

mv.NET

Version 3.5 Heads-up



A product from BlueFinity



The Main New Features in Version 3.5

Introduction

The main thrust of the new version 3.5 release of mv.NET is an extensive re-work of the core session management components in order to build upon the already sound basis of version 3.2 in the areas of system reliability, performance, resource utilization, scalability and overall robustness.

It also includes extended support for Ajax within the Web Binding Objects suite.

This document highlights the major new features of version 3.5 and also identifies the key areas where developers should focus their attention in order to adopt version 3.5 with the minimum of effort.

Session Management New Features

The session management components have been significantly enhanced to provide the following new features:

- Clustered session processes to provide improved scalability and greater system robustness
- Enhanced session monitoring to reduce performance overhead and increase the granularity and detail of monitoring information
- Session utilization statistics gathering and analysis

- Replacement of third party COM components with managed code equivalents
- Client identification of allocated sessions
- Remote Session and Connection monitoring

Session Management Internals

Developers should look out for the new internal structure of the session management components. Instead of the old structure of a Session Manager and Database Session service, there is now a Session Manager and a License Manager service.

The Session Manager service launches a number of other background processes which, together, form the session management part of mv.NET. Here is a summary of the session management executables:

mvNET.SessionManager.Service.exe – the Session Manager service. This performs a number of tasks, including the monitoring of all session management executables, the detection of hung and inactive sessions and the execution of automated database housekeeping.

mvNET.SessionManager.exe – the .NET remoting server which manages the main session pool(s). This is the process which mv.NET clients (via Core Objects) contact in order to initially acquire a database session.

mvNET.SessionCluster#n.exe – the session cluster executable. In version 3.5, the session cluster takes over from the version 3 Database Session service. There may be one or more session clusters in existence – the mvNET.SessionManager.exe process is responsible for launching session clusters processes based on demand from client applications. Each cluster can host a (configurable) number of database connections. Session clusters provide a more performant, resilient and scalable hosting container for database connections. Before launching a new cluster, the Session Manager copies the base executable to a new name and location so that the process name (as shown in the task manager) reflects the actual cluster number.

mvNET.CoreMonitor.exe – provides various monitoring services amongst the various session management components.

mvNET.LicenseManager.Service.exe – the License Manager service. This performs license management across all session management servers at a site. See section below for more details on this topic.

Session Monitoring

The Session Monitor application has been completely re-written. It can now stay active within the system tray and can also monitor session management services hosted on a remote system. The Connection Monitor has also been re-written to provide enhanced capabilities and can also monitor sessions being hosted by a session cluster on a remote system.

Configuration Database Access

In version 3 and before, the only way to access the configuration database (the place where all server, account and login profiles are stored) was to either have the configuration database sited locally or accessed through a drive mapping or file share via the ConfigurationPath file.

In version 3.5, the configuration database can now be accessed via the Session Manager. The address of the session manager can be supplied either via the ConfigurationPath file, the mvEnvironment.Login method or the mvAccount constructor. The format of the address is address:port

mvAccount Construction Streamlining

In version 3.5, the way in which mvAccount objects can be instantiated has been adjusted to make the various options more intuitive and efficient. There are now 3 different ways in which an mvAccount instance can be obtained:

1. Using the mvEnvironment.Login method. This allows a login profile name to be supplied to indicate which server/account is required.
2. Using the new mvEnvironment.Connect method. This allows a server and account profile name to be supplied.

3. Using the mvAccount constructor. This allows either a login profile name or server/account profile name to be supplied.

Much of the overhead involved in instantiating an mvEnvironment object has been removed in version 3.5. Therefore, using the mvAccount constructor technique should no longer be viewed as an inefficient way of obtaining mvAccount instances when compared to using the mvEnvironment.Login (or .Connect) methods.

Server Profile Change

The 'Non-concurrent logins' option within the server profile definition has been removed. This has been replaced by the 'Maximum concurrent session launches' field within the Data Manager's Session Manager Settings window.

Account Profile Change

The 'Maximum number of concurrent sessions' field within the account profile definition has been removed. Based on field observation, the net effect of this setting being different to the maximum pool size was to only make a high-load situation even more loaded with the overhead of constantly establishing new sessions.

Session Manager Settings Changes

The Session Manager Settings window within the Data Manager has been changed extensively.

The 'Use Session Management' option has been removed. All sessions are now controlled by the Session Manager. If no pooling of sessions is required, the minimum and maximum pool sizes should be set to 0; this will result in sessions being shutdown immediately after they have been released by clients.

The message logging setting has been streamlined to just 3 different settings: None, Normal or Verbose. For hardened production installations, message logging is not necessary, but the overhead of message logging has been greatly reduced in version 3.5 and so a setting of Normal in production environments is acceptable.

The rest of the settings in this window are documented in the Core Objects Developer Guide.

New Core Objects Class Members

`mvAccount.DebugInfo` – this allows the data flow associated with the last couple of interactions with the database server to be retrieved programmatically. This is useful for debugging or problem reporting within an application.

`mvAccount.ItemExists` – this allows the determination of whether an item exists or not without having to instantiate an `mvFile` object.

New Session Utilization Statistics

In order to analyze how efficiently sessions are being utilized, version 3.5 has the ability to gather session utilization statistics on an account by account basis and also provides a charting utility to provide an easy to understand display of the statistical data.

New Database License Management

In order to provide tighter and more resilient control of license usage, version 3.5 introduces a new method of installing and monitoring database access licensing. A new service (the License Manager Service) can be installed on either the same server or on any system reachable by the Session Manager system. Database access licences can be registered with a License Manager which will then work with one or more Session Managers to ensure that only the purchased number of concurrent sessions to a specific database installation are in existence at any one time.