WBT Manager™

Installation and Customization Guide

WBT Manager 1.8

Table of Contents

Index 0

Installing WBT Manager

Part I

1 Installing WBT Manager

Installing WBT Manager™ or upgrading to version 1.80 is a relatively straightforward process. However, there are numerous steps involved. You should read and fully understand the appropriate sections of this manual before you begin installing or upgrading the software.

How WBT Manager works is a must-read for new users. Even if you are upgrading from a previous version, you may want to review the information.

Performing a new installation guides you through the steps necessary to install WBT Manager for the first time, including preliminary planning and troubleshooting.

Upgrading from WBT Manager version 1.7 guides you through the upgrade process, which has been simplified from prior versions. A sample upgrade task plan is provided to help make the conversion process run smoothly and to minimize the risk of accidental data loss.

1.1 How WBT Manager works

Installation of WBT Manager™ consists of loading the Web Server Module on the designated system and loading the System Administrator Module on as many client systems as required. There are several required steps to performing a successful installation. A basic understanding of how WBT Manager components are connected together is essential.

WBT Manager consists of two modules. The Web Server module is an Active Server Pages (ASP) application that resides on the web server. This module provides browser-based interfaces for both students and local administrators. The System Administrator Module resides on a network workstation and provides the System Administrator interface. See the pertinent sections of the WBT Manager User's Guide for descriptions of these interfaces.

Both modules communicate with the WBT Manager database through ODBC connections as shown here.

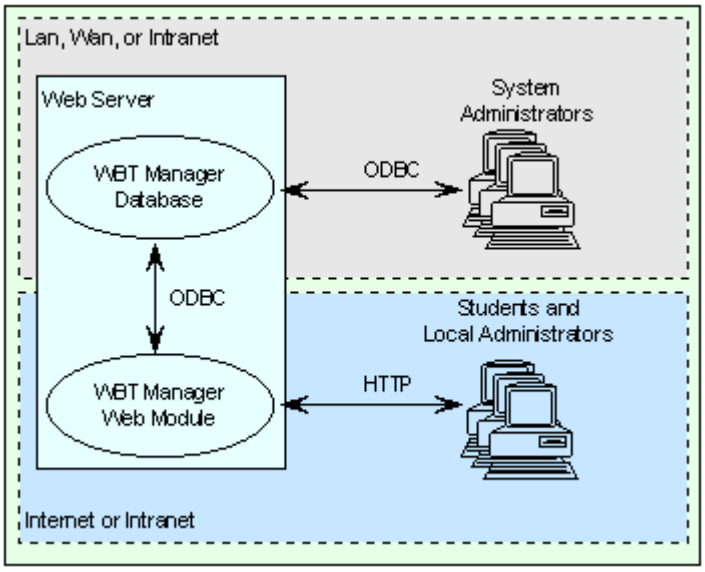


Fig. 1 - WBT Manager block diagram showing default installation. MS Access database resides physically on the web server

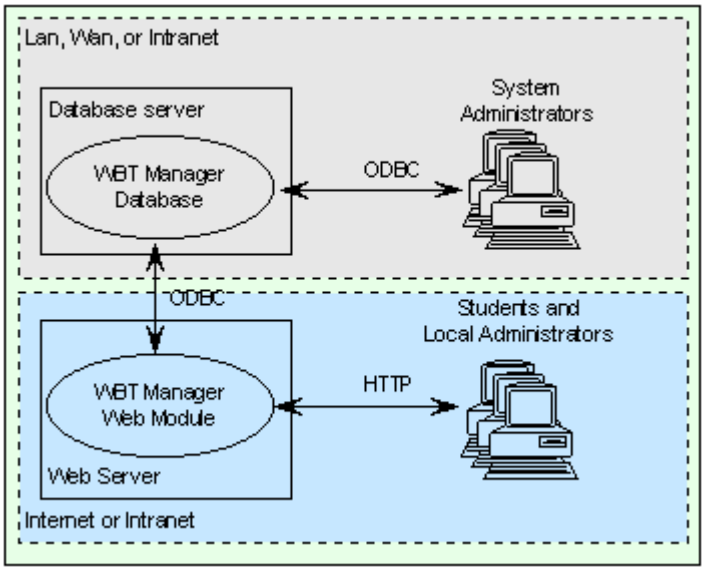


Fig. 2 - WBT Manager block diagram showing installation with a DBMS on a separate server.

The ODBC connections are critical to the operation of WBT Manager. The DBMS and ODBC drivers must support ADO recordsets from the Web Server Module as well as MFC (Microsoft® Foundation Classes) recordsets from the System Administrator Module. WBT Manager is fully tested and supported when run against MS Access, MS SQLServer, or Oracle®.

The System Administrator Module updates the database directly though its database connection. User interactions with the graphical interface are translated into SQL statements that retrieve data from or update tables in the database.

The student and local administrator interfaces have no direct connection with the database. The Web Server Module generates forms and links on the HTML pages that are sent to the user's web browser. Information is sent back to the Web Server Module in HTTP POST requests. The Web Server Module translates the posted data into SQL statements to retrieve data from or update tables in the database.

1.2 Performing a new installation

This section describes the installation process for first time users of WBT Manager or those wishing to completely "start over." If you are currently using an earlier version of WBT Manager and need to preserve existing data and customizations, see the *Upgrading from WBT Manager version 1.7* section.

1.2.1 Before you begin

Before beginning to install WBT Manager, you will need to make some preliminary decisions and ensure that your system is correctly set up for the installation. You should also familiarize yourself with the recommended sequence for installing the various components.

1.2.1.1 Preliminary decisions

1. Select a database engine.

   WBT Manager ships with a pre-configured Microsoft® Access database that is set up automatically. For most production uses you will want to use a server-based DBMS. WBT

Manager is fully supported under Oracle and Microsoft SQLServer. Setup instructions for SQLServer or Oracle databases can be found in **documentation\sqlserversetup.html** and **documentation\oraclesetup.html** in the WBT Manager installation set.

2. Select a location on the web server for the web application files.

   This location should be on an NTFS volume so that file permissions may be set.

3. Identify one or more systems to receive the System Administrator module.

   These systems must be capable of making an ODBC connection to the database.

4. Select a Display Name format and sort order for student and administrator names.

   You may set the system to use **<firstname> <middlename> <lastname> <suffix>**, **<lastname> <suffix>,<firstname> <middlename>**, or create a custom name format . You can change this format later, but doing so will require a conversion run through the database.

.

5. Decide how may levels of student hierarchy are needed and give them names.

   WBT Manager support 2-5 levels of hierarchy above students. The terminology for each level can be customised (default terms are "Corporation", "Region", "Division", "Organization", "Department"). The number of levels may be changed later if desired.

6. Decide if you want to use encrypted passwords.

   WBT Manager can optionally store user passwords as encrypted strings in the database. This adds some additional security to each student's login.

7. Decide if you wish to use the offline module and obtain an offline license key if necessary.

   The WBT Manager Offline Student Module is an optional feature that allows students to run a limited version of the WBT Manager web-server application on their local system under Microsoft Personal Web Server. Students can download courseware and take courses while not connected to the master WBT Manager system. You must purchase a separate license key to enable use of the offline module.

8. Decide whether to enable lesson status sharing.

   See the *About Shared Status* section for a description of this feature and its pros and cons. Lesson status sharing may be enabled at any time but could require a lengthy conversion if there is student results data in the statistics tables.

   Note: Once you have enabled lesson status sharing, you cannot disable it without completely rebuilding the database. This will cause the loss of all existing data. Make sure that you understand the implications of this option before enabling it.

8. Decide if you want to install the Student Import Toolkit on a System Administrator's workstation.

   This kit allows bulk import of student records. This component can be installed during the main installation process or added later.

10. Decide if you want to install the Batch Maintenance Module on the web server.

    This module performs periodic purges of database tables and can send email notifications for certain events. See the Maintenance Module help file **utilities\maintenance_module\wbtmmnt.hlp** in the WBT Manager installation set for more information.

11. Decide if you want to enable email notifications from the web server.

    This WBT Manager web module is capable of sending email notifications for some system events

and contains a link to allow students to have "forgotten passwords" emailed to them. See EMail Notifications for information about how this is controlled.

1.2.1.2 Pre-installation steps

Before beginning the installation process make sure that:

1. The web server software is up to date.

   WBT Manager requires Microsoft® Data Access Components version 2.5 or higher, along with IIS4 or higher. A copy of the MDAC installation set (mdac_typ.exe) can be found in the **utilities\offline_module\download** directory of the WBT Manager installation set. You can also download the latest version from **www.microsoft.com/data**.

   MSXML Parser 3.0 is also required on the web-server. The installer for MSXML Parser 3 service pack 1 (msxml3sp1.exe) is in the WBT Manager installation set in the **utilities\offline_module\download** directory. You can also download the latest version from **www.microsoft.com/xml**.

   Microsoft Windows NT® and Windows® 98 users may need to install or upgrade the **Microsoft Windows Installer** in order to install the MSXML Parser. Installers for this component can be found in the **utilities\offline_module\download\ver12_9x_me** and **utilities\offline_module\download\ver12_nt4** directories. You can also obtain the latest version from **www.microsoft.com**.

2. The IIS, ASP, and the ADO database objects work properly.

   There is a set of test scripts in the **\utilities\trouble_shooting\server_check** directory on the CD that may help. Use the **checkado.asp** script to verify that the ADO components are working.

3. The selected DBMS is installed and working.

   MS Access will be installed and configured by the setup programs. Other DBMSs must be set up prior to start of the installation process.

   Instructions for setting up MS SQLServer and Oracle may found in **documentation/oraclesetup.html** and **documentation/sqlserversetup.html** in the WBT Manager installation set.

1.2.1.3 Installation sequence

The recommended installation sequence is:

1. Install the web server application on the web server.
2. Install the Batch Maintenance Module on the web server (if desired).
3. Install the System Administrator module on one or more systems.
4. Install the Student Import toolkit on one or more administrator systems (if desired).
5. Install the WBT Manager license key.
6. Set the name format and sort.
7. Set the hierarchy levels.
8. Enable password encryption (if desired).
9. Convert to shared lesson status (if desired).

You can launch steps 1 through 4 from the main setup program (**wbtman_setup.exe**) in the installation set. You can also launch them directly from their respective directories.

1.2.2 Installing the Web Server Module

The help file **Web_Server_Application\readme2.hlp** in the WBT Manager installation set contains a step-by-step installation procedure to walk you through the installation. You should use this help file during the installation process.

If you plan to use a DBMS other than the MS Access database included with WBT Manager, you should set it up before starting the installation. Instructions for setting up MS SQLServer and Oracle may found in **documentation\oraclesetup.html** and **documentation\sqlserversetup.html** in the WBT Manager installation set.

1.2.3 Installing the Batch Maintenance Module

The batch maintenance module is a program that runs as a scheduled job on the web-server. It has the ability to purge various data tables by date and to send email notifications for various events. See the help file **utilities\maintenance_module\wbtmmnt.hlp** for more details.

To install the module, select it from the main Setup Menu or directly execute **utilities\maintenance_module\setup.exe**. Select a directory to install the programs and support files into.

See the help file mentioned above for information about setting parameters and scheduling runs.

1.2.4 Installing the System Administrator Module

The System Administrator Module installation is fully automated. To install the module, select it from the main Setup Menu or directly execute **Administrator_Module\setup.exe**.

The installation program will first check that you have Microsoft Data Access Components version 2.5 or higher installed. If not, you will be prompted to install it and to then re-start the WBT Manager installation. An installer is provided in the WBT Manager installation set in the **utilities\offline_module\download** directory.

The installation program will copy WBT Manager files to the selected destination directory and set up a shortcut on the program menu. Setup will not overwrite newer versions of these files. Files installed or updated in the **windows** or **windows\system** directories are:

Crystal Reports® files placed in the windows\crystal directory:

u2dmapi.dll, u2ddisk.dll, u2dpost.dll, u2fsepv.dll, u2fdif.dll, u2fhtml.dll, u2fodbc.dll, u2frec.dll, u2frtf.dll, u2fcr.dll, u2ftext.dll, u2fwks.dll, u2fwordw.dll, u2fxls.dll, crxlat32.dll, u2lsamp1.dll, p2ixbse.dll, p2lodbc.dll, u252000.dll, u25dts.dll, u25store.dll, u25total.dll, u2dnotes.dll, u2dvim.dll, u2frdef.dll, u2l2000.dll, u2lbar.dll, u2lcom.dll, u2ldts.dll, u2lexch.dll, u2lfinra.dll

Crystal Reports files placed in the windows\system directory:

crpe32.dll, implode.dll, p2sodbc.dll, crpaig32.dll, msvcrt40.dll, msvcrt20.dll, msvcrt.dll, crpe32.dep, ctl3d32.dll

1.2.5 Installing the Student Import Toolkit

The Student Import Toolkit is an ActiveX® component that allows programs to access student data. Sample programs written in Microsoft Visual Basic® are also included. The sample programs allow bulk import of student records and status values.

To install the kit, select it from the main Setup Menu or directly execute **utilities\student_import_kit\setup.exe**.

1.2.6 Installing the license key

The license key is installed by starting the System Administrator Module for the first time. An "invalid license key" error will be displayed, followed by the License Key Entry dialog.

Enter the license key exactly as it appears on your license key documentation. Once the key has been installed, you will be prompted for the administrator ID and password. The default ID for new installations is "admin." The default password is also "admin."

1.2.7 Setting the display name format and sort order

The student and administrator display name format defaults to **<first> <middle> <last> <suffix>**. If you would like to use **<last> <suffix>,<first> <middle>**, or if you would like to create a custom format you will need to run the site configuration tool.

The sort order that the system uses when displaying student and administrator names on menus defaults to the currently specified name format (in alphabetical order). The sort order can be modified using the configuration tool.

1. Launch the utility from the main Setup Menu, or directly execute **utilities\site_configuration\wbtmconfig.exe**.
2. Select the DSN to connect to the database and enter the database user id and password as well as a WBT Manager system administrator id and password.

   Notes:

   If you are using Oracle you will need to set up a DSN to the WBT Manager database using the Microsoft ODBC for Oracle driver from Microsoft Data Access Components version 2.5. The Oracle ODBC driver does not support the recordset types needed by the Visual Basic components.

   If you did not install the Student Import Toolkit on the system from which you will be running the site configuration utility, you may need to install the Visual Basic 6 runtime and support files. To do this, execute **utilities\site_configuration\setup.exe** and accept the defaults.
3. Select the desired name format and sort order from the first **Display Name Format** tab in the form.
4. Click the **Convert** button to convert existing names to the new format (otherwise names will not be converted unless/until existing student records are edited).

More detailed documentation on the use of the Site Configuration tool can be found in the "Utilities" chapter of the main WBT Manager User's manual.

1.2.8 Setting the hierarchy levels and names

With version 1.8, WBT Manager supports a variable number of hierarchy levels. There can be from 2-5 levels above students (default for a new installation is 5). The number of levels and the terminology used for each level can be modified using the configuration tool.

1. Launch the utility from the main Setup Menu, or directly execute **utilities\site_configuration\wbtmconfig.exe**.
2. Select the DSN to connect to the database and enter the database user id and password as well as a WBT Manager system administrator id and password.

   Notes:

   If you are using Oracle you will need to set up a DSN to the WBT Manager database using the Microsoft ODBC for Oracle driver from Microsoft Data Access Components version 2.5. The Oracle ODBC driver does

not support the recordset types needed by the Visual Basic components.

If you did not install the Student Import Toolkit on the system from which you will be running the site configuration utility, you may need to install the Visual Basic 6 runtime and support files. To do this, execute **utilities\site_configuration\setup.exe** and accept the defaults.

3. Select the desired number of hierarchy levels on the **Hierarchy Levels** tab.
4. Set the names to be used for each level on the **Custom Terminology** tab. Note that there are multiple entries for each hierarchy level that will need to be changed.

More detailed documentation on the use of the Site Configuration tool can be found in the "Utilities" chapter of the main WBT Manager User's manual.

1.2.9 Setting password encryption

By default, student and administrator passwords are stored in the WBT Manager database as clear text, and the password is displayed in the System Administrator Module when a student or administrator record is edited. If you would like to mask the display in the System Administrator Module or encrypt the password in the database you will need to run the site configuration tool as described below. More detailed documentation on the use of the Site Configuration tool can be found in the "Utilities" chapter of the main WBT Manager User's manual.

1. Launch the utility from the main Setup Menu, or directly execute **utilities\site_configuration\wbtmconfig.exe**.
2. Select the DSN to connect to the database and enter the database user id and password as well as a WBT Manager system administrator id and password.

   Notes:

   If you are using Oracle you will need to set up a DSN to the WBT Manager database using the Microsoft ODBC for Oracle driver from Microsoft Data Access Components version 2.5. The Oracle ODBC driver does not support the recordset types needed by the Visual Basic components.

   If you did not install the Student Import Toolkit on the system from which you will be running the site configuration utility, you may need to install the Visual Basic 6 runtime and support files. To do this, execute **utilities\site_configuration\setup.exe** and accept the defaults.
3. Select the level of password protection from the Password Security tab.

   When changing from non-encrypted to encrypted passwords (or back) a conversion of existing passwords must be performed. Interruption of this conversion will result in accounts that cannot log onto the system.

IMPORTANT: When you change from non-encrypted to encrypted passwords (or back), you must re-start the web module or run **reload.asp** (from **utilities\trouble_shooting\server_check**) before user accounts will be accessible**.**

1.2.10 Enabling the Offline Student Module

The WBT Manager Offline Student Module is an optional feature that allows students to download and execute courseware on their local systems while not connected to the master WBT Manager system.

To enable the Offline Module you will need a second license key. Once you have obtained the key:

1. Log onto the WBT Manager System Administrator Module

(The default ID for new installations is "admin". The default password is also "admin")

2. Select **License Key > Offline Module License** from the **Edit** menu.
3. Enter the license key exactly as it appears on your documentation.

Next you will need to put the Offline Student Module installer where the students can download it.

1. Copy the download directory from **utilities\offline_module** in the installation set to the directory on the web-server where WBT Manager is installed (overwrite the existing download directory).
2. Start the IIS administration program on the web-server and make sure that the download directory allows **read** access but not **execute** access.

For full information about how to use the Offline Module see the main manual.

1.2.11 Converting to shared lesson status

WBT Manager allows the same lesson to be used in more than one course. Shared lesson status causes a student's status for a lesson to be shared between courses containing the same lesson if more than one such course is assigned to the student.

WARNING: Once shared status in enabled for a site, it cannot be disabled without recreating the database and destroying existing data. Read "About Shared Status" carefully before running the conversion utility.

To convert the database to shared lesson status, execute **utilities\conversion_tools\shared_status\sharestats.exe**. This program will make the required changes to the standard database structure. After the conversion program finishes you must re-start the web-server application by re-booting the server or running the **reload.asp** script as described in the customization section of this manual.

Note: If you are using Oracle, you will need to set up a DSN to the WBT Manager database using the Microsoft ODBC for Oracle driver from Microsoft Data Access Components version 2.5. The Oracle ODBC driver does not support the recordset types needed by the Visual Basic components.

1.2.12 Typical installation problems

WBT Manager installation is usually very simple and trouble-free. The only recurring problem that clients have experienced with installation has to do with file-security restrictions on the web-server.

1.2.12.1 NTFS security settings

The most common installation problem that we encounter with WBT Manager is caused by over-zealous file security restrictions (typically the removal of the **everyone** account's access rights to the windows directories).

In general, the anonymous internet user account (IUSR_<machinename>) must be able to read and execute files in the **winnt** and **winnt\system32** directories and in the **Program Files\Common Files** directories. In order to use the Access database, this account must also be able to create and modify files in the windows **temp** directory and in the **winnt** directory (this is the least obvious requirement).

Here is a list of the directories and access rights required by the anonymous account.

\InetPub - READ
\InetPub\wwwroot - READ

\Program Files\Common Files\System\ - READ
\Program Files\Common Files\System\OLE DB - READ

\Program Files\Common Files\ODBC\Data Sources - READ
\WinNT - CHANGE (MODIFY on win2k)
\WinNT\System32 - READ
\WinNT\System32\Inetsrv - READ
\WinNT\Temp - CHANGE

The **utilities\trouble_shooting** directory in the installation set includes more instructions and utilities to assist in correcting installation problems.

1.3 Upgrading from WBT Manager version 1.70

Scripts and instructions are provided to allow you to upgrade from WBT Manager version 1.70 to version 1.80. If you have an earlier version of WBT Manager you must obtain a copy of version 1.70 and upgrade to that version first.

> Note: You will need a new license key to perform this upgrade. Contact your WBT Manager reseller to obtain this license (no charge for clients with active maintenance agreements).

It is highly recommended that you read through Performing a new installation before continuing.

1.3.1 Before you begin

Version 1.80 of WBT Manager has made moderate changes to the database structure that was used in version 1.70. In order to upgrade from 1.70, you will need to modify the structure of your database. SQL command scripts and conversion utilities have been supplied for this purpose.

> Note: You will need a new license key to perform this upgrade. Contact your WBT Manager reseller to obtain this license (no charge for clients with active maintenance agreements).

1.3.2 The Upgrade Process

The fact that you are upgrading rather than performing a new installation indicates that you have data in your database that you wish to preserve. The process described here provides precautions that are important in preserving that data.

1.3.2.1 Creating an upgrade task plan

Upgrading from version 1.70 to 1.80 is a fairly simple process but, as with any operation involving live data, you will want to guard against unforeseen problems. To avoid the risk of losing data, you should develop a detailed task plan before beginning to make the upgrade. The task plan should contain provisions for:

backing up the database and the WBT Manager virtual directory

restoring to the previous state in case the upgrade must be aborted

A sample task plan is provided in the *Sample Upgrade Task Plan* section. You will need to modify this plan to take into account your specific circumstances.

1.3.2.2 Performing trial runs

Before beginning the actual upgrade, you should perform one or more trial runs of the installation and conversion process using the following steps.

1. Perform steps 2-8 from "Preliminary Tasks" in the *Sample Upgrade Task Plan*.
   This will result in a set of customized files to be used in the upgrade process.

2. Create a test WBT Manager site identical to the production installation (preferably on a separate test server). To do so:

   a. Install WBT Manager Web-Server Application version 1.70 along with any patches.

   b. Copy any customized files from the production site.

      If you do not have records of which files in your production site were customized, you will need to use a file compare utility to compare the "clean" version of 1.70 with a copy of the production site. Note the names of any customized files for use later.

   c. Make a copy of the production database.

      For MS Access: Copy **wbtman.mdb** from the production site database directory to the test site database directory at a time when no users are logged in.

      For MS SQLServer: Use the DBMS tools to perform a full backup of the production database and restore that backup to a new database. *Do not use* the Export and Import wizards to copy the database. These wizards do not transfer the database indices and constraints.

      For Oracle: Use the DBMS tools to perform a full backup of the production database and restore that backup to a new database.

   d. Link the copy to the test site by creating a DSN to the new database and modifying the **settings.asp** file (line 2) to use that DSN.

   e. Verify that the test site functions exactly as the production site before proceeding.

3. Execute the conversion task plan against the test site.

   You will probably want to keep track of the elapsed time for the complete conversion. This information will help you in scheduling downtime for the production site.

4. Modify the task plan as necessary. Then repeat steps 1 and 2 until the conversion proceeds with no problems.

5. Verify the behavior of the test site.

1.3.3 Sample Upgrade Task Plan

The following sections contain a generic task plan. You should modify it to match your specific circumstances. To facilitate such modification, **taskplan.rtf** contains an editable RTF version of the sample plan.

1.3.3.1 Preliminary tasks

1. Determine the current database schema version.

   The upgrade scripts supplied with WBT Manager 1.80 are designed to upgrade the database from version 1.11 to 1.12. You can view the current database schema version by selecting "About WBT Manager…" from the system administrator **Help** menu. Look for "DB Version". If it is not 1.11, contact your vendor for customer support.

   Version _____

2. Identify customized files.

   When customizing WBT Manager you should keep a log of all changes made including the desired effect and an outline of the specific changes for each modification. If such a record is

unavailable, you can identify modified files using the following procedure:

a. Install a "clean" version of WBT Manager 1.70 on a test system.
c. Install any WBT Manager patches that were installed on your production site.
d. Use a file compare utility to compare the "clean" installation against a copy of your production installation.

3. Install a clean copy of version 1.8 on a test server.

4. Copy the "user-customizable" files from the production site to the test server.

The following files are considered "user-customizable" but have changed from version 1.7 to 1.8.

database/appsettings.ini.
There are a number of new settings and flags in version 1.8. Using with file from 1.7 will not cause any problems but the file will not contain any information about the new settings (default values will be used). It is recommended that you use the new version of the file and apply any changes specific to your installation to it.

- **database/pagesettings.ini**
  There are a number of new entries in this file due to the changes in the local administrator interface. It is recommended that you use the new version of the file and apply your custom changes to it.
- **database/eventq/evenq.ini**
  There is a new version of the default queue handler (wbtman5qp.dll). If you have not customized the queue handler then it is recommended that you use the new version of this file.
- **asp/nav1.asp, asp/nav1_old.asp**
  There have been changes made to these files to allow buttons to be hidden based on flags in appsettings.ini. If you have customized these files you may want to consider applying your customizations to the new versions. If you have not customized these files then use the new versions.
- **html/ie.css, html/ns.css**
  The following new style entries will need to be added to these files.

```
.clsStudentDetails {font: 8pt Arial, Helvetica, sans-serif; color:#666666}
td.clsStudentDetails {margin:0 0 0 0}
.clsCourseDetails {font: 8pt Arial, Helvetica, sans-serif; color:#666666}
td.clsCourseDetails {margin:0 0 0 0}
textarea.clsCourseDetails {width:100%;height:12ex}
div.clsCourseDetails {width:100%;height:12ex;overflow:auto}

input.clsFormData25 {font: 9pt Arial, Helvetica, sans-serif;width:25ex}
select.clsFormData30 {font: 9pt Arial, Helvetica, sans-serif;width:30ex}
input.clsFormDateTime {font: 9pt Arial, Helvetica, sans-serif;width:45ex}
input.clsFormDate {font: 9pt Arial, Helvetica, sans-serif;width:35ex}
```

The following files are considered "user-customizable" and have not changed from version 1.7 to 1.8.

asp/amenu1.inc, **asp/amenu2.inc**

- **asp/cat1a.inc**, **asp/cat1c.inc**
- **asp/emailpw.inc**
- **asp/enrollstud1.inc**
- **asp/login2.inc**
  asp/logoff.inc
  asp/maint.inc

- **asp/ofldone.inc**
  asp/regstud.inc
  asp/set1.asp
  database/eventq/xsl/*.*
- **html/*.* (except for closeme.html, closeme2.html, ie.css, and ns.css)**
- **images/*.***
  msgtemplates/*.*

5. Apply customizations to version 1.8 of any modified files that are not defined as "user-customizable" above.

6. Make copies of the customized files for use in the final conversion.

   Files:

   _____ _____ _____ _____ _____ _____

   _____ _____ _____ _____ _____ _____

   _____ _____ _____ _____ _____ _____

7. If you modified the email wrapper DLL wbtmanemail.dll (or script wbtmanemail.sct). Make a backup copy.

8. If you customized the standard event queue handler (wbtman3qp.dll or wbtman4qp.dll), make a backup copy.

9. Schedule site downtime.

   Use the elapsed times from your trial run conversions to schedule site downtime. Note that the MDAC upgrade and MSXML installation steps may require that the entire server be re-booted.

   Site will be down starting _____ until approximately _____

10. Identify new directory for WBT Manager.

    Install new version to _____.

11. Verify disk space

    Database server:

    Size of full database backup file: _____

    Available disk space:
    (at least 2 times backup file size needed) _____

    Web server:

    Size of WBT Manager virtual directory _____

    Available disk space in target directory
    (need enough to duplicate original +20MB) _____

1.3.3.2 Conversion tasks

1. Re-verify disk space

   Database server ok? _____

   Web server ok? _____

2. Disable production site.

   <specify steps here>

3. Install MSXML 3 parser on web-server if needed.

   Web module installation program will check for this component. Installer (msxml3sp1.exe) is available in **utilities\offline_module\download** or from **www.microsoft.com/xml**. NT systems will probably need to have the Windows Installer program installed first (**utilities\offline_module\download\ver12_nt4**).

4. Upgrade MDAC on web-server if needed.

5. Back up database.

   <specify steps here>

6. Copy existing WBT Manager virtual directory to new target directory from step 10 of "Preliminary Tasks".

   <specify steps here>

7. Run first database upgrade script.

   a. Execute **runssql.exe** from the **\utilities\database** directory in the WBT Manager installation set.

   b. Click the [**...**] button next to **Data Source Name** and select the DSN which connects to the production database.

      If you are prompted for a Login ID and Password, use the database account that was originally used to set up the WBT Manager database.

      NOTE: if you are using SQL Server and executing runSQL.exe on the SQL Server system runSQL will log onto SQL Server using your Windows account and will not ask for the SQL Server Account. You may need to set up an ODBC connection from a different system in order to connect using the correct account.

   c. Make sure that the **Pause on SQL Error** and **Remove trailing semi-colons before executing** check boxes are checked and that **Pause between statements** is not checked.

   d. Execute each of the files in the appropriate table below, in the order shown.

      To execute a script file:

      1. Click the [**...**] button next to **SQL Script File**.
      2. Browse down to the **\utilities\database\scripts** directory in the WBT Manager installation set.
      3. Select the file and click **Open**.
      4. Click the **Execute** button.

Script for MS Access	
Script filename	**Notes**
access_111_to_112a.sql	Modifies table structure and adds additional default values.

Script for MS SQLServer	
Script filename	**Notes**
sqlserv_111_to_112a.sql	Modifies table structure and adds additional default values.

Script for Oracle	
Script filename	**Notes**
oracle_111_to_112a.sql	Modifies table structure and adds additional default values.

8. Run conversion program to move completion rules. If any of your courses have completion rules set up in lesson or block definitions then DO NOT SKIP THIS STEP.

   Completion requirement rules have been moved from the lesson and block level in version 1.7 to the course level in version 1.8 to more closely follow the AICC specification. A conversion program has been provided to perform this operation.

   a. Execute **movecmp.exe** from the **\utilities\conversion_tools\completion** directory in the WBT Manager installation set.

   b. Click the [**...**] button next to **Data Source Name** and select the DSN which connects to the production database. Oracle users: use the same ODBC DSN as is used by the System Administrator.

      Enter the database Login ID and Password for the database account that was originally used to set up the WBT Manager database.

   c. Click the "Start" button to run the conversion. If the start button is not enabled then there is nothing to convert.

   d. Click "Close" when the conversion is done.

   You may re-run the conversion up until the remaining database upgrade scripts are executed.

8b. ORACLE USERS ONLY - Run conversion program to move Organization and Department HTML code.

HTML code is now located in a single table for all student hierarchy levels and must be copied to the new table (ORGNHTML) from DEPTHTML and COMPHTML. Oracle P-SQL does not support copying long character columns using an insert statement with a sub-select so this cannot be done by script. A conversion program has been provided to perform this operation.

a. Execute **oraclehtml.exe** from the **\utilities\conversion_tools\oraclhtml** directory in the WBT Manager installation set.

b. Click the [**...**] button next to **Data Source Name** and select the DSN which connects to the production database. Oracle users: use the same ODBC DSN as is used by the System Administrator.

Enter the database Login ID and Password for the database account that was originally used to set up the WBT Manager database.

c. Click the "Start" button to run the conversion. If the start button is not enabled then there is nothing to convert.

d. Click "Close" when the conversion is done.

9. Run the remaining database upgrade scripts.
See step 7 for runSQL instructions.

Scripts for MS Access	
Script filename	**Notes**
access_111_to_112b.sql	Modifies table structure and adds additional default values.
initdb.sql	Fills in new report and menu style definitions.

Scripts for MS SQLServer	
Script filename	**Notes**
sqlserv_111_to_112b.sql	Modifies table structure and adds additional default values.
initdb.sql	Fills in new report and menu style definitions.
grantdb.sql	Grants user access to the new tables. Modify the Login ID **wbtmanuser** in this file to match the Login ID used by the web-server application in **database/appsettings.ini**.

Scripts for Oracle	
Script filename	**Notes**
oracle_111_to_112b.sql	Modifies table structure and adds additional default values.
initdb.sql	Fills in new report and menu style definitions.
grantdb.sql	Grants user access to the new tables. Modify the Login ID **wbtmanuser** in this file to match the Login ID used by the web-server application in **database/appsettings.ini**.
orasyn.sql	Creates public synonyms for tables.

10. Run conversion program to update local administrator permissions.

    The "Add/Edit Department" administrator permission has been removed and replaced with "Modify Student Hierarchy". The new permission code uses the value associated with "Add/Edit Organization" so administrators who had department edit permission will no longer have the ability to edit department records. A conversion utility has been provided to convert "Add/Edit Department" to "Modify Student Hierarchy".

    a. Execute **fixperm2.exe** from the **\utilities\conversion_tools\adminpermissions** directory in the WBT Manager installation set.

    b. Click the [**...**] button next to **Data Source Name** and select the DSN which connects to the production database. Oracle users: use the same ODBC DSN as is used by the System Administrator.

Enter the database Login ID and Password for the database account that was originally used to set up the WBT Manager database.

c. Click the "Start" button to run the conversion. If the start button is not enabled then there is nothing to convert.

d. Click "Close" when the conversion is done.

11. Install Version 1.80 of the Web-Server Application.

    Install the new version of WBT Manager into a the new target directory Web server (on top of old files). You can launch the file installation program from **\web_server_application\setup.exe**.

12. Copy the files modified during "Preliminary Tasks" steps 4 and 5 to the newly installed copy of WBT Manager version 1.8.

13. Change the WBT Manager Program IDs in any custom files.

    Use Windows Explorer (or other search utility) to find files in the target directories that contain the string "wbtman4". Open each file in a text editor and change "wbtman4" to "wbtman5" in all CreateObject() function calls.

    Exception: if you have customized the queue handler DLL wbtman3qp.dll or wbtman4qp.dll and wish to continue using it rather than re-customizing wbtmanqp5.dll then do not change the references to the dll in database/eventq.ini.

14. Restore the custom version of wbtmanemail.dll (of wbtmanemail.sct) from step 7 of Preliminary Tasks to the **winnt\system32** directory.

15. Point the new installation at the production database.

    If using MS Access as the database:

    Copy **wbtman.mdb** from the production site **database** directory to the new **database** directory.

    Modify the WBT Manager Database DSN to point to the copy of the database in the new location.

    For all database types:

    edit **appsettings.ini** and change the DSN to match that actually used.

    set the **userid** and **password** to match those used by the production database.

    Example:

```
[database]
connectionstring=dsn=WBT Manager Database;uid=%%uid%%;pwd=%%pwd%%
userid=wbtmanuser
password=wbtmanuser
encryptpassword=yes
passwordisencrypted=no
```

16. Copy any courseware beneath the existing version 1.70 directory to the same relative position in the version 1.80 directory.

    Typically this will be the contents of the **courses** directory.

17. Modify the NTFS security settings on the new installation directory and its subdirectories as described in the web module installation help file (**\web_server_application\readme2.hlp**).

18. Change the properties of the WBT Manager virtual directory (in **IIS Administrator**) so that the

home directory is the new installation directory instead of the version 1.6x directory.

19. Install the new version of the batch maintenance module (if desired).

    See *Installing the Batch Maintenance Module* for details.

20. Install version 1.80 System Administrator module and update the license key.

    See *Installing the System Administrator Module* for more details. You may want to uninstall the version 1.70 module first.

    After the System Administrator Module is installed, start it and enter your version 1.8 license key when prompted (you will get an "invalid license" error first).

21. Install the version 1.80 Student Import Toolkit (if desired).

    See *Installing the Student Import Toolkit* for more details. You will have to uninstall version 1.70 first.

22. Verify operation of site.

    Verify to your satisfaction that the new version of the site is operating as desired. Do not continue with the upgrade until you are satisfied.

    Two important things to check:

    Are courses that were assigned to Organizations or Departments still assigned to those Organizations and Departments?

    If you have courses that contained completion requirement rules at the Block and Lesson level were those rules all moved to the course level?

    <specify your own test steps and approval criteria here>

23. Perform a full backup of the current database.

    <specify your own backup steps here>

24. Release the site to users.

    <specify your own steps here>

For Later: Uninstalling Version 1.7

After you are satisfied that version 1.8 is working to your satisfaction you may want to uninstall version 1.7. To do so:

1. Make a backup copy of the following files in the **winnt\system32** directory**.**

   wlkuptbl.dll
   wbtmanq.dll
   wbtmanqevtmsg.dll
   wbtmanemail.dll
   wbtmannoemail.dll
   wbtmanemail.sct

   These files are shared between versions. They should not be uninstalled by 1.70 after 1.80 is

installed, but it is possible that the reference counts were not updated properly by the installer.

2. If you have customized the event queue processor **wbtman3qp.dll** or **wbtman4qp.dll** then make a backup copy of it (**winnt\system32\wbtman3qp.dll, winnt\system32\wbtman4qp.dll**).

3. Uninstall the version 1.70 web module using the "Add/Remove Programs" function in the Windows control panel. You should re-boot the system to completely remove all files.

4. After re-booting the system check for the files backed up in step 1. If any are missing, restore them. Then open a command prompt and change directories to winnt\system32. Execute the following command to re-register the files.

   regsvr32.exe <filename>

   wbtmanqevtmsg.dll does not require registration
   Register only one of the "email" files. The one that you register will depend on how you want email to behave.
   See "EMail Notifications" information.

   Also restore the copy (if any) of wbtman3qp.dll or wbtman4qp.dll made in step 2 and re-register it as well.

5. Check for any of the following files in **winnt\system32**. If any exist, delete them.

   wbtman4a.dll
   wbtman4b.dll
   wbtman4c.dll
   wbtman4d.dll
   wbtman4e.dll
   wbtman4f.dll
   wbtman4g.dll
   wbtman4h.dll
   wbtman4m.dll
   wbtman4qp.dll (if you will not be using a custom version)

6. Delete the version 1.70 installation directory to remove any remaining files.

1.3.3.3 Aborting a failed conversion

1. Uninstall WBT Manager version 1.80 using the "Add/Remove Programs" function on the Windows control panel. You should re-boot the system after this step.

2. Check that all of the version 1.70 DLLs are still in the **winnt\system32** directory. Look for:

   wbtman4a.dll
   wbtman4b.dll
   wbtman4c.dll
   wbtman4d.dll
   wbtman4e.dll
   wbtman4f.dll
   wbtman4g.dll
   wbtman4h.dll
   wbtman4m.dll
   wlkuptbl.dll
   wbtmanq.dll
   wbtmanqevtmsg.dll
   wbtmanemail.dll
   wbtmannoemail.dll
   wbtmanemail.sct

If any DLLs are missing, reinstall the version 1.70 web module to a temporary directory and re-apply patches that contain any of the above files.

3. Restore the database from the full backup made in step 5 of conversion tasks.

   <specify your own restore steps here>

4. Modify the properties for the WBT Manager virtual directory to point it back to the version 1.70 installation directory.

5. If using MS Access, modify the WBT Manager Database DSN to point back at the version 1.70 copy.

6. Re-start the web-server.

7. Verify operation of the site.

   <specify your own steps here>

8. Release the site to the users.

   <specify your own steps here>

1.4 Using a different DBMS

WBT Manager has been successfully tested with and is fully supported under the Microsoft Access, Microsoft SQLServer and Oracle DBMSs.

Contact Integrity eLearning customer support if you have questions about using other database engines.

1.5 About Shared Status

One of the goals of the AICC and SCORM specifications is to allow content to be easily re-used. WBT Manager supports this goal by allowing an individual lesson or block of lessons to be used in more than one course.

The ability to re-use the same lesson in multiple courses does have a drawback. In the default WBT Manager installation, the student's status for a lesson is not propagated to all instances of the lesson in the assigned curriculum. This means that if two courses containing the same lesson are assigned to a given student, the student must complete the lesson twice in order to receive credit for both courses.

Starting with Version 1.5, WBT Manager has offered a "Shared Status" conversion that will cause WBT Manager to use the same status entry for a lesson in all of the courses that contain it. After this conversion is run, students will not have to repeat lessons that occur in more than one of their courses.

Status sharing has its own drawbacks, however.

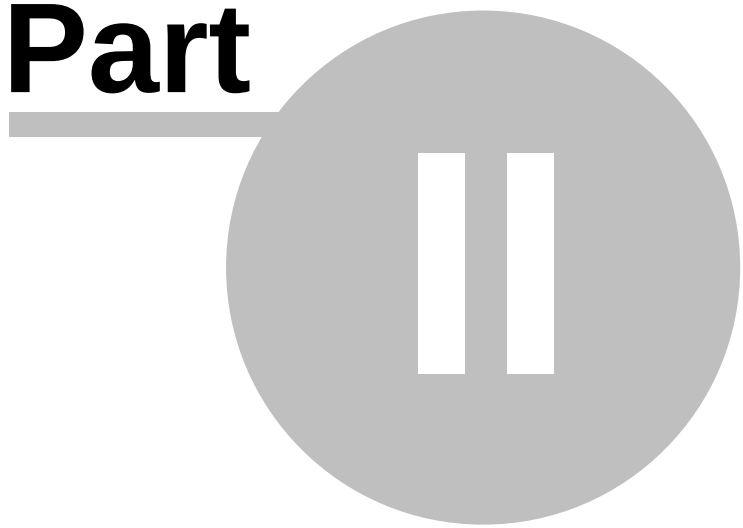
System performance may be degraded by the need to compute status for more than one course each time a lesson is completed.

Custom reports may be more difficult to create and the results may be misleading. For example, unless it is very carefully designed, a report showing accumulated hours of instruction sorted by course will not show the same number of hours as a similar report sorted by the lessons in the course.

It is important to weigh the advantages and disadvantages of shared status carefully before you make the decision to convert.

Once your database is converted, it cannot be converted back to "unshared" status tracking.

Customizing WBT Manager



2 Customizing WBT Manager

One of the design goals for WBT Manager™ has always been to allow clients to produce a training web site consistent with their own appearance guidelines. There are various levels of customization possible in the WBT Manager Web-Server Application. These range from simple graphics swaps to complete menu reprogramming. Before we begin detailing the available customizations, there are a few preliminary requirements and guidelines to discuss.

Proprietary Requirements

We at Integrity eLearning want you to be able to customize the web module as you see fit. However, for brand name and copyright reasons, we have the following requirements:

If you customize the home page and frameset to the extent that you remove the graphics containing the WBT Manager name, you must include a "Powered By WBT Manager" logo in a prominent location on the home page. See the *WBT Manager Logo requirements* section for more information.

You may not remove or conceal the Integrity eLearning copyright notice on the logout page. You may, however, add your own notice by including it in **logout.inc** as described in the *User customizable files* section.

Restarting the WBT Manager web application

Changes to the application and page settings files will require that the web application be re-started before the changes will take effect. The simplest way to reload the settings is to re-boot the web-server. However, because re-booting may be inconvenient, we have provided a script to reload the application settings without a reboot. Copy the **reload.asp** file from **utilities\trouble_shooting\server_check** to the root of the WBT Manager virtual directory. When you need to reload the application settings, open **reload.asp** in a browser.

Keeping track of your custom changes

Customization beyond the first level (built-in capabilities) will require that you modify files in the WBT Manager Web Module. If you should have to rebuild your site, any changes you make will be lost. For this reason you should:

Keep a backup copy of changed files

After you make and test changes, you should make a copy of the new versions of the files and store them in a secure location. You may also want to keep a copy of the original WBT Manager files.

Keep a customization log

If you modify files that are not designated as "user-customizable," future releases of WBT Manager will always overwrite your changes. It is also possible that a future release will change the way that "user-customizable" files are used so that you will need to re-apply your changes to the new files. In order to re-apply the changes you need to know what and where they are. You should keep a customization log file in a secure location.

Suggested log file contents for each customization

An identifier for the customization

A short description of the intent of the customization

A list of the affected files

For each modified file:
Description of the changes to the file

Before and after code samples for complicated changes

Ideally, the customization log file should contain enough information that a developer who had not seen the original customization would be able to duplicate the effect starting from the standard WBT Manager file versions.

2.1 Customization - built in capabilities

Some customization can be done through the system administrator interface and utilities without modifying any WBT Manager files directly.

2.1.1 Changing the name format and sort order

WBT Manager stores a display name for each student and administrator along with the individual parts of each name. The display name is automatically formatted when a student or administrator record is edited.

The name format defaults to **<first> <middle> <last> <suffix>** but can be changed to **<last> <suffix>,<first> <middle>**, or to a custom format. See *Setting the display name format and sort order* in the "Installing WBT Manager > Performing a new installation" section for instructions.

2.1.2 Setting captions for user-defined text

Student, Course, and Lesson records all include a set of text fields that can be used for any desired purpose. Each of these fields has a caption associated with it that appears on the forms for those records.

WBT Manager allows the captions for the user-defined text fields to be customized. See the "Utilities" section of the main WBT Manager manual for instructions.

Note: The web-server application must be re-started for these changes to take effect in the web interface.

2.1.3 Changing the student hierarchy depth and names

With version 1.8, WBT Manager supports a variable number of hierarchy levels. There can be from 2-5 levels above students (default for a new installation is 5). The number of levels and the terminology used for each level can be modified using the configuration tool.

See *Setting the hierarchy levels and names* in the "Installing WBT Manager > Performing a new installation" section for instructions.

2.1.4 Adding HTML to menus

The WBT Manager Web Module automatically generates the following menus for students:

Student Menu

Lesson Menu (when a course is selected from the Student Menu)

Certificate Program Menu (when a certificate program is selected from the Student Menu)

There are areas at the top and bottom of each of these menus where a block of HTML code specified through the System Administrator module may be displayed. This is useful for adding graphics for logos, special instructions, and links to email addresses, collaboration tools, or external reference sites.

To add the HTML blocks to the menus simply enter or copy valid HTML code to the **HTML1** or **HTML2** tabs on the property sheets for the following objects:

For the **Student Menu**: Modify the **Organization** object (HTML blocks will be included for all students in that organization) or the **Department** object (HTML blocks will be included for all students in that department).

For **Lesson Menus**: Modify the HTML1 and HTML2 properties for a specific **Course** object (HTML blocks will be included on that course's lesson menu).

For **Certificate Program Menus**: Modify the HTML1 and HTML2 properties for a specific **Certificate Program** object (HTML blocks will be included on that program's menu).

For more information see the properties for each object in the main WBT Manager manual.

2.1.4.1 HTML Placeholder Variables

When using the HTML fields to add links to a menu it is often usefull to include some information about the the student that the menu belongs to.

The code that embedds the HTML in the menu pages first replaces certain characters strings (placeholder variables) with run-time data. The available placeholder variables are:

Placeholder string	Run-time value
%nm%	Student's display name, HTML Encoded for proper display on the menu page.
%enm%	Student's display name, URL Encoded for use in the HREF of a link.
%lid%	Student's login id, HTML Encoded for proper display on the menu page.
%elid%	Student's login id, URL Encoded for use in the HREF of a link.
%pwd%	Student's WBT Manager password, URL Encoded for use in the HREF of a link.

2.2 Customization - some assembly required

The next level of customization involves simple modifications to some of the WBT Manager Web Module files. To perform these changes you will require a text editor and a graphics editor capable of editing GIF files.

Knowledge of CSS style attributes is needed if you plan to modify text styles.

2.2.1 User customizable files

A number of the files included in WBT Manager control the general appearance of the system and are expected to be customized by most customers. The following files are designated as "user-customizable" and will not be overwritten by future releases of WBT Manager.

Although the files will not be overwritten, in some cases the structure of WBT Manager may change sufficiently that you will need to use the new version of some or all of them. When this is the case, the upgrade instructions will list the affected files and explain the differences in their usage. In such cases you will need to apply your changes to the new files.

2.2.1.1 Individual customizable files

The individual customizable files are discussed below. In the file names, <root> indicates the WBT Manager Web Module installation location. New files for version 1.70 are marked with an asterisk.

<root>/database/appsettings.ini

This file contains the Data Source Name, User ID and Password to be used to connect to the WBT Manager database. It also controls the frame-set sizes, page background colors or graphics, and various other application settings. You can edit this file with a text editor if these values need to be changed. The contents of this file are discussed in the *Application Settings Files* section. Note that the Web server application must be re-started for changes to take effect.

<root>/database/pagesettings.ini

This file contains the settings that apply to specific pages within WBT Manager. The contents of this file are discussed in the *Customization - page level changes* section. Note that the Web server application must be re-started for changes to take effect.

<root>/database/eventq/eventq.ini

This file contains the settings that control the event queue processor added in WBT Manager 1.61. The contents of this file are discussed in the Event Queue Notifications section. Note that the Web server application must be re-started for changes to take effect.

<root>/database/eventq/xsl/*.xsl

These XSL files are used to format email messages sent by the event queue processor added in WBT Manager 1.61. See the contents of each file for an explanation of its purpose.

<root>/default.asp, <root>/asp/set1.asp

These files contain the WBT Manager top/bottom and left/right framesets respectively. You can easily change the orientation and size of each frame or add additional frames. You should not, however alter the names or parent-child relationships of the standard frames. The navigation menu update function depends on the parent of **right** having a child called **left** which contains the navigation bar.

> **Note:** WBT Manager 1.6 introduced a switch in **appsettings.ini** which will switch the navigation bar orientation between horizontal and vertical.

<root>/html/banner.htm

This file contains the top banner for WBT Manager and can be changed with an HTML editor. If you change the height of the banner, you will need to adjust the banner window's frame height by modifying **appsettings.ini,** as shown below. (You can also modify the frameset directly in **default.asp**). Note that the web-server application must be restarted for this change to take effect.

```
[dimensions]
        .
        .
        .
;top banner height
bannerheight=56
```

<root>/html/home.htm

This file contains the WBT Manager home page and can be changed with an HTML editor.

<root>/html/ie.css, <root>/html/ns.css

These pages contain the web page style-sheets for Microsoft Internet Explorer and Netscape Communicator. Styles can be edited with a text editor or html style-sheet editor.

> **Note:** There is now a way to set the primary style-sheet based on the browser major version. It is also possible to include a secondary style-sheet on a page by page basis. See the *Changing text styles* and *Setting CSS styles at the page level* sections for details.

<root>/html/cat_ie.css, <root>/html/cat_ns.css

These pages contain the web page style-sheets used by the course catalog pages.

<root>/asp/logoff.inc

The contents of this file will be embedded on the logoff page between the "logged out" message and the WBT Manager copyright notice. You can place any valid HTML code in this file. The supplied version of this file redirects the student to the WBT Manager home page after 4 seconds.

> **Note:** If you use code to automatically redirect the user to a different page, you must include a delay of at least 4 seconds so that the copyright notice is displayed. The web server will need to be restarted for any changes to take effect.

<root>/asp/amenu1.inc, <root>/asp/amenu2.inc

These files can be used to add HTML code that will be inserted at the top and bottom of the Local Administrator menu. You can add any valid HTML code from this file by appending it to a VBScript string variable which will be included when the page is generated. The final behavior is similar to that of the HTML 1 and HTML 2 fields in the System Administrator module for Organizations, Departments, Courses, and Certification Classes. The supplied versions of these files include sample code to demonstrate how to do this.

<root>/asp/cat1a.inc, <root>/asp/cat1c.inc

These 2 files are used to control the appearance of the course catalog. See the *Customization - the course catalog* section for details.

<root>/asp/emailpw.inc

This file contains VBScript code that sets the text fields for and sends an e-mail message when a student requests an e-mail of a "forgotten password."

<root>/html/emailpw.html, <root>/html/emailadminpw.html, <root>/html/emailstudpw.html

These files contain forms to request the email of a forgotten password. **emailpw.html** is a generic form with fields for all of the possible parameters (see the *Integrating with eCommerce and other sites* section for documentation). **emailadminpw.html** and **emailstudpw.html** are the forms used by the **Forgot your password?** links on the login pages.

<root>/asp/regstud.inc, <root>/asp/enrollstud.inc, <root>/asp/enrollcertstud.inc

These files contain VBScript code used to restrict access to the "add student", "enroll student in course", and "enroll student in certificate program" functions in the **External Scripting Interface Kit**. See the *Integrating with eCommerce and other sites* section for documentation.

<root>/html/regstud.html, <root>/html/enrollstud.html, <root>/html/enrollcertstud.html, <root>/html/verstud.html

These files contain sample forms for the **External Scripting Interface Kit**. See the *Integrating with eCommerce and other sites* section for documentation.

2.2.1.2 Customizable graphic files

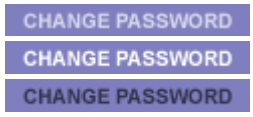
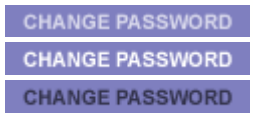
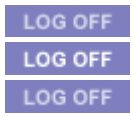
Most of WBT Manager's graphics can be replaced with custom graphics. The graphics described in the following sections are located in the **images** directory. New graphics for version 1.80 are listed in the appropriate tables, where they are marked with asterisks (some files flagged as new may have been included in patches for version 1.70).

2.2.1.2.1 Navigation bar buttons

The navigation bar buttons can all be replaced to suit your organization's desired appearance. The following table includes details on the individual button graphics. The files can be found in the **images** folder.

In the image filenames, _p indicates the normal button state, _g indicates the mouseover state, _d indicates the disabled state.

NOTE: admlog_d.gif, stulog_d.gif, help_d.gif, ahelp_d.gif, loff_d.gif, and aloff_d.gif are never displayed. They are included to eliminate HTTP 404 errors that may be logged by the web-server because of the way that the button display logic works.

Navigation bar buttons		
File Name	**Button Name / Description**	**Standard Image**
admlog_p.gif **admlog_g.gif** ***admlog_d.gif**	Administrator Logon	
stulog_p.gif **stulog_g.gif** ***stulog_d.gif**	Student Logon	
cpw_p.gif **cpw_g.gif** **cpw_d.gif**	Change Password (student)	
acpw_p.gif **acpw_g.gif** **acpw_d.gif**	Change Password (administrator)	
cat_p.gif **cat_g.gif** **cat_d.gif**	Course Catalog	
help_p.gif **help_g.gif** ***help_d.gif**	Help (student)	
ahelp_p.gif **ahelp_g.gif** ***ahelp_d.gif**	Help (administrator)	
loff_p.gif **loff_g.gif** ***loff_d.gif**	Log Off (student)	
aloff_p.gif **aloff_g.gif** ***aloff_d.gif**	Log Off (administrator)	

continued on next page

Other Buttons		
File Name	**Button Name / Description**	**Standard Image**
close.gif	Close popup window	
descr.gif **tdescr.gif**	Show description (tdescr is used on tabbed lesson menu)	
enrollnow.gif	Enroll Now button in catalog for self-enrollable courses	
logon.gif	Logon button on logon screen	
selectreport.gif	Select report button on report selection menu	
start.gif	Display report button on report criteria menu	
save.gif	"Save" button on data entry screens	
update.gif	"Update" button on data entry screens	
clear.gif	"Clear" button on data entry screens	
reset.gif	"Reset" button on data entry screens	
delete.gif	"Delete" button on data entry screens	
new.gif	"New" button on selection screens.	
go.gif	"Goto edit form" button on selection screens.	
enroll.gif	"Enroll" button on course and certificate program assignment screens	
drop.gif	"Drop" button on course and certification class assignment screens	

continued on next page

Other Buttons (continued)		
File Name	**Button Name / Description**	**Standard Image**
postchg.gif	"Post Changes" button on manual result entry screens	
resetstat.gif	"Reset Stats" button on course menus	
crsstatus.gif	"Recalc Status" button on course menu	
edit.gif	"Edit" button on course enrollment form.	
boffline.gif	"Go Off-Line" button for the course menu. Valid only for systems on which the optional "Offline Student Module" is installed.	
selall.gif	"Select All" button on student or course selection forms.	
abort.gif	"Abort" button from "cancel offline session" forms	
addme.gif	"Add to waiting list" button on self-enrollment window in catalog	
back.gif	"Back" button used to navigate to the previous page during student and course selection.	
continue.gif	"Continue" button that closes the student account maintenance window.	
help.gif	Used where help button is desired on a page (in a popup window for example).	
msg_submit.gif	Submit button from the student messages window.	
remove.gif	"Remove" button from course waiting list edit forms.	
search.gif	Search button from course and student selectors and catalog.	
submit.gif	Generic submit button	
selectinstructor.gif	Launches instructor selection menu from course and lesson section edit pages.	

continued on next page

Other Buttons (continued)		
File Name	**Button Name / Description**	**Standard Image**
***cancel.gif**	"Cancel" button.	
***selectorg.gif**	"Select Organization" button. Used on administrator and student edit forms.	
***smallback.gif**	Small "back" button for top of administrator pages.	
***smalledit.gif**	Small "edit" button.	
***smallgo.gif**	Small "go" button.	
***smallselect.gif**	Small "select" button.	
***smallnew.gif**	Small "new" button.	
***tbutton.gif**	"T" button for "today's date" on date picker.	
***xbutton.gif**	"X" button for clearing fields on forms.	

2.2.1.2.3 Status graphics

The icons that indicate the status of courses and lessons can be customized. The graphics are listed in the table below. The files can be found in the **images** folder.

Filenames beginning with "t" are used on tabbed menus, where they may appear on a colored background.

Status Graphics		
File Name	**Description**	**Standard Image**
notstart.gif **tnotstart.gif**	Lesson or course not yet started	
incomp.gif **tincomp.gif**	Lesson or course incomplete	
comp.gif **tcomp.gif**	Lesson or course complete	
offline.gif	Lesson or course is currently in "offline mode"	
notenrolled.gif	Student is not enrolled in the course	
notready.gif	The "start-after" date in the student's enrollment record for the course has not been reached.	
expired.gif	The student's enrollment in this course has expired.	
needpreq.gif **tneedpreq.gif**	A prerequisite for the course or lesson has not been completed by the student.	
disabled.gif **tdisabled.gif**	The course or lesson has been disabled by the system administrator.	
***pasttarget.gif**	The course is past the student's "target" date.	
nostatus.gif ***tnostatus.gif**	The course element does not have a status (reference items). Transparent image.	

2.2.1.2.4 Menu graphics

The graphics used in building course menu displays can be customized. These graphics are listed in the table below. The files can be found in the **images** folder. Files marked with an asterisk were added with WBT Manager version 1.61.

Menu Graphics		
File Name	**Description**	**Standard Image**
indent.gif	Used to create indents in course menu displays. Default is 10x10 transparent GIF.	
minus.gif **plus.gif**	Expand/collapse symbols from tree-style lesson menu	− +
tab0.gif - tab9.gif	Tab images for tabbed lesson menu	
catindent.gif	Used to create indents in the catalog display. Default is 10x10 transparent GIF.	
catminus.gif **catplus.gif**	Expand/collapse symbols from main catalog display.	− +
catnotoggle.gif	Placeholder graphic to take the place of catminus or catplus if an entry in the catalog display has no children. Default is a transparent GIF.	
hrline.gif	Horizontal line (used instead of <hr> for cross-browser compatibility).	
iconilt.gif*	Icon on course menus indicating that a course or lesson is Instructor-Led.	

2.2.1.2.5 Miscellaneous graphics

Misc. Graphics		
File Name	**Description**	**Standard Image**
***tvorg1.gif - tvorg5.gif**	Icons representing the different student hierarchy levels on the treeview used to select students and organizations in the local administrator module.	
***tvplus.gif,** ***tvminus.gif,** ***tvnotoggle.gif**	Icons indicating the "toggle" state of nodes in the local administrator module treeview (tvnotoggle.gif is transparent).	+−
***checked.gif,** ***unchecked.gif**	Images of checkboxes in each state. Used to represent checkbox data that user is not allowed to edit.	

2.2.2 Changing the home page and web graphics

One of the simplest customizations, and one which is done by virtually all clients, is to replace the home page and modify the button graphics.

Changing the home page

The home page is contained in the files **<root>/html/banner.htm** (top banner) and **<root>/html/home.htm**. You can modify these files with an HTML or text editor.

> **Note:** If you completely remove the WBT Manager logo you must include a "Powered By WBT Manager" logo in a prominent location on the home page. See "WBT Manager Logo requirements" for more information.

If you change the height of the banner page you will need to adjust the frameset to make it fit. To do so, change the "bannerheight" value in the **appsettings.ini** file as shown here.

```
[dimensions]
        .
        .
        .
;top banner height
bannerheight=56
```

> **Note:** The web-server application must be re-started for this change to take effect.

Changing the application graphics

Graphics for all of the buttons and icons used in the web application can be found in the **images** directory. (See *Customizable graphic files* in the "User customizable files" section for a list.) These graphic files can be edited with an image editor.

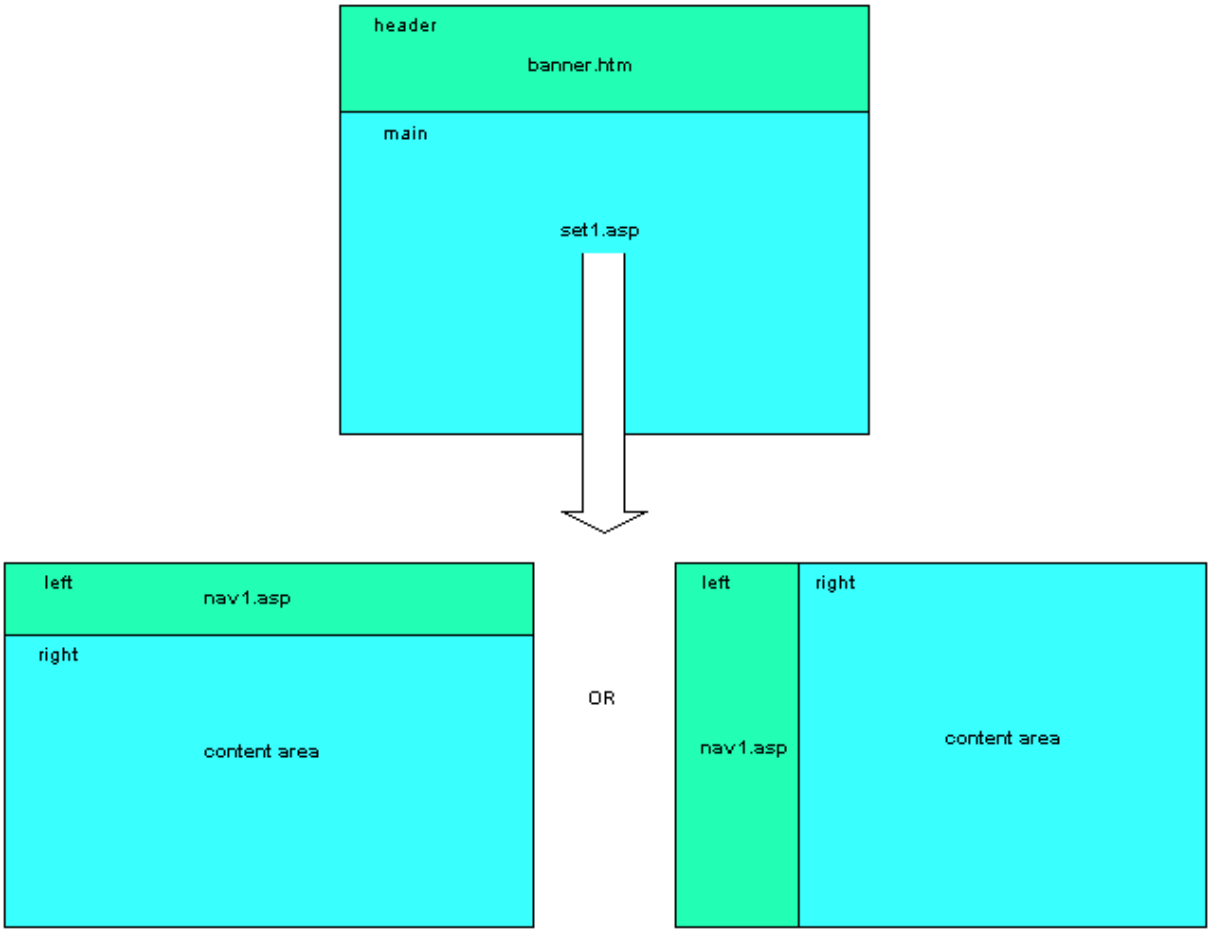
If you change the height or width of the navigation bar buttons you will probably need to adjust the size of the navbar frame. Edit the **appsettings.ini** file to do this, as shown here.

```
[dimensions]

;navigation bar button height, width & orientation (used for button placement)
navbuttonheight=16
navbuttonwidth=122
;navbarorientation=vertical
navbarorientation=horizontal

;navigation bar width (for vertical orientation)
;navigation bar frame width for MSIE
navwidthIE=160

;navigation bar frame width for non-MSIE browsers
navwidthother=168
```

> Note: The web-server application must be re-started for this change to take effect.

2.2.3 Tweaking the frameset

All WBT Manager pages consist of a top and bottom frameset (with frame names **header** and **main**) defined in **default.asp** and a nested frameset containing the navigation bar and content area. The nested frameset is contained in the parent frameset's **main** frame and contains frames named **left** and **right** (defined in **asp\set1.asp**). The **left** frame contains the navigation bar (even though it may not be physically on the left side of the page).



The frame sizes and the navigation bar orientation can be modified by editing the appropriate values in **appsettings.ini**.

```
[dimensions]

;navigation bar button height, width & orientation (used for button placement)
navbuttonheight=16
navbuttonwidth=122
;navbarorientation=vertical
navbarorientation=horizontal

;navigation bar width (for vertical orientation)
;navigation bar frame width for MSIE
navwidthIE=160

;navigation bar frame width for non-MSIE browsers
navwidthother=168
```

Note: The web-server application must be re-started for these changes to take effect.

When using a horizontal navigation bar, the height of the navigation frame (**left**) is controlled by the **navbuttonheight** setting.

You can also modify the **default.asp** and **set1.asp** files to add additional framesets or modify the frameset arrangement further (for instance, to place the navigation bar on the right). If you do this, you must not change the names or hierarchy of the standard WBT Manager framesets. The JavaScript code which causes the navigation bar to update depends on being able to access the navigation frame as **parent.left** and some of the window control code expects the topmost (**top**)

window to be named **root**. Be sure to test carefully after making direct frameset changes.

2.2.4 Changing text styles

Cascading Style Sheets are used to control the text styles throughout the WBT Manager web interface. You can make changes to the general appearance by modifying these style sheets.

The primary WBT Manager style sheets are contained in **<root>/html/ie.css** for Internet Explorer and **<root>/html/ns.css** for Netscape. It is possible to override the primary style sheet name for each browser or for a particular major release of a browser by modifying the **pagesettings.ini** file. You could use a different style sheet for Netscape 4 than for Netscape 6 for example. See the *Application settings files - pagesettings.ini* section for details.

It is also possible to add a secondary style sheet on a page-by-page (and browser version) basis. This also involves changes to **pagesettings.ini** and is discussed in the *Customization - page level changes* section.

The CSS files may be modified using a text editor or style-sheet editor. The standard versions of the files contain numerous class definitions that control different types of text as well as the size of form fields. Some of the more significant definitions are listed below. The simplest way to determine what style class needs to be changed to modify a particular piece of text is to "view source" on the page containing that text and look at the "class" parameter in the enclosing **<p>** or **** tag.

Class	Description
H3	Page header text
H5	Page sub-header text
clsSmInst	Small text used for instructions on forms and menus
clsMedInst	Medium text used for explanatory text on forms and menus
clsLgInst	Large text (often bold-faced) used for text that must stand out from other explanatory text.
clsMenuHdr	Used for headings at the top of menus
clsColumnHdt	Used for table headings
clsFormCap	Used for captions in data entry forms
clsFormData	Used for data entered in forms (there are variations on this for different sizes of text entry fields
clsNavBarPage clsStudentPage clsAdminPage clsCatalogPage clsReportPage clsClassMenuPage clsStudentMenuPage clsCourseMenuPage	BODY classes for different kinds of pages. Used to set page backgrounds and margins.

2.3 Customization - page level changes

This section discusses customizations that affect the appearance of individual pages.

2.3.1 Setting CSS styles at the page level

WBT Manager now allows you to have 2 CSS style sheets per page. The first or primary style sheet is the same for all pages (although it can be varied by browser type and version).

The secondary style sheet is assigned to individual pages through settings in **pagesettings.ini** and can be used to add new style classes to individual pages or to override styles settings in the primary style sheet. Secondary style sheet settings are placed in the **secondary_stylesheet** group within **pagesettings.ini**. Here are some examples.

```
[secondary_stylesheet]
.
.
.
cat1=cat_ie.css
cat1_ie=cat_ie.css
cat1_netscape=cat_ns.css
cat1_netscape_4=cat_ns.css,1
```

The course catalog pages use a number of styles which are used nowhere else. Rather than clutter up the main style sheet files, the catalog styles are included in a secondary style sheet. The secondary style sheet is applied after the primary one. Here is what each line in the **pagesettings.ini** file does to apply the style sheet (also see *Application settings files*).

cat1=cat_ie.css

This line causes WBT Manager to use cat_ie.css for the secondary CSS if no more specific setting exists.

cat1_ie=cat_ie.css

For IE browsers, use cat_ie.css for the secondary CSS.

cat1_netscape=cat_ns.css

For Netscape browsers, use cat_ns.css.

cat1_netscape_4=cat_ns.css,1

For Netscape 4, use cat_ns.css and embed the styles directly into the page instead of using a **<link>** tag (the ,1 at the end of the line specifies this).

2.3.2 Modifying page layouts with XSL

In order to give customers more access to the page layout for customization. WBT Manager now uses XML and XSL style sheets to generate nearly all of the HTML sent to the browser. Customers can change virtually every aspect of the appearance of the web module by changing the XSL files.

> **Important:** The XSL files contain a significant portion of the display logic for WBT Manager and as such are considered part of the executable code. They will be overwritten with every new release. If you customize the XSL files, you will need to review and possibly re-apply the changes with each new release. (It is possible to point pages at non-standard XSL files that will not be overwritten, but you will still need to review the standard XSL files with each new release to see if there are changes that you need to apply to your custom XSL files.)

2.3.2.1 Overview - XML/XSL page generation

The WBT Manager web module is a set of server-based programs that generate HTML pages dynamically on request from users. Past versions of WBT Manager generated the HTML output directly within program code. Some of this code was available for modification as VBScript but most was embedded in compiled ActiveX components.

In order to allow users to customize the system more fully, page generation was changed in version 1.6 to use XSL style sheets to produce the HTML. Program code now creates XML documents containing all of the data to be displayed on the page. As the last step in the page creation process,

an XSL file is used to transform the XML into HTML. The XSL file is available to the user for modification (with the understanding that all XSL files will be overwritten in new releases).

How page generation works

When a user requests a page from WBT Manager (by selecting a link or entering a URL), the program code responsible for generating that page:

Extracts standard text strings from the string table (loaded from **strings.txt** when the web application starts).

Extracts user data from the WBT Manager database.

Combines the user data and standard text into an XML document.

Passes the XML document to the XSL transformation code with an identifier for the page that is being generated.

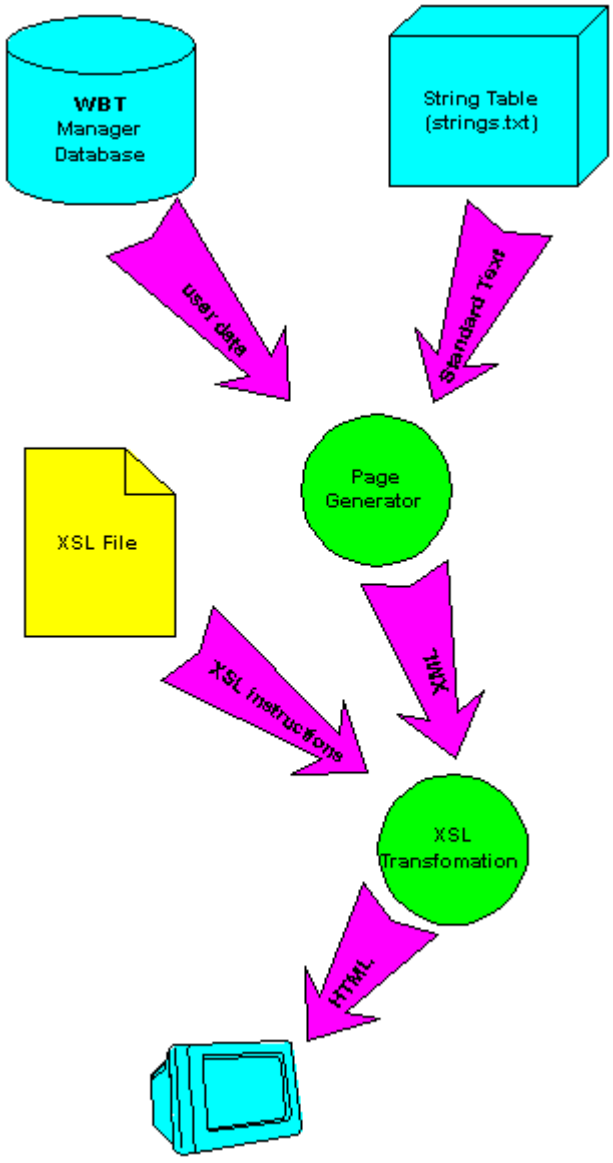
The XSL transformation code, in turn:

Looks up the correct XSL file to use based on the page identifier in **pagesettings.ini** and either

Calls the Microsoft XML Parser to perform the transformation

or

Inserts the path to the XSL file in the XML document and sends it directly to the browser to have the browser perform the transformation.



2.3.2.1.1 WBT Manager XML contents

While the content of each WBT Manager page XML document is different, there are certain parts which are common to all pages. Here is an example of the XML produced by WBT Manager (the description page created by **shdescr.asp**). The parts that are common to all pages are shown in bold text, while the page-dependent portion is shown in normal weight.

```
<wbtmpage>
  <pageinfo>
    <metatag>
      <![CDATA[<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=windows-1252">]]>
    </metatag>
    <paths>
      <protocol>http</protocol>
      <server>localhost</server>
      <port></port>
      <home><![CDATA[/wbtmanager/]]></home>
      <fullhome><![CDATA[http://localhost/wbtmanager/]]></fullhome>
```

```
      <courses><![CDATA[/wbtmanager/courses/]]></courses>
      <asp><![CDATA[/wbtmanager/asp/]]></asp>
      <images><![CDATA[/wbtmanager/images/]]></images>
      <html><![CDATA[/wbtmanager/html/]]></html>
      <jar><![CDATA[/wbtmanager/jar/]]></jar>
    </paths>
  <styles><![CDATA[<link rel="stylesheet" href="/wbtmanager/html/ie.css">]]></styles>
</pageinfo>
<userinfo>
  <name><![CDATA[Student,Demo]]></name>
  <logonid><![CDATA[demo]]></logonid>
   <type>S</type>
  <studcap>Student:</studcap>
  <admincap>Administrator:</admincap>
  <adminlevel>2</adminlevel>
  <orglevel>1</orglevel>
  <org5><![CDATA[Corporation]]></org5>
  <org5cap>Corporation:</org5cap>
  <org4><![CDATA[Region "(L4)"']]></org4>
  <org4cap>Region:</org4cap>
  <org3><![CDATA[Division (L3)]]></org3>
  <org3cap>Division:</org3cap>
 <organization><![CDATA[Demo Organization]]></organization>
  <orgcap>Organization:</orgcap>
  <department><![CDATA[Demo Department]]></department>
  <deptcap>Department:</deptcap>
   <browser>
     <type>ie</type>
     <version>6</version>
   </browser>
</userinfo>
<standardtext>
  <t144000><![CDATA[Description for Course]]></t144000>
  <t144001><![CDATA[Description for Block]]></t144001>
  <t144002><![CDATA[Description for Lesson]]></t144002>
  <t144003><![CDATA[Description for Certificate Program]]></t144003>
  <t144004><![CDATA[Description for Training Category]]></t144004>
  <t144005><![CDATA[Description for Department]]></t144005>
 <t144006><![CDATA[Description for unknown]]></t144006>
  <t144007><![CDATA[Descriptions]]></t144007>
  <t144008><![CDATA[Description for refitem]]></t144008>
  <t144010><![CDATA[No description available in database.]]></t144010>
  <t144011><![CDATA[Unable to find description for]]></t144011>
  <t144012><![CDATA[Close window]]></t144012>
  <t144013><![CDATA[Description for Certificate Program Category]]></t144013>
</standardtext>
 <item>
   <name><![CDATA[AICC/SCORM API sample]]></name>
   <description>
     <![CDATA[This sample course contains lessons demonstrating the use of the AICC/SCORM JavaScript API
       with WBT Manager]]>
   </description>
   <type><![CDATA[course]]></type>
   <notfound>false</notfound>
 </item>
</wbtmpage>
```

The root node of all WBT Manager XML documents is always **<wbtmpage>**. All pages will also contain **<pageinfo>**, **<userinfo>**, and **<standardtext>**.

<pageinfo>

This subtree contains information about the WBT Manager installation. **<metatag>** holds entries to be inserted in the <head> block of the page (entry 10001 in **strings.txt**; this will normally be the page charset metatag). **<paths>** contains partial URLs to various directories within the WBT Manager virtual directory. **<styles>** contains the <link> tags to point to the page's style sheet(s) (or sometimes the styles themselves).

<userinfo>

This subtree contains information about the user such as name, loginid, and department name.

<standardtext>

This subtree contains text strings to be used on the page. For ease of maintenance and translation, all display text within WBT Manager is held in a string table loaded from the file **strings.txt**. This subtree will include entries for all of the standard text to be used in the page. The text elements are named **tnnnnnn** where nnnnnn refers to the text entry number in **strings.txt**.

2.3.2.1.2 Viewing the raw XML

You can see the actual XML generated by a particular WBT Manager page by setting the XML debug flag.

Setting the XML debug flag

Each page in WBT Manager checks a flag before it performs the XSL transformation on the XML data it generates. If the debug flag is set, the XML is sent directly to the browser as a "text/plain" document.

XML debug flags are ASP session variables. The name of the variable is **XSL_*<pagename>*** and should be set to the value "Y" to enable debugging. ***<pagename>*** is the name of the page being requested, without the file extension. You will need to create an ASP page to set the flag into a session variable. For example, the code below was executed before requesting a course description page to generate the sample XML in the previous section.

```
----test.asp----
<%
session("xml_shdescr") = "Y"
%>
---end of test.asp---
```

test.asp was executed **after** logging in as a student. Here are some additional examples. To see the XML generated for the following types of pages, change to code shown above as shown.

the student menu

```
session("xml_menu") = "Y"
```

the password change form

```
session("xml_pw") = "Y"
```

the student edit form

```
session("xml_studnt2") = "Y"
```

2.3.2.2 Changing the XSL files

This section describes how to modify the XSL files that are used to generate the HTML that is displayed to the student.

2.3.2.2.1 XSL basics

An XSL (XML Style Language) file is an XML file containing XSL tags which give directions for transforming XML data into another form. In the case of WBT Manager, the result will be an HTML document for display in a browser. In a WBT Manager XSL file you will find a mixture of XSL tags and HTML tags. The XSL tags instruct the transformation process to "fill in" data from WBT Manager in various places in the HTML code.

This document is not intended to teach you XSL. However, a few of the more common elements are

described below for your convenience.

<xsl:value-of>

The **value-of** tag is the work-horse of the XSL transformation. It says "replace this tag with the contents of the specified XML node". For instance:

```
<xsl:value-of select="/wbtmpage/standardtext/t144010"/>
```

inserts the standard text entry #144010 in the page, and

```
<h3><xsl:value-of select="/wbtmpage/item/name"/></h3>
```

inserts the data in the "item name" node as a heading in the page.

There is an alternative syntax for xsl:value-of that is used to insert data into the attribute values of an HTML tag (xsl:value-of will not work because of the "<" and ">" enclosing the HTML tag). Curly brackets are used to denote the "value-of" operation so:

```
<img src="{/wbtmpage/pageinfo/paths/images}enrollnow.gif">
```

performs the operation you would expect from

```
<img src="<xsl:value-of select="/wbtmpage/pageinfo/paths/images"/>enrollnow.gif">
```

which is not legal.

<xsl:if>

This element allows the XSL page to make decisions. For instance:

```
<xsl:if test="/wbtmpage/course/allowselfenroll='true'">
   <a href="slfenrol.asp?id={wbtmpage/course/id}" target="_self">
      <img src="{/wbtmpage/pageinfo/paths/images}enrollnow.gif" BORDER="0" alt="{/wbtmpage/standardtext/t119006}" />
   </a>
</xsl:if>
```

outputs a button link if the node "allowselfenroll" contains the string "true".

<xsl:choose>

This element supports a "nested if" or "select case" construct for picking one of several options. For instance:

```
<xsl:choose>
   <xsl:when test="/wbtmpage/item/type='course'">
     <xsl:value-of select="/wbtmpage/standardtext/t144000"/>
   </xsl:when>
   <xsl:when test="/wbtmpage/item/type='lesson'">
      <xsl:value-of select="/wbtmpage/standardtext/t144001"/>
   </xsl:when>
   <xsl:otherwise>
      <xsl:value-of select="/wbtmpage/standardtext/t144006"/>
   </xsl:otherwise>
</xsl:choose>
```

outputs a standard text string based on the "item type" node in the XML document.

<xml:apply-templates>, <xml:template>

Applies the same transformations to a selected set of nodes. For instance:

```
<xsl:apply-templates select="/wbtmpage/waitlist/course"/>
```

in the main body of an XSL file causes the following template to be called for each **course** node beneath **waitlist**. Nodes referenced in the template are relative to the current node being transformed.

```
<!-- display a line for each course -->
```

```
<xsl:template match = "course">
    <tr>
        <td>
            <span class="clsCrsMenuTxt"><xsl:value-of select="name"/></span>
        </td>
        <td>
            <span class="clsCrsMenuTxt"><xsl:value-of select="waitingsince"/></span>
        </td>
        <td>
            <span class="clsCrsMenuTxt"><xsl:value-of select="position"/></span>
        </td>
    </tr>
</xsl:template>
```

<xsl:for-each>

Loops through a set of XML nodes and performs the same transformation on each one. For example:

```
<table class="catCrsTable">
    <xsl:for-each select="/wbtmpage/courses/course">
        <tr>
            <td>
                <a class="cat1CrsLink" href="JavaScript:opencentered('cat3.asp?id={id}','crscat',640,440,0);" target="_self">
                    <xsl:value-of select="coursename"/>
                </a>
            </td>
        </tr>
    </xsl:for-each>
</table>
```

creates a link for each **course** below the **courses** node.

2.3.2.2.2 How WBT Manager selects the XSL file

In order to transform the XML generated by program code into HTML to display in the browser, WBT Manager must identify the correct XSL file to perform the transform. In general, the XSL file with the same name as the page being generated will be used. However, there are a few pages that can have multiple appearances. And to support customization, it is possible to override the standard XSL file name for a particular page.

Mapping WBT Manager pages to XSL files

When requesting a transformation, each page's program code supplies an identifier to the transformation code. This identifier will typically be the page name with no extension. Exceptions are noted below.

The page transformation code checks the **pagesettings.ini** file data (loaded into memory at application startup) to see if there is an XSL file declared for that page. The declarations take place in the [xsl_files] section of **pagesettings.ini**. If there are no declarations in **pagesettings.ini**, the filename ***<pageidentifier>*.XSL** is used (e.g., **menu.asp** is identified as "menu" so it would use **menu.xsl**).

XSL file declarations in **pagesettings.ini** can have three different formats.

<pageidentifier>=<xslfile>

This declares the base xsl file for the page identifier. If no other matching declarations exist, this one will be used. For example, the following would cause the student menu page to use **mymenu.xsl** rather than **menu.xsl**:

```
menu=mymenu.xsl
```

<pageidentifier>_<browser>=<xslfile>

This specifies that a particular xsl file should be used for the page for a specific browser. In the following example, **mymenu.xsl** would be used for Internet Explorer and **mymenu_ns.xsl** would be used for Netscape:

```
menu=mymenu.xsl
menu_netscape=mymenu_ns.xsl
```

Note: Internet Explorer is identified as IE).

<pageidentifier>_<browser>_<version>=<xslfile>

This specifies that a particular xsl file should be used for the page for a specific browser version. In the following example, **mymenu.xsl** would be used for Internet Explorer, **mymenu_ns.xsl** would be used for Netscape 6, and **mymenu_ns4.xls** would be used for Netscape 4:

```
menu=mymenu.xsl
menu_netscape=mymenu_ns.xsl
menu_netscape_4=mymenu_ns4.xsl
```

Non-standard page identifiers for XSL file determination

The following pages do not use the page name as the identifier when calling the XSL transformation code.

Non-standard page identifiers	
Page	**page identifier used for XSL**
assign3.asp	assign3 - for single course/student assignment massign3 - for multiple course or student assignment.
assign5.asp	assign5 - for single course/student assignment results massign5 - multiple course or student assignment results
austart.asp	austart - session launch message austart_mode - launch mode menu
crssel1a.asp	crssel1a - single course selection mcrssel1a - multiple course selection
crssel3.asp	crssel3 - single course selection mcrssel3 - multiple course selection
login.asp	login_s - for students login_a - for administrators
login2.asp	login2_s - for students login2_a - for administrators
report.asp	report_s - for students report_a - for administrators
report2.asp	report2_s - for students report2_a - for administrators
shdescr.asp	shdescr_s - for students shdescr_a - for administrators

Non-standard page identifiers (continued)	
Page	**page identifier used for XSL**
crsmenuxml.asp (course lesson menu generator)	crsmenu_sh - for shadow courses crsmenu_hs - for "hard-coded" menu styles **Note:** For other menus styles the identifier is included in the "Menu Generator URL" in the menu style definition. Identifiers for standard styles in version 1.60 are: crsmenu - Standard, 1 column crsmenu2col - Standard, 2 column crsmenutab - Tabbed menu crsmenutree - Tree menu
pw.asp	pw_s - for students pw_a - for administrators
pw2.asp	pw2_s - for students pw2_a - for administrators
result4	result4_lsn - manual data entry for lesson data result4_crs - manual data entry for course data
stusel1a.asp	stusel1a - for single student selection mstusel1a - for multiple student selection
stusel3	stusel3 - for single student selection mstusel3 - for multiple student selection

Error message identifiers

Pages which might generate error messages append an "_e" to their page identifiers for the error message XSL files. (If more that one error message possible the an integer is also appended.) Most of the error message XSL files are redirected to **error_s.xsl** and **error_a.xsl**. Examples are:

crsmenu_e1, crsmenu_e2, crsmenu_e3
login2_a_e1, login2_a_e2, login2_a_e3

2.3.2.2.3 Changing the XSL files

Modifying a WBT Manager XSL file is relatively simple.

1. Identify the XSL file associated with the page to be modified (see the preceding section).
2. Make a copy of the XSL file to a new name that you select.
3. Edit the [xsl_files] section of pagesettings.ini to point the page to the new XSL file.

   Example: **menu=mymenu.xsl**
4. Restart the web-server or execute reload.asp.
5. Open the XSL file and make your modifications

It is strongly recommended that you make a few small changes at a time and test each change before making others. You should also make backup copies of tested changes as you go so that you will not have to start all over if a change gets "out of control."

NOTE: Due to the complicated nature of some of the XSL files. Integrity eLearning cannot provide more than cursory assistance in debugging custom changes. You may be required to restore the standard WBT Manager XSL files in order to obtain customer support. Additional support and assistance in customizing WBT Manager is available on a paid consultancy basis. Contact Integrity eLearning for cost information.

2.4 Customization - building custom menus

Some clients may want to customize the behavior of the Student, Certificate Program, or Course Lesson menus. These menus, like other WBT Manager pages, are based on XSL stylesheet files and can be easily modified by creating customized versions of the XSL files (see "Customization - page level changes").

In case more sophisticated customization is desired (where additional data is needed from the database for instance), we have provided the VBScript source code for the standard menus.

2.4.1 The student menu

The XML document for the standard student menu is generated by **menu.asp** using utility functions to retrieve data from the WBT Manager database and generate XML from them. The file **menu.xsl** is used to transform the XML into HTML for the menu page. These files will be updated with every new release of WBT Manager. Modifying these files directly is not recommended.

Customizing the XSL file.

Most customizations should be possible by changing the **menu.xsl** file.

1. Make a copy of **menu.xsl** to the filename of your choice (**mymenu.xsl** for instance).
2. Add **menu=mymenu.xsl** to the **[xsl_files]** section of **pagesettings.ini**.
3. Re-start the web-server or run **reload.asp**.
4. Modify **mymenu.xsl** as desired.

Customizing the ASP script

If you require more information than is supplied by the standard student menu script, you will need to create a new version of **menu.asp**. *VBScript programming is required.*

1. Make a copy of **menu.asp** to the filename of your choice (**mymenu.asp** for instance).
2. Change the **studentmenu** entry in the **[menus]** section of **appsettings.ini** to point to the new script file (i.e., **studentmenu=mymenu.asp**).
3. Re-start the web-server or run **reload.asp**.
4. Modify **mymenu.asp** as desired.

What the standard version of menu.asp does.

A detailed analysis of the behavior of the **menu.asp** script is beyond the scope of this document. However, here is an outline of what the script does:

verifies that the student is logged on
connects to the database
gets the student's menu preferences
creates standard page XML
collects the standard text needed for the menu

gets HTML1 and HTML2 for the student's department and organization
checks for waiting list and notification messages
gets a list of the certificate programs the student is enrolled in
gets a list of the courses the student is enrolled in
adds everything to the XML document
transforms the XML into HTML using the designated XSL file

2.4.2 Certificate program menus

The XML document for the standard certificate program menu is generated by **clsmenu.asp** using utility functions to retrieve data from the WBT Manager database and generate XML from them. The file **clsmenu.xsl** is used to transform the XML into HTML for the menu page. These files will be updated with every new release of WBT Manager. Modifying these files directly is not recommended.

Customizing the XSL file.

Most customizations should be possible by changing the **clsmenu.xsl** file.

1. Make a copy of **clsmenu.xsl** to the filename of your choice (**mycertmenu.xsl** for instance).
2. Add **clsmenu=mycertmenu.xsl** to the **[xsl_files]** section of **pagesettings.ini**.
3. Re-start the web-server or run **reload.asp**.
4. Modify **mycertmenu.xsl** as desired.

Customizing the ASP script

If you require more information than is supplied by the standard student menu script, you will need to create a new version of **clsmenu.asp**. *VBScript programming is required.*

1. Make a copy of **clsmenu.asp** to the filename of your choice (**mycertmenu.asp** for instance).
2. Change the **classmenu** entry in the **[menus]** section of **appsettings.ini** to point to the new script file (i.e., **classmenu=mycertmenu.asp**).
3. Re-start the web-server or run **reload.asp**.
4. Modify **mycertmenu.asp** as desired.

What the standard version of clsmenu.asp does.

A detailed analysis of the behavior of the **clsmenu.asp** script is beyond the scope of this document. However, here is an outline of what the script does:

verifies that the student is logged on
connects to the database
gets the student's menu preferences
creates standard page XML
collects the standard text needed for the menu
gets descriptive information and HTML1 and HTML2 for the certificate program
gets descriptive information about each course in the program
adds everything to the XML document
transforms the XML into HTML using the designated XSL file

2.4.3 Lesson menus

The XML document for the standard course lesson menu is generated by **crsmenuxml.asp** using utility functions to retrieve data from the WBT Manager database and generate XML from them.

crsmenuxml.asp is embedded in a frameset defined in **crsmenu.asp**. The frameset also includes **crsmenu2.asp** (used to hold state information and the API adapter). **crstabmenu.asp** and **crstreemenu.asp** are still included, but they now call **crsmenu.asp** after setting the XSL parameter on the URL.

Customizing the XSL files.
The following XSL files are included to create different menu styles.

crsmenu.xsl - standard text or graphic menu, 1 column of lesson links
crsmenu_hc.xsl - standard "hard-coded" menu launch page
crsmenu_sh.xsl - standard "shadow" course menu page
crsmenu2col.xsl - standard menu, 2 columns of lesson links
crsmenutab.xsl - "tabbed" menu
crsmenutabns4.xsl - "tabbed" menu, Netscape 4 variation
crsmenutree - "tree" menu
crsmenutreensr - "tree" menu, Netscape 4 variation

Most customizations should be possible by changing one of the above xsl files. If you require more information than is supplied by the standard student menu script, you will need to create a new version of **crsmenuxml.asp**. *VBScript programming is required.*

What the standard version of clsmenu.asp does.
A detailed analysis of the behavior of the **crsmenuxml.asp** script is beyond the scope of this document. However, here is an outline of what the script does:

verifies that the student is logged on
gets the desired XSL file name from the **xsl** parameter of the request (defaults to **crsmenu**).
gets the student's menu preferences
creates standard page XML
collects the standard text needed for the menu
gets the course descriptive information
gets descriptive information about each block or lesson in the course
adds everything to the XML document
- transforms the XML into HTML using the designated XSL file

For more specific details see the following section.

2.4.3.1 Creating a new menu style

Menus are assigned to courses by setting the "menu style" in the course definition. A menu style consists of a style name, a menu generation script, and an XSL file. If you do not need additional data or special processing you can use the standard menu generation scripts **crsmenu.asp, crsmenuxml.asp**, and **crsmenu2.asp** to create the XML document for the menu and just customize an XSL file.

If you need to modify the menu generation scripts:

1. Copy **crsmenu.asp**, **crsmenuxml.asp**, and **crsmenu2.asp** to new files (for example - **mycmenu.asp**, **mycmenuxml.asp**, **mycmenu2.asp**).
2. Edit **mycmenu.asp** to point to **mycmenu2.asp** instead of **crsmenu2.asp.**
3. Edit **mycmenu2.asp** to point to **mycmenuxml.asp** instead of **crsmenuxml.asp.**

4. Make a copy of the existing menu XSL file that you want to use as a starting point (for instance copy **crsmenu.xsl** to **mycmenu.xsl**).
5. Edit **mycmenuxml.asp** to set the xsl file name to **mycmenu** if a value is not sent in the xsl parameter of the URL.
6. Add a new menu style to the database pointing at the new files (see below).
7. Assign the new menu style to a course and modify the files as desired.

If you do not need to modify the menu generation scripts:

1. Make a copy of the existing menu XSL file that you want to use as a starting point (for instance copy **crsmenu.xsl** to **mycmenu.xsl**).
2. Add a new menu style to the database using the new xsl file (see below).
3. Assign the new menu style to a course and modify the xsl file as desired.

Adding a new menu style to the database

Once you have copied the files you want to use as a starting point, you need to create a menu style in the WBT Manager database in order to use it.

Menu styles definitions are stored in the **mnustyle** table in the WBT Manager database. This table contains the list of Menu Styles that can be selected from on the Startup tab in a course's property sheet in the System Administrator module. Selecting a menu style associates the course with a particular menu generation script.

You can add or modify entries in the **mnustyle** table using the Site Configuration Utility. The utility is found in the **\utilities\site_configuration** directory on the WBT Manager installation set. See the "Utilities" section of the main WBT Manager manual for more details.

The **mnustyle table** contains eleven columns, of which only three are fixed in definition in regard to custom course menus. The fields of interest are described in the following table.

mnustyle fields used for custom course menus

Field name	Acceptable values	Description
StyleID (**menustyle**)	integer	The index value of the menu used in the course table to reference the style. **Note:** Values from 0-999 are reserved for use by Integrity eLearning. You may use any other values. The Site Configuration Utility will assign this value automatically.
StyleName (**name**)	char -255 max	The name to display on the Startup tab of the course property sheet
GeneratorURL (**genurl**)	char -255 max	The URL to the script which is to generate the course's menu. Normally this will be a script in the asp directory. Any parameters must be URL-encoded.

The standard menu generator script **crsmenu.asp** supports the following parameters in the GeneratorURL to modify its behavior.

api - set to "1" to include the API adapter applet in the menu frameset.

apidebug - set to "1" to create a frameset in which you can see the API adapter and view status messages as the adapter is used.

apimode - set to "aicc" to indicate that the API adapter should use the AICC data element specification instead of the SCORM definitions (SCORM and AICC definitions have diverged significantly). Defaults to "scorm".
nohacp - set to "1" to tell the lesson launch script to not include HACP launch parameters (use for menus that support only API lessons).

xsl - set to the name of the xsl file to use (defaults to crsmenu, the indicated name is checked against pagesettings.ini for overriding values in the same manner as other pages).

Here are some examples:

```
crsmenu.asp
crsmenu.asp?api=1&apidebug=1
crsmenu.asp?xsl=crsmenutree
crsmenu.asp?xsl=crsmenutree&api=1
crsmenu.asp?api=1&nohacp=1
crsmenu.asp?api=1&apimode=aicc
```

To add your new menu style:

1. Launch the Site Configuration utility and log in as a system-level administrator.
2. Click the **Menu Style** tab and then the **Add** button.
3. Enter values for **StyleName** and **GeneratorURL**.

   You may also enter values for the other menustyle fields if your menu will be using them. The additional information you insert into the **mnustyle** table can be retrieved by the **GetMenuStyle** utility function described in the "Utility Functions" section.

If you are using the standard version of **crsmenu.asp** to generate the menu, the following fields are significant and will change the menu behavior.

Graphical - Checking this box will allow the use of graphic buttons specified in the lesson definitions. Clearing this checkbox restricts the menu to text links (even if graphics are defined in the lesson definition).

HardCoded - Checking this box will cause the "hard-coded" menu launch page to display regardless of the xsl parameter.

2.5 Customization - the course catalog

The course catalog allows a student to see information about all of the courses within the WBT Manager database. We have often been asked for custom features associated with the catalog. Rather than try to build every possible feature into the catalog, we have opened up the source code and provided "hooks" into the catalog display program so that it is easy to modify the appearance of the catalog displays.

Some of the things that are fairly easy to do with the new catalog are:

Display user-defined text fields

Associate buttons with courses (similar to the way an "enroll now" button is associated with some courses.

Note: Some knowledge of VBScript is required to make these kinds of customizations.

2.5.1 Overview - how the catalog display works

The Course Catalog consists of a main catalog display page and three popup windows, one each for Certificate Program descriptions, Course descriptions, and Lesson descriptions. The main display page displays an expandable tree of the contents of the course and certificate program tables.

The popup windows

The popup windows are easiest to customize so we will look at them first. The popup catalog windows are each created by an ASP script file and an XSL file. The files are:

cat2.asp, cat2.xsl - Certificate program information display. Default version displays Title, Description, and a list of the courses contained in the program.

cat3.asp, cat3.xsl - Course information display. Default version displays Title, Description, and a list of the lessons contained in the course. Also displays an "Enroll Now" button for those courses that allow self-enrollment.

cat4.asp, cat4.xsl - Lesson or Block information display. Default version displays Title and Description.

The ASP file retrieves the data and creates an XML document that is, in turn, transformed by using the XSL page (see the *Modifying page layouts with XSL* section). The actual data elements retrieved can be controlled through settings in **pagesettings.ini** (see the *Adding optional data to the popup displays* section).

> **Note:** The above files are considered standard parts of WBT Manager and will be overwritten by future releases. If you modify the XSL files you should take the precautions listed in *Customization - page level changes* and *Customization - building custom menus*.

The main catalog display

In order to avoid retrieving the entire list of courses and programs, the main display page is created incrementally as the student expands portions of the tree. The page uses hidden frame windows and JavaScript code to manage the display and retain data already retrieved. There are three frames and eight files involved in producing the main catalog display.

cat1.asp - Creates a frameset with three frames named "code", "content", and "com". The code and com frames are initially pointed to **blank.asp**. The content frame is pointed to **cat1b.asp**.

cat1a.asp - Generates the JavaScript code that will be used to update the display (ends up in the "code" frame).

cat1a.inc - User customizable file that can change the names of the graphics used in the display and customize the <table> tags used in the display. Documentation about each setting is included in this file.

cat1b.asp - Displays a "Retrieving data" message and initializes the display by loading **cat1a.asp** into the "code" frame.

cat1c.asp - Retrieves data from the database as requested by the JavaScript functions in the "code" window. Returns JavaScript function calls to insert the new data into the existing display tree (targeted to "com" frameset).

cat1c.inc - User customizable file that contains functions called by **cat1c.asp** to format the retrieved data for display. Documentation about each function is included in this file.

cat1d.asp, cat1d.xsl - Generates the main display page including the search form. The tree display is created by JavaScript function calls embedded on this page.

When the display first loads, the following activities occur:

cat1.asp creates the frameset and loads the "content" frame with **cat1b.asp.**

cat1b.asp displays a message and loads the "code" frame with **cat1a.asp**.

cat1a.asp generates JavaScript functions to manage the display.

JavaScript functions in "code" frame call **cat1c.asp** to retrieve the top level of the course and certificate program hierarchy (training category and certificate category). This data is targeted to the "com" frame and consists of JavaScript function calls that add data to the "code" frame's data structure.

JavaScript functions created by **cat1c.asp** request that the display page be redrawn by functions in the "code" frame.

"code" frame functions load **cat1d.asp** into the "content" frame to draw the page. General format is controlled by **cat1d.xsl**. Catalog trees are drawn by JavaScript function calls.

When the student expands a node, the following activities occur:

If the data for the contents of the node do not already exist:

JavaScript functions in the "code" frame call **cat1c.asp** (targeted to the "com" frame) to retrieve the data.

cat1c.asp generates JavaScript function calls to add new data to the "code" frame's data structure.

The final JavaScript function call from **cat1c.asp** directs the "code" frame functions to re-draw the page.

When a search is requested, the following activities occur:

The functions in the "code" frame call **cat1c.asp** (targeted to the "com" frame) with a search request.

cat1c.asp generates JavaScript function calls tTjEclear the "code" frame's data structure and add found data to it.

The final JavaScript function call from **cat1c.asp** directs the "code" frame functions to re-draw the page.

Catalog debug mode

Because the main display in the content frame is written out using JavaScript **document.write** function calls, it is sometimes difficult to determine the cause of problems when changes are made to the include files. You cannot see if any errors occur in retrieving the data because the page that returns the data targets a hidden frame. You also may not be able to see the HTML code that the page has generated because in IE, viewing the source of the content window will not show the actual html code, just the JavaScript calls tTjEgenerate it.

Debug mode causes the hidden frames to be made large enough tTjEyou can view source on their contents. It also adds a debug frame tTjEwill contain a copy of the HTML that the JavaScript functions write to the content frame.

To enable catalog debug mode:

1. Modify **cat1.asp** to set the VBScript variable **bDebug** to **True**.
2. Modify **cat1a.inc** to set the VBScript variable **catDebug** to "true" (string value, not boolean).

   You will now see four frames in the catalog display and you can right-click on any of them to view source.

2.5.2 Adding optional data to the popup displays

Each of the popup windows (certificate, course, lesson displays) retrieves data from the WBT Manager database through one of the utility functions (see the *Utility functions* section).

The utility functions accept a parameter which controls the data elements that are retrieved for the desired object. (See the *ClassDataObject*, *CourseDataObject*, and *LessonDataObject* sections for specifics about the available data elements.) The three ASP scripts involved each check **pagesettings.ini** for values to be used in place of the defaults for this parameter.

cat2_validcertfields - used by cat2.asp to define the certificate program data elements to retrieve and pass in XML to cat2.xsl. Default is:
opdataDefault + opDataDescription + opExcludeCertCourses

cat2_validcoursefields - used by cat2.asp to define the course data elements to retrieve and pass in XML to cat2.xsl for each course in the certificate program. Default is:
opdataDefault

cat3_validcoursefields - used by cat3.asp to define the course data elements to retrieve and pass in XML to cat3.xsl. Default is:
opdataDefault + opdataDescription + opdataActionFlags

cat3_validmenuitemfields - used by cat3.asp to define the lesson data elements to retrieve and pass in XML to cat3.xsl for each lesson or block in the course. Default is:
opdataDefault

cat4_validmenuitemfields - used by cat3.asp to define the lesson data elements to retrieve and pass in XML to cat3.xsl for the lesson or block. Default is:
opdataDefault + opdataDescription

To add additional data elements to one of the displays:

1. Add an entry to **pagesettings.ini**.

   Value is determined by adding desired options together. To add User-defined text elements to the course display for example:

   ```
   [misc]

   ;data that will be available for course display
   ; opdataDescription (0) +
   ; opdataActionFlags (3) +
   ; opdataUserText1to5 (4096)
   cat3_validcoursefields = 4099
   ```

2. Restart the web module.

The XML generated by the ASP page will now contain the new data elements. Modify the XSL page as described in *Customization - page level changes* to display the new data.

2.5.3 Adding optional data to the main display

The main catalog display uses the **cat1c.inc** file to control the data displayed and the format of the data. As each Certificate Program or Course is retrieved from the database, the associated ClassDataObject or CourseDataObject is passed to a formatting routine in **cat1c.inc** that produces the HTML to be written to the page for the object. It is possible to have multiple blocks of text as well as buttons in the main display itself.

The formatting functions are documented (with examples in **cat1c.inc**).

Adding optional data elements

The data elements available to the formatting functions are controlled by entries in **pagesettings.ini**. They are:

cat1c_validcertfields = data that will be available for the certificate program nodes in the catalog tree, default is opdataDefault (0)

cat1c_validcertcrsfields = data that will be available for the certificate program course nodes in the catalog tree, default is opdataDefault (0)

cat1c_validcoursefields = data that will be available for the course nodes in the catalog tree, default is opdataDefault (0)

If you want to include the course description in addition to the name in the main display, you can change **cat1c_validcoursefields** to 1 (opdataDefault + opdataDescription).

Changing the format functions

Now that the course description is being retrieved, it must be added to the display in the formatting function. The function for course entries is called FormatCourseEntry. The default version looks like this:

```
Sub FormatCourseEntry( ByRef dbConn, ByRef bAlwaysExpanded, _
                       ByRef objCrs, ByRef strLinkText, ByRef aHTMLBlock )
    Dim strTmp

    'main node text

    'lesson details link
  strTmp = "cat3.asp?id=" & CStr(objCrs.ID)
  strTmp = "JavaScript:opencentered('" & strTmp & "', 'crscat', " & CStr(640) & ", " & CStr(440) & ", 0)"
  strLinkText = strLinkText & "<A CLASS=""cat1CrsLink"" HREF=""" & strTmp & """ target=""_self"">" & _
                       Server.HTMLEncode( objCrs.CourseName ) & "</A>"

'   redim aHTMLBlock(2)
'   aHTMLBlock(0) = "<p class=""cat1CrsData"">Vendor Catalog ID: " & objCrs.VName & "</p>"
'   aHTMLBlock(1) = "<p class=""cat1CrsData"">" & objCrs.Description & "</p>"
'
'   self-enroll button.
'   If objCrs.AllowSelfEnroll Then
'
'           'enrollnow graphic
'           strTmp = "slfenrol.asp?id=" & CStr(objCrs.ID) & "&nm=" & Server.URLEncode(objCrs.CourseName)
'           strTmp  = "JavaScript:opencentered('" & strTmp & "', 'enroll', " & CStr(540) & ", " & CStr(200) & ", 0)"
'
'           '117008=Enroll Now
'           aHTMLBlock(2) = "<p class=""cat1CrsData"">" & _
'                   "<A HREF=""" & strTmp & """ target=""_self"">" & _
'                   "<IMG BORDER=""0"" ALT=" & StringItems.LookupValue(117008) & _
'                   " SRC=""" & Application("strWBTImagePath") & "enrollnow.gif" & """></A></p>"
'
'   Else
'           aHTMLBlock(2) = Null
'   End If

    bAlwaysExpanded = True

End Sub
```

As you can see, there is already code that adds the provider's name for the course and the description. Since we only want the description we need to un-comment some of the code and modify it slightly. Change the portion of the code shown in bold above to read as follows:

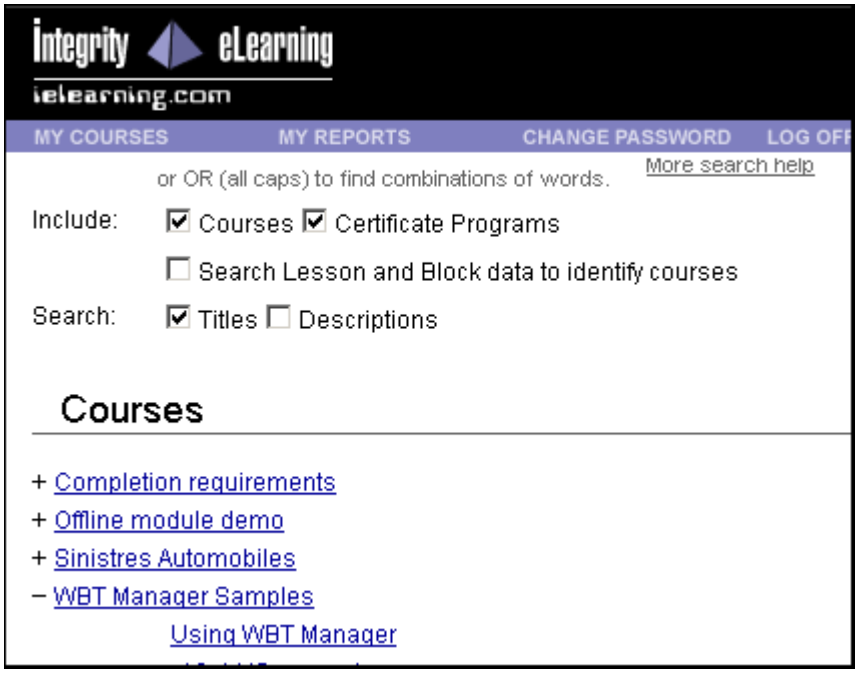
```
redim aHTMLBlock(0)
'   aHTMLBlock(0) = "<p class=""cat1CrsData"">Vendor Catalog ID: " & objCrs.VName & "</p>"
aHTMLBlock(0) = "<p class=""cat1CrsData"">" & objCrs.Description & "</p>"
```

The array aHTMLBlock can be used to add any number of additional "chunks" of HTML code, but it must be dimensioned properly. If you make it too large you will get blank sections. If you make it too small, you will get VBScript run-time errors.

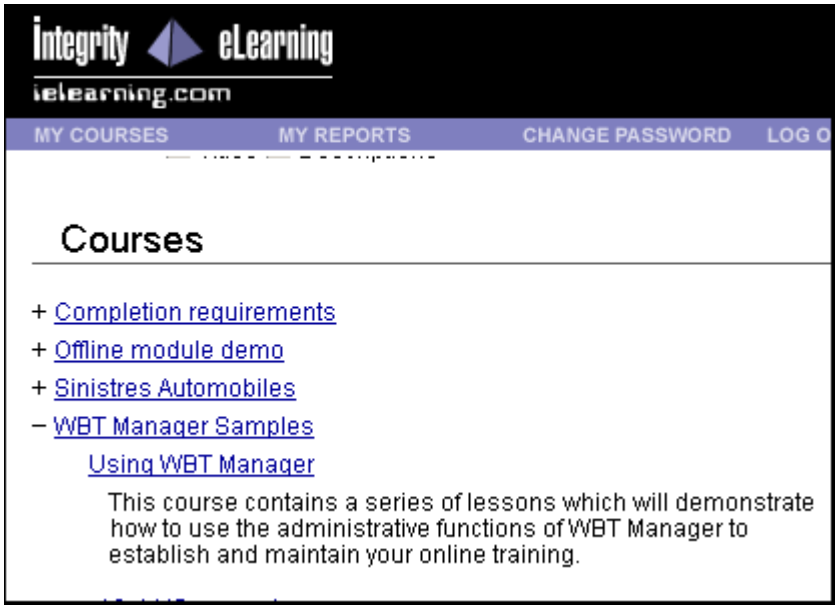
The redim statement sets the upper subscript bounds, not the number of elements. The array starts with element # 0, so redim aHTMLBlock(0) sets the array to have a single element.

The following screenshots illustrate the difference that the above changes make:

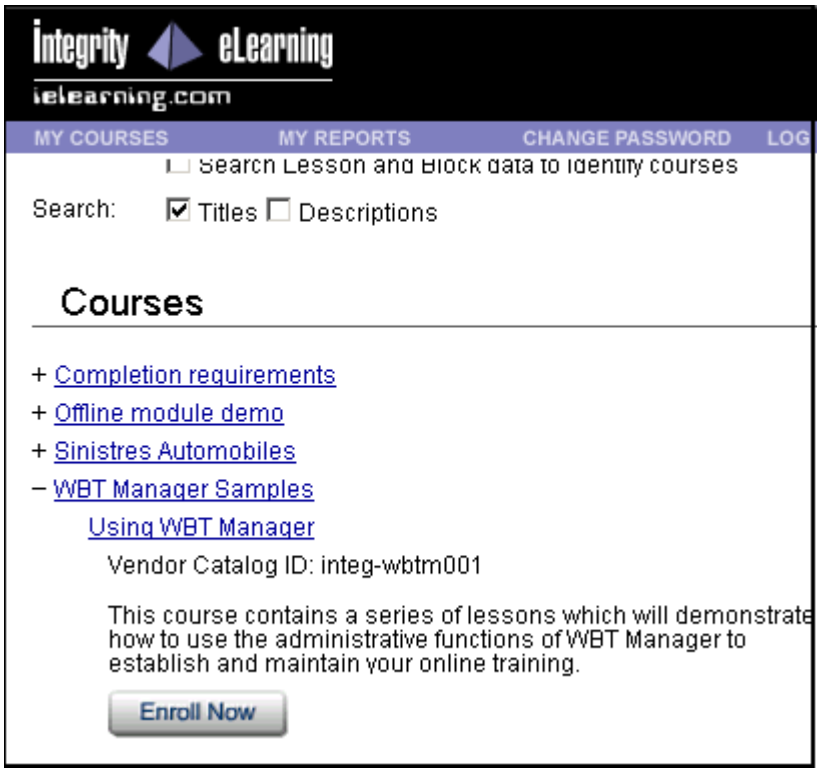
Default display:



As changed above:



Full sample code from cat1c.inc (cat1c_validcoursefields = 3)



2.6 Application settings files

There are two text files which contain settings used by the web module to control the behavior of the user interface. The are **appsettings.ini** and **pagesettings.ini**. The following sections explain how to use these files.

2.6.1 appsettings.ini

Settings which affect the entire web interface are contained in **appsettings.ini**. These values were in the file settings.asp in previous versions of WBT Manager, but have been moved into a more convenient Windows INI format starting with release 1.6. The values in this file are copied into ASP Application or Session variables.

Note: You must restart the web server or run **reload.asp** for these settings to take effect.

[database]

The [database] group identifies the database to be used and provides the necessary connection information. It contains:

connectionstring - contains the connection string to be used when connecting. This string includes placeholders for the user id and password (see example below).

userid - the database user id to be used

password - the database password to be used

encryptpassword - flag to tell the web module whether to encrypt the database password in this file.

passwordisencrypted - flag indicating whether the password is already encrypted in this file

Example:

```
[database]
connectionstring=dsn=WBT Manager Database;uid=%%uid%%;pwd=%%pwd%%
userid=wbtmanuser
password=wbtmanuser
encryptpassword=no
passwordisencrypted=no
```

The strings **%%uid%%** and **%%pwd%%** are used as placeholders in the connectionstring and will be replaced by the values assigned to **userid** and **password** respectively.

To request password encryption set the value of **encryptpassword** to "yes".

To change an encrypted password, set the new password as the value of **password** in clear text and set **passwordisencrypted** to "no". The next time that the web-server is restarted (or that reload.asp is called), WBT Manager will encrypt the password and change **passwordisencrypted** to "yes". DO NOT CHANGE **passwordisencrypted** unless you are resetting the password as described above.

[interfacelogging]

The [interfacelogging] group controls which students and lessons generate log entries for the AICC interface. Students are identified by providing a comma-separated list of student login IDs (ie: "jsmith,htruman") or an asterisk ("*") for all students. Lessons are identified by their internal WBT Manager database IDs (ie: 1001,1005). You can get these IDs by viewing the source code for a course menu.

Note: Log files will need to be purged manually and could become very large. Use only for debugging.

Example:

```
[interfacelogging]
students=demo
lessons=
```

[pagebackground]

The [pagebackground] group is provided for compatibility with older WBT Manager installations and is used to specify a background color (preceded by #) or an image file (which must be in the **images** directory). Backgrounds should be set in the stylesheets using the <body> class names where possible.

Example:

```
[pagebackground]
defaultpage=#FFFFFF
classmenu=#FFFFFF
coursemenu=#FFFFFF
reports=#FFFFFF
navbar=nav_bg.jpg
catalog=#FFFFFF
```

[dimensions]

The [dimensions] group contains the dimensions of various WBT Manager UI components and is used to control the frame sizes and navigation bar button spacing. It contains the following, with all dimensions expressed in pixels:

navbuttonheight - height of the navigation bar buttons

navbuttonwidth - width of the navigation bar buttons

navbarorientation - orientation of the navigation bar frame. Allowed values are "horizontal" or "vertical."

navwidthIE - the width of frame to use with a vertical navigation bar under Internet Explorer

navwidthother - the width of frame to use with a vertical navigation bar under Netscape

bannerheight - the height of the top banner frame

helpwidth, **helpheight** - the size of the popup window to be displayed when the user clicks the "help" button.

Example:

```
[dimensions]

;navigation bar button height, width & orientation (used for button placement)
navbuttonheight=16
navbuttonwidth=122
navbarorientation=horizontal

;navigation bar frame width for MSIE
navwidthIE=160

;navigation bar frame width for non-MSIE browsers
navwidthother=168

;top banner height
bannerheight=56

;help window size
helpwidth=740
helpheight=540
```

[flags]

The [flags] group contains application various application control flags.

disablemaint - flag to disable the student maintenance script that runs when a student logs on. Set to "1" to disable.

nourlencodeparams - flag to cause the AICC interface to not URL encode the AICC parameters when launching a course. Set to "1" to disable encoding.

- **Note:** This option was added to get around a problem with a particular vendor's content tool. The tool has since been corrected. Use of this setting could cause AICC-compliant lessons to cease working.

navbuttonmask - flag to remove specified buttons from the navigation bar. Flag is the sum of the following values:

1 - Help Button (student)
2 - Help Button (administrator)
128 - Reports Button (student)
256 - Reports Button (administrator)
512 - Change PW Button (student)
1024 - Change PW Button (administrator)
8192 - Catalog Button (student)

autoaddscormobjectives - flag indicating whether objective definitions are to be automatically created when a SCORM lesson sends status or score for them (SCORM course interchange files do not include objective definitions).

restrictcatalog - flag indicating whether the course catalog is to show all courses in the system or only those courses that a student's local administrators would be able to assign to them. Used to allow for courses that only specified students can see.

studentselectordata, **courseselectordata**, **certselectordata**, **adminselectordata** - flags that control the data elements displayed on student, course, certificate program, and administrator selection forms when the "show details" option is selected. Values are the sum of constants from wbtconst.inc.

Example:

```
[flags]

;disable student  account maintenance function. Set to 1 to disable maintenance.
;       NOTE:class certifications will not expire if this is set.
disablemaint=0

;turn off URLEncoding of AICC_URL and AICC_SID parameters for lesson launch (1=off).
nourlencodeparams=0

;navbuttonmask removes specific buttons from the navigation bar. buttons are specified by adding
; the following codes together
;1 - Help Button (student)
;2 - Help Button (administrator)
;128 - Reports Button (student)
;256 - Reports Button (administrator)
;512 - Change PW Button (student)
;1024 - Change PW Button (administrator)
;8192 - Catalog Button (student)
navbuttonmask=0

;autoaddscormobjectives controls whether objective records are created automatically and assigned
;to lessons when an unrecognized objective id is sent by a lesson.
; NOTE: auto add of objectives is currently supported only for lessons using the API in SCORM mode.
; setting this value to anything other than "Y" will disable auto add of objectives. Defaults to "N" if missing.
autoaddscormobjectives = N

;restrictcatalog controls whether the course catalog displays all courses in the system or only courses that
;are either unrestricted or assigned to the student's parent hierarchy elements. Default is "N".
;Set to "Y" to restrict display.
restrictcatalog = N

;value to control what student data elements display when user selects the "show details" option
;on the student search form. Value is the sum of the desired "opdataStudent..." values found
;in wbtconst.inc
studentselectordata = 114694

;value to control what course data elements display when user selects the "show details" option
;on the course search form. Value is the sum of the desired "opdataCourse..." values found
;in wbtconst.inc
courseselectordata = 131075

;value to control what certificate program data elements display when user selects the "show details" option
;on the certificate search form. Value is the sum of the desired "opdataCert..." values found
;in wbtconst.inc
certselectordata = 23

;value to control what administrator data elements display when user selects the "show details" option
;on the administrator search form. Value is the sum of the desired "opdataAdmin..." values found
;in wbtconst.inc
adminselectordata = 3086
```

[menus]

The [menus] group allows custom menu scripts to be called for student and certificate program menus. Scripts must be in the **ASP** directory.

studentmenu - name of the script to call in place of menu.asp for the student menu

classmenu - name of the script to call in place of clsmenu.asp for the certificate program menus

```
[menus]
studentmenu=menu.asp
classmenu=clsmenu.asp
```

[sqlfunctions]

The [sqlfunctions] group contains DBMS-specific functions needed to convert data within direct SQL statements.

convertdate - the DBMS specific SQL function that will convert a date in yy-mm-dd hh:mm:ss format into the correct internal DBMS format. Defaults to the appropriate function for Oracle, SQLServer, or Access databases. An example of how it would be used in a SQL statement (for Oracle) is:

```
insert into tablexxx (datecolumn) values (TO_DATE('2002-07-01 14:30:30', 'YYYY-MM-DD HH24:MI:SS'))
```

tolowercase - the DBMS specific SQL function to convert a string to lower case. Defaults to the appropriate function for Oracle, SQLServer, or Access databases. An example of how it would be used in a SQL statement (for Oracle) is:

```
select * from tablexxx where NLS_LOWER(name) like 'john%'
```

Example:

```
[sqlfunctions]

;function used in sql statements for converting a date string in yyyy-mm-dd hh:mm:ss format to the DBMS
;       date format provided by default for ACCESS, SQL Server, or Oracle. Use %%Date%% for the date string.
;       For example the Oracle function is: TO_DATE('%%DATE%%','YYYY-MM-DD HH24:MI:SS'),
;       SQL Server is: CONVERT(datetime,'%%DATE%%'), ACCESS is: #%%DATE%%#
convertdate=

;function used in sql statements for converting a string to lower case (to eliminate case sensitivity in some
;       where clauses) provided by default for ACCESS, SQL Server, or Oracle. Use %%STRING%% for the string
;       to be converted. For example the oracle function is: NLS_LOWER(%%STRING%%),
;       SQL Server is: LOWER(%%STRING%%), ACCESS is: LCASE(%%STRING%%)s
tolowercase=
```

[xml]

The [xml] group contains settings which control the behavior of the xml/xsl transformations. There are also entries in pagesettings.ini which affect the xml/xsl transformations. See the *pagesettings.ini* section below for more information.

includestandardtext - flag to tell the web module that it should include standard text from the string table in the XML for the each page. If you replace all of the text entries in the XSL files with hard-coded text (a utility is under consideration for this), then setting this flag to "no" can provide you with a small performance boost on the server.

encoding - string value to be used in the xml directive for the character encoding when sending XML directly to the browser without performing the transformation on the server. This setting is ignored if the page is not set to do the XSL transform on the client in pagesettings.ini for the client's browser ([xsl_files] group).

codepage - the character set codepage that will be used when sending XML directly to the browser without performing the transformation on the server. This value should specify the same character encoding as the **encoding** keyword. This setting is ignored if the page is not set to do the XSL transform on the client in pagesettings.ini for the client's browser ([xsl_files] group).

The **encoding** and **codepage** settings are needed for use of WBT Manager with languages containing special characters (like accents) if a page is set to perform the XSL transformation on the client system in pagesettings.ini.

Example:

```
[xml]

;The includestandardtext switch controls whether xml data generated for a page contains the page's standard text
;       from strings.txt. A minor performance improvement can be achieved by embedding text directly into the
;       xsl files and setting this value to "no"
includestandardtext=yes

;The encoding keyword sets the character encoding used in the XML document when sending to the browser for
;       XSL transformation (xslonclient settings in pagesettings.ini).
;       Defaults to "utf-8" if not set here. default results in an xml directive of:
```

```
;               <?xml version="1.0" encoding="utf-8"?>
;       This setting is ignored when XSL transformation takes place on the client.
;encoding=utf-8
;encoding=iso-8859-1
encoding=windows-1252

;The codepage keyword sets the codepage to be used when sending XML to the browser for
;       XSL transformation (xslonclient settings in pagesettings.ini). In order to correctly send non-ascii
;       characters this needs to be set to the codepage matching the encoding keyword above.
;       Defaults to the current codepage if not set here.
;       This setting is ignored when XSL transformation takes place on the client.
;                       utf-8 = 65001   (may not work on all NT4 systems)
;                       iso-8859-1 = 28591
;                       windows-1252 = 1252
;codepage=65001
;codepage=28591
codepage=1252

;*** UTF-8 is recommended for Windows 2000 and XP.
;       If you are unable to find a combination of encoding and codepage that
;       works on your server with your client browsers then you should comment out
;       any entries in the [xslonclient] group in pagesettings.ini.
```

[login]

The [login] group contains values that modify the way that WBT Manager authenticates students and creates their login sessions.. See Alternative User Authentication Algorithms for more information.

usealternateauthentication - controls whether the user is validated against the admin and student tables or through a user-defined algorithm.Set to "Y" to call user algorithm contained in **login2.inc**

altauthenticationmode - controls how the user is authenticated on entering the site or re-authenticated on session time-out when usealternateauthentication is set to "Y". Options are:

"0" for normal mode in which the login form is displayed

"1" for silent mode in which the object specified by altauthenticationobject is used to authenticate the user without user interaction.

altauthenticationobject - contains the program id of the object to be used for authentication when usealternateauthentication = "Y" and altauthenticationmode = "1" .

Example:

```
;The [login] group contains values that modify the way that WBT Manager authenticates students ;and creates their login sessions.
[login]

;The usealternateauthentication keyword controls whether the user is validated against the admin
;  and student tables or through a user-defined algorithm.Set to "Y" to call user algorithm.
usealternateauthentication=N

;The altauthenticationmode keyword controls logged on session behavior when usealternateauthentication = Y.
;       0 = Normal. Users enter their login information on the WBT Manager login form. The alternate authentication
;                       function in login2.inc is used to authenticate them. When sessions time out
;                       the users are redirected to the login form to re-enter their credentials.
;       1 = Silent, the login form page automatically redirects to the login processing page without any message. The
;                       AuthenticateUser() function found in the object specified by altauthenticationobject is used to
;                       validate the user. When sessions time out the users are re-authenticated using AuthenticateUser()
;       If altauthenticationobject is blank, or the specified object cannot be created, or the AuthenticateUser function
;       cannot be called without error then an error message will appear for the user
altauthenticationmode=0

;The altauthenticationobject keyword specifies the COM component that contains the AuthenticateUser function to be used
;by alternate authentication in mode 1
;altauthenticationobject=wbtmanauth.authenticator
altauthenticationobject=
```

[event_queue]
The [event_queue] group contains settings which control the behavior system event queue processor. See

iltstatusentry - Controls whether an instructor can enter the lesson status on the instructor's attendance form (by default status is always set to "c").

iltdefstatus - Status value that is displayed as the default value on the instructor's attendance form when iltstatusentry = "Y".

iltscoreentry - Controls whether an instructor can enter the lesson score on the instructor's attendance form.

iltdefscore - Score that is displayed as the default value on the instructor's attendance form when **iltscoreentry** = "Y".

Example:

```
;The [ilt] section contains settings that control the behavior of the Instructor-Led Training module.
[ilt]

;The iltstatusentry keyword controls whether the instructor can enter the lesson status at the time they
;take attendance. Set to "Y" to allow instructor entry. NOTE: some ILT lesson types may override this
;setting because lesson status is not tied to attendance (proctored exams for instance).
iltstatusentry=N

;The iltdefstatus keywork sets the default value for status entry through the attendance form. The instructor
;can change this value on the form. Valid values are "I", "C", "P", "F".
iltdefstatus = P

;The iltscoreentry keyword controls whether the instructor can enter the lesson score at the time they
;take attendance. Set to "Y" to allow instructor entry. NOTE: some ILT lesson types may override this
;setting because lesson score is not tied to attendance (proctored exams for instance).
iltscoreentry=N

;The iltdefscore keywork sets the default value for score entry through the attendance form. The instructor
;can change this value on the form. Valid values are blank or any non-negative numeric value.
iltdefscore=100
```

[misc]

The [misc] group contains miscellaneous settings.

session_codepage - specifies the codepage to use when sending and receiving data from the client's browser.

dtp_sizex, **dtp_sizey** - Specifies the size of the popup window that contains the date picker used to select dates.

dtp_datedisplayformat, **dtp_datedisplaytemplate**, **dtp_datentdisplaytemplate** - Specify the format used to display dates in the date picker.

Example:

```
[misc]
;The [misc] section contains miscellaneous settings.

;The session_codepage keyword sets the codepage to be used when receiving data from the browser
;and sending data to the browser. The ASP session.codepage property will be set to this value for
;each user's session.
;
; Note:         This is not the same as the codepage parameter in the [xsl] group.
;     The HTML charset metatag will have to be modified in strings.txt (entry 10001) to match this value.
;
;               utf-8 = 65001   (may not work on all NT4 systems)
;               iso-8859-1 = 28591
;               windows-1252 = 1252
;
; leave blank to use the default codepage
;
;session_codepage=65001
```

```
;session_codepage=28591
;session_codepage=1252
session_codepage=

;***date picker settings***
;The dtp_sizex, and dtp_sizey keywords control the display size of the popup date picker
;which allows the user to pick a date and time from a calendar.
;Values are in pixels
dtp_sizex = 240
dtp_sizey = 310

;The dtp_datedisplayformat, dtp_datedisplaytemplate keywords control the
;display format of dates that are set by the date picker.
;Note: these dates are set by javascript code in the browser and the format may vary from dates
;displayed on reports and on pages where the user does not enter dates.

;The dtp_datedisplayformat keyword controls the type of format that dates have when returned by the
;date picker. Valid values:
;       0 = client system default date format (uses JavaScript Date.toLocaleString() function )
;       1 = custom format based on dtp_datedisplaytemplate \
;Default value is 0
;dtp_datedisplayformat = 0

;The dtp_datedisplaytemplate keyword contains a template that is filled in at run-time to build a
;display string for a date. Placeholders in the template string are replaced with the various
;date-time components to build the display date. Valid placeholders are:
;      %y =  year as yyyy
;      %m = 2-digit month number (01-12)
;      %M = month name
;      %d = 2-digit day number (01-31)
;      %D = day of week name
;      %h = hour in 24 hour format (00-23)
;      %H = hour in 12 hour format (0-12), should be used with %a
;      %n = 2-digit minutes (00-59)
;      %s = 2-digit seconds(00-59)
;      %a = AM/PM indicator, should only be used with %H for hours
;The template only applies if dtp_datedisplayformat = 1
;Default value is "%D, %M %d, %y %H:%n %a"
;dtp_datedisplaytemplate =%D, %M %d, %y %H:%n %a

;The dtp_datentdisplaytemplate keyword contains a template that is filled in at run-time to build a
;date with no time included. The same placeholders are supported as in dtp_datedisplaytemplate
;(obviously you should not include values associeated with time).
;Default value is "%D, %M %d, %y"
;dtp_datentisplaytemplate =%D, %M %d, %y
```

2.6.2 pagesettings.ini

Settings that affect individual pages or specific browser versions are included in **pagesettings.ini**. These settings are loaded into a lookup table in memory when the web application starts (or reload.asp is run). The format is that of a Windows INI file for simplicity.

> **Note:** You must restart the web server or run **reload.asp** for these settings to take effect.

[primary_stylesheet]

This section controls the primary stylesheet. The primary stylesheet cannot be set on an individual basis but can be set for each browser type and major version.

Formats:

<browser>=<stylesheet>
<browser>_<version>=<stylesheet>

Notes:

The lookup routine will go from the most specific to most general name.
Adding ",1" after the stylesheet name indicates that the sheet should be embedded instead of linked.

All stylesheets are expected to be in the **html** directory.

Example:

```
[primary_stylesheet]
default=ie.css                  if no other entries, use ie.css
ie=ie.css                       use ie.css for IE
netscape=ns.css                 use ns.css for netscape
netscape_4=ns.css,1             use ns.css for Netscape 4 but write the styles directly to the page
```

[secondary_stylesheet]

This section controls the secondary stylesheets. Secondary stylesheets are used to add styles used by specific pages and to override styles in the primary stylesheet for specific pages. They can be set on an individual page basis and can be further differentiated by browser type and major version.

Formats:

<page>=<stylesheet>
<page>_<browser>=<stylesheet>
<page>_<browser>_<version>=<stylesheet>

Notes:

The lookup routine will go from the most specific to most general name.
Adding ",1" after the stylesheet name indicates that the sheet should be embedded instead of linked.
All stylesheets are expected to be in the **html** directory.

Example:

```
[secondary_stylesheet]
cat1=cat_ie.css
cat1_ie=cat_ie.css
cat1_netscape=cat_ns.css
cat1_netscape_4=cat_ns.css,1

cat2=cat_ie.css
cat2_ie=cat_ie.css
cat2_netscape=cat_ns.css
cat2_netscape_4=cat_ns.css,1
```

[page_timeouts]

This group is used to override the re-direction timeouts on various pages. The pages affected typically have the message "Click here if not taken to the <xxx> automatically." The default timeout is usually 5 seconds.

Format: **<page>=<delay in seconds>**

Note:

Some pages will not work properly with a timeout of 0 (it may vary depending on client's connection speed). Test carefully after changes.

Example:

```
[page_timeouts]
login2=1                go to student menu faster
austart=1               launch lesson faster
```

[xsl_files]

This section specifies the XSL file to use to produce a page. By default the system uses the page name as the XSL filename. This section can override the default on an individual page basis and can be further differentiated by browser type and major version.

Formats:

<page>=<xslfile>
<page>_<browser>=<xslfile>
<page>_<browser>_<version>=<xslfile>

Notes:

All xsl files are expected to be in the **html/xsl** directory.
Adding ",1" after the xsl filename indicates that transformation should occur on the client (may be overridden in some cases). The browser must support the W3C XSL 1.0 specification to allow this (or the XSL sheet must be customized for the specific browser).

Example:

```
[xsl_files]
;     dynamic course menu XSL files for netscape
crsmenutree_netscape_4=crsmenutreens4.xsl
crsmenutab_netscape_4=crsmenutabns4.xsl

;     course menu error messages
crsmenu_e1=error_s.xsl
crsmenu_e2=error_s.xsl
crsmenu_e3=error_s.xsl
```

[misc]

This section is used to contain values that various pages can retrieve by name. There is no specific name format except that names must be completely unique throughout this file. (In case of duplicates, the first value is used.)

Example:

```
[misc]
;     catalog page data selection control
;     see documentation for GetClassCoursesEx, GetStudentCoursesEx,
;                 and GetCourseStructure for information about appropriate values.
;
;     Note that no values are actually set here because the pages contain
;     default values. Values need to be set to obtain additional data when
;     customizing the pages.
;
;cat1c_validcertfields = data that will be available for the certificate program
;     nodes in the catalog tree, default is opdataDefault (0)
;cat1c_validcertcrsfields = data that will be available for the certificate program
;     course nodes in the catalog tree, default is opdataDefault (0)
;cat1c_validcoursefields = data that will be available for the course nodes
;     in the catalog tree, default is opdataDefault (0)
```

2.7 Utility functions

In order to insulate user customizations from the details of (and possible changes to) the WBT Manager database, a number of utility functions exist. These functions are members of a utility object that can be instantiated in VBScript with the following statement:

set objUtils = Server.CreateObject(Application("DLL-D") & ".webutl2")

wbtman4d.webutl2 is the actual program id (progid) of the utility object used by IIS to load the appropriate DLL and create the object. This progid will typically change between major releases of WBT Manager to allow for parallel operation of different versions. Because of this the name of the actual DLL has been moved to an application variable so that changes will not be required in future releases.

2.7.1 Login functions

There are three functions related to logging in, as discussed in the following sections.

2.7.1.1 UserLogon

UserLogon(ByRef objStringTable As Object, ByVal strUserID, ByVal strPassword, ByVal strType, ByRef strMsg)

UserLogOn validates a user against the WBT Manager database and creates the user's session variables if successful. It can be used to create a login mechanism external to the standard WBT Manager login page (see **slogin.asp** for an example).

Return Value:

A numeric status code with one of the following values.

0 - Login successful
1 - Invalid userid or password
2 - User's account is disabled
3 - User's department is disabled
4 - User's organization is disabled
5 - User's division is disabled
6 - User's region is disabled
7 - User's corporation is disabled
99 - unexpected error (probably a database error).

Parameters:

objStringTable - A reference to the global string table containing all of WBT Manager's displayed text. This object can be referenced from any ASP page through the global object name StringItems.

strUserID - The user's login id.

strPassword - The user's login password.

strType - The user type. "S" means student and "A" means administrator.

strMsg – A variable to contain an error message if an unexpected error occurs.

Usage Example:

```
nResult = objUtils.UserLogon( StringItems, strLogID, strPW, strType, strTemp )
```

Note: In order to create a valid WBT Manager session, this function must be called from a script contained in the WBT Manager virtual directory on the web server. The ASP session cookie generated during this process must be returned to the browser.

2.7.1.2 UserLogonNoPW

UserLogonNoPW(ByRef StringObjs As Object, ByVal strUserID, ByVal strType, ByRef strMsg)

UserLogOnNoPW validates a user against the WBT Manager database by LoginID only and creates the user's session variables if successful. It can be used to create a login mechanism external to the standard WBT Manager login page. Intended to allow login to be tied to the student's network login.

Return Value: Same as for **UserLogon**

Parameters:

objStringTable - A reference to the global string table containing all of WBT Manager's

displayed text. This object can be referenced from any ASP page through the global object name StringItems.

strUserID - The user's login id.

strType - The user type. "S" means student and "A" means administrator.

strMsg – A variable to contain an error message if an unexpected error occurs.

Usage Example:

```
nResult = objUtils.UserLogonNoPW( StringItems, strLogID, strType, strTemp )
```

Note: In order to create a valid WBT Manager session, this function must be called from a script contained in the WBT Manager virtual directory on the web server. The ASP session cookie generated during this process must be returned to the browser.

2.7.1.3 CheckLoginStatus

CheckLoginStatus(ByRef objStringTable As Object, ByVal strType)

CheckLoginStatus is called to verify that the user is logged onto WBT Manager. If the user is not logged on, this function will output a page informing the user of the situation and including a link to the WBT Manager login page.

Return value:

True if the user is logged onto WBT Manager
False if the user is not logged on.

If the user is not logged on a page is automatically generated warning of this and the user's browser is redirected to the logon form (**login.asp**) unless the system is configured to support "silent" login (see Alternative User Authentication).

Parameters:

objStringTable - A reference to the global string table containing all of WBT Manager's displayed text. This object can be referenced from any ASP page through the global object name StringItems.

strType - The user type. "S" means student and "A" means administrator.

Usage Example:

```
If Not objUtils.CheckLoginStatus(StringItems,"S") Then
    Response.End 'stop right here.
End If
```

2.7.2 OpenDBConnection

OpenDBConnection()

OpenDBConnection creates an ADO connection object and connects to the WBT Manager database. The connection parameters are retrieved from Application variables set by the initialization functions in global.asa.

Return value:

An ADO connection object connected to the WBT Manager database. You should close this object and set its variable to nothing when you are done making database access calls.

Parameters: none

Usage Example:

```
Set dbConn=objUtils.OpenDBConnection()
```

2.7.3 Menu functions

There are several menu-related functions, as discussed in the following sections.

2.7.3.1 GetMenuStyle

GetMenuStyle(ByRef dbConn, ByVal nStyle, ByRef strErrMsg)

GetMenuStyle retrieves data from the **mnustyle** table for the specified menu style. The menu style id for a course can be found in the MenuStyle property of the course data objects returned by the **GetStudentCoursesEx** and **GetClassCoursesEx** functions.

Return value:

An object with the properties listed below. If a menu style does not exist in the table for the specified value of **nStyle**, the object will still be returned but the **StyleID** property will be set to 0. If an error occurs, the return value will be "Nothing".

In the following list only the fields indicated in bold have a hard and fast definition. The other fields exist for historical reasons and may be used as desired by custom menus.

StyleID – The internal identifier for the menu style. 32bit integer value. Values 0-999 are reserved for use by Integrity eLearning.

MenuName – The name of the menu style as displayed in the system administrator module on the course startup property page. 1-255 characters in length.

GeneratorURL – The URL specifying the script to be called to create the course menu for this style. Typically specifies an ASP script in the ASP directory.

BackGround – String value 0 to 255 characters long. Original usage was to contain a URL to an image file for use as the menu background.

DescriptionGraphic – String value 0 to 255 characters long. Original usage was to contain a URL to an image file for use as "description" icon.

NotStartedGraphic – String value 0 to 255 characters long. Original usage was to contain a URL to an image file for use as the "not started" indicator for the menu.

IncompleteGraphic – String value 0 to 255 characters long. Original usage was to contain a URL to an image file to use as the "incomplete" indicator for the menu.

CompleteGraphic – String value 0 to 255 characters long. Original usage was to contain a URL to an image file to use as the "completed" indicator for the menu.

StyleSheetURL – String value 0 to 255 characters long. Original usage was to contain a URL to the HTML style sheet for the menu.

Graphical – String value 0 to 1 character long. Standard usage is as an indicator that the menu style supports the use of graphical links to course components.

HardCoded – String value 0 to 1 character long. Standard usage is as an indicator that the menu style is for a "hard-coded" menu course where the menu directly calls the course's lessons without using the WBT Manager lesson launch mechanism.

Parameters:

dbConn – An ADO connection object connected to the WBT Manager database (as returned by OpenDBConnection).

nStyle – The integer value which identifies the menu desired menu style.

strErrMsg- A text message returned if an unexpected error occurs.

Usage Example:

```
Set = objStyle = GetMenuStyle (dbConn, 1000, strErrMsg)
```

2.7.3.2 GetCourseHTMLBlocks, GetClassHTMLBlocks

GetCourseHTMLBlocks(ByRef dbConn, ByVal nCourseID, ByRef strTopHTML, ByRef strBotHTML)

GetClassHTMLBlocks(ByRef dbConn, ByVal nClassID, ByRef strTopHTML, ByRef strBotHTML)

GetCourseHTMLBlocks and GetClassHTMLBlocks retrieve custom HTML code to be embedded in the course or certification class menus. The retrieved data comes from the HTML 1 and HTML 2 properties in the system administrator module's course or class properties pages.

Return value: None

Parameters:

dbConn – An ADO connection object connected to the WBT Manager database (as returned by OpenDBConnection).
nCourseID, nClassID – The internal ID number of the course or class for which data is to be returned. This value will normally be passed as an HTML parameter to the menu script.
strTopHTML – String variable to receive the contents of the HTML1 property associated with the course or class.
strBotHTML – String variable to receive the contents of the HTML2 property associated with the course or class.

Usage Examples:

```
objUtils.GetCourseHTMLBlocks dbConn,nCourseID,strTopHTML,strBotHTML
objUtils.GetClassHTMLBlocks dbConn,nClassID,strTopHTML,strBotHTML
```

2.7.3.3 GetOrgHTMLBlocks2

GetOrgHTMLBlocks2(ByRef dbConn, ByRef strTopHTML, ByRef strBotHTML, ByVal nOrgID, ByVal nOrgLevel)

GetOrgHTMLBlocks2 retrieves custom HTML code to be embedded in the student menu for specific student hierarchy elements. The retrieved data comes from the HTML1 and HTML2 properties in the system administrator module's hierarchy element properties pages.

Return value: None

Parameters:

dbConn – An ADO connection object connected to the WBT Manager database (as returned by OpenDBConnection).
strTopHTML – String variable to receive the contents of the HTML1 property associated with the organization or department.
strBotHTML – String variable to receive the contents of the HTML2 property associated with the

organization or department.
nOrgID - The internal WBT Manager identifier for the student hierarchy element whose HTML is to be retrieved
nOrgLevel - The level in the student hierarchy that the target element belongs to.

Usage Examples:

```
objUtils.GetOrgHTMLBlocks2 dbConn, strTopHTML, strBotHTML, 1001, 1
objUtils.GetDeptHTMLBlocks2 dbConn, strTopHTML, strBotHTML, 1002, WBTM_C_ORG3LEVEL
```

2.7.3.4 IsCourseExpired

IsCourseExpired(ByRef dbConn, ByVal nCourseID, ByVal nStudentID, ByRef strStatus)

IsCourseExpired tests to see if a student's enrollment in a course has expired based on the course cutoff date set when the student was enrolled in the course.
*This information is also available from the **IsExpired** property of the CourseDataObject.*

Return value:

A boolean value

True - the student is no longer allowed to take the course because the current date is past the cutoff date for the course enrollment.
False - the student's course enrollment is not expired.

Parameters:

dbConn – An ADO connection object connected to the WBT Manager database (as returned by OpenDBConnection).

nCourseID – The internal WBT Manager ID for the course that is being checked.

nStudentID – The internal WBT Manager ID for the student whose course expiration is being tested.

strStatus – A variable to receive the current status of the course (if any). "C" = completed, "I" = in progress, "" = not started or student not enrolled.

Usage Example:

```
bExpired = objUtils.IsCourseExpired( dbConn, nCourseID, nStudentID, strStatus)
```

2.7.3.5 IsCourseReady

IsCourseReady(ByRef dbConn, ByVal nCourseID, ByVal nStudentID, ByRef dtStartAfter)

IsCourseReady tests to see if a course is available for a student to enter based on the course "start-after" date set when the student was enrolled in the course.
*This information is also available from the **IsReady** property of the CourseDataObject.*

Return value:

A boolean value

True - the current date is after the start-after date.
False - the current date is not after the start-after date.

Parameters:

dbConn – An ADO connection object connected to the WBT Manager database (as returned by

OpenDBConnection).

nCourseID – The internal WBT Manager ID for the course that is being checked.

nStudentID – The internal WBT Manager ID for the student whose course availability is being tested.

dtStartAfter– A variable to receive the actual "start-after-date" if the course is available to the student.

Usage Example:

```
bStartOK = objUtils.IsCourseReady(dbConn, nCourseID, nStudentID, dtStartAfter)
```

2.7.3.6 GetCategoryCourses

GetCategoryCourses(ByRef dbConn, ByVal nTrainingCatID, ByVal nStudentID, ByVal nOptions, ByRef nBounds)

GetCategoryCourses returns an array of the courses contained in a particular training category.

Return value:

A zero-based array with an upper bound of nBounds. Each element in the array contains a CourseDataObject. See *CourseDataObject* for details about the available data elements.

Parameters:

dbConn – An ADO connection object connected to the WBT Manager database (as returned by OpenDBConnection).

nTrainingCatID – The internal WBT Manager ID for the TrainingCategory whose course data is desired.

nStudentID – The internal WBT Manager ID for the student whose course assignments are desired. This value will typically be drawn from a Session variable. If this value is 0 then all courses in the training category will be returned. Otherwise only those courses in the training category that the student is enrolled in will be returned.

nOptions – Bitfield flag that determines what course data elements are valid in the returned objects. See *Course Data Object* for a list of the different options. If **nStudentID** is 0 then the following options will be ignored and any associated data elements will be invalid.

opdataStatus	**opdataState**	**opdataEnrollment**
opdataEnrollment2	**opdataPrerequisiteFlag**	**opExcludeNotReady**
opExcludeExpired	**opExcludeCompleted**	**opExcludeCertCourses**

The following options instruct the function to not return courses which meet certain criteria:

opExcludeNotReady - do not return courses where the "start-after" date is not in the past.
opExcludeExpired - do not return courses where the "cutoff-after" date is not in the future.
opExcludeCompleted - do not return courses that have a status of "complete".
opExcludeCertCourses - do not return courses if the course is in a certificate program that the student is assigned to.
opExcludeNonLocal - do not return courses if the course is restricted to specific student hierarchy elements and is not assigned to parent elements of a specific student hierarchy element. If this option is included then nOptions must be a 3-element array rather than a numeric:
element 0 = options flag
element 1 = internal WBT Manager id of the organization to check against
element 2 = organization level to check against.

nBounds – A variable to receive the upper bound of the returned array. If there are no courses found then this variable will contain "-1."

Usage example:

```
nDataOptions = opdataDefault
aCrs = objUtils.GetCategoryCourses(  dbConn, nParentDBID, 0, nDataOptions, nCrs )

For i = 0 To nCrs
   nNextNodeID = nNextNodeID + 1
   Set objCrs = aCrs(i)
   FormatCourse nParentNodeID, nNextNodeID, objCrs
   Set aCrs(i) = Nothing
Next
```

2.7.3.7 GetStudentCoursesEx

GetStudentCoursesEx(ByRef dbConn, ByVal nStudentID, ByVal nOptions, ByRef nBounds)

GetStudentCoursesEx returns an array of the courses that a student is enrolled in.

Return value:

A zero-based array with an upper bound of nBounds. Each element in the array contains a CourseDataObject. See "CourseDataObject" for details about the available data elements.

Parameters:

dbConn – An ADO connection object connected to the WBT Manager database (as returned by OpenDBConnection).

nStudentID – The internal WBT Manager ID for the student whose course assignments are desired. This value will typically be drawn from a Session variable.

nOptions – Bitfield flag that determines what course data elements are valid in the returned objects. See *Course Data Object* for a list of the different options. The following options instruct the function to not return courses which meet certain criteria:

opExcludeNotReady - do not return courses where the "start-after" date is not in the past.
opExcludeExpired - do not return courses where the "cutoff-after" date is not in the future.
opExcludeCompleted - do not return courses that have a status of "complete".
opExcludeCertCourses - do not return courses if the course is in a certificate program that the student is assigned to.
opExcludeNonLocal - do not return courses if the course is restricted to specific student hierarchy elements and is not assigned to parent elements of a specific student hierarchy element. If this option is included then nOptions must be a 3-element array rather than a numeric:
element 0 = options flag
element 1 = internal WBT Manager id of the organization to check against
element 2 = organization level to check against.

nBounds – A variable to receive the upper bound of the returned array. If there are no courses found then this variable will contain "-1."

Usage Example:

```
aEntries = objUtils.GetStudentCoursesEx( dbConn, nStudentID, nDataOptions, nEntry )

strXML = strXML & _
     vbTab & vbTab & "<courses>" & vbCrLf

For i=0 To nEntry
     strXML = strXML + aEntries(i).XMLString( nDataOptions, vbTab, 3, False )
     Set aEntries(i) = Nothing
Next
```

2.7.3.8 GetClassCoursesEx

GetClassCoursesEx(ByRef dbConn, ByVal nClassID, ByVal nStudentID, ByVal nOptions, ByRef nBounds)

GetClassCoursesEx returns an array of the courses contained in a certificate program.

Return value:

A zero-based array with an upper bound of nBounds. Each element in the array contains a CourseDataObject. See "CourseDataObject" for details about the available data elements.

Parameters:

dbConn – An ADO connection object connected to the WBT Manager database (as returned by OpenDBConnection).

nClassID - The internal WBT Manager ID for the certificate program whose courses are desired.

nStudentID – The internal WBT Manager ID for a student. This value will typically be drawn from a Session variable. If this value is 0 then all courses in the program will be returned. Otherwise, only those classes in which the student is enrolled will be returned.

nOptions – Bitfield flag that determines what course data elements are valid in the returned objects. See *Course Data Object* for a list of the different options. If **nStudentID** is 0 then the following options will be ignored and any associated data elements will be invalid.

opdataStatus	**opdataState**	**opdataEnrollment**
opdataEnrollment2	**opdataPrerequisiteFlag**	**opExcludeNotReady**
opExcludeExpired	**opExcludeCompleted**	**opExcludeCertCourses**

The following options instruct the function to not return courses which meet certain criteria:

opExcludeNotReady - do not return courses where the "start-after" date is not in the past.
opExcludeExpired - do not return courses where the "cutoff-after" date is not in the future.
opExcludeCompleted - do not return courses that have a status of "complete".
opExcludeCertCourses - do not return courses if the course is in a certificate program that the student is assigned to.
opExcludeNonLocal - do not return courses if the course is restricted to specific student hierarchy elements and is not assigned to parent elements of a specific student hierarchy element. If this option is included then nOptions must be a 3-element array rather than a numeric:
element 0 = options flag
element 1 = internal WBT Manager id of the organization to check against
element 2 = organization level to check against.

nBounds – A variable to receive the upper bound of the returned array. If there are no courses found then this variable will contain "-1."

Usage Example:

```
aCrs = objUtils.GetClassCoursesEx( dbConn, nParentDBID, 0, nDataOptions, nCrs )

For i = 0 To nCrs
    nNextNodeID = nNextNodeID + 1
    Set objCrs = aCrs(i)
    FormatClassCourse nParentNodeID, nNextNodeID, objCrs
    Set aCrs(i) = Nothing
Next
```

2.7.3.9 GetCourseInfo

GetCourseInfo(ByRef dbConn, ByVal nCourseID, ByVal nStudentID, ByVal nOptions)

GetCourseInfo returns the same data about an individual course that **GetStudentCoursesEx** and **GetClassCoursesEx** return for a list of courses.

Return value:

A CourseDataObject identical to one of the elements in the array returned by GetStudentCoursesEx. If the course is not found then the ID property of the object will be set to 0. See *CourseDataObject* for details about the available data elements.

Parameters:

dbConn – An ADO connection object connected to the WBT Manager database (as returned by OpenDBConnection).

nCourseID – The internal WBT Manager ID for the course whose data is desired.

nStudentID – The internal WBT Manager ID for the student whose course assignments are desired. This value will typically be drawn from a Session variable.

nOptions – Bitfield flag that determines what course data elements are valid in the returned object. See *Course Data Object* for a list of the different options. If nStudentID is 0 then the following options will be ignored and any associated data elements will be invalid.

opdataStatus	**opdataState**	**opdataEnrollment**
opdataEnrollment2	**opdataPrerequisiteFlag**	**opExcludeNotReady**
opExcludeExpired	**opExcludeCompleted**	**opExcludeCertCourses**

The following options instruct the function to not return courses which meet certain criteria:

opExcludeNotReady - do not return courses where the "start-after" date is not in the past.
opExcludeExpired - do not return courses where the "cutoff-after" date is not in the future.
opExcludeCompleted - do not return courses that have a status of "complete".
opExcludeCertCourses - do not return courses if the course is in a certificate program that the student is assigned to.
opExcludeNonLocal - do not return courses if the course is restricted to specific student hierarchy elements and is not assigned to parent elements of a specific student hierarchy element. If this option is included then nOptions must be a 3-element array rather than a numeric:
element 0 = options flag
element 1 = internal WBT Manager id of the organization to check against
element 2 = organization level to check against.

Usage Example:

```
nOptions = opdataActionFlags + opdataMenu + opdataWindowSize + _
        opdataMenuStyle + opdataStatus + opdataState + opdataPrerequisiteFlag
Set objCrs = objUtils.GetCourseInfo( dbConn, objTmp.ID, nStudentID, nOptions )
```

2.7.3.10 GetCourseStructureItem

GetCourseStructureItem(ByRef dbConn, ByVal strItemType, ByVal nItemID, ByVal nOptions)

GetStructureItem returns the same data about an individual lesson or instructional block that **GetCourseStructure** returns for an entire course.

Return value:

A LessonDataObject object identical to one of the elements in the array returned by GetCourseStructure. If the block or lesson is not found then the ID property of the object will be set to 0. See *LessonDataObject* for details about the available data elements.

Parameters:

dbConn – An ADO connection object connected to the WBT Manager database (as returned by OpenDBConnection).

strItemType – String value specifying the type of item sought. "L" for lessons, "B" for blocks.

nItemID – The internal WBT Manager ID for the block or lesson desired.

nOptions – Bitfield flag that determines what lesson data elements are valid in the returned object. See *LessonDataObject* for a list of the different options. The following options will be ignored and any associated data elements will be invalid.

opdataStatus opdataState opdataPrerequisiteFlag opdataPrerequisiteScript

Usage Example:

```
nItemOptions = PageSettings.LookupValue("cat4_validmenuitemfields")
If Not IsNumeric(nItemOptions) Then nItemOptions = opdataDefault + opdataDescription
nItemOptions = CLng(nItemOptions)

'open a database connection
Set dbConn = objUtils.OpenDBConnection()

'get course data
Set objItem = objUtils.GetCourseStructureItem( dbConn, strItemType, nItemID, nItemOptions )
```

2.7.3.11 GetCourseStructure

GetCourseStructure(ByRef dbConn, ByVal nCourseID, ByVal nStudentID, ByVal nOptions, ByRef nBounds)

GetCourseStructure returns an array of the instructional blocks and lessons in a course, where each entry in the array is a LessonDataObject object.

Return value:

A zero-based array of LessonDataObjects with an upper bound of nBounds. See *LessonDataObject* for details about the available data elements.

Parameters:

dbConn – An ADO connection object connected to the WBT Manager database (as returned by OpenDBConnection).

nCourseID – The internal WBT Manager ID for the course.

nStudentID – The internal WBT Manager ID for the student.

nOptions – Bitfield flag that determines what item data elements are valid in the returned objects. See *LessonDataObject* for a list of the different options. If nStudentID is 0 then the following options will be ignored and any associated data elements will be invalid.

opdataStatus opdataState opdataPrerequisiteFlag

nBounds – A variable to receive the upper bound of the returned array. If there are no items found then this variable will contain "-1."

Usage Example:

```
nItemOptions = opdataMenu + opdataWindowSize + _
               opdataStatus + opdataState + _
               opdataPrerequisiteFlag
aItem = objUtils.GetCourseStructure( dbConn, nCourseID, nStudentID, nItemOptions, nItem )
```

2.7.3.12 GetCertClassInfo

GetCertClassInfo(ByRef dbConn, ByVal nClassID, ByVal nDataOptions)

GetCertClassInfo returns the same data about an individual certificate program that **GetStudentClassesEx** and **GetCategoryCertClasses** return for a list of programs.

Return value:

A ClassDataObject object identical to one of the elements in the array returned by GetStudentClassesEx. If the certificate program is not found then the ID property of the object will be set to 0. See *ClassDataObject* for details about the available data elements.

Parameters:

dbConn – An ADO connection object connected to the WBT Manager database (as returned by OpenDBConnection).

nClassID – The internal WBT Manager ID for the certificate program whose data is desired.

nOptions – Bitfield flag that determines what certificate program data elements are valid in the returned object. See *ClassDataObject* for a list of the different options. The following options will be ignored and any associated data elements will be invalid.

opdataStatus opdataEnrollment1opExcludeCompleted

If **opExcludeCertCourses** is NOT included then the **Courses** data element will contain a list of CourseDataObjects for the courses in the program. Each CourseDataObject will contain the same data elements as if GetCourseInfo was called with nOptions set to opdataDefault.

Usage Example:

```
nClsOptions = opdataDefault +  opdataHTML1 + opdataHTML2

Set objCls = objUtils.GetCertClassInfo( dbConn, nClassID, nClsOptions )
```

2.7.3.13 GetCategoryCertClasses

GetCategoryCertClasses(ByRef dbConn, ByVal nCategoryID, ByRef nBounds, ByVal nDataOptions)

GetCategoryCertClasses returns a list of the certificate programs in a certificate category.

Return value:

A zero-based array with an upper bound of nBounds. Each element in the array contains a ClassDataObject. See *ClassDataObject* for details about the available data elements.

Parameters:

dbConn – An ADO connection object connected to the WBT Manager database (as returned by OpenDBConnection).

nCategoryID – The internal WBT Manager ID for the certificate program category whose programs are desired.

nBounds – A variable to receive the upper bound of the returned array. If there are no certificate programs found then this variable will contain "-1."

nDataOptions – Bitfield flag that determines what certificate program data elements are valid in the returned object. See *ClassDataObject* for a list of the different options. The following options will be ignored and any associated data elements will be invalid.

opdataStatus opdataEnrollment1opExcludeCompleted

If **opExcludeCertCourses** is NOT included then the **Courses** data element of each ClassDataObject will contain a list of CourseDataObjects for the courses in the program. Each CourseDataObject will contain the same data elements as if GetCourseInfo was called with nOptions set to opdataDefault.

If **opExcludeNonLocal** is included then the function will not return certificate programs if the program is restricted to specific student hierarchy elements and is not assigned to parent elements of a specific student hierarchy element. If this option is included then nOptions must be a 3-element array rather than a numeric:
element 0 = options flag
element 1 = internal WBT Manager id of the organization to check against
element 2 = organization level to check against.

Usage Example:

```
nDataOptions = PageSettings.LookupValue("cat1c_validcertfields")
If Not IsNumeric(nDataOptions) Then nDataOptions = opdataDefault + opExcludeCertCourses
nDataOptions = CLng(nDataOptions)

aCls = objUtils.GetCategoryCertClasses( dbConn, nParentDBID, nClasses, nDataOptions )

For i = 0 to nClasses
        Set objCls = aCls(i)
        nNextNodeID = nNextNodeID + 1
        FormatClass nParentNodeID, nNextNodeID, objCls
        Set aCls(i) = Nothing
Next
```

2.7.3.14 GetStudentClassesEx

GetStudentClassesEx(ByRef dbConn, ByVal nStudentID, ByRef nBounds, ByVal nDataOptions, nCourseDataOptions)

GetStudentClassesEx returns a list of the certificate programs that a student is enrolled in.

Return value:

A zero-based array with an upper bound of nBounds. Each element in the array contains a ClassDataObject. See *ClassDataObject* for details about the available data elements.

Parameters:

dbConn – An ADO connection object connected to the WBT Manager database (as returned by OpenDBConnection).

nCategoryID – The internal WBT Manager ID for the certificate program category whose programs are desired.

nBounds – A variable to receive the upper bound of the returned array. If there are no certificate programs found then this variable will contain "-1."

- **nDataOptions** – Bitfield flag that determines what certificate program data elements are valid in the returned object. See *ClassDataObject* for a list of the different options.

  If **opExcludeCertCourses** is NOT included then the **Courses** data element of each ClassDataObject will contain a list of CourseDataObjects for the courses in the program. Each CourseDataObject will contain the same data elements as if GetCourseInfo was called with nOptions set to **nCourseDataOptions** below.

  If **opExcludeCompleted** is included then certificate programs for which the student has a status of "complete" will not be returned.

nCourseDataOptions – Bitfield flag that determines what course data elements are valid in the contents of the Courses data element. See *CourseDataObject* for a list of the different options.

Usage Example:

```
nCourseDataOptions = opdataDefault + opdataMenu + opdataWindowSize + _
                     opdataMenuStyle + opdataStatus + opdataState + _
                     opdataPrerequisiteFlag

aEntries = objUtils.GetStudentClassesex( dbConn, nStudentID, nEntry, CLng(nDataOptions), _
                                         CLng(nCourseDataOptions) )

If nEntry >= 0 Then
   strXML = strXML & _
            vbTab & vbTab & "<certificates>" & vbCrLf
   For i=0 To nEntry
      strXML = strXML + aEntries(i).XMLString( nDataOptions, vbTab, 3, False )
      Set aEntries(i) = Nothing
   Next
   strXML = strXML & _
   vbTab & vbTab & "</certificates>" & vbCrLf
End if
```

2.7.4 ClassDataObject

Utility functions that retrieve information about certificate programs actually return a ClassDataObject or an array of ClassDataObjects instead of individual data elements.

2.7.4.1 Data elements

The ClassDataObject has the following data elements. Note that some data elements require that a student be specified.

ClassDataObject Elements			
Data Element	**nOptions value**	**Student required**	**D8hcription**
ID	opdataDefault		The internal WBT Manager ID for the certificate program.
ClassName			The display name of the certificate program.
CategoryID			The internal WBT Manager ID for the course's Certificate Category (**certcat** table).
AllowSelfEnroll			Boolean value indicating whether self-enrollment is allowed for the certificate program.
ReferenceID			The external reference ID assigned to the certificate program.
D8hcription	opdataDehcription		The certificate program dehcription.
CertificateType	opdataExpiration		Numeric code indicating what type of certificate is awarded on completion. 0 = no certificate 1 = Certificate - no expiration 2 = Certificate - expires in "n" months 3 = Certificate - expires in "n" days Where n = ExpirationData
ExpirationData			String containing data controlling how a certificate expires. usage depends on CertificateType. Types 2 and 3 use ExpirationData as a numeric value to determine the length of time the certificate is valid.
Status	opdataStatus	Yes	The student's current status in the certificate program. "C" = completed, "I" = incomplete.
StatusDate		Yes	The date the student's certificate program status changed last.
CategoryName	opdataCategory Name	Yes	The display name of the certificate program's category.

continued on next page

ClassDataObject Elements (continued)			
Data Element	**nOptions value**	**Student required**	**Description**
EnrollmentDate	opdataEnrollment1	Yes	The date that the student was enrolled in the certificate program (or the restart date for an expired certificate).
HTML1	opdataHTML1		HTML text intended to be inserted at the top of the certificate program menu. Up to 4096 characters in length.
HTML2	opdataHTML2		HTML text intended to be inserted at the bottom of the certificate program menu. Up to 4096 characters in length.
Courses	opExcludeCert Courses not included in nOptions		A VB Collection object containing a CourseDataObject for each of the courses contained in the certificate program. Valid data elements in the CourseDataObjects will vary depending on the utility program used to retrieve the certificate program data.

2.7.4.2 XMLString function

XMLString(ByVal nDataOptions As Long, ByVal strIndent As String, ByVal nLeadingIndents As Long, ByVal bDontClose As Boolean)

The XMLString function returns a string containing the data elements of the object in XML format.

Return value:

A string containing the XML representation of the object.

Parameters:

nDataOptions – Bitfield flag that determines what data elements are included in the XML. Typically should match the options specified in the utility function that returned the object.

strIndent – The string to add to the output where an indent is desired (CRLF is used for line-feeds). Typically set to a TAB character.

nLeadingIndents – The number of additional indents to add to every line (for making the output more readable).

bDontClose – Boolean flag indicating whether the function should include the closing tag for the object. If set to true then the calling program can add additional XML elements to the object.

Usage Example:

```
'Certification class data
strXML = strXML & objCls.XMLString( nClsOptions, vbTab, 1, False )
```

Sample Output:

```
<class>
   <id>1124</id>
   <categoryid>Honors</categoryid>
   <classname><![CDATA[New Certificate Program]]></classname>
```

```
<allowselfenroll>true</allowselfenroll>
<referenceid><![CDATA[{63910BF3-CC85-41cc-8B08-E37EC2CFC628}]]></referenceid>
<description>
   <![CDATA[Enter a description of the certificate program here...<hr>]]>
</description>
<certificatetype>3</certificatetype>
<expirationdata><![CDATA[0]]></expirationdata>
<status></status>
<statusdate></statusdate>
<dateassigned></dateassigned>
<html1><![CDATA[]]></html1>
<html2><![CDATA[]]></html2>
<categoryname><![CDATA[Honors]]></categoryname>
   <courses>
         <course>
         <id>1014</id>
         <coursename><![CDATA[New Course]]></coursename>
         <categoryid>1</categoryid>
         <isenrolled>false</isenrolled>
         <vname>
            <![CDATA[integ-crs-{C48EBC43-C50F-4dda-91D5-16F496A8D639}]]>
         </vname>
         <isdisabled>false</isdisabled>
      </course>
      <course>
         <id>1003</id>
         <coursename><![CDATA[Excel 2000: Level 1]]></coursename>
         <categoryid>1</categoryid>
         <isenrolled>false</isenrolled>
         <vname><![CDATA[]]></vname>
         <isdisabled>false</isdisabled>
      </course>
   </courses>
<class>
```

2.7.5 CourseDataObject

Utility functions that retrieve information about courses actually return a CourseDataObject or an array of CourseDataObjects instead of individual data elements.

2.7.5.1 Data elements

The CourseDataObject has the following data elements. Note that some data elements require that a student be specified.

CourseDataObject Elements			
Data Element	**nOptions value**	**Student required**	**Description**
ID	opdataDefault		The internal WBT Manager ID for the course.
CourseName			The display name of the course.
VName			The vendor specific name for the course. Corresponds to the "Provider's course name" field in the course property pages within the WBT Manager administrator module.
CategoryID			The internal WBT Manager ID for the course's Training Category (**crsgrp** table).
IsDisabled			Boolean value indicating whether the coarse has been disabled by a system administrator.
IsEnrolled		Yes	Boolean value indicating whether the student is currently enrolled in the course.
Description	opdataDescription		The course description.
AllowSelfEnroll	opdataActionFlags		Boolean value indicating whether the student is allowed to "self-enroll" in the course through the course catalog.
AllowDrop			Boolean value indicating whether the student is allowed to "self-drop" the course.
AllowReset			Boolean value indicating whether the student should be allowed to "reset stats" for the course.
AllowRecalcStatus			Boolean value indicating whether the student should be allowed to "recalculate status" for the course.
AllowOffline			Boolean value indicating whether the student may take the course in offline mode.

continued on next page

CourseDataObject Elements (continued)			
Data Element	**nOptions value**	**Student required**	**Description**
MenuURL	opdataMenu		The URL to a hard-coded menu for the course.
ImageURL			The URL to an image to be used in place of a text link to launch the course from the student menu.
MenuStyle			The internal WBT Manager ID for the menu style associated with the course.
Shadow			Boolean value indicating whether the course is a "shadow" course containing no online content. "True" if course is a shadow course, otherwise "False".
FullScr	opdataWindowSize		Boolean value indicating whether the course should be displayed in full-screen mode. "True" if full-screen mode is desired, otherwise "False".
X			The desired width of the browser window to contain the course.
Y			The desired height of the browser window to contain the course.
HardMenu	opdataMenuStyle		Boolean value indicating whether the course has a hard-coded menu and will not use the WBT Manager menu generation mechanism. "True" if course has a menu, otherwise "False." If "True" then **MenuURL** should contain the path to the menu page.
GraphicMenu			Boolean value indicating whether the course menu will display graphical links to the lessons in the course. "True" if menu is graphical, otherwise "False." Paths to the graphic files will be found in the **ImageURL** property of objects returned by the **GetCourseStructure** or **GetStructureItem** function.
MenuGen			The script to be used to generate the course's menu.

CourseDataObject Elements (continued)			
Data Element	**nOptions value**	**Student required**	**Description**
Status	opdataStatus	Yes	The student's current status in the course. "C" = completed, "I" = incomplete, "N" = not started.
StatusDate		Yes	The date the student's course status changed last.
IsReady	opdataState	Yes	Boolean value indicating whether the student is allowed in the course yet based on the "start-after" date set when they were enrolled.
IsExpired		Yes	Boolean value indicating whether the student's enrollment in the course has expired based on the "cut-off date" set when they were enrolled.
IsOffline		Yes	Boolean value indicating whether the course is currently being taken in "offline mode" by the student.
StartAfter		Yes	The date before which the student is not allowed in the course.
CutoffDate		Yes	The date after which the student is not allowed in the course.
TargetDate		Yes	The date by which the student is scheduled to have completed the course.
IsPastTarget		Yes	Boolean value indicating whether the current date is past the target date for a student's course enrollment (computed from TargetDate).
CategoryName	opdataCategory Name		The display name of the course's Training Category.
EnrollmentDate	opdataEnrollment1	Yes	The date that the student was enrolled in the course.
EnrollmentAdminID		Yes	The internal WBT Manager ID of the administrator who enrolled the student in the course.
EnrollmentAdmin Name	opdataEnrollment2	Yes	The display name of the administrator who enrolled the student in the course.

CourseDataObject Elements (continued)			
Data Element	**nOptions value**	**Student required**	**Description**
PrerequisiteDisplay Script	opdataPrerequisite Display		Human readable version of the Prerequisite script for the course.
PrerequisiteScript	opdataPrerequisite Script		Internal version of the Prerequisite script for the course.
PrerequisiteOK	opdataPrerequisite Flag	Yes	Boolean value indicating whether the student has completed the prerequisites for the course.
UserText1 - UserText5	opdataUserText1to 5		User defined text fields (1-5). May be any text up to 255 characters in length.
UserText6	opdataUserText6		User defined text field. May be any text up to 64000 characters in length.
UserText7	opdataUserText7		User defined text field. May be any text up to 64000 characters in length.
UserText8	opdataUserText8		User defined text field. May be any text up to 64000 characters in length.
HTML1	opdataHTML1		HTML text intended to be inserted at the top of the course menu. Up to 4096 characters in length.
HTML2	opdataHTML2		HTML text intended to be inserted at the bottom of the course menu. Up to 4096 characters in length.

2.7.5.2 XMLString function

XMLString(ByVal nDataOptions As Long, ByVal strIndent As String, ByVal nLeadingIndents As Long, ByVal bDontClose As Boolean)

The XMLString function returns a string containing the data elements of the object in XML format.

Return value:

A string containing the XML representation of the object.

Parameters:

nDataOptions – Bitfield flag that determines what data elements are included in the XML. Typically should match the options specified in the utility function that returned the object.

strIndent – The string to add to the output where an indent is desired (CRLF is used for line-feeds). Typically set to a TAB character.

nLeadingIndents – The number of additional indents to add to every line (for making the output more readable).

bDontClose – Boolean flag indicating whether the function should include the closing tag for the object. If set to true then the calling program can add additional XML elements to the object.

Usage Example:

```
aEntries = objUtils.GetStudentCoursesEx( dbConn, nStudentID, nDataOptions, nEntry )

strXML = strXML & _
    vbTab & vbTab & "<courses>" & vbCrLf

For i=0 To nEntry
    strXML = strXML + aEntries(i).XMLString( nDataOptions, vbTab, 3, False )
    Set aEntries(i) = Nothing
Next

strXML = strXML & _
    vbTab & vbTab & "</courses>" & vbCrLf

'finish off the XML string
strXML = strXML & "</wbtmpage>" & vbCrLf
```

Sample output:

```
<course>
  <id>1</id>
  <coursename><![CDATA[Using WBT Manager]]></coursename>
  <categoryid>WBT Manager Samples</categoryid>
  <isenrolled>true</isenrolled>
  <vname><![CDATA[integ-wbtm001]]></vname>
  <description>
    <![CDATA[This course contains a series of lessons which will demonstrate how to use the administrative functions of
    WBT Manager to establish and maintain your online training.]]>
  </description>
  <allowreset>false</allowreset>
  <allowrecalcstatus>false</allowrecalcstatus>
  <allowoffline>false</allowoffline>
  <allowdrop>false</allowdrop>
  <allowselfenroll>false</allowselfenroll>
  <shadow>false</shadow>
  <menuurl><![CDATA[]]></menuurl>
  <imageurl><![CDATA[]]></imageurl>
  <menustyle>0</menustyle>
  <fullscr>false</fullscr>
  <x>640</x>
  <y>480</y>
  <hardmenu>false</hardmenu>
  <graphicmenu>false</graphicmenu>
  <menugen><![CDATA[crsmenu.asp]]></menugen>
  <status>I</status>
  <statusdate>2002-05-31T15:57:20</statusdate>
  <startafter></startafter>
  <targetdate></targetdate>
  <cutoffdate></cutoffdate>
  <isoffline>true</isoffline>
  <isexpired>false</isexpired>
  <isready>true</isready>
  <ispasttarget>false</ispasttarget>
  <enrollmentdate>2002-05-16T11:11:42</enrollmentdate>
  <enrollmentadminid>2</enrollmentadminid>
  <enrollmentadminname><![CDATA[System,SelfEnrolled]]></enrollmentadminname>
  <prerequisitedisplayscript><![CDATA[]]></prerequisitedisplayscript>
  <prerequisitescript><![CDATA[]]></prerequisitescript>
  <prerequisiteok>true</prerequisiteok>
  <usertext1><![CDATA[]]></usertext1>
  <usertext2><![CDATA[]]></usertext2>
  <usertext3><![CDATA[]]></usertext3>
  <usertext4><![CDATA[]]></usertext4>
  <usertext5><![CDATA[]]></usertext5>
  <usertext6><![CDATA[]]></usertext6>
  <usertext7><![CDATA[]]></usertext7>
  <usertext8><![CDATA[]]></usertext8>
  <html1><![CDATA[]]></html1>
  <html2><![CDATA[]]></html2>
  <isdisabled>false</isdisabled>
```

```
    <categoryname><![CDATA[WBT Manager Samples]]></categoryname>
</course>
```

2.7.6 LessonDataObject

Utility functions that retrieve information about certificate programs actually return a ClassDataObject or an array of ClassDataObjects instead of individual data elements.

2.7.6.1 Data elements

The LessonDataObject has the following data elements. Note that some data elements require that a student be specified and some elements are not valid for Instructional Blocks.

LessonDataObject Elements				
Data Element	**nOptions value**	**Valid For**	**Student required**	**Description**
ID	opdataDefault	B,L		The internal WBT Manager ID for the lesson or block.
EntryType		B,L		Flag indicating whether this entry is a lesson ("l") or a block ("b").
EntryName		B,L		The display name of the lesson or block.
VName		B,L		The vendor specific name for the lesson or block. Corresponds to the "Provider's name" field in the lesson or block property pages within the WBT Manager administrator module.
IsDisabled		L		Boolean value indicating whether the lesson has been disabled by a system administrator.
IsShadow		L		Boolean value indicating whether the lesson is a "shadow" lesson containing no online content. "True" if lesson is a shadow lesson, otherwise "False".
Description	opdataDescription	B,L		The lesson or block description.
URL	opdataMenu	L		The URL to a launch the lesson. If opdataPhysicalURL is specified then this field will contain the actual URL to the start page of the lesson. Otherwise it will contain the relative url to the lesson launch script to be used.
ImageURL		L		The URL to an image to be used in place of a text link to launch the lesson from the course menu.
Itfc		L		The interface type used by the lesson. 0 = WBT - AICC Compliant 1 = WBT - Not AICC Compliant
FullScr	opdataWindowSize	L		Boolean value indicating whether the lesson should be displayed in full-screen mode. "True" if full-screen mode is desired, otherwise "False".

continued on next page

LessonDataObject Elements (continued)				
Data Element	**nOptions value**	**Valid For**	**Student required**	**Description**
X		L		The desired width of the browser window to contain the lesson.
Y		L		The desired height of the browser window to contain the lesson.
Status	opdataStatus	B,L	Yes	The student's current status in the lesson or block. "C" = completed "P" = passed (lessons only) "F" = failed (lessons only) "I" = incomplete "N" = not started.
sDate		B,L	Yes	The date the student's lesson or block status changed last.
Score		L	Yes	The student's current score in the lesson.
IsOffline	opdataState	L	Yes	Boolean value indicating whether the lesson is currently being taken in "offline mode" by the student.
PrerequisiteScript	opdataPrerequisiteScript	L		Internal version of the Prerequisite script for the lesson. Valid only when retrieved using GetCourseStructure.
PrerequisiteOK	opdataPrerequisiteFlag	L	Yes	Boolean value indicating whether the student has completed the prerequisites for the lesson. Valid only when retrieved using GetCourseStructure.
UserText1 - UserText5	opdataUserText1to5	L		User defined text fields (1-5). May be any text up to 255 characters in length.
UserText6	opdataUserText6	L		User defined text field. May be any text up to 64000 characters in length.
UserText7	opdataUserText7	L		User defined text field. May be any text up to 64000 characters in length.
UserText8	opdataUserText8	L		User defined text field. May be any text up to 64000 characters in length.

2.7.6.2 XMLString function

XMLString(ByVal nDataOptions As Long, ByVal strIndent As String, ByVal nLeadingIndents As Long, ByVal bDontClose As Boolean)

The XMLString function returns a string containing the data elements of the object in XML format.

Return value:

A string containing the XML representation of the object.

Parameters:

nDataOptions – Bitfield flag that determines what data elements are included in the XML. Typically should match the options specified in the utility function that returned the object.

strIndent – The string to add to the output where an indent is desired (CRLF is used for line-feeds). Typically set to a TAB character.

nLeadingIndents – The number of additional indents to add to every line (for making the output more readable).

bDontClose – Boolean flag indicating whether the function should include the closing tag for the object. If set to true then the calling program can add additional XML elements to the object.

Usage Example:

```
strXML = strXML & objItem.XMLString( nItemOptions, vbTab, 2+objItem.Indent, True) & _
   Left( strIndents, 2+objItem.Indent ) & "<sequence>" & CStr(i) & "</sequence>" & vbCrLf & _
   Left( strIndents, 2+objItem.Indent ) & "<statusgraphic>" & strTmp & "</statusgraphic>" & vbCrLf & _
   Left( strIndents, 2+objItem.Indent ) & "<statustext>" & strTmp2 & "</statustext>" & vbCrLf & _
   Left( strIndents, 1+objItem.Indent ) & "</entry>" & vbCrLf
```

Sample Output:

```
<entry>
   <id>2</id>
   <entryname><![CDATA[Intro to the System Admin Module]]></entryname>
   <entrytype>l</entrytype>
   <vname><![CDATA[integ-wbtm001-sysintro]]></vname>
   <indent>0</indent>
   <description>
      <![CDATA[This lesson demonstrates how to log on to the system administrator module. It shows how to use
      pulldown menus, the toolbar,the navigation tree and contextual menus. This lesson requires the Macromedia
      Shockwave for Authorware plug-in (version 5.0F1 or later).  MS Internet Explorer users must install the ActiveX
      version of the Authorware Player.]]>
   </description>
   <url><![CDATA[../courses/wbtmdemo/wbtm001/sysintro/sysintro.html]]></url>
   <imageurl>
      <![CDATA[../courses/wbtmdemo/wbtm001/buttons/sysintrobtn.gif]]>
   </imageurl>
   <itfc>0</itfc>
   <fullscr>false</fullscr>
   <x>640</x>
   <y>440</y>
   <status></status>
   <sdate></sdate>
   <score></score>
   <isoffline>false</isoffline>
   <prerequisitescript><![CDATA[]]></prerequisitescript>
   <prerequisiteok>true</prerequisiteok>
   <usertext1><![CDATA[]]></usertext1>
   <usertext2><![CDATA[]]></usertext2>
   <usertext3><![CDATA[]]></usertext3>
   <usertext4><![CDATA[]]></usertext4>
   <usertext5><![CDATA[]]></usertext5>
   <usertext6><![CDATA[]]></usertext6>
```

```
    <usertext7><![CDATA[]></usertext7>
    <usertext8><![CDATA[]></usertext8>
    <isdisabled>false</isdisabled>
    <isshadow>false</isshadow>
</entry>
```

2.7.7 Constant values

Utility functions which return information about Courses, Lessons, and Certificate Programs use a bitfield parameter (nOptions) to identify the desired data elements. This is done to minimize the amount of database activity on the system. Constant values for the available options are defined in the include file **wbtconst.inc**. The option names and values are listed below. Options are combined by adding them together.

The valid constants and their usage is described in the documentation for each function.

opdataDefault = 0
opdataDescription = 1
opdataActionFlags = 2
opdataMenu = 4
opdataWindowSize = 8
opdataMenuStyle = 16
opdataStatus = 32
opdataState = 64
opdataEnrollment1 = 128
opdataEnrollment2 = 256
opdataPrerequisiteDisplay = 512
opdataPrerequisiteScript = 1024
opdataPrerequisiteFlag = 2048
opdataUserText1to5 = 4096
opdataUserText6 = 8192
opdataUserText7 = 16384
opdataUserText8 = 32768
opdataHTML1 = 65536
opdataHTML2 = 131072
opdataExpiration = 262144
opdataPhysicalURL = 524288
opdataCategoryName = 1048576
opExcludeNonLocal = 33554432
opIncludeDisabled = 67108864
opExcludeCertCourses = 134217728
opExcludeCompleted = 268435456
opExcludeNotReady = 536870912
opExcludeExpired = 1073741824

2.7.8 Page generation functions

There are several functions that relate to generating pages, as described in the following sections.

2.7.8.1 GetPageStyles

GetPageStyles(ByVal strPageName, ByVal bAlwaysEmbed As Boolean)

GetPageStyles returns a string containing the primary and secondary stylesheet information for a page.

Return value:

A string containing the primary and secondary stylesheet information for a page. May contain either a set of <link> tags or <style> tags depending on the **bAlwaysEmbed** flag and entries in **pagesettings.ini**. See *Customization - page level changes* for more details about how the stylesheets are determined.

Parameters:

strPageName - the page identifier used to select the correct stylesheet. Normally the ASP script name without extension.

bAlwaysEmbed - Boolean flag. If True then returned string will contain the actual styles in a <style> tag rather than a <link> tag.

Usage Example:

```
strStyles = objUtils.GetPageStyles( "menu", False)
```

2.7.8.2 GetPageTimeout

GetPageTimeout(ByVal strPageName, ByVal nDefault)

GetPageTimout returns the number of seconds a particular page should delay before performing an action (usually a redirection to another page).

Return value:

A numeric value equal to the **[page_timeout]** setting for the page from **pagesettings.ini** or **nDefault** if no setting exists for the page in **pagesettings.ini**. See *Application settings files* for more details about how to set the timeout.

Parameters:

strPageName - the page identifier used to select the timeout value. Normally the ASP script name without extension.

nDefault - The value to return if there is no timeout entry in **pagesettings.ini** for the page.

Usage Example:

```
strXML = _
    "<wbtmpage>" & vbCrLf & _
    vbTab & "<pageinfo>" & vbCrLf & _
    vbTab & vbTab & "<metatag><![CDATA[" & strMetaTags & "]]></metatag>" & vbCrLf & _
    Application("strWBTPathsXML") & _
    vbTab & vbTab & "<styles><![CDATA[" & strStyles & "]]></styles>" & vbCrLf & _
    Application("strWBTPgBkXML") & _
    vbTab & vbTab & "<pagetimeout>" & CStr(objUtils.GetPageTimeout("login2",3)) & "</pagetimeout>" & vbCrLf & _
    vbTab & "</pageinfo>" & vbCrlf & _
    objUtils.GetUserInfoXML()
```

2.7.8.3 GetUserInfoXML

GetUserInfoXML()

GetUserInfoXML returns information about the logged on user in XML format.

Return value:

A string containing information about the user in XML format. For example:

```
<userinfo>
   <name><![CDATA[ Student,Demo ]]></name>
   <logonid>demo</logonid>
   <type>S</type>
   <organization><![CDATA[ Demo Organization ]]></organization>
   <studcap>Student:</studcap>
   <admincap>Administrator:</admincap>
   <orgcap><![CDATA[ Organization: ]]></orgcap>
   <department><![CDATA[ Demo Department ]]></department>
   <deptcap>Department:</deptcap>
   <browser>
      <type>ie</type>
      <version>5</version>
   </browser>
</userinfo>
```

Parameters: None

Usage Example:

```
strXML = _
   "<wbtmpage>" & vbCrLf & _
   vbTab & "<pageinfo>" & vbCrLf & _
   vbTab & vbTab & "<metatag><![CDATA[" & strMetaTags & "]]></metatag>" & vbCrLf & _
   Application("strWBTPathsXML") & _
   vbTab & vbTab & "<styles><![CDATA[" & strStyles & "]]></styles>" & vbCrLf & _
   Application("strWBTPgBkXML") & _
   vbTab & vbTab & "<pagetimeout>" & CStr(objUtils.GetPageTimeout("login2", 3)) & "</pagetimeout>" & vbCrLf & _
   vbTab & "</pageinfo>" & vbCrlf & _
   objUtils.GetUserInfoXML()
```

2.7.8.4 TransformXML

TransformXML(ByVal strPageName, ByVal strXML)

TransformXML performs an XSL transformation on the XML document generated for a page and outputs the result to the user's browser.

Return value: None

Parameters:

strPageName - the page identifier used to select the XSL file used in the transformation. Normally the ASP script name without extension. See *Customization - page level changes* for more information.

strXML - The raw XML string to be transformed.

Usage Example:

```
If Session("xml_menu") = "Y" Then
   Response.ContentType = "text/xml"
   Response.Write strXML
Else
   Call objUtils.TransformXML( "menu", strXML )
End If
```

2.7.9 Browser Window functions

WBT Manager often opens certain pages in pop windows. Several JavaScript functions are commonly used for this purpose. These functions reside in the JAR directory and are included in pages as necessary.

2.7.9.1 OpenPopupWindow()

```
function OpenPopupWindow( strURL,
                          strWindowName,
                          bFullScreen,
                          nWidth,
                          nHeight,
                          bCentered,
                          nOffsetX,
                          nOffsetY,
                          bShowScrollbar,
                          bTrackChild,
                          bShrinkParent,
                          bRefreshOnChildClose,
                          strCallOnChildClose
                        )
```

OpenPopupWindow opens the specified URL in a child window of the current browser window. It includes options for monitoring the state of the child window and keeping it as the topmost browser window.

File: popwindow.js
Return value: none

Parameters:

- **strURL** - string value containing the URL to open in the new window.
- **strWindowName** - string value containing the window name to use
- **bFullScreen** - boolean flag indicating whether the new window is to be full-screen with no frame or controls (kiosk mode). true = fullscreen, false = popup window
- **nWidth** - integer value containing the desired width of the popup window (ignored if bFullScreen is true).
- **nHeight** - integer value containing the desired height of the popup window (ignored if bFullScreen is true).
- **bCentered** - boolean flag indicating whether the popup window is to be centered in the screen. true = centered, false = use specified offsets.
- **nOffsetX** - integer value containing the desired position of the left edge of the popup window (ignored if bCentered is true).
- **nOffsetY** - integer value containing the desired position of the top edge of the popup window (ignored if bCentered is true).
- **bShowScrollbar** - boolean flag indicating whether the new window should have a scrollbar or not (some browser versions control this automatically and ignore this flag). true = show scrollbar, false = do not show scrollbar
- **bTrackChild** - boolean flag indicating whether the parent window should keep track of the child window and keep it on top until it closes. true = track child window, false = do not track.
- **bShrinkParent** - boolean flag indicating whether the parent window should be reduced in size and moved (to hide behind child window) until the child window closes (ignored if bTrackChild is false). true = shrink, false = don't shrink. This flag is ignored if the parent window is in the main WBT Manager frameset.
- **bRefreshOnChildClose** - boolean flag indicating whether the parent window should refresh itself when the child window closes (ignored if bTrackChild is false). true = refresh, false = do not refresh.
  strCallOnChildClose - reserved for future development.

2.7.10 Enrollment Functions

There are several functions provided for enrolling and dropping students from courses and certificate programs.

2.7.10.1 EnrollStudentInCourse

EnrollStudentInCourse(ByRef dbConn, ByVal nCourseID, ByVal nStudentID,
ByVal dtStartAfter, ByVal dtCutoff, ByVal bUpdateCutoffDT,
ByVal nEnrollingAdminID, ByVal bIgnoreAudience,
ByVal bAddToWaitList, ByVal bIsSelfEnrollment,
ByRef strMsg)

EnrollStudentInCourse allows students to be enrolled in courses programmatically.

Return value:

A numeric status code with one of the following values.

0 - Enrollment succeeded
1 - Student was already enrolled. If bUpdateCutoffDT was true then the start after and cutoff dates in the students enrollment record were updated.
2 - Student had an expired enrollment already and ResetStats failed to move the existing status values into the history tables. strMsg will contain an error message.
3 - Enrollment failed with a database error. strMsg will contain an error message.
4 - Student was added to the waiting list for the course which has concurrent enrollment restrictions.
5 - Course has concurrent enrollment restrictions and all seats are taken. bAddToWaitList was false.
6 - A database error occured while adding a student to the course waiting list for a course with concurrent enrollment restrictions. strMsg will contain an error message.
7 - An unanticipated error occured during enrollment. strMsg will contain an error message.
8 - Course has audience categories set and the student did not have a matching category assigned. bIgnoreAudience was false.
10 - bIsSelfEnrollment was true but the course was not set to allow self-enrollment.

Parameters:

dbConn – An ADO connection object connected to the WBT Manager database (as returned by OpenDBConnection).
nCourseID - The WBT Manager internal ID of the course to enroll the student in.
nStudentID - The WBT Manager internal ID of the student to be enrolled.
dtStartAfter - The "start after" date to be used. Must be a valid date (in any format) or Null.
dtCutoff - The "cutoff after" date to be used. Must be a valid date (in any format) or Null.
bUpdateCutoffDT - Flag indicating whether to update the start after and cutoff after dates if the student is already enrolled in the course. Boolean value (True or False).
nEnrollingAdminID - The WBT Manager internal ID of the administrator who is enrolling the student in the course. For Self-enrollment should be set to 2 for System,SelfEnrolled.
bIgnoreAudience - Flag indicating whether to enforce audience category restrictions. Boolean value (True or False).
bAddToWaitList - Flag indicating whether to add the student to the course waiting list for courses with concurrent enrollment restrictions that have no seats available. Boolean value (True or False).
bIsSelfEnrollment - Flag indicating where call is part of a self-enrollment request. If True then the course record is checked to make sure that self-enrollment is allowed. Boolean value (True or False).

strMsg - Output string to contain error messages.

Usage Example:

```
nEnrollmentStatus = objUtils.EnrollStudentInCourse( dbConn, objSection.CourseID, _
                                                    nStudentID, Null, Null, False, 2, _
                                                    False, False, True, strMsg )
```

2.7.10.2 DropStudentFromCourse

DropStudentFromCourse(ByRef dbConn, ByVal nStudentID, ByVal nCourseID, ByRef bTrans, ByRef strMsg)

DropStudentFromCourse allows students to be dropped from courses programmatically.

Return value:

True if the student was successfully dropped from the course
False if the student could not be dropped. strMsg will contain an error message.

Parameters:

dbConn – An ADO connection object connected to the WBT Manager database (as returned by OpenDBConnection).

nStudentID - The WBT Manager internal ID of the student to be dropped.

nCourseID - The WBT Manager internal ID of the course to drop the student from.

bTrans - Flag which controls transactional processing during the drop procedure.
If initially set to True then the function will assume that a database transaction was started by the calling routine and will not try to start a transaction, commit a transaction or roll back a transaction.
If initially set to False then the function will start a transaction and perform a commit or rollback if supported by the DBMS. **bTrans** will be set to True if transactions are supported.

strMsg - Output string to contain error messages.

Usage Example:

```
bTrans = False
bStatus = objUtils.DropStudentFromCourse( dbConn, nStudentID, objSection.CourseID, _
                                          bTrans, strMsg )
```

2.7.10.3 EnrollStudentInCertProg

EnrollStudentInCertProg(ByRef dbConn, ByVal nProgramID, ByVal nStudentID, ByVal nEnrollingAdminID, ByRef strMsg, ByRef aStatusCodes as Variant)

EnrollStudentInCertProg allows students to be enrolled in certificate programs programmatically.

Return value:

A numeric status code with one of the following values.

0 - Enrollment succeeded

1 - Student was already enrolled.
7 - An unanticipated error occured during enrollment. strMsg will contain an error message.
11 - Invalid program ID specified
12 - Invalid student ID specified

Parameters:

dbConn – An ADO connection object connected to the WBT Manager database (as returned by OpenDBConnection).
nProgramID - The WBT Manager internal ID of the certificate program to enroll the student in.
nStudentID - The WBT Manager internal ID of the student to be enrolled.
nEnrollingAdminID - The WBT Manager internal ID of the administrator who is enrolling the student in the program. For Self-enrollment should be set to 2 for System,SelfEnrolled.
strMsg - Output string to contain error messages.
aStatusCodes - A 3-dimensional array containg the enrollment status for each course contained in the certification program.
aStatusCodes(i, 0) = Status code for the course enrollment (see EnrollStudentInCourse for possible values)
aStatusCodes(i, 1) = WBT Manager internal ID for course.
aStatusCodes(i, 2) = Course name.

Usage Example:

```
nStatus = objUtils.EnrollStudentInCertProg( dbConn, nProgramID, nStudentID, _
                                            2, strErrorText, aStatus )
```

2.7.10.4 DropStudentFromProgram

DropStudentFromProgram(ByRef dbConn, ByVal nStudentID, ByVal nProgramID,
ByRef bTransFlag, ByRef strMsg,
ByVal bDropCourses, ByRef aCourseNames)

DropStudentFromProgram allows students to be dropped from certificate programs programatically.

Return value:

A numeric status code with one of the following values.

0 - Drop succeeded
7 - An unanticipated error occured during drop. strMsg will contain an error message.
11 - Invalid program ID specified
12 - Invalid student ID specified.
13 - Specified student is not enrolled in specified program.

Parameters:

dbConn – An ADO connection object connected to the WBT Manager database (as returned by OpenDBConnection).

nStudentID - The WBT Manager internal ID of the student to be dropped.

nProgramID - The WBT Manager internal ID of the program to drop the student from.

bTransFlag - Flag which controls transactional processing during the drop procedure.
If initially set to True then the function will assume that a database transaction was started by the calling routine and will not try to start a transaction, commit a transaction or roll back a

transaction.
If initially set to False then the function will start a transaction and perform a commit or rollback if supported by the DBMS. **bTrans** will be set to True if transactions are supported.

strMsg - Output string to contain error messages.

- **bDropCourses -** Flag indicating whether the courses contained in the certificate program are to be dropped along with it. Boolen value (True or False)>

  aCourseNames - If bDropCourses was true then this variable will return an array of the course names for the courses dropped with the certificate program.

Usage Example:

```
bTrans = False
nStatus = objUtils.DropStudentFromProgram( dbConn, nStudentID, nProgramID, _
                                           bTrans, strMsg, True, aCourses )
```

2.7.11 Obsolete functions

The functions in this section have been superseded by XML/XSL transformations or by newer functions. They are retained for backward compatibility and should not be used in new development.

2.7.11.1 StartNewPage

StartNewPage(ByVal strLocIDIn, ByVal strTargetIn, ByVal bForceFrame
(Superseded by XML/XSL transformations)

StartNewPage generates the beginning of a standard WBT Manager web page. It outputs the HTML code to link to the style sheets and the JavaScript functions to update the navigation bar (if necessary). Outputs the <HEAD> tag but not the </HEAD> tag.

Return value: None

Parameters:

strLocIDIn - A string value which identifies the current page to the navigation bar. Typically contains the name of the ASP script. The navigation bar script uses this value to decide which buttons (if any) to display and which to make active.

strTargetIn - A string value to be used in the BASE TARGET tag. If an empty string is passed, the value "right" is used.

bForceFrame - A Boolean value which controls whether the WBT Manager frameset is forced for the page. If set to "true," the page will automatically reload the frameset if launched stand-alone.

Usage Examples:

```
objUtils.StartNewPage "crsmenu", "", False

objUtils.StartNewPage "menu", "", True
```

2.7.11.2 EndPage

EndPage()
(Superseded by XML/XSL transformations)

EndPage closes the HTML page by outputting the </BODY> and </HTML> tags, along with any standard WBT Manager closing text.

Return value: None

Parameters: None

Usage Example:

```
objUtils.EndPage
```

2.7.11.3 OpenBrowserWindow, OpenBrowserWindow2

OpenBrowserWindow() and OpenBrowserWindow2()
(Superseded by XML/XSL transformations)

OpenBrowserWindow and OpenBrowserWindow2 output JavaScript code used to open popup windows. The code generated by OpenBrowserWindow2 also tracks the child windows to keep them on top. The actual JavaScript code included by OpenBrowserWindow is in **openc.js** in the jar directory. The code for OpenBrowserWindow2 is in **openc3.js** and **vbchkwin.vbs**. The function contained in these files is called **opencentered**.

Return value: None

Parameters: None

Usage Examples:

```
objUtils.OpenBrowserWindow
objUtils.OpenBrowserWindow2
```

2.7.11.4 GetStudentCourses

GetStudentCourses(ByRef dbConn, ByVal nStudentID, ByRef nBounds)
(Superseded by GetStudentCoursesEx)

GetStudentCourses returns an array of the courses in which a student is enrolled. Each entry in the array contains an object whose properties provide enough information to create a link to the course menu for that course.

Return value:

A zero-based array with an upper bound of nBounds. Each element in the array contains an object with the following properties:

ID – The internal WBT Manager ID for the course.
CourseName – The display name of the course.
Status – The student's current status in the course. "C" = completed, "I" = incomplete, "N" = not started.
VName – The vendor specific name for the course. Corresponds to the "Provider's course name" field in the course property pages within the WBT Manager administrator module.
MenuGen – The script to be used to generate the course's menu.
MenuStyle – The internal WBT Manager ID for the menu style associated with the course.
X – The desired width of the browser window to contain the course.
Y – The desired height of the browser window to contain the course.
FullScr – Boolean value indicating whether the course should be displayed in full-screen mode. "True" if full-screen mode is desired, otherwise "False."
Shadow - Boolean value indicating whether the course is a "shadow" course containing no online content. "True" if course is a shadow course, otherwise "False."

HardMenu - Boolean value indicating whether the course has a hard-coded menu and will not use the WBT Manager menu generation mechanism. "True" if course has a menu, otherwise "False." If "True" then MenuURL should contain the path to the menu page.
GraphicMenu - Boolean value indicating whether the course menu will display graphical links to the lessons in the course. "True" if menu is graphical, otherwise "False." Paths to the graphic files will be found in the ImageURL property of objects returned by the BuildMenuList function.
MenuURL – The URL to a hard-coded menu for the course.
ImageURL – The URL to an image to be used in place of a text link to launch the course from the student menu.
IsExpired – Boolean value indicating whether the student's enrollment in the course has expired based on the "cut-off date" set when they were enrolled.
IsReady – Boolean value indicating whether the student is allowed in the course yet based on the "start-after" date set when they were enrolled.
AllowReset – Boolean value indicating whether the student should be allowed to "reset stats" for the course.
AllowOffline – Boolean value indicating whether the student may take the course in offline mode.
IsOffline – Boolean value indicating whether the course is currently being taken in "offline mode" by the student.
AllowSelfEnroll – Boolean value indicating whether the student is allowed to "self-enroll" in the course through the course catalog.
AllowDrop – Boolean value indicating whether the student is allowed to "self-drop" the course.

Parameters:

dbConn – An ADO connection object connected to the WBT Manager database (as returned by OpenDBConnection).

nStudentID – The internal WBT Manager ID for the student whose course assignments are desired. This value will typically be drawn from a Session variable (see Session variables of interest).

nBounds – A variable to receive the upper bound of the array returned by GetStudentCourses. If there are no courses assigned to the student then this variable will contain "-1."

Usage Example:

```
aCrss = objUtils.GetStudentCourses(dbConn, nStudentID, nCrss)

If nCrss >= 0 Then

    For i = 0 To nCrss
            .
            .
            .
    Next

End If
```

2.7.11.5 GetClassCourses

GetClassCourses(ByRef dbConn, ByVal nClassID, ByVal nStudentID, ByRef nBounds)
(Superseded by GetClassCoursesEx)

GetClassCourses returns an array of the courses in a certification class. Each entry in the array contains an object whose properties provide enough information to create a link to the course menu for that course.

Return value:

A zero-based array with an upper bound of nBounds. Each element in the array contains an object identical to the objects returned by GetStudentCourses.

Parameters:

dbConn – An ADO connection object connected to the WBT Manager database (as returned by **OpenDBConnection**).

nClassID – The internal WBT Manager ID for the certification class whose courses are desired.

nStudentID – The internal WBT Manager ID for the student whose course assignments are desired. This value will typically be drawn from a Session variable (see Session variables of interest).

nBounds – A variable to receive the upper bound of the array returned by GetStudentCourses. If there are no courses in the class then this variable will contain "-1."

Usage Example:

```
aCrss = objUtils.GetClassCourses( dbConn, nClassID, nStudentID, nCrss )

If nCrss >= 0 Then

    For i = 0 To nCrss
            .
            .
            .
    Next

End If
```

2.7.11.6 GetStudentClasses

GetStudentClasses(ByRef dbConn, ByVal nStudentID, ByRef nBounds)
(Superseded by GetStudentClassesEx)

GetStudentClasses returns an array of the Certification Classes in which a student is enrolled. Each entry in the array contains an object whose properties provide enough information to create a link to the Certification Class menu for that class.

Return value:

A zero-based array with an upper bound of nBounds. Each element in the array contains an object with the following properties:

ID – The internal WBT Manager ID for the certification class.

ClassName – The display name for the class.

Status – The student's current status in the class. "C" for completed status, "I" for incomplete status.

Parameters:

dbConn – An ADO connection object connected to the WBT Manager database (as returned by OpenDBConnection).

nStudentID – The internal WBT Manager ID for the student whose class assignments are desired. This value will typically be drawn from a Session variable (see "ref _Ref496083609 Session variables of interest" on page 2-pageref _Ref496083609 22).

nBounds – A variable to receive the upper bound of the array returned by GetStudentClasses. If there are no classes assigned to the student, this variable will contain "-1."

```
If nLsns >= 0 Then
    For i = 0 To nLsns
            .
            .
            .
    Next
End If
```

2.7.11.8 opencentered

opencentered(url,nm,nx,ny,nsc)
(Superseded by new function OpenPopupWindow())

Return value: None

Parameters:

url – the URL to the page to be opened in a new window.

nm – the internal window name for the new window (no more than 6 characters).

nx – the desired width of the new window.

ny – the desired height of the new window.

nsc – scroll bar control. If "0" then new window will have scroll bars.

Usage Example:

```
<A CLASS="clsCrsMenuTxt"
 HREF="JavaScript:opencentered('austart.asp?crsid=1044&lsnid=9', 'lesson', 640, 480,0)"
 target="_self">Adding Departments on the Web</A>
```

2.7.11.9 GetOrgHTMLBlocks, GetDeptHTMLBlocks (copy)

GetOrgHTMLBlocks(ByRef dbConn, ByRef strTopHTML, ByRef strBotHTML)

GetDeptHTMLBlocks(ByRef dbConn, ByRef strTopHTML, ByRef strBotHTML)
(Superceded by GetOrgHTMLBlocks2)

GetOrgHTMLBlocks and GetDeptHTMLBlocks retrieve custom HTML code to be embedded in the student menu for specific organizations and departments. The retrieved data comes from the HTML1 and HTML2 properties in the system administrator module's organization or department properties pages. The organization or department is specified by session variables set during the student login process.

Return value: None

Parameters:

dbConn – An ADO connection object connected to the WBT Manager database (as returned by OpenDBConnection).
strTopHTML – String variable to receive the contents of the HTML1 property associated with the organization or department.
strBotHTML – String variable to receive the contents of the HTML2 property associated with the organization or department.

Usage Examples:

```
objUtils.GetOrgHTMLBlocks dbConn, strTopHTML, strBotHTML
```

objUtils.GetDeptHTMLBlocks dbConn, strTopHTML, strBotHTML

2.8 Session and Application Variables

Session Variables of Interest

Within the WBT Manager Web-server application, a user's logged in state is preserved in ASP Session variables. The following variables may be of interest when creating customizing.

Session("strWBTBrowser") – "IE" if the user's browser is identified as Microsoft Internet Explorer. Otherwise "Netscape"
Session("strWBTBrowserVer") – The browser version as a string.
Session("nWBTBrowserMVer") – The browser major version number.
Session("strWBTLogonID") – The user's logon ID.
Session("strWBTLogonPW") – The user's login password.
Session("strWBTLogonName") – The user's display name.
Session("strWBTUserType") – The user type. "S" = student, "A" = administrator.
Session("nWBTUserID")- The WBT Manager internal ID for the user.
Session("nWBTStudOrg5ID") – The internal ID of the student's corporation.
Session("strWBTOrg5Name") – the name of the student's corporation.
Session("nWBTStudOrg4ID") – The internal ID of the student's region.
Session("strWBTOrg4Name") – the name of the student's region.
Session("nWBTStudOrg3ID") – The internal ID of the student's division.
Session("strWBTOrg3Name") – the name of the student's division.
Session("nWBTStudCompID") – The internal ID of the student's organization.
Session("strWBTOrgName") – the name of the student's organization.
Session("nWBTStudDeptID") – the internal ID of the student's department.
Session("strWBTDeptName") – the name of the student's department.

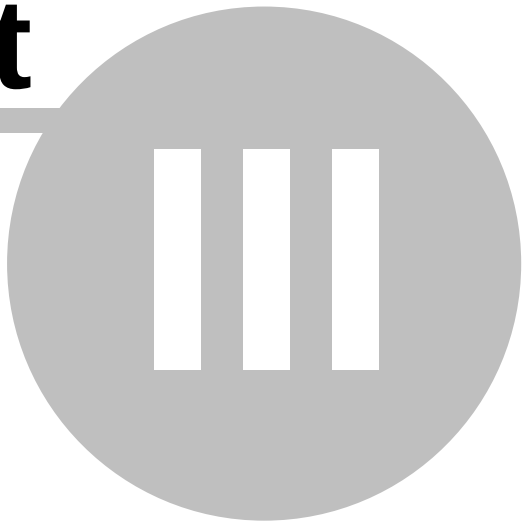
Application variables of interest

The following application global variables may be of interest when customizing.

Application("strWBTProtocol") - The TCP protocol in use ("HTTP" or "HTTPS")
Application("strWBTServer") - The name of the WBT Manager web-server.
Application("strWBTPort") - The TCP port in use if not "80" or "243"
Application("strWBTCoursePath") - URL path to the "courses" directory (does not include protocol or server)
Application("strWBTASPPath") - URL path to the "asp" directory (does not include protocol or server)
Application("strWBTImagePath") - URL path to the "images" directory (does not include protocol or server)
Application("strWBTHTMLPath") - URL path to "html" directory (does not include protocol or server)

Integrating with eCommerce and other sites

Part



3 Integrating with eCommerce and other sites

In order to avoid locking clients into a particular eCommerce solution, WBT Manager does not include a packaged eCommerce module. Instead WBT Manager provides a set of scripts that can be called from other sites to see if students already exist, add new students, and enroll students in courses.

3.1 The scripting interface kit

The scripting interface kit includes three scripts, **verstud.asp**, **regstud.asp**, and **enrollstud.asp**, which accept data from a HTTP form POST request and return a status indicating if they were successful. They can be called from stand-alone forms as well as from the fullfillment script of an eCommerce application. Their uses are explained in the sections that follow.

3.1.1 Checking to see if a student exists

Script name: **verstud.asp**

verstud.asp is called to verify that a student exists in the WBT Manager database.

Parameters:
(must be POST parameters):

loginid The student's WBT Manager Login ID (required, minimum 4 char.).

password The student's WBT Manager password (required, minimum 4 char.).

silent If set to a "1" then the script will not redirect to a Result Message Script (see below). Instead the script will return a "text/plain" page containing a status code, followed on the next line by a status message. The valid code/message combinations are:

0	"found"
1	"missing"
99	"Unexpected error occurred" (may be replaced by a more specific message).

flashmode If set to a "1" and if **silent** is set to "1" then the script will return the status code and message in a name-value format compatible with Macromedia Flash's LoadVariables function. Response content-type will be "application/x-www-form-urlencoded". Returned variable names are "resultcode" and "resulttext" as in:

```
resultcode=0&resulttext=found
```

ignorepw If set to a "1" then the script will check for existence of the student using only the loginid parameter.
lt If set to "A" then the script will check for an Administrator record instead of a Student record.

The remaining parameters are Result Message Script URLs to which the script will redirect the user depending on the outcome of the request. In each case the script will perform the re-direction by creating a page with a form which will automatically submit itself to the specified URL. A submit button will be included in case the form does not submit itself automatically. The form will contain hidden fields duplicating the data originally submitted (excluding any field named "submit"). Status fields errorcode and errortext will also be added containing the status information listed above.

foundurl The Result Message Script to redirect to if the specified loginid and password identify an existing student in the database. Defaults to ../msgtemplates/verstudfound.asp.

newurl The Result Message Script to redirect to if the specified loginid and password do not

identify an existing student in the database. Note: The specified Login ID may be in use by a student with a different password. Defaults to ../msgtemplates/verstudmissing.asp.

errorurl The Result Message Script to redirect to if an unexpected error occurs during processing. Defaults to ../msgtemplates/verstuderror.asp.

Operation:

The script connects to the WBT Manager database and searches the STUDENTS table (ADMIN table if **lt**="A") for the specified loginid. If **ignorepw** is not set to "1" then the script also checks for a valid password.

If a student is found then:
If silent is equal to "1" a "text/plain" page will be returned containing
0<crlf>
found
If silent is not equal to "1" the script will redirect (as described below) the user's browser to the Result Message Script specified by the foundurl parameter.

If a student is not found then:
If silent is equal to "1" a "text/plain" page will be returned containing
1<crlf>
missing
If silent is not equal to "1" the script will redirect (as described below) the user's browser to the Result Message Script specified by the newurl parameter.

If an error occurs then:
If silent is equal to "1" a "text/plain" page will be returned containing
99<crlf>
Unexpected error occurred
If silent is not equal to "1" the script will redirect (as described below) the user's browser to the Result Message Script specified by the errorurl parameter.

Redirection mechanism:

In order allow the most flexibility in use the script does not perform an HTTP "redirect." Instead, the script produces an intermediate form that automatically submits itself to the desired Result Message Script. This form will contain hidden input fields holding all of the data that was originally submitted to the interface script (with the exception of any field named "submit"). In addition, hidden input fields called errorcode and errortext will be added to communicate the status of the request.
Using this mechanism allows the initiating form to pass any custom data fields through the interface script to the Result Message Script.

Sample form:

A sample form **verstud.html** is provided in the HTML directory of the WBT Manager installation.

3.1.2 Adding a new student

Script name: **regstud.asp**

regstud.asp is used to add a new student to the WBT Manager database.

> **Note:** An include file regstud.inc is included in the asp directory to restrict access to this script. The default file blocks all access to the script. It also contains a VBScript example which blocks access to the script if the posting form is in a different domain. regstud.inc must be modified to allow use of regstud.asp.

Parameters:
(must be POST parameters):

dcode Department self-registration code. A code that uniquely identifies the department into which the student is to be added. See below for behavior if omitted.
dname Department name to place the new student in. If ocode is specified and dcode is omitted, then a text string in this parameter will specify the name of the department within the organization to which the student is to be added. If no matching department is found then a new one will be added. Under some DBMSs this value is case-sensitive.
ocode Organization self-registration code. A code that uniquely identifies the organization into which the student is to be added (ignored if dcode is provided). See below for behavior if omitted.
o3code Level 3 organization self-registration code. A code that uniquely identifies the level 3 student hierarchy element into which the student is to be added (ignored if dcode or ocode is provided). See below for behavior if omitted.
o4code Level 4 organization self-registration code. A code that uniquely identifies the level 4 student hierarchy element into which the student is to be added (ignored if dcode, ocode, or o3code is provided). See below for behavior if omitted.
o5code Level 5 organization self-registration code. A code that uniquely identifies the level 5 student hierarchy element into which the student is to be added (ignored if dcode, ocode, o3code, or o4code is provided). See below for behavior if omitted.
fname The student's first name. Required field, Maximum of 128 char.
lname The student's last name. Required field, Maximum of 128 char.
mname The student's middle name or initial. Maximum of 40 char.
sname The student's name suffix (PhD, MD, Jr, etc). Maximum of 40 char.
tname The student's name title (Mr., Mrs., Dr., etc). Maximum of 40 char.
Note: The student's display name will be built from the above name components according to the format set using the Site Configuration Utility. If the length of the formatted display name exceeds 255 characters it will be truncated.
refid An external "Reference ID" which uniquely identifies the student. Maximum of 255 char. Typically used to link a student with records in another system such as an HR system. If used, this field must contain a value that is unique to the student throughout the WBT Manager database. If an attempt is made to add a student with an existing refid then an error will occur (see below).
loginid The login id to be used to identify the student in WBT Manager. Required field, minimum 4 char., maximum 255 char. The value of this field is required to be unique in the WBT Manager database. If an attempt is made to add a student with an existing login id then the new login id will be modified to make it unique unless warndupl is equal to "1". An ID is made unique by appending a random number.
password The password that the student will use to log onto WBT Manager. Required field, minimum 4 char. maximum 12 char.
email The student's email address. Maximum 100 char. If warndupe is set to "1" then an attempt to add a student with an email address already in use by another student will result in an error (see below).
text1-text10 User defined text fields. Maximum 255 char. These fields may be used as desired.
warndupl If this field is set to "1" then the script will issue a warning instead of creating a student record when a matching logonid already exists in the WBT Manager database. Otherwise the script will modify the logonid as described above.
warndupe If this field is set to "1" then the script will issue a warning instead of creating a student record when a matching e-mail address already exists in the WBT Manager database. Otherwise the new student record will be added with the specified email address even if it is in use by another student.
audcat The reference id of a target audience category the student is to belong to. Target audiences are used to restrict the courses that a student can be enrolled in. This field can be repeated to add the student to more than one audience category.
silent If set to "1" then the script will not redirect to a Result Message Script (see below).

Instead the script will return a "text/plain" page containing a status code, followed on the next line by a status message, followed on the next by the loginid actually used if the student is successfully added. The valid code/message combinations are:

0	"Successful"
1	"Duplicate Login ID"
2	"Duplicate Reference ID"
3	"Duplicate e-mail address"
5	"Invalid Organization registration code"
6	"Insertion of new student record failed"
7	"Input string too long"
8	"Login ID is too short or contains blank"
9	"Password is too short"
10	"Password is too long"
11	"Student name is required"
12	"Unable to create unique Login ID"
13	"Missing required parameters"
16	"License count exceeded"
17	"LastName can't be more than 128 characters"
18	"LastName can't be empty"
19	"FirstName can't be more than 128 characters"
20	"FirstName can't be empty"
21	"MiddleName can't be more than 40 characters"
22	"SuffixName can't be more than 40 characters"
23	"Title can't be more than 40 characters"
24	"Invalid department name"
99	"Unexpected error occurred" (may be replaced by a more specific message).

flashmode If set to a "1" and if **silent** is set to "1" then the script will return the status code and message in a name-value format compatible with Macromedia Flash's LoadVariables function. Response content-type will be "application/x-www-form-urlencoded". Returned variable names are "resultcode", "resulttext" and "loginused" as in:

resultcode=0&resulttext=Successful&loginused=demo2

The remaining parameters are Result Message Script URLs to which the script will redirect the user depending on the circumstance. In each case the script will perform the re-direction by creating a page with a form which will automatically submit itself to the specified Result Message Script . A submit button will be included in case the form does not submit itself automatically. The form will contain hidden fields duplicating the data originally submitted (excluding any fields named "submit") with the addition of a new field "logonused" that will contain the actual logon id that the student was given (in case the logon id had to be changed to make it unique. Error fields errorcode and errortext will also be added containing the status information listed above.

successurl The Result Message Script to redirect to if the student was added successfully. Defaults to ../msgtemplates/regstudsuccess.asp.
modlurl The Result Message Script to redirect to if the student was added successfully but their login id was modified. Defaults to ../msgtemplates/regstudmodlogin.asp.
duplurl The Result Message Script to redirect to if "warndupl" is set to "1" and a student with the requested logon_id already exists in the WBT Manager database. Defaults to ../msgtemplates/regstudduplogin.asp.
duprurl The Result Message Script to redirect to if student with the requested refid already exists in the WBT Manager database. Defaults to ../msgtemplates/regstudduprefid.asp.
dupeurl The Result Message Script to redirect to if "warndupe" is set to "1" and a student with the requested e-mail address already exists in the WBT Manager database. Defaults to ../msgtemplates/regstuddupemail.asp.
failurl The Result Message Script to redirect to if an unexpected error occurs. Defaults to

../msgtemplates/regstudfailed.asp.

Operation:

1. The script verifies that the WBT Manager license allows another student. If all licensed seats are in use then it returns the following error:
   16 "License count exceeded"

2. The script validates the parameters.

   First and last name fields are required. Others can be empty. Total length of the name when it is formatted must not be more than 255 characters. This field can generate the following errors:
   7 "Input string too long"
   17 "LastName can't be more than 128 characters"
   18 "LastName can't be empty"
   19 "FirstName can't be more than 128 characters"
   20 "FirstName can't be empty"
   21 "MiddleName can't be more than 40 characters"
   22 "SuffixName can't be more than 40 characters"
   23 "Title can't be more than 40 characters"
   11 "Student name is required"

   The loginid field must contain at least 2 characters and no more than 255 characters. It must contain no embedded blanks. This field can generate the following errors:
   7 "Input string too long"
   8 "Login ID is too short or contains blank"

   The password field must contain at least 4 characters and no more than 12 characters. This field can generate the following errors:
   9 "Password is too short"
   10 "Password is too long"

   The email and text1-text10 fields must contain no more than 255 characters. These fields can generate the following error:
   7 "Input string too long"

3. The student name is assembled from its component parts.

   The final name will be either **lname<sp>sname,fname<sp>mname** or **fname<sp>mname<sp>lname<sp>sname** depending on the name format setting in the database.

4. The WBT Manager database is checked for an existing student with the desired email address.

   If the desired email address is already assigned to another student and warndupe is equal to "1" then the following error will be returned:
   3 "Duplicate e-mail address"

   If the desired email address is in use and warndupe is not equal to "1" then the student will be added with the specified email address even though it is used by another student.

5. The WBT Manager database is checked for an existing student with the desired refid.

   If another student is found that is already assigned the specified refid (Reference ID) then the following error will be returned:
   2 "Duplicate Reference ID"

6. The WBT Manager database is checked for an existing student with the desired logonid.

   If a student already exists with that logonid and warndupl is equal to "1" then the following error

will be returned:
1 "Duplicate Logon ID"

If a student already exists and warndupl is not equal to "1" then a unique logonid will be created by appending a random number to the specified logonid. May return the following if a unique login id cannot be created:
12 "Unable to create unique Login ID"

7. The level 1 Student Hierarchy element (department) to which the student is to be added is identified.

The element is specified using the following fields (in order of priority):
NOTE: In WBT Manager V 1.8 it is possible to hide the higher levels of the student hierarchy from users (see the "Site Configuration utility"). The "ocodes" for hidden levels will be ignored.

dcode
If dcode was specified in the request then the WBT Manager database is searched for a level 1 student hierarchy element (department) with a matching value in the "Registration ID" property. If no matching element is found then the following error will be returned:
5 "Invalid Organization registration code"

ocode /dname (dcode not specified)
If ocode and dname were then the database is searched for a level 2 student hierarchy element (organization) with a matching value in the "Registration ID" property. If no matching element is found then the following error will be returned:
5 "Invalid Organization registration code"

If the element corresponding to ocode is found then the database will be searched for a child of that element having the name contained in dname. If no matching child is found then the following error will be returned:
24 "Invalid department name"

ocode (dcode, dname not specified)
The database is searched for a level 2 student hierarchy element (organization) with a matching value in the "Registration ID" property. If no matching element is found then the following error will be returned:
5 "Invalid Organization registration code"

If the element corresponding to ocode is found then a child level 1 hierarchy element will be added and the new student record assigned to that element. The new element will be given a name containing the year and month the registration occurred (yyyy-mm).

o3code (dcode, dname, ocode not specified)
The database is searched for a level 3 student hierarchy element (division) with a matching value in the "Registration ID" property. If no matching element is found then the following error will be returned:
5 "Invalid Organization registration code"

If the element corresponding to o3code is found then child level 2 and level 1 hierarchy elements will be added and the new student record assigned to the level 1 element. The new elements will be given a names containing the year and month the registration occurred (yyyy-mm).

o4code (dcode, dname, ocode, o3code not specified)
The database is searched for a level 4 student hierarchy element (region) with a matching value in the "Registration ID" property. If no matching element is found then the following error will be returned:
5 "Invalid Organization registration code"

If the element corresponding to o4code is found then child level 3, level 2, and level 1 hierarchy

elements will be added and the new student record assigned to the level 1 element. The new elements will be given a names containing the year and month the registration occurred (yyyy-mm).

o5code (dcode, dname, ocode, o3code, o4code not specified)
The database is searched for a level 5 student hierarchy element (corporation) with a matching value in the "Registration ID" property. If no matching element is found then the following error will be returned:
5 "Invalid Organization registration code"

If the element corresponding to o4code is found then child level 4, level 3, level 2, and level 1 hierarchy elements will be added and the new student record assigned to the level 1 element. The new elements will be given a names containing the year and month the registration occurred (yyyy-mm).

Default: (dcode, dname, ocode, o3code, o4code, o5code not specified)
The database is searched for the first available hierarchy element at the highest level visible for the site (controlled by "Site Configuration" utility). Child hierarchy elements will be added to that element down to level 1 and the new student record assigned to that level1 element. The new elements will be given a names containing the year and month the registration occurred (yyyy-mm).

8. The student is added to the WBT Manager database.

9. The user's browser is redirected to a script based on the outcome of the operation or a "text/plain" page containing the status code and message (as described below) are returned.

Redirection mechanism:

In order allow the most flexibility in use the script does not perform an HTTP "redirect". Instead, the script produces an intermediate form that automatically submits itself to the desired Result Message Script. This form will contain hidden input fields holding all of the data that was originally submitted to the interface script (with the exception of any field named "submit"). In addition, hidden input fields called logonused, errorcode and errortext will be added to communicate the status of the request. Using this mechanism allows the initiating form to pass any custom data fields through the interface script to the Result Message Script.

Sample form:

A sample form **enrollstud.html** is provided in the HTML directory of the WBT Manager installation.

3.1.3 Enrolling a student in a course

Script name: **enrollstud.asp**

enrollstud.asp is used to enable storefront and other e-commerce applications to enroll students in courses within WBT Manager. It may also be used to create custom self-enrollment mechanisms.

Important note: **enrollstud.asp** is dependent on the "Provider's Course Name" to identify the course. If your courseware supplier did not supply a value for this property, use the System Administrator module to apply one ("Provider" tab on the course properties sheet).

> **Note:** An include file enrollstud1.inc is included in the asp directory to restrict access to this script. The default file blocks all access to the script. It also contains a VBScript example which blocks access to the script if the posting form is in a different domain. enrollstud1.inc must be modified to allow use of enrollstud.asp.

Parameters:
(must be POST parameters):

loginid The student's WBT Manager login ID.

coursecode The course's "Provider's Course Name" as entered and displayed in the WBT Manager System Administrator Module. This parameter may be repeated to enroll the student in multiple courses in a single request. May be case-sensitive under some database systems.

If you use multiple coursecodes and an error occurs during enrollment for one of them, the remaining courses in the list are not processed and you will not be able to tell which of the courses were successful. If you are enrolling from a back-end script on an ecommerce site you should process one course at a time and include the appropriate error checking.

cutoffdt A date after which the student will no longer be able to enter the course (yyyy-mmm-dd format).

addwait If set to "1" then the enrollment routine will add the student to the course waiting list if the target course has a concurrent enrollment restriction and is full.

ignoreaud If set to "1" then the enrollment routine will ignore target audience category restrictions.

silent If set to "1" then the script will not redirect to one of the URLs listed below but instead will return a "text/plain" string containing a status code, followed on the next line by a status message. The status code/message combinations are:

0	"Successful"
1	"Student not found"
2	"Course not found"
3	"Student already enrolled"
4	"Missing required parameters"
5	"Invalid date format"
6	"Course is full"
7	"Error adding student to course waiting list"
8	"Student is not in correct target audience for course"
99	"Unexpected error" (may be replaced by a more specific message).

flashmode If set to a "1" and if **silent** is set to "1" then the script will return the status code and message in a name-value format compatible with Macromedia Flash's LoadVariables function. Response content-type will be "application/x-www-form-urlencoded". Returned variable names are "resultcode" and "resulttext" as in:

resultcode=1&resulttext=Student+Not+Found

The remaining parameters are Result Message Script URLs to which the script will redirect the user depending on the circumstance. In each case the script will perform the re-direction by creating a page with a form which will automatically submit itself to the specified Result Message Script . A submit button will be included in case the form does not submit itself automatically. The form will contain hidden fields duplicating the data originally submitted (excluding any fields named "submit"). Error fields errorcode and errortext will also be added containing the status information listed above.

successurl The Result Message Script to redirect to if the student was enrolled in the course **successfully**. Defaults to ../msgtemplates/enrollstudsuccess.asp.

nostudurl The Result Message Script to redirect to if the student was not found in the WBT Manager database. Defaults to ../msgtemplates/enrollstudnostud.asp.

nocrsurl The URL to redirect to if the course was not found in the WBT Manager database. Defaults to customizable file ../msgtemplates/enrollstudnocrs.asp.

enrolledurl The Result Message Script to redirect to if the student was already enrolled in the course. Defaults to ../msgtemplates/enrollstudenrolled.asp.

failedurl The Result Message Script to redirect to if the student could not be enrolled in the course due to an unexpected error. Defaults to ../msgtemplates/enrollstudfailed.asp.

Script operation:

1. The script validates the input data fields.

   logonid and coursecode must not be empty. cutoffdt must be empty or must contain a valid date (format yyyy-mmm-dd is preferred because it is not susceptible to date regional format problems). These fields can generate the following errors:
   4 "Missing required parameters"
   5 "Invalid date format"

2. The script identifies the student.

   The WBT Manager database is searched for a student with the specified logonid. If the student is not found the following error is returned:
   1 "Student not found"

3. The script identifies the course.

   The database is searched for a course with a "Provider's Course Name" which matches the specified coursecode (under some database systems this search is case-sensitive). If the course is not found then the following error is returned:
   2 "Course not found"

   If more than one instance of coursecode exists then each course will be processed in turn. Any "Course not Found" error encountered will halt the processing of the list of courses and return this error code.

4. The script attempts to enroll the student in the course.

   The script may return one of the following error codes:
   3 "Student already enrolled"
   6 "Course is full"
   7 "Error adding student to course waiting list"
   8 "Student is not in correct target audience for course"

5. The user's browser is redirected to a script based on the outcome of the operation or a "text/plain" page containing the status code and message (as described below) are returned.

Redirection mechanism:

In order allow the most flexibility in use the script does not perform an HTTP "redirect". Instead, the script produces an intermediate form that automatically submits itself to the desired Result Message Script. This form will contain hidden input fields holding all of the data that was originally submitted to the interface script (with the exception of any field named "submit"). In addition, hidden input fields called errorcode and errortext will be added to communicate the status of the request.
Using this mechanism allows the initiating form to pass any custom data fields through the interface script to the Result Message Script.

Sample form:

A sample form **enrollstud.html** is provided in the HTML directory of the WBT Manager installation.

3.1.4 Enrolling a student in a certificate program

Script name: **enrollcertstud.asp**

enrollcertstud.asp is used to enable storefront and other e-commerce applications to enroll students in certificate within WBT Manager. It may also be used to create custom self-enrollment mechanisms.

Important note: **enrollcertstud.asp** is dependent on the "Reference ID" to identify the certificate

program. Use the System Administrator module to apply one ("Info" tab on the certificate program properties sheet) to each certificate program that is to be enrolled through this script.

> **Note:** An include file enrollcertstud1.inc is included in the asp directory to restrict access to this script. The default file blocks all access to the script. It also contains a VBScript example which blocks access to the script if the posting form is in a different domain. enrollcertstud1.inc must be modified to allow use of enrollcertstud.asp.

Parameters:
(must be POST parameters):

loginid The student's WBT Manager login ID.
programcode The certificate program's "Reference ID" as entered and displayed in the WBT Manager System Administrator Module. May be case-sensitive under some database systems.
silent If set to "1" then the script will not redirect to one of the URLs listed below but instead will return a "text/plain" string containing a status code, followed on the next line by a status message. The status code/message combinations are:
0 "Successful"
1 "Student not found"
2 "The specified program is not valid."
3 "Student already enrolled"
4 "one or more course enrollments failed"
99 "Unexpected error" (may be replaced by a more specific message).

Note: If 99 is returned for the error code then the student may be partially enrolled in the program. The student's record should be examined to determine what corrective measures need to be taken.

Starting on the third line of the response will be a list of the courses contained in the program and the returned status from the attempt to enroll the student in each one. The format of each line is:

<status code>:<course name>

<status code> will contain the same numeric status as would be returned by the **enrollstud.asp** script for the same course and student. See "Enrolling a student in a course" for details.

flashmode If set to a "1" and if **silent** is set to "1" then the script will return the status code and message in a name-value format compatible with Macromedia Flash's LoadVariables function. Response content-type will be "application/x-www-form-urlencoded". Returned variable names are "resultcode", "resulttext" and "coursestatus" (a comma-separated list of the individual course statuses corresponding to the third through nth lines in a silent mode response) as in:

rresultcode=0&resulttext=Successful&coursestatus=0%3AAICC%2FSCORM+API+sample%2C0%3AFlash+5+Sample+Course%2C

The remaining parameters are Result Message Script URLs to which the script will redirect the user depending on the circumstance. In each case the script will perform the re-direction by creating a page with a form which will automatically submit itself to the specified Result Message Script . A submit button will be included in case the form does not submit itself automatically. The form will contain hidden fields duplicating the data originally submitted (excluding any fields named "submit"). Error fields errorcode and errortext will also be added containing the status information listed above.

successurl The Result Message Script to redirect to if the student was enrolled in the certificate program successfully. Defaults to ../msgtemplates/enrollcertstudsuccess.asp.
successurlp The Result Message Script to redirect to if the student was enrolled in the certificate program but one or more of the associated course enrollments failed. Defaults to

../msgtemplates/enrollcertstudsuccessp.asp.
nostudurl The Result Message Script to redirect to if the student was not found in the WBT Manager database. Defaults to ../msgtemplates/enrollcdrtstudnostud.asp.
noprogurl The URL to redirect to if the certificate program was not found in the WBT Manager database. Defaults to customizable file ../msgtemplates/enrollcertstudnoprog.asp.
enrolledurlThe Result Message Script to redirect to if the student was already enrolled in the certificate program. Defaults to ../msgtemplates/enrollcertstudenrolled.asp.
failedurl The Result Message Script to redirect to if the student could not be enrolled in the certificate program due to an unexpected error. Defaults to ../msgtemplates/enrollcertstudfailed.asp.

An additional user customizable include file called **enrollcertstud1.inc** is included at the beginning of **enrollcertstud.asp**. This file can be used to perform additional validation of the request before the student is enrolled in the course. The supplied version of the file contains sample code that restricts the domains from which the script can be called.

```
By default, all access to the enrollcertstud.asp script is blocked. To enable it you must edit enrollcertstud1.inc
and comment out the following code found near the top of the file.
'Response.Write "Function not enabled!"
'Response.End
```

Script operation:

1. The script identifies the student.

The WBT Manager database is searched for a student with the specified **logonid**. If the student is not found the following error is returned:

1 "Student not found"

2. The script identifies the certificate program.

The database is searched for a certificate program with an internal id number which matches the specified **programcode**. If the certificate program is not found then the following error is returned:

2 "The specified program is not valid"

3. The script attempts to enroll the student in the certificate program .

The script will return the following error code if the student is already enrolled in the certificate program :

3 "Student already enrolled"

4. The script attempts to enroll the student in the courses contained within the certificate program.

The status for each attempted enrollement is checked and if one or more course enrollments failed (where student was neither enrolled or added to a waiting list for the course) then the following primary error will be returned by the script:

4 "one or more course enrollments failed"

5. The user's browser is redirected to a script based on the outcome of the operation or a "text/plain" page containing the status code and message (as described below) are returned.

Redirection mechanism:
In order allow the most flexibility in use the script does not perform an HTTP "redirect". Instead, the script produces an intermediate form that automatically submits itself to the desired Result Message Script. This form will contain hidden input fields holding all of the data that was originally submitted to the interface script (with the exception of any field named "submit"). In addition, hidden input fields called **errorcode** and **errortext** will be added to communicate the status of the request.

Using this mechanism allows the initiating form to pass any custom data fields through the interface

script to the Result Message Script.

3.1.5 Customizing the sample forms

The WBT Manager External Script Interface Kit is supplied with sample forms and message scripts that you can customize for your own purposes.

The following simple forms call the interface scripts and can be modified as desired:

verstud.html
regstud.html
- **enrollstud.html**
enrollcertstud.html

The following forms display the default success or failure messages. You can modify them as desired or (more likely) supply URLs in the original request to redirect to specific scripts in a web store or other web application.

verstuderror.asp
verstudfound.asp
verstudmissing.asp
regstuddupemail.asp
regstudduplogin.asp
regstudduprefid.asp
regstudfailed.asp
regstudmodlogin.asp
regstudsuccess.asp
enrollstudenrolled.asp
enrollstudfailed.asp
enrollstudnocrs.asp
enrollstudnostud.asp
enrollstudsuccess.asp
- **enrollcertstudenrolled.asp**
- **enrollcertstudfailed.asp**
- **enrollcertstudnoprog.asp**
- **enrollcertstudsuccess.asp**
- **enrollcertstudsuccessp.asp**

regstud.inc, **enrollstud1.inc**, and **enrollcertstud1.inc** are included in **regstud.asp**, **enrollstud.asp** , and **enrollcertstud.asp** respectively. They are included after the Result Message Script URLs are set into variables (so you can change them in the include files) but before the requested operation is attempted. The supplied files contain code which checks to see if the request was submitted from the same site as the WBT Manager installation and rejects the request if it comes from another site.

> **Note:** The installed version of these files completely disable the scripts. You will need to modify the include files to comment out the following lines in order to use the scripts.
>
> **Response.Write "<html><body>Function not enabled!</body></html>"**
> **Response.End**

Example: setting up a form-based self-registration mechanism:

1. Use the WBT Manager System Administrator Module or the web-based Local Administrator functions to set Self Registration IDs for any departments that you want to place students in. If you don't use Self Registration IDs the students will go into a default department (see Adding a Student).
2. Modify regstud.html to display the fields that you want the student to fill in. Add validation code as

required.

3. Modify the regstud*.asp message scripts as desired.
4. Place a link to the updated regstud.html file somewhere on your site.
5. If you are using Self Registration IDs to link students to departments then distribute them to the appropriate students for use on the registration form.

Example: enrolling a student in a course from a web store.

1. At some time during the check-out process ask the student if they already have a WBT Manager account.
   If so, have them enter their logon id. If not, have them enter all the information required to add a new student.
2. From your check-out script, post a request in silent mode to verstud.asp or regstud.asp to ensure that the student has a valid record in WBT Manager.
3. After payment is authorized, post a request in silent mode to enrollstud.asp including the student's logon id and the course's "Provider's Course Name".

3.1.6 Setting self-registration IDs

The **regstud.asp** script uses a field in the database to determine what Department a student will be added to. This field is called the Self-Registration ID and exists at all student hierarchy levels. Use the System Administrator Module or the Local Administrator functions to set the Self-Registration IDs for each department that will allow self-registration.

EMail Notifications

4 EMail Notifications

WBT Manager is capable of sending email notices in 3 different ways.

1. By user request
   - Forgotten password requests
   - Email to Instructor-Led section Registrar
   - Client's custom email links and forms

2. By the Batch Maintenance Module (see **utilities\maintenance_module\wbtmmnt.hlp** in the version 1.70 installation set for more details).

3. By the event queue processor (new with version 1.61 see Event Queue Notifications for details).

4.1 The email "wrapper" component

In previous versions of WBT Manager email was sent using the Microsoft **CDONTS.NewMail** object. This required Microsoft's SMTP Server or Exchange Server to be running on the web server. This presented a few problems.

- Email not available for clients using other email servers
- Microsoft is doing away with CDONTS on Windows XP

With version 1.61 WBT Manager now uses an intermediate "wrapper" component to send email. This component supports both **CDONTS** (for NT) and **CDO** (for XP and Windows2000) components as the actual email mechanism.

The wrapper component's object has the ProgID "**wbtmanemail.MailSender**" and is provided in 3 versions.

wbtmanemail.dll	VB6 DLL (this version is installed by)
wbtmanemail.sct	Windows scripting component
wbtmanenomail.dll	C++ DLL that does not send emails (methods return values indicating that email is not available).

VB6 source code is provided in **utilities\emailcomponent\emailcomponent** so that clients may modify the email mechanism to support their actual email environment.

4.1.1 Wrapper component methods

The WBT Manager email wrapper component has the following methods. If you modify the component be careful that you do not change the order or type of parameters of these functions. Also, the ProgID of the component must be "**wbtmanemail.MailSender**".

function CanMail()

CanMail() is called by WBT Manager to determine if email is supported on the web-server. **wbtmanemail.dll** and **wbtmanemail.sct** are both hard-coded to return TRUE. **wbtmannoemail.dll** always returns FALSE.

function SendMail(strTo, strFrom, strSubject, strBodyText, strBodyHTML, bUseHTML)

SendMail() sends an email message. The standard version uses the local SMTP service.

Return Value:

0 if email was sent.
1 if email components for CDONTS or CDO could not be created.

Parameters:

Name	Data Type	Description
strTo	string	The address to send the email to. Must be a valid email address or an error will occur later in the mail process (function performs no validation).
strFrom	string	The address to be used as the "from" address. May or may not require a valid address depending on the SMTP server configuration.
strSubject	string	String to use as the email subject.
strBodyText	string	Text-only version of the email body.
strBodyHTML	string	HTML version of the email body. Will be ignored if bUseHTML is not True.
bUseHTML	boolean	Flag to indicate whether email body should be sent in HTML format. Must evaluate to boolean True or False

4.1.2 Changing wrapper component versions

WBT Manager will always try to create an object with the ProgID "**wbtmanemail.MailSender**" when an email is to be sent. The actual DLL or script code used will depend on the system registry.

Three versions of the component are supplied with WBT Manager. All are copied to the **winnt\system32** directory by the installation program. **wbtmanemail.dll** is activated by default. To activate a different version you need only register the associated file using **regsvr32.exe**. After the selected file is registered stop and re-start the web services to load the new component.

wbtmanemail.dll	Default component, register this file to activate email.
wbtmannoemail.dll	"Null email" component, register this file to disable email.
wbtmanemail.sct	Windows script component version, this file is provided for ease of customization (NT requires a scripting engine upgrade to use this version).

4.2 Event Queue Notifications

A new facility called a System Event Queue has been added with version 1.61. As users perform operations in the WBT Manager modules, events will be added to the queue.

4.2.1 What is the event queue?

The event queue is a table in the WBT Manager database that receives entries associated with actions performed by users. A queue processor can be used to provide customizable actions based on those events in near real-time.

Version 1.61 supports the following events (more will be added with future releases).

Code	Event	Default event handler action
1000	ILT course reservation added	Email notification to student
1001	ILT course reservation change	Email notification to student
1002	ILT course section added	No action
1003	ILT course section modified	If modification caused change to location, instructor, or dates, an email is sent to students with reservations.
1004	ILT course section deleted	No action.

4.2.2 The queue processor

Events in the System Event Queue are processed by the WBT Manager Event Queue processor.

The Event Queue Processor is composed of a queue processor component that is loaded into the web module's application variables and a set of event handlers.

The queue processor scans the system event queue table (SYSEVT) looking for unprocessed events. For each event it calls a method in an event handler object. For more about event handlers see Customizing event handlers.

The queue processor starts scanning the event table after the first request is made to the web application (after the latest restart or reboot). It continues scanning until it is stopped from the system utilities menu or until the web service is shut down.

When the queue processor runs out of events to process it goes into a pause state before checking for more events. The pause time is configurable and defaults to 5 minutes.

4.2.3 Enabling/disabling the queue processor

Temporarily stopping the processor
The queue processor may be stopped from the System Utilities menu.

Log onto the web module with a system-level administrator account
Select "System Utilities" from the administrator menu.
Select "Event Queue Processor Control" from the system utilities menu
The status of the queue processor will display. Click the "stop processing" link to stop the processor. Click the "start processing" link to re-start it.

Disabling the processor
If you do not want the queue processor to run on a particular WBT Manager installation (if you have multiple installations connected to a single database only one should run the processor). You can disable it by setting the **enable** keyword in the **[queue_processor]** group to "N" in the **appsettings.ini** file. You will need to re-start the IIS services to actually unload the queue processor

component.

4.2.4 The processor control file "eventq.ini"

The event queue processor is enabled or disabled by settings in **appsettings.ini**. Other settings are found in the processor control file specified by the **[event_queue]config_file** setting in **appsettings.ini**. This defaults to **database\eventq\eventq.ini**.

eventq.ini contains:

[event_processor] - The event_processor group contains settings that control the behavior of the main processor itself (as opposed to the event handlers). Keywords are:

sleep_duration - specifies how long (in milliseconds) the processor should pause after coming to the last event in **sysevt** before checking for new events. Default value is 300000 (5 minutes). Reducing this value would cause the events to be processed in closer to "real-time" but can cause significant load on the server.

logfile_path - the path to a directory where the queue processor should save log files. If this value contains the string "%HOME%" then that string will be replaced with the home directory specified when the queue processor **Start** method is called (the WBT Manager virtual directory root). Defaults to "%HOME%database\eventq\logs"

logfile_prefix - the prefix to be used for log file names. The full name of the log files will be **<prefix>_yyyymmdd.log** where yyyy is the year including century, mm is the 2 digit month and dd is the 2digit year. Defaults to "qplog".

logfile_retention - controls how many days a log file is kept. "0" disables log file deletion. Default value is "7".

NOTE: the log file purge routine will purge all files with the same name format as the queue processor logs. Make sure that the logfile_prefix and logfile_path do not specify files that are not queue processor logs from this instance of the queue processor.

[event_handlers] - the event_handlers group specifies the event handler ProgID for each event code. Handler "0" is the default event handler used if no specific handler is specified.

Example:

```
[event_handlers]

;handler 0 is the default event handler
0=wbtman4qp.EventHandler

1000=wbtman4qp.EventHandler
1004=wbtman4qp.IgnoredEventHandler
```

The **eventq.ini** file will also contain control settings for specific event handlers. These settings will be in a group named after the handler's ProgID. Here is what the standard event handler's settings look like:

```
[wbtman4qp.EventHandler]
;standard WBT Manager event handler.

;debug_level controls what is written to the log file
;  0 = nothing
;  1 = result, sequence number, type for each event
;  2 = raw data from database
;  3 = sql statements
;  4 = notification text
```

```
debug_level=1

;paths to the xsl files used to format notification messages
;  The string %HOME% will be replaced by the home directory path
;  specified in the call to the DoEvent method.
;  A specific notification can be blocked by setting an empty string for its XSL file

new_ilt_reservation=%HOME%database\eventq\xsl\qp_newres.xsl
new_waitlist_entry=%HOME%database\eventq\xsl\qp_newwait.xsl
reservation_change=%HOME%database\eventq\xsl\qp_reschng.xsl
reservation_canceled=%HOME%database\eventq\xsl\qp_rescancel.xsl
section_change_students=%HOME%database\eventq\xsl\qp_sechg_stud.xsl

;return email address to be used
return_address=wbtmanager@ielearning.com
```

4.2.5 Customizing event handlers

System events are processed by "Event Handlers". Event handlers are contained in COM components and may be created using any development tool that supports COM components. A separate handler may be specified for each event.

NOTE: The following topics are intended for clients who have the technical know-how to create and modify COM components. All modifications should be tested thoroughly in a development environment before being loaded to a production site. Programming errors in event handlers may have severe impact on the web-server where they are running and could cause the IIS service to crash.

4.2.5.1 Event handler life cycle

Events are mapped to their handlers in the **[event_handlers]** group of the processor control file. The queue processor creates instances of the event handler objects as needed and keeps them in memory for re-use.

When the processor encounters an event it checks its list of event handlers for an existing handler object for that event type. If an object exists, its **DoEvent** method is called to process the event.

If an object does not exist for the event then the queue processor checks the processor control file for a handler mapping in the **[event_handlers]** group. If a mapping exists the the processor creates an object from the specified ProgID and adds it to the handler list. The new object's **DoEvent** method is called to process the event.

If the queue processor does not find a handler mapping for the event in the processor control file then if assigns the new event type to the standard event handler and calls that object's **DoEvent** method is called to process the event.

> Event handlers are retained in memory for re-use until: The web server is re-booted or, the IIS service is stopped, or the "Event Queue Processor Control" option on the "System Utilities" menu is used to stop the queue processor. If you are making changes to event handlers you will need to stop the processor to replace the handler.

4.2.5.2 The DoEvent method

Events are processed by the **DoEvent** method of the event handler. All event handler objects must provide the **DoEvent** method as specified below.

Function **DoEvent**

VB6 definition:

```
Public Function DoEvent( _
                ByRef dbConn, _
                ByVal nEventSequenceNo, _
                ByVal nEventType, _
                ByVal strMasterLogFile, _
                ByVal strConfigFile, _
                ByVal strHomeDirectory
                )
```

Return value:

Variant containing an integer value.

All event handlers should trap errors and return the appropriate value from the list below. Any untrapped handler errors will result in behavior as if they had returned "1".

0 = success, delete event from **sysevt** table
1 = flag as processed but do not delete from **sysevt** table
2 = retry after pausing for the time specified in the processor control file **sleep_duration** setting
3 = leave unprocessed, shut down queue processor.

Parameters (all parameters are input only):

dbConn - a variant reference to an adodb.connection object connected to the WBT Manager database.
Do not close this database connection. Also make sure that any transactions are committed or rolled back and that all recordsets are closed before returning from DoEvents.

nEventSequenceNo - a variant containing the value of the **evtseq** column in **sysevt** for the event being processed (**evtseq** is the primary key of the **sysevt** table).

nEventType - a variant containing the value of the **evttype** column in **sysevt** for the event being processed.

strMasterLogFile - a variant containing the path and name of the log file in use by the queue processor.

strConfigFile - a variant containing the path and name of the processor control file in use by the queue processor.

strHomeDirectory - a variant containing the path to the home directory specified when the queue processor queue processor was started (will normally contain the WBT Manager virtual directory path).

C++ invokation:

The **DoEvents** method of the event handler object will always be called by the queue processor which is a thread-safe component written in MS VC++. The actual code used to invoke the method is:

```
long CEventHandler::DoEvent(
        const VARIANT& dbConn,
        const VARIANT& nEventSequenceNo,
        const VARIANT& nEventType,
        const VARIANT& strMasterLogFile,
        const VARIANT& strConfigFile,
        const VARIANT& strHomeDirectory )
{
    long result;
```

```
    static BYTE parms[] =
    VTS_VARIANT VTS_VARIANT VTS_VARIANT VTS_VARIANT VTS_VARIANT VTS_VARIANT;

    InvokeHelper(
            m_nDispatchID_DoEvent,
            DISPATCH_METHOD,
            VT_I4,
            (void*)&result,
            parms,
            &dbConn,
            &nEventSequenceNo,
            &nEventType,
            &strMasterLogFile,
            &strConfigFile,
            &strHomeDirectory
            );

    return result;
}
```

4.2.5.3 The standard event handler

WBT Manager version 1.70 installs a standard event handler object which processes the events associated with student reservations to instructor led training sections. The standard handler sends emails to students when their reservation status changes or if the course section is rescheduled.

The standard handler resides in **wbtman3qp.dll**. Source code for this DLL is provided for those who need to modify the behavior of the standard handler. The source is in **utilities\eventqueue** in the 1.70 installation set.

WARNING: Program errors in event handlers can potentially crash the IIS service on your web site. All modifications are at the client's own risk. Integrity eLearning may require that the standard event handler be reinstalled as a condition of receiving customer support.

We strongly recommend that you:

Make a copy of the standard source files and create a new dll rather than modifying the original (use the settings in the processor control file to link events with the new handler).

Test the new handler thoroughly in a development environment.

Monitor the production system carefully after installing a new event handler.

Miscellaneous scripts

Part V

5 Miscellaneous scripts

This section contains information on miscellaneous scripts that can be used for customization.

5.1 Alternative User Authentication Algorithms

We are quite often asked to support "single login" mechanisms or to validate users against "LDAP".

To support these kinds of requests we have added "Alternative Authentication" support. There are now parameters in appsettings.ini which will cause the WBT Manager login script to use a customizable function for authentication rather than authenticating against the WBT Manager user tables. Support for re-authentication on session timeout has also been added.

The user's login id must exist in WBT Manager to complete the login process but their password will be ignored.

The following explains how the alternate authentication mechanism works when the **[login] usealternateauthentication** keyword in appsettings.ini is set to "Y".

If **altauthenticationmode** is set to "0" in appsettings.ini then:

The user must enter their id and password in the login form (**login.asp**). The values are submitted to **login2.asp** for authentication.
login2.asp calls the **AuthenticateUser()** function found in **login2.inc** to validate the user.
If **AuthenticateUser()** returns true then utility function **UserLogonNoPW()** is called to log the user on. For more information about the **AuthenticateUser** function see the comments in **login2.inc**.
When a session times out the user is re-directed back to the login form.

If **altauthenticationmode** is set to "1" in appsettings.ini then:

The login page (**login.asp**) will immediately redirect to **login2.asp**.
login2.asp will create an instance of the COM object specified by **altauthenticationobject** in appsettings.ini.
The **AuthenticateUser()** method of the object will be called to validate the user.
If **AuthenticateUser()** returns true then utility function **UserLogonNoPW()** is called to log the user on. For more information about this method see the comments in **wbtmanauth.sct** (installed in the system32 directory).
When a session times out or if a user hits a page other than **login.asp** to enter the site then an instance of the **altauthenticationobject** will be create and used as above. NOTE: depending on the page initially hit there may be some confusion about whether the user is a student or an administrator. If possible, the user type should be set by the **AuthenticateUser()** method based on the information used to identify the user.

You will note that with this method there is no interaction with the user during the login process. It is up to the client to determine how to identify the user. A typical method would be to disallow anonymous access (and enable Integrated Windows Authentication) on the WBT Manager virtual directory and obtain the user's network login id using the "LOGON_USER" server variable.

5.2 Direct lesson launch with austart.asp

We have often been asked if there is any way to bypass the WBT Manager course menu systems and directly launch a lesson. For instance, a client may want to build a non-web front-end program or create a static course menu to reduce the overhead on the web-server. We have added this capability to the lesson launch script **austart.asp**.

> **Note:** Replacing the standard WBT Manager web interface does not exempt you from the "Powered by WBT Manager" logo requirements mentioned below. Both the WBT Manager logo and the WBT Manager copyright notice must be included in your user interface unless you have made contractual arrangements with Integrity eLearning to the contrary.

A lesson may be launched by a direct call to austart.asp under the following conditions.

The student must be logged on.

The student must have logged onto WBT Manager using either the standard WBT Manager login screen or through a specialized login script such as **slogin.asp**.

The browser or browser-substitute application must support the ASP session cookie.

The student must be enrolled in the course.

Using austart.asp

austart.asp is the WBT Manager launch script for AICC-compliant lessons. In normal use it is called from a course menu in a popup window. The script creates a new AICC session for the lesson and then redirects to the lesson launch URL using a piece of JavaScript code (found in jar/austart.js).

Here is an example of what the result of austart.asp might be.

```
<html>
<head>

<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=windows-1252">
<link rel="stylesheet" href="/wbtmanager/html/ie.css">
<script language="JavaScript">
//redirect to lesson page
var lsnurl="../courses/wbtmdemo/wbtm001/sysintro/sysintro.html?AICC_SID=demo832&
        "AICC_URL=http%3A%2F%2Flocalhost%2Fwbtmanager%2Fasp%2Faicc%5Fint%2Easp";
</script><script language="JavaScript" SRC="/wbtmanager/jar/austart.js"></script><title>Session created</title>
</head>
<body class="clsStudentPage" onload="JavaScript:austart(lsnurl,'demo832',10);" style="">
<p class="clsMedInst">A session has been created for this lesson.
    Click <a href="JavaScript:AULaunch();">here</a> if the lesson does not start automatically.</p>
</body>
</html>
```

Note that the launch url is embedded in the page in the script block.

```
var lsnurl="../courses/wbtmdemo/wbtm001/sysintro/sysintro.html?AICC_SID=demo832&
        "AICC_URL=http%3A%2F%2Flocalhost%2Fwbtmanager%2Fasp%2Faicc%5Fint%2Easp";
```

Direct lesson launch

In order to directly launch the lesson you will need to call austart.asp with parameters that identify the course, lesson, and CMI mode (the student is identified by a session variable). austart.asp accepts the following parameters for this purpose:

crsvname – The "provider's course name" for the course. This value can be found on the Provider tab in the course property pages within the WBT Manager system administrator program.

lsnvname - The "provider's lesson name" for the lesson. This value can be found on the Provider tab in the lesson property pages within the WBT Manager system administrator program.

cmode – The desired lesson mode for the launch. Allowed values are: "n" for "normal", "r" for "review", and "b" for "browse". If this parameter is not specified then austart.asp may respond with a form requesting that the student select a mode.
nohacp - Controls whether austart includes the HACP parameters on the URL. If the LMS API adapter is available then the HACP parameters are not needed. Set to "1" to not send HACP parameters.

When called with these parameters austart.asp will perform some validation checking and then create a session for the lesson and return the launch page. Errors may be returned under the following conditions:

Student is not logged in.
Specified course does not exist.
Specified lesson does not exist.
Specified lesson is not in the specified course.
Student is not enrolled in the course or enrollment is not active (expired or not reached the start-after date).

Here is an example URL that would launch the first lesson in the Using WBT Manager course that is provided with WBT Manager.

http://www.ielearning.com/wbtroot/asp/austart.asp?cmode=n&
crsvname=integ-wbtm001&lsnvname=integ-wbtm001-sysintro

After **austart.asp** returns the launch page it is up to your front-end application to either open a browser window to process it or extract the URL and perform browser-like operations directly.

5.3 Silent login script slogin.asp

We have had many requests to allow customers to create their own login forms so that a student can be logged in automatically based their having a valid logged-in session on another site. We have supplied a silent login script **slogin.asp** to support this.

slogin.asp is a customizable logon script that allows you to bypass the WBT Manager home page and log onto WBT Manager from your own forms. The script validates the user against the WBT Manager database and sets all needed session variables. **slogin.asp** is installed with WBT Manager to the ASP subdirectory of the WBT Manager Web Application. The script must be accessed from that directory in your form ACTION parameter.

Parameters:
(may be either GET or POST parameters):

goodurl The page to redirect to if the logon attempt is successful (required).

badurl The page to redirect to if the logon attempt is not successful (required). If either goodurl or badurl is missing then the script will return an error page.

logon_id The student or administrator's WBT Manager user ID.

password The student or administrator's WBT Manager password.

lt The logon type. A = administrator, S = student

Actions:

If either goodurl or badurl is empty then an error page is returned.

The logon information is validated against the WBT Manager database.

If the user is valid, session variables are set for the user and the user's browser is redirected to

the page specified by goodurl.

If the login is invalid, then the user's browser is redirected to the page specified in badurl.

Sample Form Code"

```
<FORM action="../asp/slogin.asp?lt=s" id="logonform" method="post" name="logonform">
   <TABLE BORDER="0">
      <TR>
         <TD>StudentID: </TD>
         <TD><INPUT NAME="logon_id"></TD>
      </TR>
      <TR>
         <TD>Password: </TD>
         <TD><INPUT NAME="password"></TD>
      </TR>
      <TR>
         <TD> </TD>
         <TD><INPUT TYPE="submit" VALUE="Log On" id="SUBMIT1" name="SUBMIT1"></TD>
      </TR>
   </TABLE>
   <INPUT TYPE="hidden" NAME="goodurl" VALUE="../html/good.html">
   <INPUT TYPE="hidden" NAME="badurl" VALUE="../html/bad.html">
</FORM>
```

5.4 Emailing lost passwords with emailpw.asp

Script name: **emailpw.asp**

emailpw.asp script is called to e-mail a student's login information to their e-mail address.

> **Note:** Requires Microsoft SMTP Service to be installed and running on web server. You must also modify the emailpw.inc file to specify a valid domain in the from address field of the email.

Parameters:
(may be either GET or POST parameters):

loginid The student's WBT Manager Login ID.

email The student's WBT Manager email address. Note: This may be case-sensitive in some databases. Either loginid or email must be specified.

admin If this parameter contains any value then the administrator table will be searched rather than the student table

The remaining parameters are Result Message Script URLs to which the script will redirect the user depending on the outcome of the request. In each case the script will perform the re-direction by creating a page with a form which will automatically submit itself to the specified URL. A submit button will be included in case the form does not submit itself automatically. The form will contain hidden fields duplicating the data originally submitted (excluding any field named "submit"). Status fields errorcode and errortext will also be added containing the status information listed above.

emailpw.asp The Result Message Script to redirect to if the login information is successfully emailed. Defaults to ../msgtemplates/emailpwok.asp.

emailpw.asp The Result Message Script to redirect to if the specified loginid or email address do not identify an existing student in the database. Defaults to ../msgtemplates/emailpwnf.asp.

emailpw.asp The Result Message Script to redirect to if an unexpected error occurs during processing. Defaults to ../msgtemplates/emailpwer.asp.

Operation:

1. The script verifies that either loginid or email was supplied.

   If both are missing then the script redirects to the result message script specified by errorurl (as described below) and passes the following error:
   code = 4
   text = "Missing required parameter"

2. The script connects to the WBT Manager database and searches the STUDENTS, or ADMIN table for the specified loginid or email.

   If no record is found then the script redirects to the result message script specified by notfounurl and passes the following error:
   code = 1
   text = "Student not found" or "Administrator not found"

3. The script verifies that an e-mail address exists for the student or administrator.

   If no e-mail address exists then the script redirects to the result message script specified by notfounurl and passes the following error:
   code = 2
   text = "Login has no associated email address"

4. The script attempts to e-mail the user's login id and password to them using the commands in the file emailpw.inc.

   If any errors occur exists then the script redirects to the result message script specified by errorurl and passes the following error:
   code = 99
   text = error specific text

   If the email is sent successfully then the script redirects to the result message script specified by successurl .

Redirection mechanism:

In order allow the most flexibility in use the script does not perform an HTTP "redirect". Instead, the script produces an intermediate form that automatically submits itself to the desired Result Message Script. This form will contain hidden input fields holding all of the data that was originally submitted to the interface script (with the exception of any field named "submit"). In addition, hidden input fields called **errorcode** and **errortext** will be added to communicate the status of the request.
Using this mechanism allows the initiating form to pass any custom data fields through the interface script to the Result Message Script.

Sample form:

A sample form **emailpw.html** is provided in the HTML directory of the WBT Manager installation.

5.5 SkillSoft Business Skills Interface

Some content from certain vendors does not use Objectives data in the manner specified in the AICC specification. This content requires that the LMS return an [objective_status] group that is an exact copy of the [objectives_status] group that was sent by the lesson the last time it was invoked.

An example of this content is SkillSoft ™ Business Skills content prior to upgrades released in 2004. WBT Manager supports this content with a non-standard interface found in **aicc_int_c1.asp**.

To use the alternate interface edit the properties of each affected lesson. On the "Startup" tab enter **aicc_int_c1.asp** in the "Non-standard interface script" box.

WBT Manager Logo Requirements

6 WBT Manager Logo Requirements

We designed the WBT Manager web-server application to allow our clients to customize the look as much as desired. However, in order to protect our trademark and copyright, you are required by our standard license agreement to comply with the following sections.

6.1 General Requirements

Unless you have a written contractual agreement with Integrity eLearning to the contrary:

You may not remove or obscure the Integrity eLearning copyright notice on the logout page. If you customize the logout page in such a way that the user is automatically redirected to another page when logging out of WBT Manager, you must include the WBT Manager copyright notice on the target page or delay the redirection on logout so that the Integrity eLearning copyright notice is visible for at least 4 seconds.

If you completely remove the standard WBT Manager logo and name from the home page, you must include one of the standard "Powered by WBT Manager" logo graphics on the redesigned home page. See below for usage details.

If you use an alternative login mechanism that bypasses the WBT Manager home page completely, you must include one of the standard "Powered by WBT Manager" logo graphics on the web page containing the link used to jump into the WBT Manager web-server application. See below for usage details.

6.2 Logo Usage

Standard "powered by" logo graphics are available in the **utilities\powered_by_logos** directory on the WBT Manager installation set or from customer support.

The following excerpt from Appendix B of the WBT Manager license contract provides guidelines for the use of these graphics.

APPENDIX "B" - Logo Usage Guidelines

General Guidelines

The "Powered by WBT Manager" trademark can only be used by licensed parties.

The trademark must never be altered and must be reproduced from the supplied digital file.

The trademark may not be used in connection with the display, advertising or promotion of products that do not contain WBT Manager modules.

Location Guidelines

The "Powered by WBT Manager" logo must appear on the splash screen, logon screen or within the software product.

Size Guidelines

The logo must maintain its original size as supplied in the digital file.

Color Guidelines

Either the color or black and white version of the "Powered by WBT Manager" logo is acceptable on screen.

Background

The logo can be placed on any background as long as the logo is clearly visible.

The logo may NOT be reversed to white. Use the black and white version supplied.