



Technical Overview

This overview provides a technical backdrop to the mv.NET product. It discusses the main architectural aspects of the product and examines the main components that implement this architecture.

Introduction

The mv.NET product provides a 100% native .NET interface to all major MultiValue database platforms, allowing .NET developers to access all aspects of MultiValue systems – both data and program code – from within their .NET application. A full list of the currently supported MultiValue database platforms can be found at www.BlueFinity.com.

The mv.NET 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 MultiValue developers to harness the full power of both their MultiValue system and the .NET platform.

The mv.NET Suite

The mv.NET product suite as a whole is comprised of the following packages:

- **Core Objects** – object oriented native .NET access to MultiValue databases.
- **Binding Objects** – high performance databinding technology enabling standard .NET controls to become fully MultiValue-aware. Binding Objects links closely with Core Objects to provide its functionality.

- **Adapter Objects** – Comprehensive ADO.NET managed data provider implementation.

Core Objects

The architecture of Core Objects comprises three separate tiers:

- Client Interface Tier
- Session Pooling Tier
- MultiValue Server Tier

Each of these tiers is designed to be capable of residing on physically distinct systems separated by remote connections, but it may be that some or all tiers run on the same system – it all depends on the particular requirements of a specific installation.

The **Client Interface Tier** contains all of the .NET programmer visible objects, i.e. the programming interface components that the developer works with in order to produce an application. It is very likely that most of your development effort will go into creating the contents of the client interface tier. The Client Interface Developer module of Core Objects contains many features to assist you in this task – see next section.

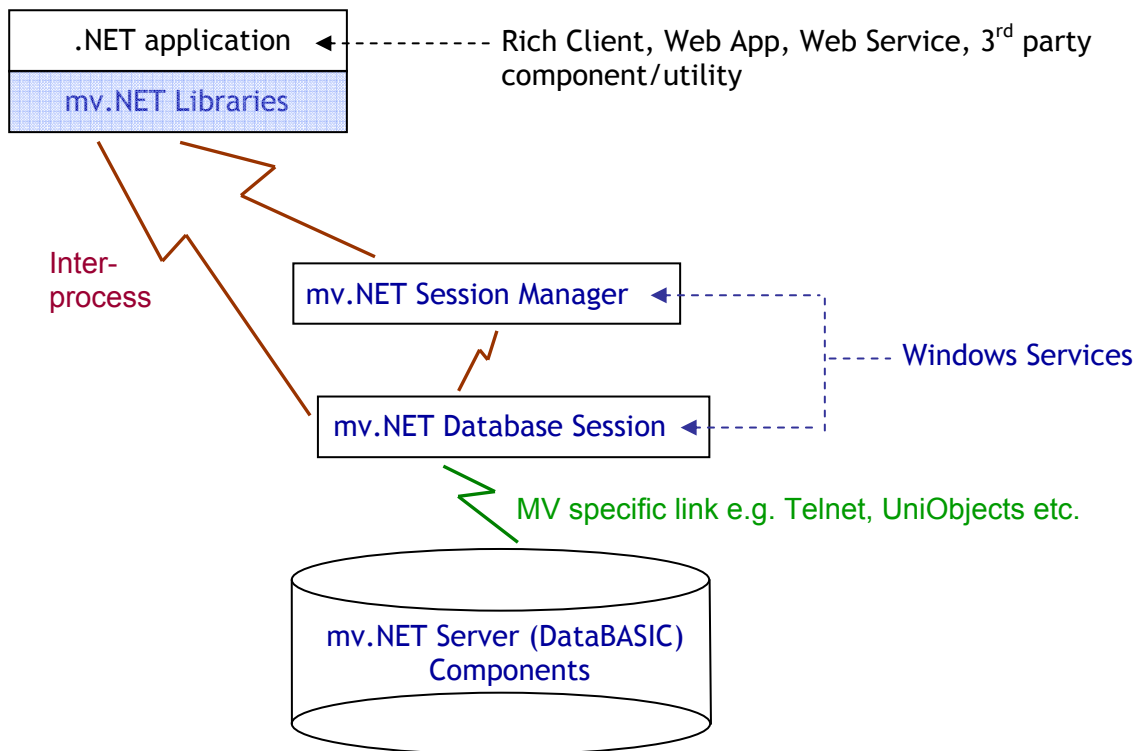
Typically, the Client Interface Tier will consist of the code that you write (using, for example, VS.NET), the Core Objects dll files that implement the Core Objects functionality plus any additional support files/assemblies that your application utilizes.

The **Session Pooling Tier** acts as the point of contact for Client Interface Tiers wishing to access one or more remote MultiValue systems. The Client Interface Tier to Session Pool Tier communications link utilizes .NET Remoting, which provides high a performance, loosely coupled connection to allow flexibility in the siting of these components.

The **MultiValue Server Tier** resides within the MultiValue database system. It is the point of contact for one or more Client Interface/Pooled sessions. The link to the MultiValue Server is capable of utilizing a variety of technologies, depending on what flavor of MultiValue system is being used.

The MultiValue Server tier is written in MultiValue DataBASIC and, therefore, a version of this tier (specifically coded and tuned) for each flavor of MultiValue database platform is provided.

The following diagram summarizes the interrelationships between these three tiers.



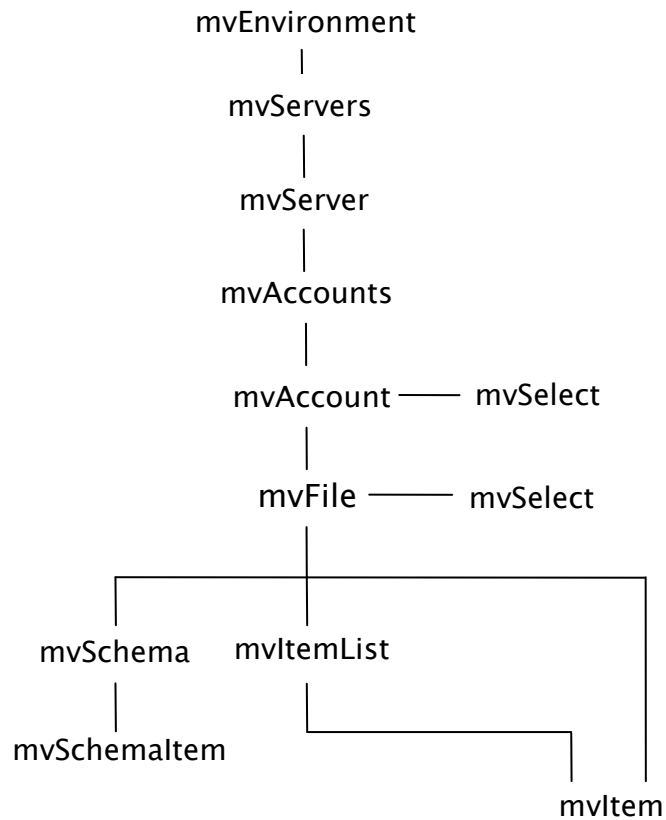
Having described the architecture the Core Objects product then consists of three separately installable packages:

- Client Interface Developer, which includes the Data Manager, Configuration Database and Session Pooling.
- Runtime Deployment Kits
- MultiValue Server Module

Client Interface Developer

The Client Interface Developer module should be installed on any workstation wishing to utilize Core Objects in conjunction with a development environment (such as Visual Studio.NET). It provides the programming interface into the Core Objects technology, along with plug-in extensions to Visual Studio.NET to allow management of many aspects of a MultiValue system from within the VS.NET native IDE.

The primary feature of the Client Interface Developer is a class library holding the series of classes that allow a developer to connect to and work with a MultiValue database. The diagram below shows the basic hierarchy of the Core Objects classes:



As part of the Client Interface Developer module, an application (written using VB.NET and utilizing many aspects of Core Objects) is provided to allow the maintenance of MultiValue systems – such as file and dictionary creation. This application is known as the Data Manager. The Data Manager provides the following capabilities:

- Maintenance of the Configuration Database (see below)
- Account maintenance
- File maintenance
- Dictionary maintenance
- Item data maintenance
- Index maintenance
- Terminal emulation
- Intra and inter-system data transfer

The Data Manager is provided in the form of both a VS.NET Addin and also as a standalone application.

In order to connect into a MultiValue system Core Objects needs to know a number of things; namely:

- what MultiValue systems are available for connection
- the address of a particular system
- what communications transport medium to use when talking to a system
- how to login to a system

To that end, a Configuration Database is used to hold all of the following configuration information:

- The list (and details) of all known local MultiValue servers (Server Profiles)
- The list (and details) of all known accounts on all known local servers (Account Profiles)
- Session pooling configuration settings

The Configuration Database's contents are maintained using the Data Manager and setting up the initial contents of this database is one of the first tasks that you will need to do when starting to use Core Objects.

Each client application needs access to a Configuration Database (either sited locally or shared centrally over the LAN) in order to either connect into a server.

As well as being able to maintain the contents of the Configuration Database using the Data Manager application, the Core Objects class library contains classes to allow the Configuration Database to be maintained programmatically by your own application code (this is, in fact, exactly what the Data Manager does).

The mv.NET Session Manager provides two key capabilities:

- To provide connection pooling across one or more individual processes or workstations. Note, the terms *connection pooling* and *session pooling* are used interchangeably within this guide.
- To monitor the presence and activity of an active database session

Connection Pooling is an essential feature for stateless applications, e.g. Web applications. It allows a database connection to be held open after an application has finished using it and then subsequently reallocated to another or same application instance, thus saving on the time and resources consumed in the initial creation of a database connection.

You should, therefore, utilize the Session Manager whenever there is a need to hold open database connections in order to avoid repeatedly incurring connection creation overheads.

The Session Manager can display a connection window for any active session, allowing the traffic flowing through that session to be monitored. For Telnet connections, a terminal emulator window is provided to allow keyboard generated input from the client to be entered.

As well as, obviously, being essential for Web-based applications, session pooling is also very useful in a software development environment. In such an environment it is very typical for a programmer to continually start and stop their application as part of the debugging/testing cycle. On MultiValue systems where login can take a number of seconds to negotiate, the ability for the Session Manager to hold a disconnected session open, ready for re-use, is a valuable time-saving feature for the developer.

The Session Manager communicates with both application and session(s) via efficient inter-thread mechanisms.

Runtime Deployment Kits

Runtime deployment kit (RDK) modules are required by an organization to enable deployment of the relevant mv.NET components necessary to run an application developed using Client Interface Developer licenses. The RDK's are supplied as part of Developer Licenses in two versions; one for client-systems where no local session pooling is required and one for server systems where session pooling is required. The relevant RDK will typically be incorporated within the developer's own product installation script and will install all of the components needed to support the Core Objects class library at runtime.

A limited version of the Data Manager utility is also supplied as part of the runtime license.

MultiValue Server Module

An essential part of Core Objects is the set of DataBASIC programs that reside on the MultiValue system. These programs are responsible for servicing all client generated requests and reside in an account typically named 'MV.NET' on the MultiValue database system.

These components are written in MultiValue DataBASIC and, therefore, a version for each flavor of MultiValue database is provided. Each of these versions has been specifically tuned for optimum performance on the target platform. The server components are downloaded onto the MultiValue system via the Data Manager application.

In order to access the data in an application account, the account has to be mv.NET enabled. This 'enabling' process creates a number of file pointers into the MV.NET account and also catalogs all of the server programs (if necessary).

The Data Manager application can be used to enable an application account, or this can be done at account command level using a command provided by Core Objects

Binding Objects

Complementing the Core Objects components, mv.NET's Binding Objects package provides state-of-the-art data binding technology, allowing rapid application development from within the .NET environment.

All aspects of the Binding Objects package are fully MultiValue data structure aware, right down to subvalue level. From the simplest file maintenance form to the most complex multi-database, multi-file transaction screen, Binding Objects provides the easy-of-use sophistication capable of meeting the most demanding needs of today's business application developer.

The Binding Objects components are built from the ground upwards to leverage the maximum application development potential from the MultiValue data model. The following features provide a sample of this MultiValue-awareness:

- Databinding right down to individual subvalue level
- Multi-item, multivalued and subvalue grid support
- Associated multivalued attribute support
- Multivalued 'foreign' item ID attribute support for linking to related data files

Binding Objects provides the developer with the choice of using either native .NET databinding, or mv.NET's extension of this; the latter providing a richer, more intuitive databinding data model, allowing more sophisticated use of the MultiValue data model and a greater range of business application development features.

The Binding Objects components utilize the native database and mv.NET's 'extended' dictionary definition data to the fullest extent possible, providing intelligent collection, formatting, validation and refreshing of both input and display-only data.

The Bindings Objects components can be used outside of the Visual Studio environment, but really come into their own when used in conjunction with mv.NET's

extensive VS.NET addin technology. Dockable toolbars providing databinding summary information along with sophisticated auto positioning of automatically created bound controls act to save the developer valuable time in their task of creating state-of-the-art applications.

Adapter Objects

mv.NET's Adapter Objects product provides the developer with a range of components designed to allow efficient ADO.NET based access to multivalued databases.

In order to provide a comprehensive ADO.NET solution, Adapter Objects provides the following 2 groups of components:

- Multivalued database specific Implementations of the ADO.NET classes/interfaces
- Visual Studio.NET addin components to aid developer productivity in the use of Adapter Objects

In order to present a multivalued oriented ADO.NET managed data provider, Adapter Objects provides the developer with the following mv.NET specific classes, most of which inherit from the corresponding .NET framework IDbxxx interface:

mvCommand
mvConnection
mvConnectionString
mvDataAdapter
mvDataReader
mvParameter
mvParameterCollection
mvTransaction

In order to ease the use of Adapter Objects within the Visual Studio IDE, Adapter Objects provides a range of VS.NET extensions which are used in various places within the IDE:

- mvDataAdapter creation wizard – invoked when an mvDataAdapter is dropped onto a form or when an existing mvDataAdapter is reconfigured.
- DataSet schema generation – invoked when the Generate Typed DataSet option within the Properties window is clicked.
- Customer property designers for a range of class properties.

About BlueFinity

BlueFinity (www.bluefinity.com) supplies leading-edge software development tools and consultancy services to the MultiValue database and Microsoft developer communities. Founded in 2002 by a group of highly experienced software developers with expertise in all flavours of MultiValue databases, BlueFinity has created a series of products that offer superb programmer productivity along with components encompassing feature rich, business application centric functionality. Its flagship product – *mv.NET* – is a comprehensive solution for developers wishing to access MultiValue databases from within Microsoft's .NET environment.

The mv.NET product and logo are Copyright © 2005 BlueFinity International Limited.