount of time indicated by this field it will be gracefully terminated. A value of zero here indicates that a session's lifespan is unlimited (unless otherwise terminated)

Terminate pooled sessions after n seconds of inactivity: Allows you to indicate the maximum number of seconds that a pooled session may remain inactive within the session pool before it is automatically terminated. A value of zero here indicates that sessions are to be kept open indefinitely. A value of 1 indicates that as soon as a session becomes free it will be terminated - in effect preventing session pooling.

Free-up allocated sessions after n seconds of inactivity: Allows you to indicate the maximum number of seconds that a pooled session may remain in an 'Allocated' state with no activity. If a non-zero value is entered here, the status of an allocated session will be set to 'Free' if no activity occurs on the session for the specified number of seconds. This field should be used with care. It is primarily provided to allow the session manager to automatically recover from situations where an application has terminated in a non-graceful manner, i.e. without performing a session disconnection action (mvAccount.Logout). A value of zero here indicates that no automatic session freeing is to be performed.

Session sharing mode: This dropdown list allows you to specify how pooled sessions are to be shared across multiple client processes/threads. The following options are available:

None - No session sharing will be allowed. That is, once a client process has been allocated a session for this account from the connection pool, it will have exclusive access to that session until it explicitly releases it.

Single - All processes requesting access to a session that is connected to this account will be 'channeled' through a single session. This option should be used only in a development environment where it is likely that processes will crash or will be manually halted prematurely before releasing a session.

Multiple - All processes requesting access to a session that is connected to this account will be allocated a free session for the duration of a single server request (round-trip) only. After a response is received back from the server, the session will be automatically placed back into the session pool and made available for other clients. It is important to note here that the client

application will still 'think' that it has a permanent connection to the database - the session allocate/free activity happens automatically. Multiple session sharing allows a greater degree of connection sharing amongst clients but does place a slightly higher load on the Session Manager.

The Multiple session sharing option is typically used to allow multiple rich client applications to be multiplexed through a relatively small number of database connections. However, there are a number of restrictions in this setting:

1. O/S level pessimistic locking is not available. To work around this, mv.NET supports a concept of 'soft' locks, which essentially means that mv.NET operates the pessimistic item lock table for an application as opposed to the O/S. This allows pessimistic locking to be used in a situation where several discrete clients share the same physical database process (PIB). It should be noted that soft locking should only be used when mv.NET is the only method being used to lock data. It will not ensure lock integrity with other applications using traditional READU item locking. Soft locking is activated on the "Other" tab of the account profile definition (see below).
2. Index-based data access is not available. That is, the use of the mvFile.IndexSelect method is not allowed.

Auto release sessions: This field is only enabled when the session sharing mode is set to "Multiple" and allows you to specify (in milliseconds) the length of time that the automatically acquired database connection will be retained before being released. If you require the connection to be released as soon at the database round-trip is completed, this value should be set to zero. Using a non-zero value here allows you to reduce the session allocate/release burden on the Session Manager but at the cost of clients retaining connections for longer periods of time.

Queue for free session: This checkbox allows you to specify whether a process will wait for a session to become available if all current sessions are allocated and the maximum concurrent session limit of the session pool had been reached. If this option is not checked, an exception will be raised if the above set of circumstances arises. The **Queuing Poll Interval** field allows you to define how frequently a process will poll for a free session if it has been placed in a wait queue. A process will poll for a free session 50 times before an exception is raised.

Housekeeping Settings

The second tab on the account profile maintenance window allows you to define the server-side housekeeping activities that mv.NET can perform, as shown below:

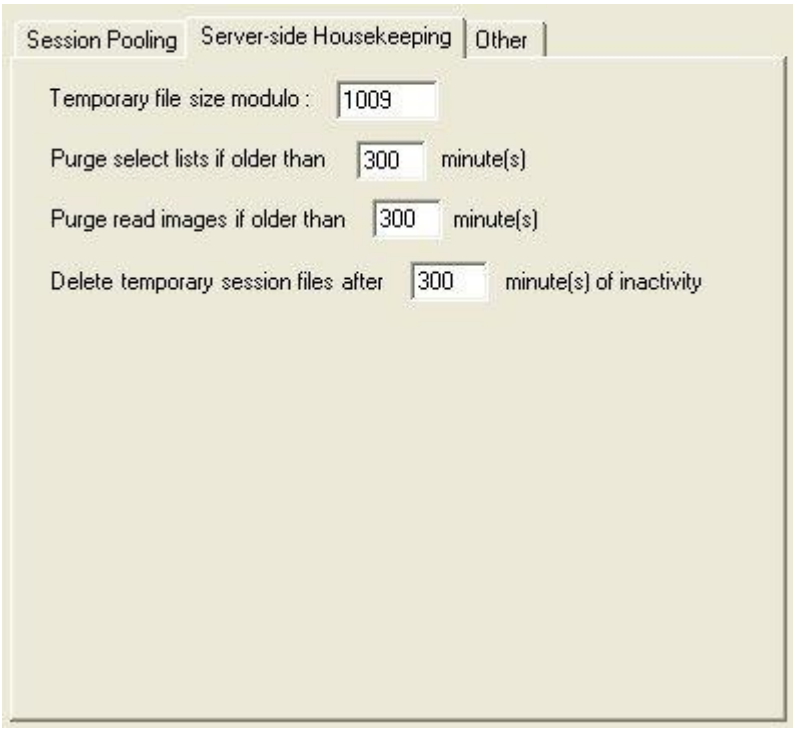

Diagram 6: Server-side Housekeeping Settings

All housekeeping activity is coordinated by the session manager service and thus the frequency of housekeeping is set at the session manager level. This can be done by double-clicking the Session Manager Settings node within the Data Manager treeview. At the foot of the resulting settings window, you are able to specify both the timing method for housekeeping (either to be run every x minutes or at a specific time each day) and the interval/time setting itself.

The account profile housekeeping settings allow you to control when temporary data on the server is to be deleted, as well as controlling the default size of temporary files created automatically by mv.NET.

The *Purge select lists* setting allows you to define when server-side select lists are to be deleted. When a select list is created by mv.NET as part of its internal activity (which will, of course, have been triggered by some application request) it is given a timestamp. When the housekeeping process executes, it examines the timestamps of all select lists created by mv.NET for this account and will delete any that are older than the age specified within the account profile.

The *Purge read images* setting allows you to define when optimistic lock read images taken on the server are to be deleted. When an item is locked in optimistic mode a copy (read image) of the current item is placed into the MVNET.READIMAGES file. Each read image is given a timestamp so that when the housekeeping process executes it examines the timestamps of all read images for this account and will delete any that are older than the age specified within the account profile.

The *Delete temporary session files* setting allows you to define when the temporary files for stateless connections are to be deleted. If an application passes an application GUID (session ID) into the connection request call, a dedicated temporary file will be created for that session ID so that information can be persisted for that session across invocation boundaries. mv.NET keeps track of when a session ID last contacted the server and if, when the housekeeping process executes, it finds that a session has not contacted the server for a period greater than that specified within the corresponding account profile the temporary session file will be deleted.

Other Account Profile Settings

The third tab on the account profile maintenance window allows you to define various miscellaneous account profile settings, as shown below:

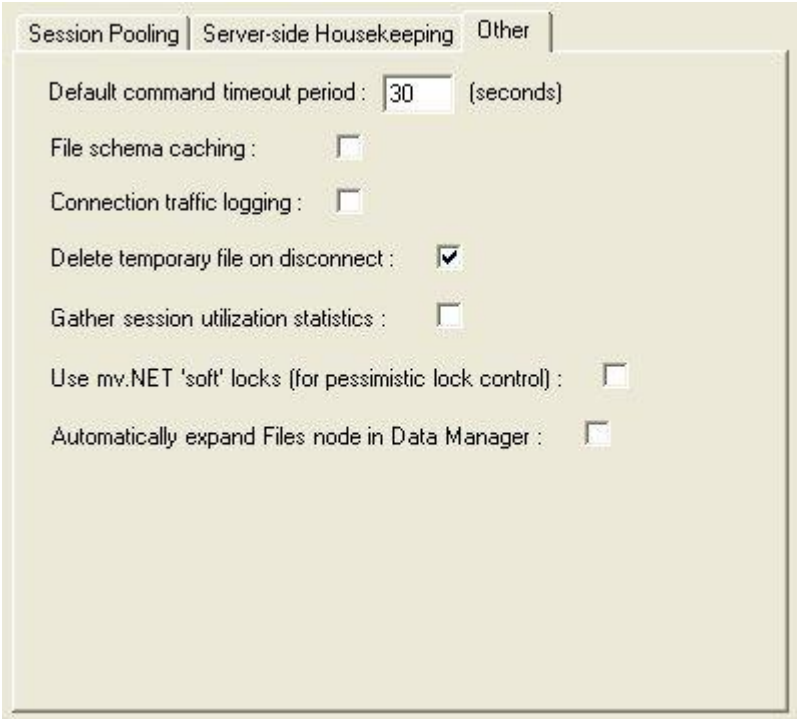

Diagram 7: Other Account Profile Settings

The *Default command timeout period* field allows you to specify the default length of time (in seconds) that an application will wait for a response from the server after issuing a request. This may be overridden programatically using the mvAccount.CommandTimeoutperiod property.

The *File schema caching* checkbox allows you to indicate whether you wish file schema data to be cached by the client to reduce round-trips to the server. This option should only be activated if the schema is not being actively developed/changed on a regular basis. Note, this option will only come into effect when new sessions are established. Also note that the Session Monitor window allows you to manually clear the schema cache without terminating a session.

The *Connection traffic logging* checkbox allows you to activate the logging of all messages passing between client and server for any sessions established using this account profile. Session logs are created in the following folder:

C:\Documents and Settings\All Users\Application Data\BlueFinity\mv.NET\Version4.0\Message Area\Live\Connections

Note, on Vista systems, **C:\Documents and Settings\All Users\Application Data** is known as **C:\ProgramData**

Within this folder will be one folder per session log. The folder name will reflect the session id, the profile names used and the date and time the log was produced. Within each session log folder will be multiple files holding a record of all message traffic.

The *Delete temporary file on disconnect* checkbox allows you to indicate whether the port-based temporary file automatically created by mv.NET at the start of a session (if it does not already exist) is to be deleted when a session terminates.

The *Gather session utilization statistics* checkbox allows you to indicate whether the Session Manager should record statistical information relating to the usage of sessions connected using this profile. This statistical information can then be viewed using the Statistical Analysis tool installed as part of the CIDSetup routine - a shortcut to which is created on mv.NET's Windows Start menu.

The *Use mv.NET 'soft' locks* checkbox allows you to indicate whether soft locking should be used for all clients connecting via mv.NET to this account. When soft locking is activated it results in mv.NET operating the pessimistic item lock table for an application as opposed to the O/S. It is typically used in situations where Multiple Session Sharing has been activated for an account profile which will, by definition, result in multiple clients potentially sharing the same physical database process (PIB); thus rendering traditional READU item locking as unusable. It should be noted that soft locking should only be used when mv.NET is the only method being used to lock data. It will not ensure lock integrity with other applications using traditional READU item locking.

Please refer to the "mv.NET Soft Locks" section within the Data Manager chapter of the Core Objects Developer Guide for more information on soft locks.

The *Automatically expand Files node in Data Manager* checkbox allows you to indicate that when you connect into an account within the mv.NET Data Manager utility, the list of available files is to be automatically assembled and displayed. You should only tick this box if you are sure that assembling the list of files will not be a time-consuming process.

Testing Connections

Once you have created server and account profiles and also downloaded the server components onto your Multi-value server, you are ready to test whether the Data Manager (and hence your own application) is able to successfully connect into the account.

Invoking a Connection Test

To run a connection test right-click an account profile node with the Data Manager explorer tree and select the Test Connection menu option. A Connection Test window will then be displayed along with a series of trace messages.

For IP-based connections, a successful connection will result in the message 'AppGUID=[...] Port=[...]' being displayed towards the foot of the test window. For non-IP-based connection, the word 'OK' or similar message will be displayed upon successful connection.

All unsuccessful connection attempts will result in an error dialog being displayed.

Use the Connection\Close menu option within the test connection window to close the window when you have finished.

If you experience difficulties getting the connection test to work, please contact your mv.NET support representative.

Defining Login Profiles

A login profile provides the means to assign a logical name to a server and account profile pairing. As well as aiding convenience of use, this has the benefit of preventing the names of server and account profiles being hard-coded into applications, which can result in problems when the time comes to deploy an application. For this reason alone, it is strongly recommended that, wherever possible, login profile names are used to identify the server and account profile to be used when establishing a database connection. In fact, in some areas of mv.NET (i.e. Binding objects and Adapter Objects) you may only define connection details in terms of login profile names.

Accessing Login Profiles

All the current login profiles are listed beneath the 'Logins' node of the Data Manager explorer tree. Right-clicking this node presents a context menu which allows new login profiles to be created. Right-clicking an existing login profile name allows an existing login profile to be amended.

Defining a Login Profile

In addition to a name, the definition of a login profile is simply a server profile name along with the name of an account profile within that server profile. The login profile maintenance window allows these 2 settings to be created and amended as necessary.

Product Licensing and Activation

Core Objects is licensed as part of the mv.NET suite as a whole. This chapter explains how mv.NET is licensed and how you may use the Data Manager application to enter licensing details so that product activation codes can be requested.

Summary of mv.NET Licensing

mv.NET is licensed in 2 parts. Firstly, if you have purchased developer licenses, each developer's workstation needs activating. Secondly, each server for which you have purchased runtime licenses will need the relevant number of database access licenses enabling.

Developer Product Licensing

To activate the full functionality of the Client Interface Developer (CID) product, you need to run the Data Manager and select the *License Control/Request CID License* top menu option. On selecting this option, the following screen will be displayed:

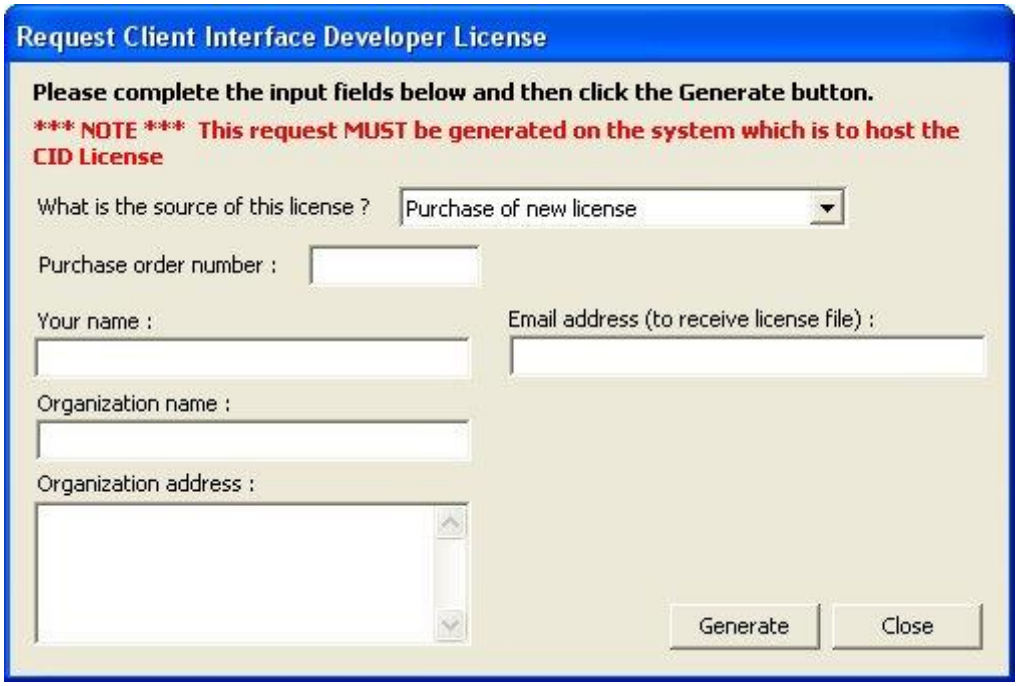

Diagram 8: Client Interface Developer License Entry

You need to enter the correct details into all of the input fields within this form and then click the Generate button. After doing so, at the foot of the window, a set of activation request details will be displayed. You need to copy and paste these details into an email which then needs to be sent to your mv.NET supplier.

Please refer to the section below Installing Licenses for instructions on what to do when you receive your license file.

CID Evaluation Licensing

On a system which has just had the CID product installed, the first usage of the Data Manager will automatically initiate a 30-days evaluation period

Please note, the Data Manager will display a 'Connected' message (with a green tick alongside it) as opposed to the contents of the account when you connect into an account using a system which does not have either an evaluation or permanent CID license.

Database Server Product Licensing

The basic concept of database access licensing within mv.NET is that an access license for a set number of concurrent connections to a specific database installation on a specific database server needs to be purchased and installed into a License Manager. For the mv.NET Session Manager to establish a connection (session) to a database, it must be able to contact a License Manager that has the appropriate access license installed. The License Manager checks the number of active sessions vs. the number of licensed sessions to decide whether the Session Manager is to be allowed to establish a new connection.

The Role of the License Manager

Each concurrent mv.NET session to a specific database instance consumes an mv.NET database access license (it also, by the way, consumes a raw database, D3, UniVerse, UniData, jBASE etc., license). You may purchase the required number of licenses for each of the database instances to which you require access via mv.NET. Each database access license is registered with a specific License Manager; a Session Manager must have access to a License Manager to successfully establish database connections. The License Manager ensures that (as a maximum) only the purchased number of concurrent sessions to a specific server is in effect at any one time.

The License Manager Service

The License Manager manifests itself as a Windows service. This service will be automatically installed as part of the CIDSetup or SRDKSetup installation routines.

Specifying the License Manager Address

The Session Manager needs to know where to find the License Manager. This can be done using the Session Manager Settings node within the Data Manager explorer treeview. This window allows both a primary and secondary system name to be supplied. If only one License Manager is in operation, only the primary address should be entered. If no names are entered, the Session Manager will assume that the License Manager can be found at localhost. The License Manager listens on the remoting port number used by the Session Manager minus one.

Applying for Database Access Licenses

To receive a database access license, you first need to request it. This can be done using either the Data Manager (mvNET.DataManager.exe) or the License Request utility (mvNET.LicenseRequest.exe). The License Request utility is provided as a standalone utility and is intended for use by system integrators as a way of pre-requesting database access licenses in advance of a pending mv.NET installation.

NOTE, when applying for database access licences, the Data Manager or License Request utility MUST be run on the system which is going to or which is currently running the License Manager service.

Using the Data Manager, you need to right-click the appropriate server profile and select the 'Request Database Access License' option. On doing this, the following window will be displayed:

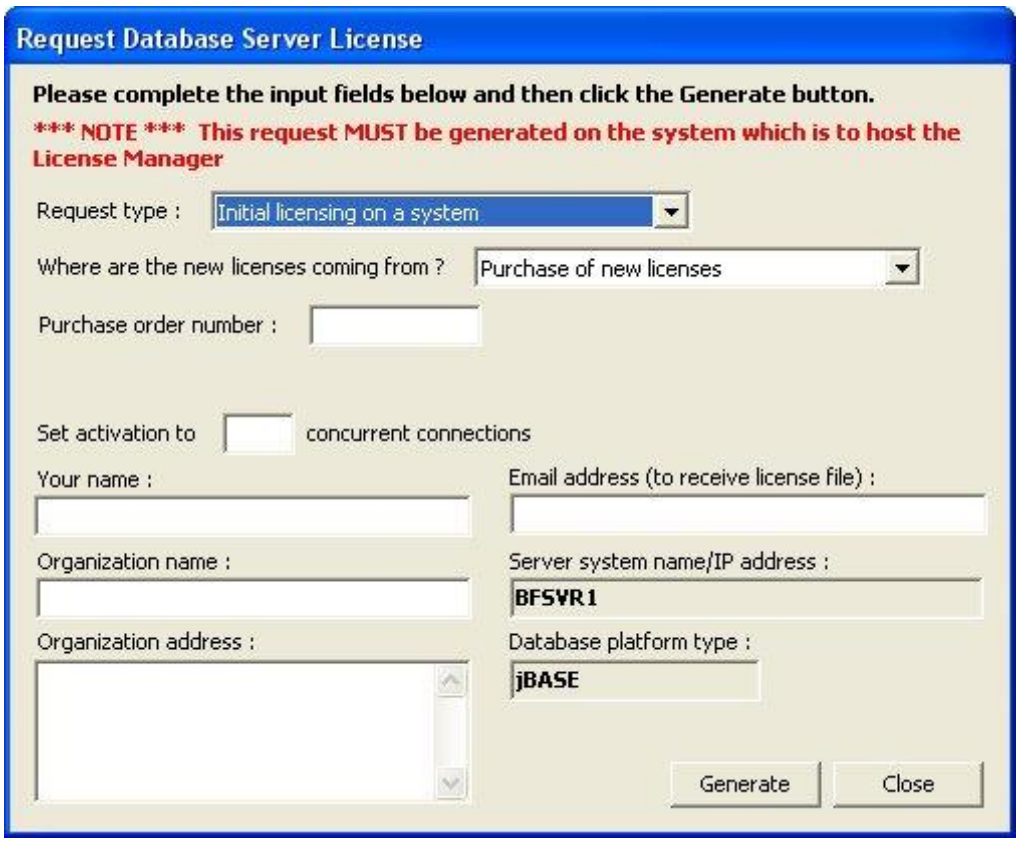

Diagram 9: The Database Access License Request Window

On entering the appropriate details into the above input fields, you should click the 'Generate' button to be shown the text which you should email to your mv.NET supplier in order to receive the relevant database access license.

The same window will be shown if you use the License Request utility, except that you will need to manually enter the database platform type.

Installing Licenses

Once a product license has been requested, you will receive an email containing a license file. On current server systems, this file should be saved into the following location:

C:\ProgramData\BlueFinity\Licences

On legacy systems, this file should be saved into the following location:

C:\Documents and Settings\All Users\Application Data\BlueFinity\Licences

Note, for Database Access Licenses, you need to perform this on the system which is hosting the License Manager service. For CID licenses, you need to do this on the system on which you have installed the CID product.

After saving the license file, please run the License Registration Utility – a shortcut to which will have been placed on your Start\BlueFinity\mv.NET menu by the mv.NET setup routine. This registration utility displays the following screen:

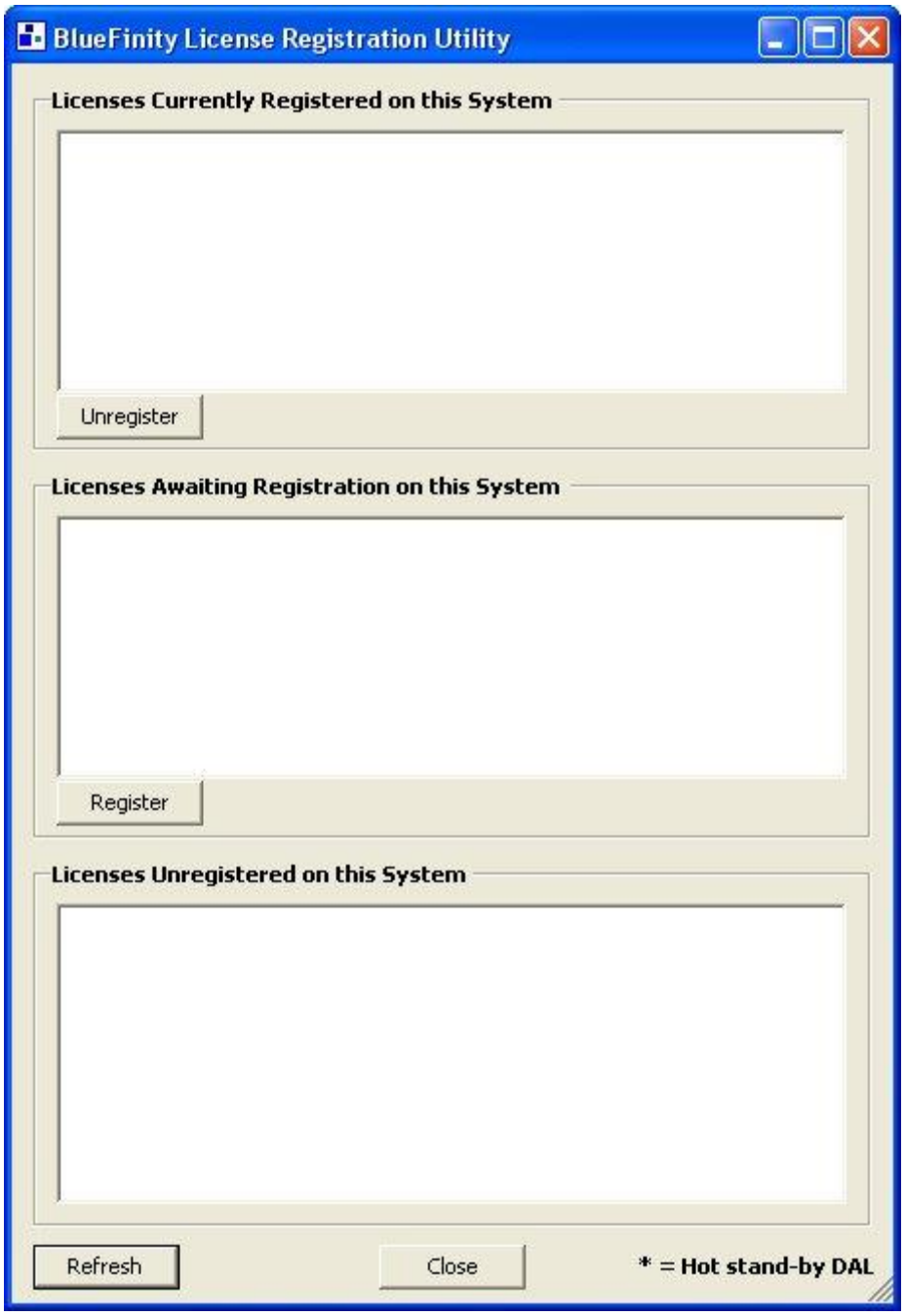

Diagram 10: The License Registration Utility Window

If you already have a previously registered license for the same database (for example a previously activated evaluation license), you will need to highlight it within the top listbox and click the Unregister button to remove it. Next, highlight the new license within the Licenses Awaiting Registration listbox and click the register button. At this point your new license will be active.

Note, the License Manager does not need to be restarted for a new DAL license to be detected.

NOTE, it is essential that you do not amend the contents of any license files.

Viewing Installed Database Access Licenses

The database access licenses which are currently installed within the License Manager may be viewed using the Session Monitor application. The 'Database Licensing' tab displays both the database access licenses installed within the License Manager and the licences currently in use.

License Manager Evaluation Mode

When the License Manager is first installed, it will, by default, allow up to 2 concurrent connections to any database installation for up to 30 days from the time of installation. Any database access licenses applied during this evaluation period will, naturally, override this behavior. After the 30 days evaluation period has expired, access to databases will only be allowed if the appropriate permanent database access license is installed.