



DataPipe Architectural Overview

EHS Data Base Management Software

2/1/2008

M. Douglas

 **DataPipe™**
The Complete Information Management Software Solution



**OCCUPATIONAL HEALTH
AND MEDICINE**

**ENVIRONMENT AND
WASTE MANAGEMENT**

**INDUSTRIAL HYGIENE
AND SAFETY**

Microsoft®
CERTIFIED

Partner

ISV/Software Solutions

Table of Contents

- INTRODUCTION 1
- OVERVIEW 1
- RUNTIME ENVIRONMENT SCENARIOS 2
 - Stand-Alone 2
 - Client/Server 2
 - Three-Tier 2
 - N-Tier 2
- DEPLOYMENT 3
 - Database Tier 3
 - Middle Tier 3
 - Client Tier 3
 - Auto-Deploy 3
 - Standard Windows Application 4
 - Standard Windows Application on Network Share 5
- SYSTEM REQUIREMENTS 5
 - Software 5
 - Client 5
 - Server 6
 - Database 6
 - Hardware 6
 - Client 6
 - Server 6
 - Database 7
- GLOSSARY 8

INTRODUCTION

This document is intended to provide Information Technologists, System Administrators, System Architects and other Information Technology specialists with an architectural overview of the .NET version of DataPipe. DataPipe was designed from the onset to be a truly distributed application. It can be deployed and run in numerous environments, ranging from stand-alone computers to multi-tier environments. There are three basic components that are needed in order to run DataPipe, regardless of the environment. They are: a client (a PC running Windows), an application server (a Windows machine running IIS 5.0 or later) and a database server (SQL Server, Oracle or DB2). It is important to distinguish between the logical and physical architectural structure of DataPipe. Logically speaking, DataPipe uses a multi-tier architecture, meaning that the application is broken up into components that run on different tiers: presentation, application server and database. Physically, however, DataPipe can run on one, two, three or more physical tiers and hybrid combinations of them.

OVERVIEW

The client (presentation) and middle (application server) tiers of DataPipe are written in Microsoft Managed Code. As such they require the .NET Framework. The client portion can be run as a .NET Windows executable. It is a .NET Managed User Control that can run within a .NET Windows executable. While at first glance this may seem to imply that client changes would need to be deployed manually to all clients, such is not the case. As discussed later in this document, there are numerous deployment options that eliminate the need to deploy updates to every machine that runs DataPipe. DataPipe offers the best of both worlds: easy deployment *and* a rich user experience. The client communicates to the middle tier via HTTP using DataPipe Web Services packaged in SOAP messages. The application tier requires Internet Information Server (IIS), which processes and returns Web Service requests. Depending on the deployment model, IIS may also serve as the deployment mechanism. Because HTTP is a stateless protocol and DataPipe needs to maintain state information (e.g. users have session duration and can time-out) we developed the DataPipe User Manager. The User Manager is a Windows Service that resides on a server somewhere (typically it would be the application server) and manages currently logged in user information. There can only be one copy of the User Manager, as there is only one set of users per DataPipe system. By default the User Manager is assumed to be on the application server. In a Web Farm scenario each of the web servers needs to know where the User Manager is. This is easily accomplished by specifying the IP address and port of the machine that is running the DataPipe User Manager in a configuration file on each machine in the Web Farm. The middle tier (IIS) Web Services talk to the data tier via a .NET Managed Provider. The DBMSs currently supported are SQL Server, Oracle and IBM's DB2.

RUNTIME ENVIRONMENT SCENARIOS

Stand-Alone

In this scenario everything (client, application server and database server) is installed on a single PC. We typically demo DataPipe from a laptop computer that has Windows, Internet Information Server (IIS) and SQL Server 2000 on it. TCP and HTTP services need to be running in order to communicate with the DataPipe User Manager and the IIS server.

Client/Server

In its heyday, Client/Server implied, for the most part, a fat client talking directly to a database. A constant connection to the database was needed in order for the application to work. Because of the need for a constant connection this approach couldn't really scale well.

DataPipe can be configured so that clients access a web server that contains both the application server and the database server. Although this may sound like a classic Client/Server configuration there is a major difference in how DataPipe works in this scenario. The difference is that no constant connection is required: DataPipe is stateless, and thus doesn't have the scalability issues inherent in a legacy Client/Server environment. The client machine needs an HTTP or HTTPS (SSL) connection to the web server. The server needs to be running TCP in order to communicate to the DataPipe User Manager.

Three-Tier

Three tier architecture simply means that there is a physical database tier, application server tier and client tier(s). The database tier may reside on a Windows server, but this is not necessary. As long as the application server has a way to communicate to the database server (whether it's running on Unix, Linux, a mainframe, etc.) DataPipe should work. The client needs HTTP or HTTPS (SSL) access to the application server. The server needs to be running TCP.

N-Tier

If DataPipe needs to scale beyond a three tier architecture there are numerous possibilities. There are two ways an application can scale: up and out. Scaling up simply refers to adding more or better hardware to a server so that it can handle more of a load. Unfortunately, regardless of Moore's law, there is only so far that one can scale up. Scaling out means adding more computers to a given tier and load balancing incoming requests so that they are distributed most efficiently to the machine that is least busy. Theoretically, one could scale out infinitely. There are two tiers that can be scaled out in DataPipe, the application tier and the database tier. Scaling out the application tier essentially means setting up a web garden or farm. DataPipe was designed to run in such an environment. Scaling out the database tier means using clustering technology offered by the particular DBMS vendor. How this works is transparent to DataPipe. Just as in the three-tier scenario the client

needs HTTP or HTTPS (SSL) access to the application server. Each web server needs to be running TCP.

DEPLOYMENT

Database Tier

It is assumed that the database will be installed and set up appropriately by a Database Administrator (DBA). We will provide the DBA with a set of Data Definition Language (DDL) script files to generate the database structure needed for DataPipe. The DBA will set up any desired security, user permissions, views, etc. in the database.

Middle Tier

This involves setting up a Virtual Root in IIS with the appropriate settings and directory structure. The necessary application files can be put on the v-root via an MSI install or a simple XCOPY.

Client Tier

There are numerous client deployment options available with DataPipe. Before we go through the deployment options, it is important to get a quick primer on deployment in general. Most people are familiar with Windows 32-bit applications. You probably use multiple such applications every day. They are typically installed on your local drive and offer a rich user experience. The biggest headache of traditional Win32 applications has been deployment. If an application file changed (say a DLL), it would need to get distributed to all client machines that ran the application. While there are tools like System Management Software (SMS) and Active Directory (AD) as well as third party distribution tools designed to ease this process, the traditional client deployment model puts a burden on System Administrators. Basic HTML-based web applications (those that don't use ActiveX controls or other such technologies) alleviated this problem, as any necessary change was simply deployed to a web server. Everyone automatically had the latest version of the app the next time they ran it. Deployment was to one machine, not many. While this made the lives of System Administrators easier, there was a downside. Strictly HTML and JavaScript-based apps can't offer the richness of a true Windows client-based application. End-users of such applications are often forced to take a step back with regard to what they had become accustomed to in straight Windows applications. DataPipe has addressed the deployment issue by providing numerous options, described below.

Auto-Deploy

One can simply run DataPipe by doing something like `http://WebServer/DataPipe.exe`. Magically, DataPipe runs on the client machine. **Web Server** would typically be an Intranet site within your organization. What happens when such a request is made is .NET checks

whether the exe file requested and its associated application files are in the client's download cache. If .NET doesn't find a file there it downloads it. If an update to an application file needs to be posted, it simply needs to be placed on the web server. Each time a client requests a file .NET checks if a newer version is on the web server. If so, it downloads and caches it automatically. The files in the cache are persistent; they remain there until they are removed.

Advantages

- All needed application files are downloaded and cached automatically.
- No need to deploy application files to every client machine.
- Application files don't get downloaded every time. They get downloaded and remain on the client machine. The only time a file is downloaded is if it's requested and it isn't in the client's download cache, or if a newer version is available.

Disadvantages

- Policy files must be deployed to client machines, because the default policy settings for the Intranet zone (the typical location that files would be auto-deployed from) don't have sufficient permissions to run DataPipe.
- If desired, an application icon would need to be set up.
- No "Windows-aware" Uninstall or Repair (i.e. from Control Panel) can be done.

Using the Auto Deploy model for the DataPipe client deployment also requires policy file changes. Specifically, you need to specify Full Trust to three Strong Names or to the URL where DataPipe is being Auto Deployed from. There are multiple ways to deploy the necessary policy files. They include:

- Using the Microsoft .NET Framework Configuration Tool from the client machine(s)
- Active Directory (AD)
- Systems Management Server (SMS)
- E-mail

If there are only a few clients and you have easy access to them, the simplest way to update their Security Policy is to use the Microsoft .NET Framework Configuration Tool. This gets installed as part of installing the .NET Framework and appears under the Administrative Tools folder in the Control Panel. Run the tool and you should see a Runtime Security Policy folder in the left-hand pane of the application. Expand the folder and you'll see that there are three levels where Security Policy can be set: Enterprise, Machine and User.

If running the Microsoft .NET Framework Configuration Tool from each machine isn't an option, there are several ways to automate the deployment of policy files. The Microsoft .NET Framework Configuration Tool has the ability to generate MSI files that, when run or distributed using AD or SMS, will apply security policies.

Standard Windows Application

This is typically how traditional Windows apps are installed. An MSI is run and all client components get installed to the local drive.

Advantages

- Typically done as an MSI install.

- No policy files need to be deployed as applications on a local drive are awarded Full Trust by default.
- Application icon and “Windows-aware” Uninstall (i.e. from Control Panel) can be done.

Disadvantages

- Deployment of client tier updates must be to every client machine. System Management Server (SMS), Active Directory (AD) or a third party tool can ease the deployment process.

Standard Windows Application on Network Share

Advantages

- Deployment is to one place, namely a network share.

Disadvantages

- Application files don't persist on the client machine. They get downloaded every time and aren't cached.
- In low bandwidth environments this can degrade the user's experience.
- Policies need to be deployed to each client machine as the network share is treated as the Intranet Zone which, by default, doesn't have the minimum requirements needed to run DataPipe.

SYSTEM REQUIREMENTS

Software

Client

These requirements may change over time. Refer to Microsoft's official [.NET Framework Requirements site](#) for current requirements.

- Windows (one of the following):
 - Windows 98
 - Windows 98 SE
 - Windows ME
 - Windows NT Workstation or Server (SP6a or later)
 - Windows 2000 (Professional, Server, Advanced Server, Datacenter Server)
 - Windows XP (Home, Professional)
 - Windows Server 2003 family
- NET Framework 1.1 Service Pack 1
- IE 5.01 (or IE 5.5, depending on the deployment model)
- Windows Installer 2.0 or later

Server

These requirements may change over time. Refer to Microsoft's official [.NET Framework Requirements site](#) for current requirements.

- Windows (one of the following):
 - Windows 2000 Professional with Service Pack 2.0
 - Windows 2000 Server with Service Pack 2.0
 - Windows 2000 Advanced Server with Service Pack 2.0
 - Windows 2000 Datacenter Server with Service Pack 2.0
 - Windows XP Professional
 - Windows Server 2003 family
- .NET Framework 1.1 Service Pack 1
- Microsoft Internet Information Server (IIS) 5.0 or later on Windows 2000, Windows XP Professional or Windows Server 2003
- Access to DBMS through Managed Provider
- The DataPipe User Manager (a Windows Service) will need to be installed on this or another server. TCP/IP will be needed to communicate between the web server(s) and the User Manager.

Database

- SQL Server (tested and verified for SQL Server 2000 on Windows 2000 Server), or
- Oracle (tested and verified for Oracle 8.1.7.0.0 and Oracle 9i on Windows 2000 Server), or
- IBM DB2 (tested and verified for DB2/NT 8.1.0 on Windows 2000 Server)
- Contact DBMS vendor for specific software requirements

Hardware

Client

- Required Processor – Pentium 90 MHz *
- Required RAM – 32 MB *
- Recommended Processor – Pentium 90 MHz or faster *
- Recommended RAM – 96 MB *

Server

- Required Processor – Pentium 133 MHz *
- Required RAM – 128 MB *
- Recommended Processor – Pentium 133 MHz or faster *
- Recommended RAM – 256 MB *

Database

- Contact DBMS vendor for specific hardware requirements

***Or the minimum required by the operating system, whichever is higher. While Microsoft makes these hardware requirements and recommendations, it has been Knorr Associates' experience that running modern versions of Windows and associated applications realistically requires more processing power and RAM than those listed above.**

GLOSSARY

.NET Framework – The programming model of the .NET environment for building, deploying, and running Web-based applications, smart client applications, and XML Web services. The .NET Framework provides the core services of .NET, similar to the Windows API in that it can be used by many Windows-based programs.

Active Directory (AD) – A Microsoft product that allows organizations to centrally manage and share information on network resources and users while acting as the central authority for security.

Client Tier – Also known as presentation tier. A logical layer of a distributed system that typically presents data to and processes input from the user, sometimes referred to as the front end. Usually, the client tier requests data from a server based on input, and then formats and displays the result.

Common Language Runtime (CLR) – The common language runtime is responsible for run time services such as language integration, security enforcement, memory, process, and thread management.

Data Definition Language (DDL) – A language for creating, altering and dropping data objects and integrity rules.

Data Tier - A logical layer that represents a computer running a DBMS, such as a SQL Server database, Oracle or IBM DB2.

Database Administrator (DBA) – An individual responsible for the design, development, operation, safeguarding, maintenance, and use of a database.

Distributed Application - A program written so that the processing can be divided across multiple computers over a network. Typically, a distributed application is divided into presentation, business logic, and data store layers, or tiers.

Internet Information Server (IIS) – Microsoft's web server product.

Microsoft Installer (MSI) – A Microsoft technology for installing, repairing, updating and uninstalling applications.

Middle Tier – Also known as application server or business logic tier. The logical layer between a user interface or Web client and the database. This is typically where the web server resides, and where business objects are instantiated. The middle tier is a collection of business rules and functions that generate and operate upon information. They accomplish this through business rules, which can change frequently, and are thus encapsulated into components that are physically separate from the application logic itself.

Scale Out – The ability to grow by adding instances/computers to provide acceptable service levels.

Scale Up - The ability to continue to grow within a single computer and continue to provide acceptable service levels.

SOAP – Simple Object Access Protocol. Provides a simple and lightweight mechanism for exchanging structured and typed information between peers in a decentralized, distributed environment using XML.

Stateless – HTTP is a stateless protocol, which means that it does not automatically indicate whether a sequence of requests is all from the same client or even whether a single browser instance is still actively viewing a page or site. As a result, building Web applications that need to maintain some cross-request state information (shopping carts, user information, and so on) can be extremely challenging without additional infrastructure help.

System Management Software (SMS) – A Microsoft product that delivers cost-effective, scalable change and configuration management for Microsoft Windows®-based desktop and server systems.

Virtual Root (v-root, application root) – When setting up a web application, the starting point directory on a web server.

Web Farm - A collection of multiple web servers that can run the same web application across different Web Servers and distribute the load evenly across the Web Servers.

Web Garden – One web server with multiple Processors which can run multiple instances of the web server worker process.