Environment Settings Manager
Usage Guide

Version 1.6

Updated July 7, 2010

Hosted on CodePlex at http://EnvSettingsManager.codeplex.com

1. Introduction

The best way to introduce the Environment Settings Manager is to explain the motivations behind its creation:

1. Most software projects have dynamic configuration settings, and their values usually vary per deployment environment (development, test, production, etc.).
2. Maintaining the values of many settings across many environments is a difficult, error-prone task.
3. Configuration settings and their values must be understood and communicated across departmental boundaries, for instance from development to IT support.
4. It is common to store configuration settings in XML, whether they exist as an XML file in the file system or as XML stored in a database column.
5. Configuration settings must be synchronized with ever-changing program code, usually in a source control/versioning tool.

The Environment Settings Manager consists of an Excel spreadsheet and an associated command-line export utility. Here's how the Environment Settings Manager's spreadsheet and exporter can help with the issues described above:

1. Settings are maintained with a well-known, user-friendly tool – Microsoft Excel 2003 or newer.
2. The spreadsheet can store hundreds of settings and their values for dozens of environments in an easy-to-read tabular format.
3. Cell comments allow descriptions to be entered along with the setting name, which can provide helpful context and meaning to the setting name itself.
4. Settings may be saved in either XML or native Excel binary format – the editing experience in Excel is identical either way.
5. The settings spreadsheet (in XML or binary format) can be easily shared between interested parties.
6. Excel's cell locking allows selected setting values or names to be easily protected from editing by other users.
7. The command-line exporter tool makes it easy to export the settings and their values to a number of different XML formats, ready for consumption by a .NET configuration file, Windows Installer XML (WiX) project or Loren Halvorson's XmlPreprocess (http://xmlpreprocess.sourceforge.net/) tool.

Please be sure to read the Credits section for more on Loren Halvorson and the inspiration behind this project!

2. Software Requirements

The settings spreadsheet simply requires Microsoft Excel 2003 or newer.

The exporter utility requires the Microsoft .NET Framework 2.0 or greater. If you are exporting from a binary Excel 2003 XLS file, then you also need to install the Microsoft

JET OLEDB 4.0 driver. If you are exporting from a binary Excel 2007+ XLSX file, then you also need to install the Microsoft Office 2007 or 2010 Data Connectivity Components.

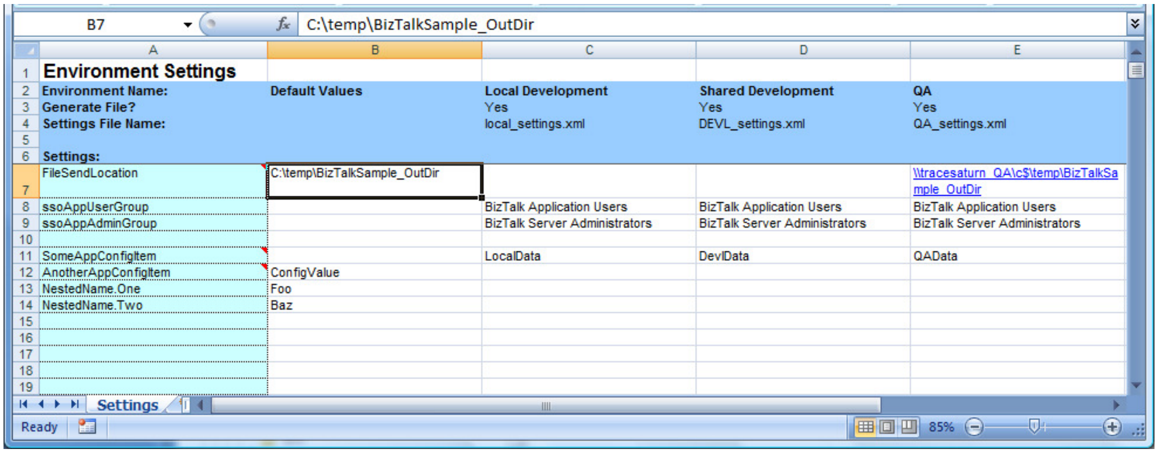
The exporter’s solution and project files are in Visual Studio 2008 format.

3. The Spreadsheet

This section includes a review of the structure of the spreadsheet and explains how to get started with your own copy.

3.1. Spreadsheet Elements

Let’s take a look at the spreadsheet, shown below in Excel 2007.



	A	B	C	D	E
1	**Environment Settings**				
2	**Environment Name:**	**Default Values**	**Local Development**	**Shared Development**	**QA**
3	**Generate File?**		Yes	Yes	Yes
4	**Settings File Name:**		local_settings.xml	DEVL_settings.xml	QA_settings.xml
5					
6	**Settings:**				
7	FileSendLocation	C:\temp\BizTalkSample_OutDir			\\tracesaturn_QA\c$\temp\BizTalkSample_OutDir
8	ssoAppUserGroup		BizTalk Application Users	BizTalk Application Users	BizTalk Application Users
9	ssoAppAdminGroup		BizTalk Server Administrators	BizTalk Server Administrators	BizTalk Server Administrators
10					
11	SomeAppConfigItem		LocalData	DevlData	QAData
12	AnotherAppConfigItem	ConfigValue			
13	NestedName.One	Foo			
14	NestedName.Two	Baz			
15					
16					
17					
18					
19					

Figure 1 - Sample Settings Spreadsheet

The layout is purposefully very straightforward. Let’s walk through the components.

Starting with column C, the columns correspond to your deployment environments, one per column. In the picture, column C corresponds to the local development workstation, column D corresponds to the shared/integration development environment, and column E corresponds to the QA testing environment. You can continue to use as many columns as you need for each of your environments.

Down the left side, starting at row 7, you’ll find the names of various runtime configuration settings. Each row from row 7 down corresponds to one particular setting, so combined with the columns defining deployment environments, we have a simple matrix of setting values to deployment environments.

Column B is a special column that defines default values for each setting. Values may be entered from row 7 down. In a particular row corresponding to one setting, if no value is provided in an environment-specific column, then the default value in column B will be used instead. In the picture, the default value at B7 will be used for the setting

"FileSendLocation" for the environments Local Development (column C) and Shared Development (column D), because the cells C7 and D7 do not contain values.

Rows 2-4 provide information to the settings exporter utility. The exporter will read these values to determine how many XML files to create and what to name them.

For the .NET <appSettings> and XmlPreprocess output formats, the exporter generates one XML file per environment (columns C and above) where "Generate File?" is set to "Yes" in row 3. It uses the filenames defined in row 4. In the picture, the exporter would generate three XML files named local_settings.xml, DEVL_settings.xml and QA_settings.xml (columns C-E). Each XML file would contain values for seven settings (rows 7-9 and 11-14).

For the WiX CustomTable output format, the exporter generates a single XML file named EnvironmentSettings.wxi. It will contain the settings from all environments where "Generate File?" is set to "Yes".

In the picture, cells A7, A11 and A12 have tiny red triangles in the upper-right corners of the cells. These are standard Excel cell comments. The settings exporter utility will write cell comments found in column A to the XML files along with the setting values (XML format workbooks only).

3.2. Starting a New Spreadsheet

The only choice required before you begin creating a new settings spreadsheet is which file format you prefer: Excel binary or Excel XML (SpreadsheetML). In most cases, XML is recommended because it provides virtually the same functionality as the binary format, can be used with common file-comparison tools and is branch/merge friendly in source control systems.

The Environment Settings Manager includes two templates that correspond to the two file formats: EnvironmentSettingsTemplate.xml and EnvironmentSettingsTemplate.xls. Simply make a copy of the template file of the format you prefer.

It is a good idea to customize the environment columns first. Open the workbook file in Excel and delete any environment-specific columns that you do not need. To add additional environments, make a copy of one of the existing environment-specific columns and paste it in a blank column to the right side of the existing columns.

Next, you'll probably want to delete the sample setting and values and begin adding your own setting names down the left side, starting at row 7.

4. The Exporter

The settings exporter is a command-line program that can read the settings spreadsheet in either Excel Binary or Excel SpreadsheetML XML format and export the configuration settings and values to one or more XML files.

The exporter can export to three different XML formats:

1. XmlPreprocess format
2. .NET <appSettings> format
3. WiX include format with a <CustomTable> definition

Here's an example of the XmlPreprocess XML format. This example corresponds to the "Local Development" environment shown in the picture of the spreadsheet above.

```
<?xml version="1.0" encoding="utf-8"?>
...
<settings>
  <property name="FileSendLocation">C:\temp\BizTalkSample_OutDir</property>
  <property name="ssoAppUserGroup">BizTalk Application Users</property>
  <property name="ssoAppAdminGroup">BizTalk Server Administrators</property>
  <property name="SomeAppConfigItem">LocalData</property>
  <property name="AnotherAppConfigItem">ConfigValue</property>
  <property name="NestedName.One">Foo</property>
  <property name="NestedName.Two">Baz</property>
</settings>
```

The format is very simple and easy to consume by your own utility programs or by the XmlPreprocess utility mentioned earlier.

Here's an example of the .NET <appSettings> XML format. This example again corresponds to the "Local Development" environment shown in the picture of the spreadsheet above.

```
<?xml version="1.0" encoding="utf-8"?>
...
<appSettings>
  <add key="FileSendLocation" value="C:\temp\BizTalkSample_OutDir" />
  <add key="ssoAppUserGroup" value="BizTalk Application Users" />
  <add key="ssoAppAdminGroup" value="BizTalk Server Administrators" />
  <add key="SomeAppConfigItem" value="DevlData" />
  <add key="AnotherAppConfigItem" value="ConfigValue" />
  <add key="NestedName.One" value="Foo" />
  <add key="NestedName.Two" value="Baz" />
</appSettings>
```

This format can be included directly into a parent .NET configuration file by using the <appSettings> element's file attribute:

```
<appSettings file="QA_settings.xml" />
```

Here's an example of the WiX CustomTable XML format. The WiX format includes all environments in one XML file, so this example includes the settings from all environments shown in the spreadsheet above.

```
<?xml version="1.0" encoding="utf-8"?>
...
<include>
  <CustomTable Id="EnvironmentSettings">
```

```
    <Column Id="Id" Category="Identifier" PrimaryKey="yes" Type="int" Width="4"
/>
    <Column Id="Environment" Category="Text" Type="string" PrimaryKey="no" />
    <Column Id="Key" Category="Text" Type="string" PrimaryKey="no" />
    <Column Id="Value" Category="Text" Type="string" PrimaryKey="no"
Nullable="yes" />
    <!--Environment: Local Development-->
    <!--Cell comments will be exported to the output XML files-->
    <Row>
      <Data Column="Id">1</Data>
      <Data Column="Environment">Local Development</Data>
      <Data Column="Key">SampleSetting</Data>
      <Data Column="Value">LocalDevValue</Data>
    </Row>
    <!--Environment: Shared Development-->
    ...
    <!--Environment: QA-->
    ...
    <!--Environment: Production-->
    ...
  </CustomTable>
</include>
```

The great thing about this model is that the spreadsheet consolidates the setting values for many environments into one place, with a familiar GUI editor in Excel. That same spreadsheet can be directly consumed by scripts or automated build processes to create simple XML files that can then be merged into a configuration file template, loaded into a database, etc.

The exporter command-line utility, EnvironmentSettingsExporter.exe, is very simple to use:

```
EnvironmentSettingsExporter.exe <ExcelFile.xls or ExcelFile.xml> <OutputPath>
[/F:<XmlPreprocess/AppSettings/WixCustomTable>]
```

The first parameter is the full path to the configuration settings spreadsheet, either in binary or XML format. Remember to surround the path in double-quotes if it contains spaces.

The second parameter is the full path to a folder that will hold the exported XML files.

The third, optional, parameter specifies the output XML format: XmlPreprocess, AppSettings or WixCustomTable. The default is XmlPreprocess.

The exporter looks at the “Generate File?” and “Settings File Name” values in the spreadsheet to determine whether or not to export each environment, and what to name each exported XML file.

5. Examples

Here are a few ways to use the Environment Settings Manager. Keep in mind that the spreadsheet and exporter can be used in a variety of situations, so these are just examples.

5.1. Simple Settings Management

The most basic way to use the Environment Settings Manager is simply to use the spreadsheet by itself. Even if you don't use the exporter utility, the spreadsheet by itself is a great way to keep track of your environments and configuration settings. Depending on how your organization works, the IT staff could manage the spreadsheet, the development staff could manage it, or they could share the responsibility.

5.2. .NET Application Configuration #1

Another way to use the Environment Settings Manager is to keep track of your settings and environments in the spreadsheet and merge the values directly into a single configuration file template. Have you tried maintaining a separate copy of your configuration file for every environment? It is extremely hard to keep all of the copies in sync and not make mistakes. A much better approach is to create one configuration file as a template, and automatically populate the template for each environment.

This is quite easy to do. First, copy the settings spreadsheet in your choice of binary or XML format and populate it with your environments, settings and values. Second, set up a script that takes an environment name as a parameter. The script will first execute the exporter utility to generate environment-specific XML files from the spreadsheet. Next, the script can use the XmlPreprocess utility (see Credits) to merge the settings from one of those XML files into a template configuration file, resulting in a complete, environment-specific copy of the configuration file.

I highly recommend using XmlPreprocess in conjunction with the Environment Settings Manager. It comes with simple documentation that describes how to set up your configuration file (any XML file) as a template.

5.3. .NET Application Configuration #2

An alternative method for using the Environment Settings Manager with .NET applications is to export the settings into <appSettings> format with the /F switch, and then include the generated file directly into your existing app/web.config. The .NET configuration schema allows you to place the entire contents of your <appSettings> section in a separate file, and simply specify a file path in the <appSettings> element's file attribute.

5.4. BizTalk Server Application Configuration

BizTalk Server is built in part on the .NET Framework, and BizTalk applications also have the need for dynamic configuration settings.

The Deployment Framework for BizTalk, also hosted on CodePlex, tightly integrates the Environment Settings Manager's Excel workbook and exporter. The Deployment Framework can automatically merge setting values from the spreadsheet into a BizTalk binding file template, and provides access to the spreadsheet values at runtime.

5.5. Windows Installer XML Configuration

Windows Installer XML (WiX) is an open-source toolkit that makes it easy to declaratively create MSI installers using XML files. You may want your MSI to dynamically update web.config or app.config XML files at install time using your settings spreadsheet as the configuration source.

You can easily create a WiX <include> file using the /F:WixCustomTable switch with the exporter. The settings from all environments will be exported into a single file: EnvironmentSettings.wxi. You can include this file directly into your WiX project.

At install time, you can read the values from the CustomTable, load them into properties and use the <util:XmlFile> task to push the values into any XML files that need to be updated.

For the specifics on how to implement this, please see this blog: http://blogs.technet.com/alexshev/archive/2008/02/14/from-msi-to-wix-part-6-customizing-installation-using-custom-tables.aspx

6. Credits

The Environment Settings spreadsheet was inspired by one that was originally created by Loren Halvorson (http://weblogs.asp.net/lorenh/), who also created the XmlPreprocess tool (http://xmlpreprocess.sourceforge.net/) mentioned above.

I first encountered XmlPreprocess and the original settings spreadsheet in the Deployment Framework for BizTalk (http://biztalkdeployment.codeplex.com). In that implementation, the script loads one of the exported environment-specific XML files into the BizTalk SSO database, an encrypted configuration store, from which the settings can be accessed at runtime inside BizTalk.

I initially created a spreadsheet that looked a bit different from Loren's original layout, but I didn't want to create a bunch of rework for those currently using it with BizTalk. As a result, the layout is basically Loren's original format, which allows an easy copy and paste from an existing spreadsheet into my version. However, the new spreadsheet has no macros or embedded scripting as did the original. The default output XML format used by the exporter is also identical to Loren's original structure, which was done specifically to ensure that the files would work with XmlPreprocess.

So, many thanks to Loren for the inspiration of his original spreadsheet, and for his XmlPreprocess utility!

7. The Author

As Enterprise Consultant for Digineer, Inc.'s Technology Solutions Group, Thomas Abraham helps a wide array of firms address their most challenging business software issues. Thomas has an extensive background in software development, architecture, configuration management and systems engineering, helping to build high-performance, mission-critical applications for companies including Nasdaq, Best Buy and Wells Fargo.

Over the last 13+ years, Thomas has worked with software technologies ranging from C/C++ to BizTalk Server to Exchange and .NET, was the lead author of the book "Visual Basic .NET Solutions Toolkit" from Wrox Press and a presenter at the 2006 SOA & Business Process Conference in Redmond, WA. Thomas holds a number of Microsoft certifications, including MCSD, TS for both BizTalk 2004 and 2006, TS for .NET 2.0, MCPD and MCT.

Thomas maintains a blog at http://www.tfabraham.com.