Web FINANCIER™
Installation Guide

Version 1.19
March 7, 2007

Introduction

Purpose

This manual describes the installation process of the web-based FINANCIER™ product.

It is assumed that the users of this manual will be capable of performing basic Oracle® installation and administration functions. Since there are multiple methods for accomplishing these tasks, for example, Oracle®'s DBA Studio or Enterprise Manager versus using a "command line" tool, the actual commands are not specified in this manual. Instead, you are instructed to accomplish a task such as "create a role" and are given the necessary details for input to whatever tool you are using.

Conventions used in this manual

Input is illustrated in a fixed Courier font, for example:

```
Create table test_table (
Col_one varchar2(10));
```

Names of database objects, roles, storage and similar items are in italics.

Software Installation Overview

Introduction

There are three main parts to the FINANCIER™ application: the database, the "business logic", and the presentation layer. Conceptually, the database defines and stores the data, the business logic layer provides the data processing functions, and the presentation layer communicates with the user.

Most of the business logic in the FINANCIER™ application is implemented in Oracle®'s PL/SQL programming language and therefore resides in the database. Data display and input validation are the main application components that are outside of the database.

The Database

The database is Oracle®, with the minimum (earliest) requirement being version 9i. Of course, more recent versions of Oracle® are fine.

The Business Logic Layer

For the business logic layer, WolffPack uses Oracle®, PHP (the open source server scripting language), and Internet Explorer (for JavaScript-based field validation). The web portion of this may be hosted under Apache or Microsoft's IIS.

The Presentation Layer

The presentation layer covers the end-user interaction with the system. Primarily, that will be through the web browser, Internet Explorer version 6 or greater. In addition, you may want to use a reporting tool of your choice. Any reporting software that can access Oracle® data may be used.

We take advantage of absolute positioning and style sheet capabilities in the browser to provide an attractive and flexible presentation of forms. The validation of user input is done using JavaScript in the browser to minimize network traffic. In order to do this, we have had to limit our initial browser support to Internet Explorer.

Summary of Software Requirements

Here is a summary of the software requirements for FINANCIER™.

Database/application server software:

- Oracle® 9i or greater
- Apache web server software version 1.3 or greater **or** Internet Information Services (IIS) version 5.0 or greater. Note that Apache is bundled with the Oracle® database distribution, or it may be downloaded from http://www.apache.org/.
- PHP version 4.4 or greater, which may be downloaded from http://www.php.net/. At the present time we do not recommend using PHP version 5.

Desktop software:

- Internet Explorer version 6.0 or greater

Optional software:

- Reporting tools that access an Oracle® database

Structure of the Installation CD or FTP directory

The installation CD (or FTP directory) contains:

- an HTML version of this file
- a Microsoft Word™ version of this file
- a self-extracting ZIP file containing an Oracle® export (.dmp) file of the FINANCIER™ database and programs
- a self-extracting ZIP file containing source files for Windows
- sample php.ini and httpd.conf files
- a compressed tar file containing source files for Unix/Linux (optional, since the ZIP file contents are identical to this)
- an Excel spreadsheet that assists in sizing the database

Prerequisites for Installation of FINANCIER™

Before installing any software from our installation media, you should install an Oracle® database instance, either Apache or IIS as the web server, and PHP.

No matter which you choose to do first, you will likely have to adjust parameters and configuration information for it when you install the other components. We'll begin here with the web server installation.

Web Server Installation

FINANCIER™ may be run in an Apache web server environment or an IIS web server environment. It is not in the scope of this document to detail all the nuances of these environments. However, there are certain considerations in each of these environments that we will touch upon here.

Introduction

We will refer to these environments as "web servers". Some clients prefer the term "application server". Depending upon what is meant by "application server", this is more or less correct. FINANCIER™ has much of its business logic stored and executed in the Oracle® database. As such, it may be somewhat misleading to refer to the web server as an "application server" in the classic sense.

The chosen web server must be configured to support PHP, which is quite straightforward. Any web server **not** using the Oracle® Apache distribution must be configured to be able to communicate with the Oracle® database. Oracle®'s distribution of Apache already has the necessary configuration information in place for database communications.

Apache Web Server

Apache is bundled with the Oracle® database distribution. This version of Apache is generally used when the database and the application server reside on the same physical machine. The Apache distribution available from www.apache.net is also usable for our purposes, though it does not have knowledge of Oracle® built into its configuration file at initial installation time.

Oracle® 's distribution

The two biggest benefits of using the Oracle® Apache distribution are:

1. Apache is installed using the Oracle® installer, like the rest of Oracle®.
2. Apache, once installed, in pre-configured for Oracle® database access.

If you plan to use Oracle®'s Apache, simply make sure to install it when you install the Oracle® database software. For Oracle® 9 it is found on the database install media. For

Oracle® 10 it resides on the "companion" media. Later, we'll describe how to tell any Apache installation to recognize PHP scripts.

Note that the Oracle® 9i distribution may refer to the Apache listener service as "Oracle® http listener", or something to that effect, in the services window. For Oracle® 10g, Apache is part of the "PracessManager" service. Do not be alarmed that no service actually refers to "Apache" in its name in the services list.

Important Oracle® 10g note: As of Oracle® 10g, the Apache installation uses an "ORACLE_HOME" that is different from the one established when the database is installed. This complicates the database connection issue, as Apache (and by inference, PHP) will look in a different folder for the tnsnames.ora file than the rest of Oracle®'s database tools. So, be sure to add the appropriate tnsnames entries in both the network\admin folder rooted from the database's ORACLE_HOME and the companion (Apache) ORACLE_HOME. Failure to do so will usually result in a 12154 error from Oracle® after logging in to web FINANCIER™.

Standard distribution of Apache

If you are using a non-Oracle® distribution of Apache, just follow the instructions regarding its installation. After installation, adjust the httpd.conf configuration file as follows to enable Oracle® support. You will need to research the requirements via the Apache web site.

It is recommended that, if you choose to use Apache, that you use Oracle®'s distribution of it, which is bundled with Oracle®.

These examples assume you have installed PHP in a directory called c:\php.

Enabling PHP in Apache

No matter what distribution you used to install Apache, the changes to the httpd.conf file that allow PHP scripts to be recognized are the same. Note that WolffPack provides a sample httpd.conf for reference. Your httpd.conf will be different from this but the sample *should be reviewed* for insights as to necessary parameter settings.

For php4:

```
LoadModule php4_module              c:\windows\system\php4apache.DLL

AddModule mod_php4.c
```

For php 5:

```
LoadModule php5_module              c:\php\php5apache.DLL
```

```
AddModule mod_php5.c
```

Place these where similar statements are located in the httpd.conf file. There will usually be quite a few `LoadModule` and `AddModule` statements.

Note: PHP 5 changed the directory structure of php somewhat, and added more support for leaving the necessary DLLs in the php directories rather than moving them to a system level directory. Be sure to add the PHP main directory to your path statement. Make sure that you set the extension_dir parameter to point to where you have stored your extensions. Some installation notes recommend moving the extensions you do use up from the php/ext to the php directory. It is possible that simply setting the extension_dir in the php.ini file to the php/ext directory would work as well.

IIS Web Server

The IIS Web Server is bundled into many of the Microsoft Windows® OS environments. Depending upon your particular version of operating system software and/or the web applications you may have deployed, you may already have IIS installed.

Most adjustments to the IIS configuration are made in dialogs accessed from Control Panel/Admin Tools/Internet Services Manager.

Enabling PHP in IIS

Note: *It is possible that some of the dialogs and tabs have been renamed somewhat from version to version of IIS.*

These examples assume you have installed PHP in a directory called c:\php.

From the Internet Services Manager dialog, choose Properties, then Home Directories, then App Mapping.

For older PHP 4 versions, add:

.php c:\php\sapi\php4sapi.dll

For newer PHP 4 installations (that have the file php4isapi.dll), add:

.php c:\php\php4isapi.dll

For PHP 5 installations, add:

.php c:\php\php5sapi.dll

Alternatively, the dialog may be “Properties/Home Directory/Configuration (button)/App Mapping”. Obviously, the directory path will be dependent upon where php was installed.

If this produces an error when PHP has yet to be installed, do it again after PHP exists on the machine.

You should change the AppSettings execute permissions to “script and executables”.

Enabling Oracle® access from PHP in IIS

Place a copy of the php Oracle® dlls in your windows/system32 or equivalent directory. Alternatively, put your php directory in the system path variable. They are:

- php_oci8.dll (this file may disappear from more recent versions of Oracle®.)
- php_Oracle®.dll

Change the permissions on the Oracle® “bin” directory to allow the “anonymous” user to have read access. You might need to add the anonymous user at this time if it does not yet exist. The process for doing that should be fairly simple to figure out.

PHP initialization parameters

There are several PHP parameters in php.ini that are of particular note to WolffPack’s use of PHP. They may or may not be set to the values we need in the initial php.ini file that is included in the PHP download.

`magic_quotes_gpc = On` This ensures that input data that includes quotes and apostrophes will be handled correctly.

`register_globals = On` This helps in the storage of session variables for use throughout the application.

`session.save_path = "/tmp"` This should be set to a location where temporary files are kept. Session information for individuals who are logged into the application will be kept here. The “tmp” directory is just an example of what might be used here.

Database Installation

Oracle®

You should install Oracle® if you do not already have Oracle® on the server that will host the FINANCIER™ application. Whether you place the FINANCIER™ database in an existing Oracle® database instance or create a new Oracle® instance is your own choice. There are advantages and disadvantages to each approach, as listed below. In either case, you will want to have a test and a production instance.

Oracle® initialization parameters

As a base installation, we require almost no modification of Oracle®'s initialization parameters. Any exceptions to that rule are listed here.

job_queue_processes

Job_queue_processes may be zero by default. We will need a value here that corresponds to the maximum number of background jobs that should be active at one time. A value of 10 or 20 should be sufficient in most environments. Jobs will queue up behind the currently running ones if no open slot is available. Do keep in mind, however, that a few interactive processes submit background jobs to do tasks such as rebuilding triggers or constraints. It is best to have a parameter value large enough to handle all of the usual background job load.

Oracle® instance – new or shared?

Sharing an instance with other application systems

Advantages

- If your student system is in the same instance, accessing student data from within FINANCIER™ will be the least complex and will likely have better performance.
- Since each instance uses hardware resources, sharing may result in less overall hardware cost.

Disadvantages

- The performance of FINANCIER™ will be more directly affected when other applications within the same instance experience high volumes of processing.
- Tuning becomes more complex as more applications share an instance.

- Security can be more complex due to a larger overall population of users and their use of multiple applications.
- More applications are affected when an instance is unavailable due to maintenance or other issues.

Using a new instance

The advantages and disadvantages are reversed from the "sharing an instance" situation as described above.

Creating a role for necessary privileges

Roles are used within Oracle® to control the privileges granted to users. Users created in the Oracle® database are always assigned a role called 'connect'. The owner of the FINANCIER™ data and objects will need several additional privileges beyond those included in the pre-defined 'connect' role. No other pre-defined Oracle® role contains just these needed privileges, so you will create one.

You need to have the privileges necessary to create a role. An Oracle® user such as SYS or SYSTEM generally has such privileges.

The suggested new role name is *fin_resource*, because it describes additional resources and capabilities we need. The role *fin_resource* must contain these privileges:

- Create any directory
- Create indextype
- Create library
- Create operator
- Create procedure
- Create trigger
- Create type

Note: This role needs to be defined only once per database instance, no matter how you ultimately choose to support multiple versions (eg. test and production).

Preparing to install FINANCIER™ data and objects

Creating a tablespace

FINANCIER™ data and objects should be placed in a separate tablespace for easier control and maintenance of the data. A possible name for that tablespace might be *financier_ts*. The test data and the objects that comprise FINANCIER™ will occupy approximately 50 megabytes initially. The actual space you will need on a long-term basis in the tablespace depends primarily upon the size of your student population.

Sizing the tablespace

See the spreadsheet delivered with the product for sizing assistance.

Creating a user/owner

FINANCIER™ data and objects are designed to be owned by a single user/schema. As delivered, that username is *wpfinancier*. When we refer to *wpfinancier* in the following sections, we are referring to this user. If you use a different username as the owner, you will need to give that user additional privileges to allow the import to work because the import expects the importing user name to be the same as the exporting user name, which was *wpfinancier*. You might need to temporarily assign the "dba" privilege to the target user if you are importing to a user other than *wpfinancier*. Also, a small but important section of code within the application will need to be adjusted to know the owner's username. **Note:** the end users of the system will not need to know this username. They will log into the system under their own assigned usernames.

The steps for creating the Oracle® user who will own the FINANCIER™ data are as follows:

- Create the user *wpfinancier* that will own FINANCIER™'s data and objects.
- Assign the role *fin_resource* to this user.
- Grant the privileges *create any directory, create trigger* and *create procedure* to this user. Note that these privileges must be explicitly granted in addition to being included in the *fin_resource* role. The reason for this is Oracle® 's inability to use privileges from within a role under certain conditions, such as when triggers are regenerated within FINANCIER™ during the implementation process.
- Assign the user to the tablespace created above (typically, *financier_ts*) as their default tablespace. You can use *temp* as the temporary tablespace to be used. *Temp* is usually the default for such objects in a typical Oracle® installation.
- Give this user unlimited quotas on the *financier_ts* and *temp* tablespaces and zero quotas on the other tablespaces in the database instance. It might be handy to use Oracle®'s DBA Studio tool or Enterprise Manager to perform this task. Log in as this user and create symbolic directories that will be accessible from within PL/SQL. See the detailed description of creating symbolic directories below.

Schema owner versus logged in user

As delivered, FINANCIER™ expects that the owner of the schema will be the same as the user id of the "internal" log in within the application. As mentioned above, the name supplied upon delivery is *wpfinancier*.

It is possible to alter the system in either of two ways:

1. Import the schema to Oracle using a schema owner other than wpfinancier. If you do this, you will need to modify the /inc/inc_db_access.php file to reflect the correct log on information for that user.

2 . Have one schema own the data while another user id is used for logging in within the application. This is more complex because the owner and the logged in user will not be the same. In this case, synonyms will need to be created for every table, view and sequence in the system so that the "log in" user can reference these objects by name rather than schema.name. In addition, you must grant the appropriate access to all the schema's objects to the "log in" user. *This is not recommended unless you are experienced with the issues of synonyms and privileges*.

Installing FINANCIER™ data and objects

Overview

Whatever choice you make for sharing or creating new instances, the process of installing the FINANCIER™ data and associated objects into the Oracle® database is the same.

FINANCIER™'s database data and objects are delivered in an Oracle® Export "dmp" file produced by WolffPack and must be imported with the Oracle® Import utility. Prior to importing that file, you should have a tablespace and a user defined to Oracle® that will own the data, as described above.

In the sections that follow, there are many references to filenames that are accessible to the web server. These will use *"DocRoot"* to indicate that portion of the full filename that is the root for the web server. When you use the example statements, substitute your path in place of *"DocRoot"*.

For example, here as some possible paths for the web server document root:

- Apache from Oracle®: ***c:\Oracle\ora9i\Apache\Apache\htdocs***
- Apache on Unix/Linux*: **/usr/local/httpd/htdocs/financier***
- IIS web server: ***c:\Inetpub\wwwroot***

- A root you chose and specified in the configuration of the server: ***c:\myDocs***, for example

Creating symbolic directories

Oracle® provides a symbolic way to point to directory paths on the database server where PL/SQL will be able to read and write text files created by FINANCIER™.

We need three directories, created by the user (e.g. *wpfinancier*) that was defined in the previous section. The following examples show the required directory names and sample paths. These paths will have to be actual directories on your server, and these directories must be accessible to the username under which Oracle® was installed (usually, this user is 'Oracle®'). As mentioned above, you will substitute your own path in place of *"DocRoot"*. The commands are issued wherever you can execute ddl, for example, in SQL*Plus.

Windows example, using Apache or IIS:

- `Create directory fin_base_dir as 'DocRoot\financier';`
- `Create directory fin_data_dir as 'DocRoot\financier\vp\data';`
- `Create directory fin_output_dir as 'DocRoot\financier\vp\output';`

Unix/Linux example:

- `Create directory fin_base_dir as 'DocRoot/financier';`
- `Create directory fin_data_dir as 'DocRoot/financier/vp/data';`
- `Create directory fin_output_dir as 'DocRoot/financier/vp/output';`

Important note about symbolic directories

One common need for most FINANCIER™ installations is having a production and a test version available at the same time. This is accomplished in one of two ways:

1. Use a completely different database instance for each.
2. Share a database instance and have a separate schema (user) for each.

Both of these options are readily accomplished, but they have the same issues previously mentioned in the discussion of having a unique instance for FINANCIER™ separate from your student or other applications. In addition to those pros and cons, there is another important issue with the symbolic directories.

If you choose to have a separate database instance, just follow this installation guide for each one.

If you choose to have test and production in the same instance, you must do the following:

- Create another user in addition to *wpfinancier* (or whatever user you do the first install as). We'll call this user *wpfintest* in this example.
- Import the *financier_nnn.dmp* into user *wpfintest*.
- Create another set of symbolic directories while logged in as the other user, prefix them by something (such as *test_*), and point them to their own specific file location.
- Modify the *wplss_gbls* package body in the schema *wpfintest*, and supply the prefix in the field called *dir_prefix*.
- Compile *wplss_gbls* in the schema *wpfintest*.

For example, if we are creating a test user, we might choose *test_* as the prefix. Then our directory creation statements might look like this (again, replacing *DocRoot* with your path):

- `Create directory test_fin_base_dir as 'DocRoot\testfinancier';`
- `Create directory test_ fin_data_dir as 'DocRoot\testfinancier\vp\data';`
- `Create directory test_ fin_output_dir as 'DocRoot\testfinancier\vp\output';`

Note also that the actual directory paths must be unique to this set of directory statements. Again, remember that the *wpfintest* user must run the directory creation statements.

Why is all this necessary? Oracle®'s directory statements are not owned by the creating user's schema, but by SYS. So you must create unique directory names across the entire instance. However, even though they appear to be owned by SYS, they are only accessible to the user who created them, to the best of our knowledge.

Enabling E-Mail from FINANCIER™

FINANCIER™ uses Oracle's SMTP interface package to provide the means to send e-mail from within the application. For example, volume processes within FINANCIER™ can be instructed to e-mail one or more users when a job completes. In order to do this, messages must be routed through the appropriate mail server at your site.

The package *wplss_gbls* contains two constants, "default_mail_server" and "default_mail_domain" which must be set properly in order for e-mail messaging to succeed. The values for these two constants will be specific to your site and should be fairly obvious to system administrative personnel.

Once these values are set as required, re-compile the package *wplss_gbls*.

Creating the FINANCIER™ error object

We will need to manually create the FINANCIER™ error object and associated sequence number. Oracle® will attempt to automatically compile the delivered packages (see Importing the FINANCIER data and objects below) and we will need to have the error objects pre-defined to allow for the compilation to execute correctly.

Note that these steps are required for each defined user (e.g. *wpfinancier*, *wpfintest*). The following may be submitted to SQL*Plus to create these objects.

Step 1) Create sequence number:

```
CREATE SEQUENCE FIN_KEY_SEQ START WITH 1000000 INCREMENT BY 1;
COMMIT;
```

Step 2) Create error object:

```
CREATE OR REPLACE
Type         WPL_ERROR_THREAD
AS OBJECT
(
   err_count       integer,
   exec_ident      number,

  MEMBER PROCEDURE store_error
     ( p_rtn           IN OUT integer,
       p_wp_msgno      IN number,
       p_wp_module     IN varchar2,
       p_wp_module_loc IN varchar2,
       p_wp_rt_text    IN varchar2,
       p_db_table      IN varchar2,
       p_db_sqlcode    IN number,
       p_db_sqlerrm    IN varchar2)
); -- Type Specification WPL_ERROR_THREAD
/
CREATE OR REPLACE
Type Body WPL_ERROR_THREAD as

  MEMBER PROCEDURE store_error
     ( p_rtn           IN OUT integer,
       p_wp_msgno      IN number,
       p_wp_module     IN varchar2,
       p_wp_module_loc IN varchar2,
       p_wp_rt_text    IN varchar2,
       p_db_table      IN varchar2,
       p_db_sqlcode    IN number,
       p_db_sqlerrm    IN varchar2)
     is
     pragma autonomous_transaction;
     begin

       -- Null exec ident means this is our initial interaction with
this instance,
       -- and we must establish an exec ident for this instance of the
error object.
       if exec_ident is null then
```

```
            select fin_key_seq.nextval into exec_ident from dual;
            err_count := 0;
          end if;

          err_count := err_count + 1;
          insert into rt_msg
            (exec_ident,
             error_seq,
             wp_msgno,
             wp_module,
             wp_module_loc,
             wp_rt_text,
             db_table,
             db_sqlcode,
             db_sqlerrm,
             cdate) values
            (exec_ident,
             err_count,
             nvl(p_wp_msgno,0),   -- cannot be null
             p_wp_module,
             p_wp_module_loc,
             p_wp_rt_text,
             p_db_table,
             p_db_sqlcode,
             p_db_sqlerrm,
             sysdate);

          p_rtn := 0;
          commit;

        exception
           when others then
             p_rtn := sqlcode;
             rollback;  -- rollback the autonomous transaction
        end store_error;


END;        -- Type Body WPL_ERROR_THREAD
COMMIT;
/
```

Importing the FINANCIER™ data and objects

Use the Oracle® import utility to import the file named *financier_nnn.dmp* from the distribution media, where nnn is the release and version number, for example, financier_101.dmp would be for version 1.01 of Web FINANCIER™.

The standard distribution contains a set of support tables, such as a dictionary, and some sample test data.

Report import errors to WolffPack, except in the following case.

IMP-00041: Warning: object created with compilation warnings

"CREATE FORCE VIEW "WPFINANCIER"."BUDGET_COMP_SUMMARY"

This warning is expected and may be ignored. Note that the database will begin compiling source code packages at this time, so the display may appear to "freeze" for five to ten minutes. Unfortunately, no message is given by the import utility when compiling begins, so the lack of output on the display can be disconcerting.

Source code Installation

Overview

A large portion of the application code is stored in the database in PL/SQL objects called *packages*. The previous step of importing the file from the distribution media into the database includes the loading of these objects into the database. Thus, at this point in the installation process, much of the system's application code is already installed.

Additional code is written in PHP and JavaScript, and is found in the files webfin.tar.Z (Unix) or webfin.exe (Windows) on the distribution media. It is this source code that you will now install.

Installing PHP and JavaScript files

The PHP and JavaScript files are stored in subdirectories according to their functional application purpose. These subdirectories must be direct descendants of a parent folder that is accessible from the web server, be it Apache or IIS. That is commonly referring to as the document root, and has been described in more detail above. We refer to it as *"DocRoot"* in the examples.

The important thing is that the FINANCIER™ PHP and JavaScript files are intended to reside in a set of subdirectories directly beneath this *"DocRoot"* location, whatever it is. **Note:** You may need to adjust the permissions of the directory structure so that a normal user account can add files and directories to this *"DocRoot"*. When using these examples, substitute your actual document root path for *"DocRoot"*.

Example, Windows:

- *DocRoot\financier*
- *DocRoot\financier\award*
- *DocRoot\financier\application*
- and so on

Example, Unix/Linux:

- *DocRoot/financier*
- *DocRoot/financier/award*
- *DocRoot/financier/application*
- and so on

The installation CD includes both the Unix and Windows compressed files of the php source code of FINANCIER™.

For Windows:
Source is in *webfin.exe,* a self-extraction ZIP executable

1. Copy this file into the Apache or IIS root directory for FINANCIER™. Unzip the file from there, making sure that *"Use Folder Names"* is checked in the zip prompt window. This will ensure that the appropriate directory structure is created beneath the root.

For Unix/Linux (if you do not have a means to unzip the webfin.exe file):
Source is in *webfin.tar.Z*

1. Copy this file into the Apache root directory
2. `uncompress webfin.tar.Z` (this will produce the uncompressed tar file *webfin.tar*.)
3. `tar xvf webfin.tar financier` (this will create the files in the appropriate directory structure.

Post-installation Tasks

Overview

Most of the tasks that follow the installation of FINANCIER™ are part of the project implementation process and are documented there. However, there are some specific technical tasks involved in going live with the system. Those are addressed here.

Site-specific constants

The package *wplss_gbls* contains a set of constants that will need to be set to appropriate values for your site. The following is an excerpt of code that illustrates the site-specific items.

```
-- Default mail server address
default_mail_server CONSTANT varchar2(200) := '161.58.169.86';

-- Default mail domain
default_mail_domain CONSTANT varchar2(200) := 'wolffpack.com';
```

```
-- The following constant must contain the correct prefix for the Oracle
-- directory statements that point Oracle to the proper location for files to
-- be read and written by PL/SQL programs in this schema.
--
-- This allows for multiple FINANCIER versions in the same instance, in
-- different schemas.
-- Note that this is necessary because the DIRECTORY specification in Oracle is
-- system-wide rather than just schema-wide.

dir_prefix  CONSTANT varchar2(50) := null;

schema_owner CONSTANT varchar2(50) := 'WOLFFPACKDEVL';
```

Change these to appropriate values for your installation and re-compile the *wplss_gbls* package and package body.

Preparing the delivered data for your own use

The standard distribution contains a set of support tables, such as a dictionary, and some sample test data. You will ultimately discard the sample test data and modify and add to the support tables in the processing of implementing the system.

The project implementation process teaches you how to modify and add to the dictionary and other support tables. We will not go into that detail here.

Once you have taken a look around the supplied test data and are confident that the system is properly installed, you can remove the test data. In other words, you are "cleaning out" the database instance to prepare it for your own data.

WolffPack provides scripts to clean out the test tables. In general, these scripts will completely empty the affected tables, making them ready for your data. These scripts will not affect the dictionary and support tables that you set up in the implementation process. Be sure to follow the project guidelines for the exact appropriate time for this "clean out" to occur.

See data_cleanup_golive_prep.doc for details.