Web+Center
For Linux
Programmers Guide
Supplement

Please Note: Platform independent Web+Center programming information can be found in the Programmers Guide

Version 5.0

February 2005

Table of Contents

Programmer Notes for use with The Linux Platform

Introduction

The Web+Center product was originally designed and built to operate on a Windows platform. However, because it was written using VBScript and using ASP (Active Server Pages) as its primary implementation language, it was feasible to “migrate” the product to the Linux platform with relatively few changes, although there were enough to make this a non-trivial exercise. This document is provided to give some insight as to the critical differences in the code between the two platforms - especially those that you might need to be aware of if you are contemplating any modification or additions to the released version of the code.

Other Web+Center programming information that is not Linux Platform specific can be found in the Web+Center Programmers Guide located in the documentation directory in the file programmersguide.htm and programmersguide.doc.

Platform Requirements for Linux:

Development and test of the Linux variant was performed using the following major components:

- **Linux** (Red Hat, Version: 7.3)
- **Sun Microsystems / Chilisoft** Version***:*** 3.6.2 (Including CDONTS/Chili!Mail and FileUpload)
- **Apache** Webserver Version: 1.3.27
- **MySQL** Version: 3.23.49

Major Differences that required code modification for Linux

The following are the main areas that needed to be addressed – each will be described in greater detail below

- Case sensitivity of directory and file names in all URL references
- Case sensitivity in any references in the code involving Table-names (using MySQL)

- Dependencies on Window’s “Virtual Directory” Mechanism

- Dependencies upon exact behavior of “Global.asa”

Case Sensitivity in all URL references

The components of a URL are in large part references to directory paths and the names of files residing within those directories. Because all such names are case sensitive in a UNIX environment, so any URL type reference must be “correctly spelled” in terms of this case sensitivity. Thus, given a file called “MyFile.asp”, a reference to it that is spelled as “Myfile.asp” will not be successful.

Consider the following html tag:

```
<a href=Webcenter/customer/login.asp><img src=images/MyLogo.GIF>
```

In this example, all 5 “names” (Webcenter”, “customer”, “login.asp”. “images”, and “MyLogo.GIF”) must be spelled “correctly”. The name “MyLogo.GIF” is not the same as “MyLogo.gif”.

Case Sensitivity in any references in the code involving Table-names

Because tables in MySQL are manifested as files (from the perspective of the Linux Operating System) they too are case sensitive – and due care must be taken to spell them correctly (in terms of case sensitivity) in all references. The names of columns within a table are **not** case sensitive – unless of course the column reference is of the type tablename.columnname in which case the tablename component must still be spelled correctly.

Consider the following SQL statement:

```
SELECT Survey.Case_Number, Survey.Date, Cases.customer_id
FROM Survey
JOIN Cases ON Survey.Case_Number = Cases.case_number
```

In this example, all instances of “Survey” and “Cases” must be spelled exactly correctly because they refer to table names, whereas “customer_id” and “case_number” will be acceptable (even though their correct spelling might be “Customer_id” and “Case_number”) because column names are not case sensitive.

Dependencies on Window’s “Virtual Directory” Mechanism

In the original Windows-based implementation of Web+Center, extensive use is made of the “Virtual Directory” capability to provide maximum flexibility in configuring which directories go where. This mechanism essentially allows one to declare a “virtual” name for a directory (much like an alias), which is automatically mapped to the name of the real underlying directory by the web server whenever it needs to be resolved by the server. Thus a directory might appear to an outside user (as used in a URL for example) to have the name "tech40" whereas the actual underlying directory might be named as "TechCenter".

While similar mechanisms exist in Linux, they may not be directly accessible for use by the Web+Center Systems Administrator – particularly if the product is being installed and used on a remote-hosted system that supports many separate and distinct applications for different customers. Accordingly, the use of virtual directories has been replaced by "relative references" within the code when it is deployed in a Linux operating environment.

Consider the following line of code inside an ASP module, which is typical of the way references are made to images such as buttons –

```
<img  src=/<%=WCLanguageDirectory%>/submit.gif>
```

In this line, a substitution occurs at run time whereby the content of the variable called WCLanguageDirectory becomes part of the path to the image file. If the content of this variable was "XXXXXX" for example, then the reference at run time would appear as:

```
<img src=/XXXXXX /submit.gif>
```

When operating in a Windows environment, the content of this variable is set to the name of a "virtual directory" as described above, namely "wclanguage" – and this will then be mapped by the server to the name of the real directory where the image file "submit.gif" is actually located.

In a Linux environment however, the content of this variable is set to the literal value "./Language". When this literal is substituted, the path to the image file becomes a "relative" reference that points to the directory where image file is actually located.

Specifically, by the time the user's web-browser sees the generated html code, it will appear differently depending on which operating platform is being used thus:

```
<img src=/wclanguage /submit.gif>        (In a Windows environment)

<img src=/./Language /submit.gif>        (In a Linux environment)
```

How these two different values are initially set will be described in greater detail below under the heading "Critical Environmental Variables and Their Setting"

Dependencies upon exact behavior of "Global.asa"

In the original Windows environment, the module called "Global.asa" performs a critical role. Its name is not arbitrary – the name "Global.asa" is explicitly recognized by the

Windows web-server to cause special actions to be taken based on pre-specified events. Specifically, if this module contains a subroutine (written in VBScript) that is named with the key name “Application_OnStart”, then this subroutine will be triggered for execution whenever “…the application starts up….”. The interesting question is what that latter phrase means in a Linux environment – particularly in a shared remote-hosted situation where the Web+Center code might be seen as an individual application or might be seen as just part of the Apache server application.

Global.asa is used to set certain critical environmental variables in the application’s memory area (i.e. Application level variables of which there is a single copy for all users, as opposed to the Session variables where there is a separate memory space for each user).

To evade problems in this area, such Application level variables are set not only by the module called Global.asa (which might or might not get invoked automatically - depending on the specific operating environment) but also by a module called SetGlobal.asp which is called explicitly as explained in the next paragraphs. This way, it can be assured that the critical environmental variables can be set correctly no matter what definition is used for “…whenever the application starts up….”.

The module called “SetGlobal.asp” might appear to be a freestanding ASP module based on it having the file extension “.asp” at the end of its name.

It isn’t. In fact it is copied into the body of certain other modules at execution time by means of an

```
<!-- #INCLUDE FILE="SetGlobal.asp" -->
```

directive in those modules.

The reason for giving it a .asp file extension rather than the .inc file extension (more usually employed for such purposes) is for security purposes. This module contains certain sensitive information (such as for example the DSN string with the username and password parameters for accessing the database). If this were to be placed in a conventionally named .inc file and an end-user managed to somehow discover the filename (e.g. by provoking a failure that issued a diagnostic naming the file) then it would be a simple matter for that user to point his or her browser to it and cause its contents to be displayed in clear text on the user’s screen. (This is because a browser will treat a file with the .inc extension as a simple text file by default.). By naming this file with a .asp extension however, such direct invocation from an end-user’s browser will instead cause it to be treated by the server as it would any other .asp file – it is assumed to contain VBScript code, which is executed, and the output of that code is transmitted to the browser. In this case, executing the module "SetGlobal.asp" will merely cause the global environmental variables to be set (again) and nothing (sensitive or otherwise) will appear on the user’s screen.

The above <#INCLUDE> statement is contained as the first executable statement in each module that a user might use when entering the system afresh. For example, in DoTechLogin, but also a number of other modules where “initial entry” might occur. An example of such a module is DCF.asp which provides entry into the system when a “customer” clicks on a URL contained within an automatically generated email, but there are a number of others as well.

Critical Environmental Variables and Their Setting

In modifying or debugging the code as it is manifested in the Linux environment, it is important to understand the role played by a number of environmental variables, how they affect execution, and how/where they are set.

There are three of these variables, namely:

- **MyOS** This contains either the value “Windows” or the value “Linux” and is examined at run-time in a number of situations to decide of which of two paths should be taken in the code. For example, whenever the code needs to send email, it will check this variable and decide accordingly which email package to invoke.

- **MyDSN** This contains the DSN string that identifies the database to be connected to via the odbc connector together with the required username and password values. Used any time a connection needs to be established with the dbms.

- **WCLanguageDirectory** This contains a string that is used to complete directory path references as described in the section entitled “Dependencies on Window’s Virtual Directory Mechanism” above. As released, it is set to contain either the string “wclanguage” or the string “./Language” depending on whether it is operating under a Windows or Linux environment respectively.

These three variables are initially set as “application level variables” by code in either Global.asa and/or SetGlobal.asp (as described above under the heading “Dependencies upon exact behavior of Global.asa”.) They are set as “application level” variables so there is only one copy per system as opposed to “session level” variables where there would be one copy per connected user.

This is their permanent home from which copies are retrieved as required at run-time. They are retrieved from application level storage to local storage within each module as it executes by means of a simple assignment statement for each, e.g. of the form:

```
MyDSN = Application("MyDSN")
```

Because this code has to be executed by almost all modules, it is placed in a single <#INCLUDE> file which is invoked near the beginning of each module by means of a statement of the form:

```
<!-- #INCLUDE FILE="DB_define.inc" -->
```

Use of this <#INCLUDE> file is not new for this release of Web+Center. It was used similarly in previous releases, but now it establishes more than just the DSN for the database.

Thus the sequence of events may be summarized as follows

1. At application start-up, Global.asa is executed and as a result sets the values of the three environment variables in their corresponding "application" variables.

2. Subsequently, as a safety measure (to cater for the case of a shared Linux environment) SetGlobal.asp is executed whenever a new user first connects to the system (e.g. via Login.) This code checks to see if the application variables are present, and if not, sets them. This code is executed because it is an <include> file that is contained in any/all modules where initial entry by a user may occur.

3. Then whenever a module is invoked for execution, its executes an <include> of the module "DB_define.inc" which causes the contents of the application level variables to be copied into "local" variables within the memory of the executing module, thus making them available for subsequent use within the code.

Given the nature of what these three environmental variables do, any problems encountered in either accessing the database or resolving directory paths (usually manifested by "missing gif files") should make one immediately suspect that something is not occurring correctly as described above.

Installation Guidelines for these Environmental Variables.

There are three of these environmental variables

MyOS
WCLanguageDirectory
MyDSN

Of these, a user installing the system need only be concerned with setting the appropriate DSN string for the variable names MyDSN. The other two are set automatically as follows:

MyOS

This is set to the value of either "Windows" or "Linux" by means of a dynamic run-time test. This is accomplished by capturing the file-path of the currently executing module by means of setting a variable to Server.MapPath("."). This filepath is then examined for the presence of a "\" (reverse slash) anywhere within it. If one or more reverse slashes exist in the string, then this is determined to be a Windows environment. If no reverse slashes are found in this string, then this is determined to be a Linux environment. Thus nothing further is required for this variable when installing a system.

WCLanguageDirectory

Once the value of MyOS is determined, then it is a simple matter for the system to set the value of WCLanguageDirectory to either the value “wclanguage” or the value “./Language” depending on whether we are in Windows or Linux respectively.

Windows installations that wish to use something else besides the virtual directory “wclanguage” for residence of the “Language” folder, can modify this value as appropriate by setting it within the Global.asa module. (Because we can rely on Global.asa being executed at the correct moment in a Windows environment, no further modifications need be made for this variable – the code in SetGlobal.asa will detect the presence of this variable as already established and not touch it further.

MyDSN

No matter whether operating in a Windows or in a Linux environment, this variable needs to be set to the appropriate value at installation time.

In a Windows environment, it need only be set in one place, namely in the module “Global.asa” – although note that this module is replicated in each of the four application folders TechCenter, BusinessCenter, CustomerCenter and PocketCenter. Thus it is probably best to set it first and check for correct operation in TechCenter and then replicate it in the other three folders when you are satisfied that it is working correctly. Even though this variable is also set in the module “SetGlobal.asp” this can be ignored when operating in a Windows environment as the code will detect dynamically that the variable has already been set and not further modify it.

However, in a Linux environment, the value for MyDSN needs to be set in both Global.asa and in SetGlobal.asp as either one or the other may be executed at run-time depending on how exactly the Apache web-server is set up to interpret “application start-up”. Again, both of these files are replicated over all four application directories, thus it is probably best to set it first and check for correct operation in TechCenter and then replicate both of these files in the other three folders when you are satisfied that it is working correctly.

Migrating existing Web+Center databases from Windows to Linux

A set of database migration tools have been developed to take existing Web+Center database under Windows (Access or SQL) and migrate all of the data to a Web+Center for Linux Database using MySQL. Please refer to the document called LinuxDatabaseMigrationGuide.doc or LinuxDatabaseMigrationGuide.HTML for more information on how to migrate windows databases to MySQL.