

mv.NET Core Objects

Getting Started



A product from BlueFinity



Copyright Notices

Copyright BlueFinity International 2004 onwards

Document ref: mvNet_GS

Revision 2.0

All rights reserved BlueFinity International 2004

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
Telephone: +44 (0) 1442 450 435
Address: Hamilton House
111 Marlowes
Hemel Hempstead
Hertfordshire HP1 1BB
United Kingdom

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

mv.NET Core Objects Getting Started	1
Copyright Notices	2
Contacting Us	2
Trademark Acknowledgements	2
Welcome to mv.NET	1
The mv.NET Family of Products	1
Feature Overview	2
The mv.NET Suite.....	3
Getting Started Guide Contents	3
Installation Procedure Summary	4
Client-side Installation	6
The Client Interface Developer Setup Routine	6
Installed Components List.....	7
Defining Server Profiles	8
Running the Data Manager.....	8
The Server Profile Maintenance Window.....	10
Connection Specific Details	11
Defining Connection Negotiations.....	11
Communication Characteristics	13
The Server Console Window	15
Functionality Overview	16
Opening a Server Console Window	16
Terminal Emulation.....	16
Server Component Download	16
Account Enabling.....	17
SOP Demo Account Download	18

Defining Account Profiles	19
Accessing Account Profiles.....	19
Account Profiles Settings.....	20
Housekeeping Settings.....	23
Testing Connections	25
Invoking a Connection Test.....	25
Defining Login Profiles	26
Accessing Login Profiles.....	26
Defining a Login Profile.....	26
Product Licensing and Activation	27
Summary of mv.NET Licensing	27
Developer Product Licensing	27
Database Server Product Licensing	28
Evaluation Licensing	30

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 multivalued database developer wishing to create .NET based application interfaces to their current or new multivalued database file system.

Core Objects not only provides the underlying framework upon which all other mv.NET products are based – it also provides, in its own right, 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 multivalued developer to combine the power and flexibility of proven multivalued technology with the start-of-the-art, feature rich .NET environment. Its design also enables and encourages the developer to leverage, wherever possible, previously acquired multivalued skills.

BlueFinity's team of software engineers has huge knowledge and experience of using both multivalued 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 mv database platforms, allowing .NET developers to access all aspects of mv 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 mv developers to harness the full power of both their mv system and the .NET platform.

Core Objects also has strong integration with Microsoft's Visual Studio.NET product, allowing the mv 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, multivalued 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 multivalued 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 multivalued platforms.

- Support for stateless applications, e.g. Web Services, featuring optimistic locking, automated state retention/reconnection and database connection pooling.
- Sophisticated Visual Studio.NET add-in, featuring both systems management and developer-centric functionality.
- A fully compliant ADO.NET managed data provider implementation allowing mv 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 as a whole comprising of:

- **Core Objects** – object oriented native .NET access to mv databases.
- **Binding Objects** – high performance databinding technology that enables standard .NET controls to become fully mv-aware. Binding Objects links with Core Objects to provide its functionality.
- **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 mv database server(s).

[The Server Console Window](#)

This chapter explains the process of installing the Core Objects server-resident components on the mv 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 mv server(s).

[Testing Connections](#)

This chapter explains how to test the connection to your mv 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 a number of 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 mv 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 of 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 in a position 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 Core Objects 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.

Part-way through the install procedure, you will be asked whether you wish to install mv.NET's Visual Studio integration components. If you do not use Visual Studio on the system upon which you are running the installation, you should answer 'no' to this question.

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 support assemblies)
- 2 Windows services used to control session pooling and session sharing
- 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)

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 mv 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 an mv 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\mv.NET menu. If you installed mv.NET's VS.NET integration components, a version of the Data Manager which runs within the VS.NET IDE will also have been installed.

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

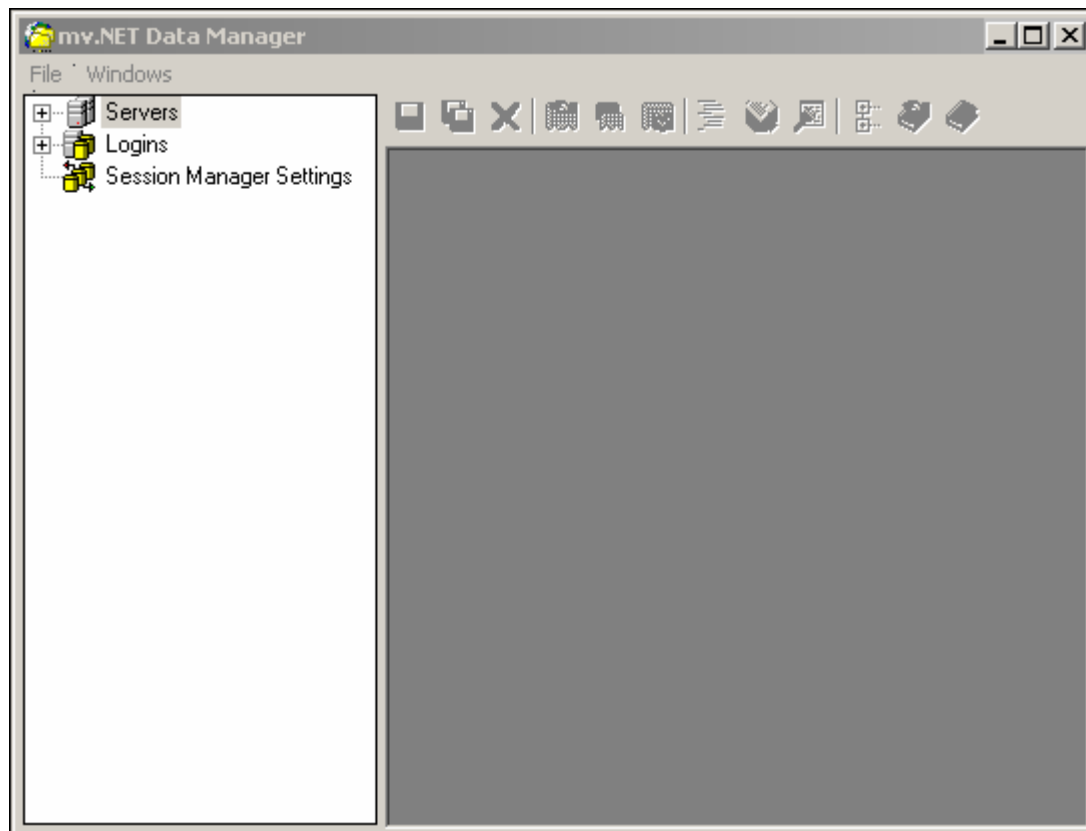


Diagram 1 : The Data Manager Explorer

The left-hand section of the Data Manager form contains a treeview comprising of several types of nodes. The node of interest to us at this stage is the Servers node. At the center 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.

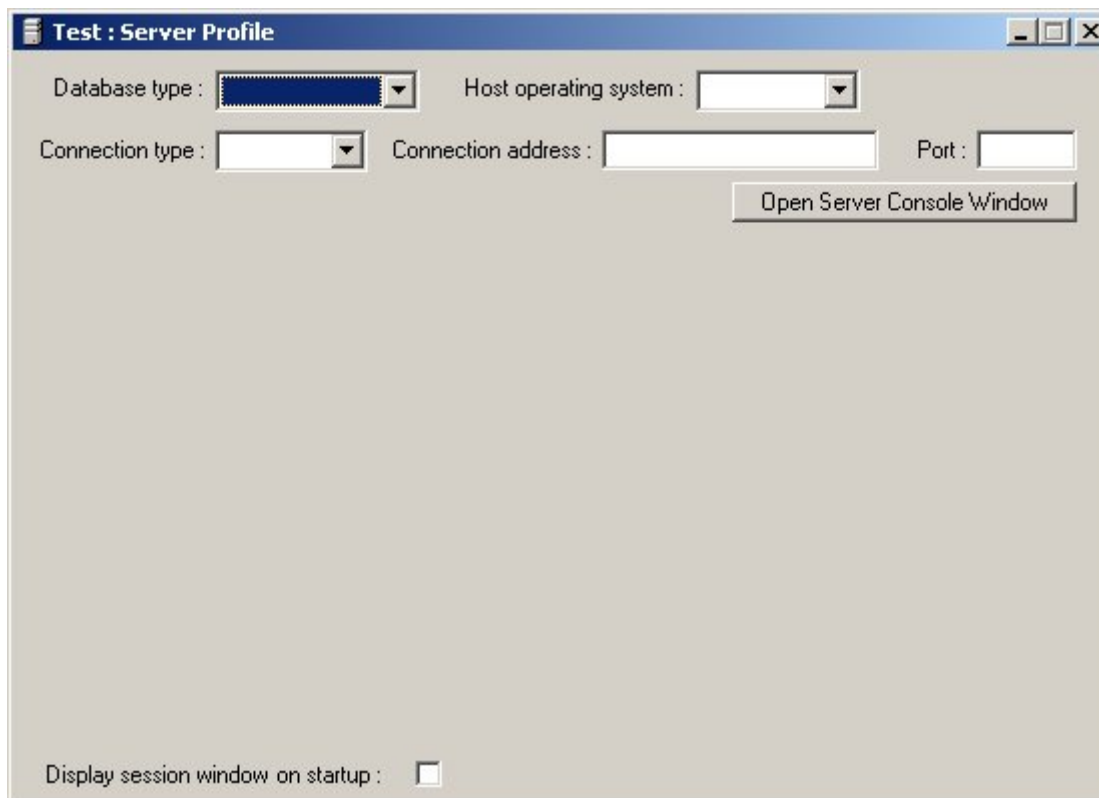
The image shows a screenshot of a software window titled "Test : Server Profile". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. The main area contains several configuration fields: "Database type" (a dropdown menu with a blue selection), "Host operating system" (a dropdown menu), "Connection type" (a dropdown menu), "Connection address" (a text input field), and "Port" (a text input field). Below these fields is a button labeled "Open Server Console Window". At the bottom left, there is a checkbox labeled "Display session window on startup" which is currently unchecked.

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 mv 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.

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 the Port field should be used depending on the selected connection type.

Connection Type	Description
IP	The port number of the Telnet listener. Defaults to 23 if left blank.
UniObjects	Not required
SAC	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 series of input fields will be presented in the lower half of the form:

Connection Negotiation		Communication Characteristics	
Send :	<input type="text"/>	Wait for :	<input type="text"/>
Send :	<input type="text"/>	Wait for :	<input type="text"/>
Send :	<input type="text"/>	Wait for :	<input type="text"/>
Send :	<input type="text"/>	Wait for :	<input type="text"/>
Send :	<input type="text"/>	Wait for :	<input type="text"/>
Send :	<input type="text"/>	Wait for :	<input type="text"/>
Send :	<input type="text"/>	Wait for :	<input type="text"/>
Send :	<input type="text"/>	Wait for :	<input type="text"/>
		Timeout period whilst waiting for Wait for string :	<input type="text" value="5"/> (seconds)

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 in order to gain access to the required account. After each send string you may specify a character sequence that must be received within the stream of characters coming back from the server before sending the next send string. In such a way you are able to 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 mv system displays during the connection process.

Communication Characteristics

In addition to specifying connection negotiation details, if a connection type is negotiated a second tab allowing 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.

Keep alive tick: This input field allows you to indicate that the communications link needs activity in order to keep it connected during periods of non-use. By specifying a non-zero number in this field, a small number of characters will be transmitted from client to host every so often in order to prevent the link from timing out.

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 in order 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 in order 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 mv platform.

Server Component Download

Before a client application can perform any form of communication with the mv server, you must download the mv.NET server components onto your database system. This, basically, involves the download, compile and catalog of 60 or so

DataBASIC routines onto the server, along with the creation of a handful of control files. In order to do this, you will need to follow the following series of steps:

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 only needed for a small number of legacy mv 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 extended delay after compile action – use this option if your mv server takes a prolonged period (> 2 seconds) to compile/catalog programs. If the download process freezes part way, it is likely that you will need to use this 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 with the data account back to the mv.NET account, and the cataloging of the downloaded DataBASIC routines (if necessary).

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. The 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 check 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

In order to run some of the sample applications provided with Core Objects, you will need to download a demo data account onto your mv server. Before doing this, you will need to create a new (empty) account on your mv 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 mv server within the SOP account. You need to complete the on-screen prompts in order to generate data within the files that the download process will have created.

Defining Account Profiles

Each server profile holds the definition of the main aspects controlling how Core Objects clients locate and connect into an mv database servers. However, this definition is not complete, as it needs augmenting with the details of a particular 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 a number of 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 6 input fields allows you to specify data that can be used to control Core Objects' login process in order 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 by the use of 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 <i>Account</i> field should be set to the path of the directory holding the required account. The <i>User</i> field should be set to the Windows/Unix user that is to be used for the login process. <i>Password</i> should be set to the user's password.
SAC	(UniVision only) The <i>Account</i> field should be set to the name of the database account that is to be connected into.

Session Pooling: This group of 8 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. A value of zero here will, effectively, prevent any session pooling from occurring; a value greater than zero indicates the maximum number of active sessions that are to be pooled. If this number of active sessions is reached, sessions will be closed upon release until the maximum pool size is attained.

Maximum number of concurrent sessions: Allows you control the absolute number of concurrent sessions that may be opened to the account at any one time. If this number of active is reached, the Session Manager will any further connection requests until existing connection become free. The purpose of this setting is to allow you set an upper bound on the number of database connection that can be consumed by an application.

Terminate pooled sessions after n minutes of delay: Allows you to indicate the maximum number of seconds that a pooled session may remain inactive within the session pool before it is automatically closed. A value of zero here indicates that sessions are to be kept open indefinitely.

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 process has been allocated a session that is connected to this account, it will have exclusive access to that session until it explicitly releases it.

Single Session – All processes requesting access to 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 the session.

Multiple Sessions – All processes requesting access to session that is connected to this account will be allocated a free session for the duration of a single server request only. After a response is received back from the server, the session will be automatically placed back into the session pool. 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.

The Multiple Sessions 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 environment:

1. Pessimistic locking is not available.
2. Index-based data access is not available. That is, the use of the `mvFile.IndexSelect` method is not allowed.

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:

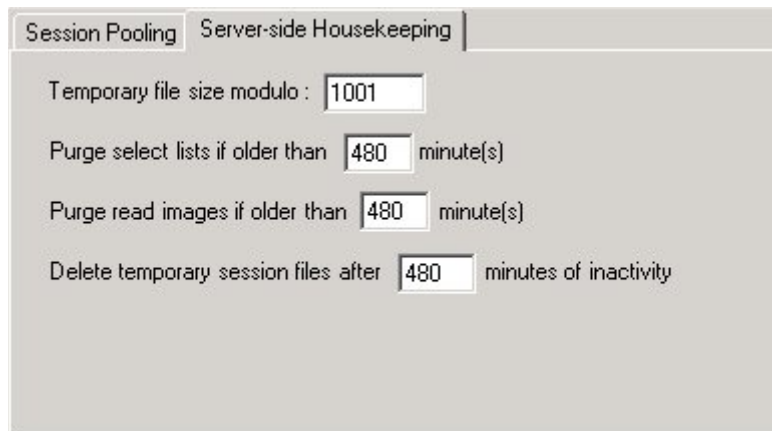


Diagram 6 : Server-side Housekeeping Settings

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.

Testing Connections

Once you have created server and account profiles and also downloaded the server components onto your mv 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 and a series of trace messages will be displayed.

A successful connection will result in the message 'AppGUID=[...] Port=[...]' being displayed at the foot of the test window.

Close the connection test 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 of 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. The last section of this chapter covers how to request evaluation licenses.

Summary of mv.NET Licensing

mv.NET as a whole 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 concurrent connections enabling.

Developer Product Licensing

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

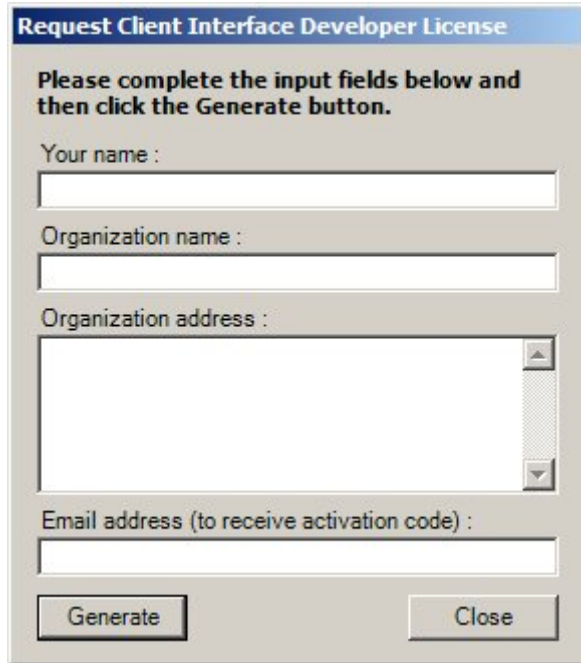


Diagram 7 : Client Interface Developer License Entry

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.

Upon receiving these details your supplier will verify them and then send back an activation code. On receiving this code you need to select the *License Control/Client Interface Developer/Enter Activation Code* top menu option. This will present a form where you may enter the supplied activation code.

Database Server Product Licensing

mv.NET supports the concept of concurrent database connection licensing. This is operated at a database installation level. Therefore, you need to activate the relevant number of concurrent connections on each of your database installations.

Before you can do this, you must use the Data Manager to create a server profile and at least one account profile for that server in order to allow the Data Manager to connect into it and transmit licensing details.

So, after creating a server and account profile for the server (please refer to the first 5 chapters of this guide for instructions on how to do this), you need to

connect into the account within the Data Manager. Once you have done this, select the *License Control/Database Server/Request Activation Code* top menu option. On selecting this option, the following screen will be displayed:

The screenshot shows a dialog box titled "Request Database Server License". The dialog contains the following elements:

- Title Bar:** Request Database Server License
- Instruction:** Please complete the input fields below and then click the Generate button.
- Server:** A dropdown menu.
- License Request Type:** Radio buttons for "Increase" (selected) and "Decrease".
- Organization name:** A text input field.
- Current license count:** A text input field.
- Increase license count to:** A text input field.
- Your name:** A text input field.
- Email address (to receive activation code):** A text input field.
- Organization address:** A large text area.
- Buttons:** "Generate" and "Close".

Diagram 8 : Server License Entry

Firstly, you need to select the correct entry from the **Server** dropdown list. After doing this, the number of concurrent connections currently licensed for the server will be displayed in the box below.

mv.NET supports the concept of transferring concurrent connection licenses from one server to another, therefore, you need to choose the relevant option from the **License Request Type** frame. If you are transferring licenses, you first need to request a license count decrease on the server which is relinquishing licenses and then, as a completely separate action, request a license count increase on the other server.

The input field beneath the license request type frame allows you to enter the number of licenses to add/remove.

You also need to enter the correct details into all of the other input fields on the form. After doing so, click the Generate button and 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.

Upon receiving these details your supplier will verify the details and (for license increase requests) will send back an activation code. On receiving this code you need to connect into the relevant server within the Data manager and then select

the *License Control/Database Server/Enter Activation Code* top menu option. This will present a form where you may select the name of the relevant server and enter the supplied activation code.

Evaluation Licensing

If you wish to evaluate mv.NET, you may activate a temporary evaluation license in order to perform your evaluation activity.

To enter the evaluation code for the Client Interface Developer (which you will need to do on each developer's workstation that has had the CIDSetup.exe installed), select the *License Control/Client Interface Developer/Enter Activation Code* top menu option within the Data Manager application. In the subsequent form enter 'eval' as the activation code. This will activate a 30-day evaluation period.

By default, mv.NET's server components will allow 2 concurrent connections, which may be activated up to 50 times. Therefore, you do not need to perform a licensing task in order to activate your server connections during your evaluation period.