# The Application Configuration File

***Please do not alter this file without fully understanding the ramifications of change. And, back it up so that you can get back to a working starting point.***

We apologize and we understand that as an administrator you understand all of this “stuff”. But it needs to be written somewhere so that for those of us that occasionally need a refresher, it is available. This file, the configuration file, is provided so application behavior can be modified at run time without change to the underlying application executable. Building configurable applications is not a new concept. The idea of applications relying on configuration parameters to dictate behavior at run time has been around for many years. For those of us who were around Microsoft-based programming prior to .NET, the most common example is the use of INI files. INI files are simple text-based files with key and value pairs; the key is used to fetch the value, and the value is then used in some way to influence the application settings and/or resulting behavior. This way, you can modify the configuration file to drive program behavior without having to recompile the application.

INI files generally worked great, except for the following drawbacks:

* Everyone wrote their own parsers for dealing with them.
* You had to have them on the machine where the application was running.
* They required disk operations to read their files.

Microsoft moved away from INI files and towards storing everything in the Registry, which made application deployment much more difficult. As a result, Microsoft moved back to the concept of an "xcopy deployment," by which programmers deployed applications by simply copying files. Redmond used XML and some objects built into the Microsoft .NET Framework to bring new life to our old friend the application configuration file.

## System.Configuration BAsics

The System.Configuration namespace provides the functionality for reading configuration files. Microsoft released an XML schema dictating the format for configuration files that can be read using the System.Configuration API (ECM Library uses this extensively), enabling the accessing object to automatically consume [*read and apply*] the configuration files. This way, we can allow configuration files to be read without having to develop a bunch of plumbing to read the file and find the desired setting. Multiple types of configuration files are relevant to .NET, but this article focuses exclusively on the configuration files containing application-specific settings

The name and storage location of an application configuration file depends on the application type with which it is being used. ***We provide an automatic menu item to get you to this rather well hidden location if you are an administrator. If you are not an admin, please do not even look for this file.*** A configuration file for an executable (.exe) is located in the same directory as the application. The file is the name of the application with a .config extension. For example, notepad.exe would have a configuration file of notepad.exe.config. A configuration file for an ASP.NET application is called web.config and is located in the root of the application's virtual directory.

appSettings Section

An application configuration file follows a specific XML schema. The appSettings section is a predefined section of the configuration file designed to make it very easy to retrieve a value based on a given name. This is the easiest way to add application-specific settings into an application configuration file. The appSettings section of the configuration file consists of a series of "add" elements with "key" and "value" attributes. While the appSettings section is predefined, it is not included in a configuration file by default and must be manually added. A simple example of a configuration file would be the following:

<?xml version="1.0" encoding="utf-8" ?>

<configuration>

<appSettings>

<add key="ApplicationTitle" value="Sample Console Application" />

<add key="ConnectionString"

value="Server=localhost;Database=Northwind;Integrated

Security=false;User Id=sa;Password=;" />

</appSettings>

</configuration>

The AppSettings property of the ConfigurationSettings object in the System.Configuration namespace is used to get the settings. For example, to read either of the settings above, the following sample code would do the trick:

// Read appSettings

string title = ConfigurationSettings.AppSettings["ApplicationTitle"];

string connectString =

ConfigurationSettings.AppSettings["ConnectionString"];

Now, you could easily set a dynamic title value or read database connection string information from the application configuration file. This gives you the freedom to adjust settings without having to recompile any code.

### Customized Configuration Sections

If you have distinct groupings of configurations to be included in the application configuration file, consider creating a custom configuration section in lieu of the appSettings section. This will allow you more organizational structuring around the storage of various settings. You include a custom section by including the configSections element in the configuration file and a sub-element called section that defines the custom section and the handler for reading it. The .NET Framework contains an IConfigurationSectionHandler interface that dictates the required functionality handlers provide by allowing you to use custom handlers with the Configuration if you so choose. This example just uses handlers the .NET Framework already provides. Continuing the example from the previous section, the following configuration file contains a custom section and specifies which handler to use for reading:

<?xml version="1.0" encoding="utf-8" ?>

<configuration>

<configSections>

<section name="sampleSection"

type="System.Configuration.SingleTagSectionHandler" />

</configSections>

<sampleSection ApplicationTitle="Sample Console Application"

ConnectionString="Server=localhost;Database=Northwind;

Integrated Security=false;User Id=sa;

Password=;" />

<appSettings>

<add key="ApplicationTitle"

value="Sample Console Application" />

<add key="ConnectionString"

value="Server=localhost;Database=Northwind;

Integrated Security=false;User Id=sa;Password=;" />

</appSettings>

</configuration>

The following code reads the configuration settings into a dictionary, where you can easily access the values:

// Read customSection

System.Collections.IDictionary sampleTable = (IDictionary)

ConfigurationSettings.GetConfig("sampleSection");

string title = (string)sampleTable["ApplicationTitle"];

string connectString = (string)sampleTable["ConnectionString"];