



Spectrum^{NG} – Architecture White Paper

<u>INTRODUCTION</u>	2
<u>TRUE MULTI-TIER ARCHITECTURE</u>	2
<u>BASIC ARCHITECTURE</u>	2
<u>DATABASE SERVER</u>	2
<u>APPLICATION HOST</u>	2
<u>REMOTING SERVER</u>	3
<u>BATCH QUEUE SERVER</u>	4
<u>CLIENT APPLICATION</u>	4
<u>SPECTRUM^{NG} ADVANTAGES</u>	5

Introduction

Spectrum^{NG} has many unique and innovative ideas behind its conception and implementation. It combines decades of experience in the industry with the latest technology to produce a software suite that is paralleled by none.

True Multi-Tier Architecture

Spectrum^{NG} can run as easily on one desktop machine as it can run in an enterprise-wide configuration. It employs a flexible configuration paradigm and state of the art remoting technology that allows it to stretch as little or as far as necessary.

In a multi-user environment, each of these components can be placed on a single machine, or broken up to different servers in order to provide load balancing.

Basic Architecture

Spectrum^{NG} requires five basic components:

- Database Server
- Application Host
- Remoting Server
- Batch Queue Server
- Client Application

Each component can be run on a single machine, or spread out to any number of configurations to take advantage of separate processes and memory requirements.

Database Server

SQL Server is the basis for data storage. Currently SQL Server 2000 and SQL Server 2005 are supported, in addition to the desktop versions of both server products. SQL Server provides a good balance between cost and scalability, with an optimum level of quality.

Customers who choose to allow CSI Software to host their application servers will share database services with other customers under normal circumstances. However, the configurability of the system allows customers to actually use separate databases while still accessing the same version of the software. This allows CSI to adjust the load to specific databases so that customers will not get bogged down when there are too many customers accessing a specific server at the same time. This may also be necessary if a customer has a security requirement to isolate its data.

Customers who wish to run the application in a local environment may use the desktop version of SQL Server.

Customers who wish to provide their own hosting environment in an enterprise- or LAN-wide architecture may do so. This will require the Customer to provide SQL Server and all of the architecture components mentioned in this section.

Application Host

The application host is used primarily for customers who will remote their data. This applies to customers that host their applications through CSI Software and customers who decide to host their own applications.

The application host server contains a copy of the application's code library. Spectrum^{NG} uses Microsoft's Updater Application Block technology to provide on-demand updates to the client. See <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpag2/html/updaterv2.asp> for more information on the mechanics of this system.

In a CSI Software-hosted environment, this allows the CSI Software team to provide timely updates and enhancements. When a user opens the application, the application communicates with the Application Host and reviews all files downloaded locally. This process only takes a few seconds.

If version differences are discovered, the application delays loading in order to download the update. The user is kept apprised of progress throughout the process. On completion of the update, the application loads with a new version.

This means that all users whose data is hosted by CSI Software can always be guaranteed to be running the latest version.

Companies who choose to host their own application can benefit from this process as well. CSI Software provides updated files to a self-hosted company, who then choose the time to update their own Application Host. That company's client applications are then be updated in the same manner as described above.

This would not apply to clients who run the application in a local desktop environment where the database and code library resides on the same machine. Updates in this case would occur via a standard Windows installation package.

Remoting Server

The remoting server is one of the major highlights of the Spectrum^{NG} architecture. This allows users to run the application in the feature-rich Windows forms environment, but receive the benefits of having data stored at remote locations, as in a web environment.

Remoting is very much a client/server scenario, yet without a direct connection to the data server as you would have in a classic client/server setting. The remoting server acts as an interface between the client and the data server. The client connects to the remoting server, which accepts requests for retrieval and persistence of data from the client. The remoting server then passes these requests on to the data server. At the end of the call, the remoting server collects any responses from the data server and returns these responses to the client application.

This technology is similar to Microsoft Web Services, but is actually more hardy and flexible. Web Services communicates with a client via XML remoting, which actually adds some bloat to remote calls and limits the flexibility of object-oriented development. Spectrum^{NG} utilizes binary remoting through IIS (Internet Information Services). While this is not as lean as pure binary remoting (a small overhead is added via the HTTP calls accessed through IIS), it is as fully flexible as pure binary remoting and the overhead is considerably better than Web Services.

This paradigm immediately benefits any company that doesn't like the idea of investing in infrastructure in order to run the latest and greatest technology. The minimum requirements for Spectrum^{NG} are very reasonable. Most companies should be able to meet minimum requirements with ease, or with very little investment.

Many applications that are as scalable and take advantage of such cutting edge technology require their owners to provide large investments in servers, networks, maintenance and technicians.

This is not necessary with Spectrum^{NG}. Along with minimal computer and operating requirements, the only other necessity is an internet connection.

Batch Queue Server

The Batch Queue server is another piece of the Spectrum^{NG} architecture that allows for maximum scalability. Spectrum^{NG} uses the Batch Queue Server to provide an asynchronous method for running processes that take a large amount of time to perform.

Monthly billing is probably the highest-profile example of how Spectrum^{NG} uses Batch Queue. There are three main processes to billing: Validation, Processing, and Acceptance. The length of time required to perform each of these operations depends on the number of clients to be billed. Although the current billing process for Spectrum^{NG} is approximately 40 times faster than the previous versions of Spectrum, customers with tens of thousands of accounts will experience some wait time.

In this case the client application submits a processing request to the Batch Queue Server; for the entire company. The client application is handed a unique identifier that allows it to request status updates on the job being processed.

The job will continue until it has finished without any required intervention from the client application. This allows the user to perform other tasks instead of tying up the machine for the period during which billing is running. If the user wants to get an update on the billing status, the client application requests the status from the Batch Queue server, which is displayed to the user.

Batch Queue is also used heavily in conjunction with large reporting requirements. For example, when a user requests an A/R Aging report, the Batch Queue is invoked to perform the aging logic on the data, providing a fresh and up to date view of the data.

Client Application

The Client Application is the visual and mechanical aspect through which end users will experience Spectrum^{NG}. The client application is based on the latest Microsoft development technology.

Spectrum^{NG} was created using Microsoft's Visual Studio 2003 in conjunction with other 3rd party user interface tools, gauges and charts. The combined result gives users an advanced visual look and an efficient operating tool. Items such as dynamic toolbars, tree menus that can be pinned and unpinned to provide more display area, and a rich set of GDI+ components place Spectrum^{NG} far ahead of the competition in terms of usability and functionality.

Using the Application Host (via Microsoft's Updater Application Block), users need never wonder whether or not they have the latest version of the software. Enhancements need not be held to major releases, but can more easily be planned along customer's needs.

Spectrum^{NG} Advantages

- Spectrum^{NG} uses Microsoft development tools that take advantage of the .Net Framework.

The .Net framework is a comprehensive set of libraries that sit on top of the standard Win32 API, providing a “wrapper” that allows developers to get at deeper Windows functionality easier than ever before in the history of application development. This allows developers to write slicker, more functional applications quicker than previously possible.

The .Net framework also provides Just-In-Time (JIT) compiling. JIT compiling means that an application is not actually compiled into machine level code until the first time it is used on a specific machine. In this manner, compilation is able to take advantage of specific conditions of the current environment and produce an application that will run as fast as possible on each machine on which the application is run.

There has been a wealth of third-party library development since .Net was introduced. Spectrum^{NG} takes advantage of these products to enhance the user experience even farther than the stock .Net tools allow.

- Spectrum^{NG} uses OOP (Object Oriented Programming) design concepts.

OOP has been around for quite some time. Using its benefits, however, has not always been a clear-cut decision. Much more time must be taken up front to design an effective strategy or OOP design could be worse than using older, more linear methods of programming.

However, OOP provides many advantages that help to outweigh any such disadvantages.

Code is more centralized. Functionality can be broken down into individual pieces. Each piece can be unit-tested to ensure that it upholds its requirements. Updates are easier to make in such an environment, as well as tracking down and fixing any issues that may arise.

The .Net framework itself is almost completely objected-oriented. It provides a number of rich features that make OOP much easier, and therefore more practical, to implement.

- Spectrum^{NG} uses flexible configuration and is therefore truly scalable.

Many applications try to make the claim that they are truly scalable, but Spectrum^{NG} fully delivers on that promise. Spectrum^{NG} can be run literally as an entire system, on a single machine; it can also be run on an enterprise-wide system that is separated physically with the only connection point being an internet connection. In either scenario, once the components are installed, all that is required to make it function in either capacity is a few settings in a configuration file.