

Oracle® Forms Developer

WebUtil User's Guide

10g (9.0.4) for Windows and UNIX

July 2004

Copyright © 2003, 2004 Oracle. All rights reserved.

Primary Author: Duncan Mills, Grant Ronald, Orlando Cordero

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Contents

1 Introduction

1.1	What is WebUtil?	1-1
1.2	The Principles of the WebUtil Design.....	1-2
1.3	The Technology behind WebUtil.....	1-2

2 Installing WebUtil

2.1	WebUtil Prerequisites	2-1
2.2	The WebUtil Components and Directory Structure	2-2
2.2.1	The Webutil ZIP/TAR File	2-2
2.2.2	Design Time Components	2-2
2.2.3	Run Time Components	2-3
2.3	Configuring your Database for WebUtil	2-3
2.4	Configuring your HTTP Server for WebUtil	2-4
2.5	WebUtil Entries in the formsweb.cfg File.....	2-4
2.6	WebUtil Entries in the Environment File	2-5
2.7	The webutil.cfg File	2-6
2.8	The webutil.properties File.....	2-6

3 WebUtil Configuration Entries in Detail

3.1	The formsweb.cfg File	3-1
3.1.1	Required Configuration in the formsweb.cfg File	3-1
3.1.2	Optional WebUtil Parameters in formsweb.cfg	3-2
3.2	The webutil.cfg File	3-3
3.2.1	Logging Options	3-3
3.2.2	Installation Options	3-3
3.3	File Upload and Download Options.....	3-6
3.3.1	Enabling Transfer.....	3-6
3.4	Access Control	3-6
3.4.1	Defining Read and Write Directives	3-6
3.4.2	The Work Area	3-7
3.5	Translatable Strings	3-7
3.6	Untranslatable Strings.....	3-7

4 Using WebUtil in Your Applications

4.1	Adding The Required Objects	4-1
4.2	Seeing the PJC at Design Time	4-2
4.3	Adding WEBUTIL Code	4-2
4.4	Using OLE Commands	4-3

5 Functions in WebUtil

5.1	Client Server Parity APIs	5-1
5.1.1	Running HOST Commands	5-2
5.2	Ported D2KWUTIL Functions	5-2
5.3	WebUtil Public Functions	5-3
5.3.1	WebUtil_ClientInfo	5-4
5.3.2	WebUtil_C_API.....	5-4
5.3.3	WebUtil_File	5-4
5.3.4	The File transfer package.....	5-6
5.4	WebUtil_Host	5-8
5.4.1	WebUtil_Session	5-9
5.4.2	WebUtil_Browser.....	5-10
5.4.3	WebUtil_SeparateFrame	5-11
5.4.4	WebUtil_Core.....	5-11
5.5	Utility Functions.....	5-12
5.5.1	DelimStr	5-12
5.5.2	Show_WebUtil_Information	5-12
5.5.3	WebUtil_Util.....	5-12
5.5.4	Internal APIs.....	5-12

6 Tracing and Diagnostics

6.1	Connection Logging	6-1
6.2	Error Handling	6-1
6.3	Diagnostic Tracing	6-2

7 Performance Tuning Considerations

7.1	Optimizing your use of WebUtil	7-1
-----	--------------------------------------	-----

8 Customizing WebUtil

8.1	Customizing Client-Side Error Messages.....	8-1
8.2	Customizing Server-Side Error Messages	8-1

A Appendix A – Runtime Install Checklist

B Appendix B – WebUtil Error Reference

Introduction

This manual documents the principals and usage of the WebUtil utility with Oracle Forms Applications. You should read and understand the sections on installing and configuring WebUtil before using the utility.

This chapter contains the following sections:

- [What is WebUtil?](#)
- [The Principles of the WebUtil Design](#)
- [The Technology behind WebUtil](#)

1.1 What is WebUtil?

WebUtil is a pre-packaged set of components that can be used to add a great deal of extra functionality to Web-deployed Forms applications. WebUtil is primarily focused at solving the more common challenges faced by Forms developers who wish to build applications which integrate tightly with the “Client Browser” machine - the machine that the end user is actually sitting at.

Traditionally Forms has provided the means of integration with the machine that the Forms executable is running on, but in the Web-deployed scenario this can be remote from end users and may even be a different operating system from the systems that are being used to host the application’s user interface. WebUtil gives Forms developers the means to interact with the client browser machines with PL/SQL, without having to learn any new technology. As such, WebUtil is particularly suited to programmers who are migrating client server applications from Microsoft Windows desktops to the Web, but still need some integration between their Forms applications and external packages such as the Microsoft Office Suite running on the client browser machines.

1.2 The Principles of the WebUtil Design

The aim of WebUtil is to provide a utility that any Forms developer can use to carry out complex tasks on the client browser machine by simply coding in PL/SQL. Although WebUtil itself uses Java extensively, there is no need for the developer to have any understanding of this aspect of the utility. Everything the developer needs to do is exposed through a standard PL/SQL library `webutil.pll`.

All of the components that are needed to support WebUtil are likewise exposed through a simple object group.

For a Forms application to take advantage of WebUtil you only need to attach the WebUtil library and subclass the Object Group. No setup or configuration in code is required as WebUtil is self-configuring and where necessary, self-installing.

Some features of WebUtil, such as client side OLE integration, require code to be installed onto each client machine. One of the key features of WebUtil is its ability to “self-install” in these cases. There is no need for an administrator to configure each and every machine because WebUtil detects which client-side pieces are missing and automatically installs them.

Another aim of WebUtil is to simplify porting of client server integration code to work on the client browser machine. To achieve this, WebUtil contains PL/SQL APIs mimic the behavior of the equivalent native Forms functions.

An example of this is the Forms `TEXT_IO` package. This package provides the developer with facilities to read and write text files on the application server. WebUtil provides an alternative implementation, `CLIENT_TEXT_IO`, which is identical to `TEXT_IO` except that calls are implemented on the client browser machine rather than on the middle tier. Migrating existing code to operate on the client can be achieved using a simple search and replace operation in the PL/SQL code to changes references to point at the alternative “CLIENT” implementations provided by WebUtil.

The final goal of WebUtil design is to add value to Oracle Forms as a product. WebUtil adds capabilities to Forms which have simply not existed before in either web or client server deployments.

1.3 The Technology behind WebUtil

WebUtil uses the extensibility features of Oracle Forms, both in the Java client using the Pluggable Java Component (PJC) mechanism, and on the application server using the Forms interfaces to Java. Nothing in WebUtil is secret; it uses published APIs and documented techniques to provide all of its functionality. Of course one of the key benefits of the utility is that it then wraps up those complex activities inside a series of simple PL/SQL APIs.

WebUtil is designed to be extended, to add further functionality, as it is required. Much basic infrastructure, such as error diagnosis and self-installation, is provided as part of the WebUtil kernel, and customers can use these APIs to expand WebUtil with additional functionality as requirements dictate.

Installing WebUtil

This chapter describes the basics of setting up WebUtil for use by Forms. [Appendix A](#) also features a checklist that you can use to take you through the installation and setup process.

Before you begin installing WebUtil, read the *Webutil Release Notes* document that is part of this distribution. You can also read the *Webutil Release Notes* at <http://otn.oracle.com/products/forms>.

This chapter contains the following sections:

- [WebUtil Prerequisites](#)
- [The WebUtil Components and Directory Structure](#)
- [Configuring your Database for WebUtil](#)
- [Configuring your HTTP Server for WebUtil](#)
- [WebUtil Entries in the formsweb.cfg File](#)
- [WebUtil Entries in the Environment File](#)
- [The webutil.cfg File](#)

2.1 WebUtil Prerequisites

In order to use WebUtil there are several version prerequisites:

1. Forms Version: WebUtil is designed to run with Oracle Forms 10g (9.0.4). No testing has been done with previous versions.
2. Client Java Virtual Machine (JVM): WebUtil requires either JInitiator 1.3.1.9 and higher, or the Sun Java Plug-in Version 1.4.1.01 or higher. WebUtil uses certain Java 1.3 features that are not available in earlier versions of the JVM.

3. Application Server Java Virtual Machine – For its Java operations on the middle tier WebUtil needs a JVM of version 1.2.2 or higher. The JVMs supplied with Oracle Developer Suite 10g and Oracle Application Server 10g are ideal.
4. Database Version – The file transfer facilities within WebUtil that post and get files from the database need to be installed into a 9.0.1 database or higher. Using WebUtil against an older database will mean that this type of file transfer is not available to you.

2.2 The WebUtil Components and Directory Structure

The following sections contain:

- [The Webutil ZIP/TAR File](#)
- [Design Time Components](#)
- [Run Time Components](#)

2.2.1 The Webutil ZIP/TAR File

When unpacked onto disk, the WebUtil archives will create a top-level “webutil” directory with the following sub-directories

- doc
- forms
- lib
- server
- src
- util

The /doc directory contains this manual and *WebUtil Release Notes*.

The /forms directory contains the webutil.pll file which defines the WebUtil APIs. The webutil.olb file supplies the required components for adding WebUtil to your forms. The create_webutil_db.sql creates the necessary database objects.

The /lib directory contains the compiled WebUtil Java classes in the Jar file webutil.jar and several operating system specific libraries that will be downloaded to clients on demand.

The /server directory contains the main WebUtil configuration file webutil.cfg plus sample templates for formsweb.cfg entries and template HTML files

The /src directory contains all of the WebUtil Java source code, if you are using a distribution supplied with source code.

The /util directory contains scripts to help you sign Jar files that provide functionality to WebUtil. For more information on how to use these scripts, see the *WebUtil Release Notes* that are part of the WebUtil distribution, or at <http://otn.oracle.com/products/forms/>.

2.2.2 Design Time Components

For design time use WebUtil consists of just two modules: the webutil.pll PL/SQL library and the webutil.olb object library. The PLL contains all of the PL/SQL APIs required to call WebUtil and the OLB contains the definitions of the Bean Areas that

the WebUtil subcomponents use. Both of these files exist in the webutil/forms subdirectory.

Also for design time use you may optionally include the webutil.jar file in your FORMS90_BUILDER_CLASSPATH environment variable/Registry entry. Adding this file to the CLASSPATH will prevent a warning message from appearing if you open the WEBUTIL_CANVAS in the layout editor and it will also allow you to view the version numbers of each of the WebUtil subcomponents.

2.2.3 Run Time Components

At runtime the primary components used for WebUtil are the webutil.jar file and webutil.pll. Some features of WebUtil such as OLE integration also require extra operating system libraries, which will be downloaded to the client on demand such as the jacob.dll file for OLE integration. These operating system-specific files are then stored in the webutil/lib directory.

JACOB source code and licensing details can be obtained at <http://danadler.com/jacob/>.

Also, see the *Webutil Release Notes* document that is part of WebUtil distribution. You can also read the *Webutil Release Notes* at <http://otn.oracle.com/products/forms/>. The *Webutil Release Notes* contains information about downloading and signing JACOB libraries.

Note: To confirm the ...webutil/lib directory has been mapped successfully with the HTTP Server, do the following:

1. Place a simple .gif file in your ../webutil/lib directory. Say it is called hello.gif.
 2. Enter <http://machine.domain:port/forms90/webutil/hello.gif>. (port is your http server port)
 3. hello.gif should display in the browser. If it doesn't either of the following is most likely the cause:
 - You need to stop and start the HTTP server after the modifications.
 - There is a typo in either the orion-web.xml or forms90.conf files
-
-

2.3 Configuring your Database for WebUtil

Some of the functions that WebUtil provides for file transfer in the WEBUTIL_FILE_TRANSFER package require the database package WEBUTIL_DB. The script called create_webutil_db.sql is supplied to create this package. This script uses the following database packages, which must be available to the schema that is installing the WEBUTIL_DB package:

- DBMS_LOB
- UTL_ENCODE
- UTL_RAW

Oracle recommends that you create a special user “webutil” on your database as the owner of this package.

Once the WEBUTIL_DB package has been created and compiled on the database you will need to make it available to the relevant users of your application. Additionally, you will have to create a synonym of WEBUTIL_DB for this package if you are

running from a schema other than the one that you installed it with. The Forms PL/SQL code refers to the package as WEBUTIL_DB without a schema prefix.

The WEBUTIL_DB package is executed with invokers' rights so that the package can refer to schema objects that the currently connected user can see, and the owner of the WEBUTIL_DB package itself does not have to be granted access to every possible table that contains a BLOB column to upload to.

2.4 Configuring your HTTP Server for WebUtil

WebUtil needs a single virtual directory to be defined in order to operate. This virtual directory is used to locate both the WebUtilJar file at runtime as well as the various operating system-specific files that WebUtil may need to download. You will need to create a virtual directory called

j2ee/DevSuite/application-deployments/forms/forms90web/webutil, which maps onto the webutil/lib directory in the WebUtil directory structure.

For an Oracle Developer Suite installation, add the following line to the file

j2ee/DevSuite/application-deployments/forms/forms90web/orion-web.xml under the Oracle Developer Suite Oracle Home:

(The following assumes that you have installed WebUtil into a directory c:\webutil)

```
<virtual-directory virtual-path="/webutil"
real-path="c:/webutil/lib" />
```

On a OracleAS install, define the same virtual directory in the forms90/server/forms90.conf file:

```
AliasMatch ^/forms90/webutil/(.*) "c:/webutil/lib/$1"
```

2.5 WebUtil Entries in the formsweb.cfg File

To use your applications with WebUtil you will need to create some specific settings in your formsweb.cfg file. These may be applied to a specific profile within the formsweb.cfg or may be defined for all profiles. A full list of valid WebUtil parameters is supplied in [Chapter 3, "WebUtil Configuration Entries in Detail"](#).

1. Define the name and location of the of the WebUtil Jar file.

At runtime the server must have access to the webutil.jar. This should be present in the /forms90/webutil directory. You define the name and location of this jar file using the config parameter webUtilArchive:

```
webUtilArchive=/forms90/webutil/webutil.jar
```

2. Define the HTML templates to use with JInitiator and the Java Plug-in

WebUtil uses its own HTML template files, which contain all of the required parameters that can be sent to WebUtil. There are three such templates, supplied in the /webutil/server directory. Any formsweb.cfg profile that uses WebUtil will need to use these templates rather than the supplied base.htm, basejini.htm and basejpi.htm.

Note: If you currently use a customized template to launch your applications, you can copy the WebUtil templates and customize them to your requirements.

Set the following parameters in your configuration to direct the Servlet to use the WebUtil specific templates (amend the physical location as required, or copy the templates into your forms90/server directory, in which case you will need no path, just the template name):

- baseHTMLjinitiator=c:\webutil\server\webutiljini.htm
- baseHTMLjpi - Should point to the baseHTML file for the Java Plug-in e.g. baseHTMLjpi=c:\webutil\server\webutiljpi.htm.
- baseHTML - Should point to the baseHTML for WebUtil, e.g. baseHTML=c:\webutil\server\webutilbase.htm.

Note: For Oracle Application Server on a UNIX server, you must regenerate webutil.pll before using it; otherwise you'll encounter ORA-06508 errors when running a form with the attached library. To generate, use the following command:

```
f90genm webutil.pll <webutil/webutil@dbconnect> Module_Type=LIBRARY
Compile_All=YES
```

User name can be any user with the necessary privileges as described in ["Configuring your Database for WebUtil"](#)

2.6 WebUtil Entries in the Environment File

In order to work, WebUtil requires several changes to your environment file (usually default.env stored in your forms90/server directory). You can of course create a separate .env file especially for WebUtil applications (a sample "webutil.env" file for Win32 servers is supplied in the webutil/server directory).

1. The FORMS90_PATH must include the directory where a copy of webutil.pll exists. You can either explicitly add the webutil/forms directory to this path, or you can of course copy the webutil.pll file to an existing directory that is mapped into the FORMS90_PATH or the ORACLE_PATH.

Note: Ensure that the total length of your FORMS90_PATH string does not exceed 255 characters. This can cause problems on Windows based installs. If necessary, use ORACLE_PATH as an additional variable to define extra long search paths.

2. The second requirement within the .env file is for an environment variable WEBUTIL_CONFIG. This variable must point at the name and location of your webutil.cfg file. By default this will be in the /webutil/server directory e.g.

```
WEBUTIL_CONFIG=c:\webutil\server\webutil.cfg
```

3. The webutil.jar (including its physical location) file must be included in the CLASSPATH, along with the Java runtime Jar rt.jar. e.g.

```
CLASSPATH=c:\webutil\lib\webutil.jar;c:\ids9i\jdk\jre\lib\rt.jar
```

Note: Some of the file transfer functions of WebUtil use Java integration on the application server. The steps concerning Java set-up in the .env file are not required if this functionality will not be used.

4. The PATH and library load path (on UNIX platforms) must be defined either in the general environment or specifically in the .env file to allow Forms to call Java. On a Win32 platform this will involve adding the <DevSuiteHome>\jdk\jre\bin\client directory of a 1.3 or 1.4 Java install to the PATH. This enables Forms to find the jvm.dll file. A suitable Java home will already exist under the Developer Suite or OracleAS ORACLE_HOME directory structure.

On Unix platforms, Forms needs to locate the libjava.so file so the LD_LIBRARY_PATH on most UNIX platforms including Linux, or SHLIB_PATH on HP/UX or LIBPATH on AIX, will need to contain the directory that holds this file. For instance on Linux this would be \$JAVA_HOME/jre/lib/i386/native_threads.

Again the Java runtime install that is part of OracleAS or OracleDS is ideal

2.7 The webutil.cfg File

The webutil.cfg file is the master configuration file for WebUtil. It allows the administrator to control features such as logging and file transfer permissions. The basic webutil.cfg supplied is sufficient for initial needs but if you want to enable file transfer functionality or logging then you will have to edit this file. The settings within this file are discussed in detail in the next chapter.

Note: As a security precaution, the default settings in the webutil.cfg disable the WebUtil file transfer features. To use file transfer to or from the database or the application server you must explicitly enable these features and define the access control parameters as required.

2.8 The webutil.properties File

When libraries are downloaded, WebUtil creates the webutil.properties file which is located in the Java home directory. If this file exists, downloading does not take place.

WebUtil Configuration Entries in Detail

This chapter describes the configuration of WebUtil in detail. Two configuration files are important in the configuration of the utility:

- [The formsweb.cfg File](#)
- [The webutil.cfg File](#)
- [File Upload and Download Options](#)
- [The Work Area](#)

3.1 The formsweb.cfg File

You're already using the formsweb.cfg file to configure your Forms applications. WebUtil has some requirements for existing settings within the configuration that you are defining, and some optional new parameters, which can be added to the file, to customize certain aspects of WebUtil behavior.

You can look at a sample_formsweb_cfg_entry.txt in the /server directory to configure formsweb.cfg.

3.1.1 Required Configuration in the formsweb.cfg File

- WebUtilArchive - Location and name of the webutil.jar file If you have followed the installation instructions this parameter should look like:
webUtilArchive=/forms90/webutil/webutil.jar
- EnvFile - If you use a named environment file or the default.env ensure that the FORMS90_PATH that you have set includes the webutil/forms directory and that you have set the WEBUTIL_CONFIG variable to define the physical location of your webutil.cfg file

- BaseHTMLjinitiator - Should point at the WebUtil JInitiator template file, e.g. BaseHTMLJinitiator= c:\webutil\server\webutiljini.htm
- baseHTMLjpi - Should point to the baseHTML file for the Java Plug-in, e.g. baseHTMLjpi=c:\webutil\server\webutiljpi. baseHTMLjpi is used when JInitiator cannot be used, such as with non-Windows platforms.
- baseHTML - Should point to the baseHTML for WebUtil, e.g. baseHTML=c:\webutil\server\webutilbase.htm. baseHTML is for running WebUtil using Internet Explorer's native JVM.

3.1.2 Optional WebUtil Parameters in formsweb.cfg

The following parameters can be set to configure WebUtil behavior but are not required. They can all be passed on the browser URL as well to change the behavior without changing the configuration entry.

- WebUtilLogging - This parameter can be used to switch on logging in WebUtil. It can take one of the following values (case is not important):

Value	Description
Off	WebUtil does no logging. This is the default value for this parameter
Console	Log messages are written to the Java console on the browser machine
Server	Log messages are written to the WebUtil log file on the server
All	Log Messages are written to both the Java Console and the Server log file

- WebUtilLoggingDetail - Defines the level of logging used if logging has been enabled. There are only two levels of detail: **Normal** and **Detailed**. Normal is the default setting.
- WebUtilErrorMode - This parameter can be used to control how errors are displayed. Errors will always be displayed on the Java console, additionally you can specify one of the following for the WebUtilErrorMode parameter (case is not important):

Value	Description
Console	Error messages are written to the Java console on the browser machine. This is the default value for this parameter (there is no way to completely switch errors off)
Server	Error messages are written to the WebUtil log file on the server
Alert	Error messages pop up in a dialog as well as the console - This is a good mode to use during testing
All	Error messages appear in an alert and are written to the console and the Server log file

- `WebUtilDispatchMonitorInterval` - Used with the `WebUtil_Session` package to control how often the monitor thread checks to see if the Forms session is still alive. The value is measured in seconds and the default value is 5 seconds.
- `WebUtilMaxTransferSize` - When transferring files using WebUtil the file is transferred in segments, the size of which defaults to 16k. You can change the size of the transfer segments using this parameter to define the number of bytes required. Note that 16k is the maximum value accepted.

3.2 The webutil.cfg File

The `webutil.cfg` file is the key configuration file that controls how WebUtil works. The file is structured as a conventional Java properties file and is divided into two sections:

- [Logging Options](#)
- [Installation Options](#)

3.2.1 Logging Options

Because logging from WebUtil can be requested on the URL, `webutil.cfg` provides a way to ultimately control logging through the utility. This prevents “Denial of Service” type attacks attempting to disrupt the server by swamping the logging mechanism. By default all logging is disabled. The following parameters exist in the `webutil.cfg` to control logging:

Parameter	Description
<code>logging.file</code>	Defines the name and location of the log file to use. The file should be writable. WebUtil does no automatic log file maintenance and the log will continue to grow until manually deleted or truncated.
<code>logging.enabled</code>	Master switch that tells WebUtil to allow logging. Can be set to TRUE or FALSE
<code>logging.erroronly</code>	Tells WebUtil to only log error messages not diagnostic messages. Can be set to TRUE or FALSE (Note: Errors will only be logged if the <code>WebUtilError</code> mode in the <code>formsweb.cfg</code> is set to <code>SERVER</code> or <code>ALL</code> to send the errors to the server in the first place)
<code>logging.connections</code>	This switch is separate from the main <code>logging.enabled</code> switch, and instructs WebUtil to keep track of all Forms sessions that use WebUtil. This log record includes the connecting machine IP address, the operating system information of the client and the running module.

3.2.2 Installation Options

One of the key features of WebUtil is its ability to auto-install any client side libraries that it requires. This feature is used by WebUtil internally for its own requirements, but can also be used by the WebUtil administrator to install application specific libraries as well.

The key setting in this section is `install.syslib.location`, which points to the virtual directory that contains the various WebUtil libraries which you’ve defined as

/webutil. This path needs to either be an absolute URL of the machine or the virtual directory mapped to the webutil/lib directory (e.g. `http://apps-1/forms90/webutil`), or a virtual directory relative to the documentbase `<server>/forms90/` (e.g. `/webutil`).

Libraries to be downloaded are then listed in the entries with the Format `syslib.<os>.<package>.<n>` where:

Code	Operating System
0	Windows 32 bit (XP, Win2000 etc.)
1	Linux
2	Solaris
3	HP
4	AIX
5	Mac

- `<os>` is a number representing the client operating system family that this library is destined for. These are the applicable codes
- `<package>` is an identifier representing the WebUtil component that uses this library. To add your own library to the download list you should use “user” for this value.
- `<n>` a sequence number starting at 1 and incrementing by 1 – there must be no gaps in this sequence for a particular platform and package combination.

The library definition string itself consists of four segments delimited by a vertical bar. The format is `name|size|version|showDownloadDialog`:

- **Name** – the name of the library file including its extension e.g. `jacob.dll`, `cfunc.so` etc.
- **Size** – the size in bytes of the library. This is used to checksum the download and must be correct
- **Version** – a version string, which WebUtil checks to see if it has already downloaded this library. If this version number is changed the new library will be automatically downloaded. (This is simply based on a string comparison of the version number not a numerical comparison of the version level)
- **ShowDownloadDialog** – Whilst auto-installing WebUtil can be asked to display a progress bar `TRUE` switches this on, `FALSE` suppresses it.

As an example for an application that uses the default WebUtil settings and has an additional two custom Windows DLLs to download this section might look like:

```
install.syslib.location=/webutil
## Change 2nd value to correct file size to avoid raising an error
install.syslib.0.7.1=jacob.dll|94208|1.0|true
install.syslib.0.9.1=JNIsharedstubs.dll|65582|1.0|true
install.syslib.0.9.2=d2kwut60.dll|192512|1.0|true
#Application custom libraries
install.syslib.0.user.1=scanner.dll|23056|1.0|true
install.syslib.0.user.2=mortgagecalc.dll|100230|1.0|true
```

3.3 File Upload and Download Options

This section contains the following:

- [Enabling Transfer](#)
- [Access Control](#)

As in the case of logging you may want to control access to the file transfer functions offered by WebUtil. To accomplish this WebUtil provides switches to Enable file transfer to and from the database and to and from the application server disk. As a further refinement the administrator can define lists of directory roots on the application server that are readable and writable. Applications will only be allowed to read and write from these directories and an error will be raised at runtime if access elsewhere is requested.

3.3.1 Enabling Transfer

Transfer is controlled by two master switches:

- `transfer.database.enabled=TRUE | FALSE`
- `transfer.appsrv.enabled=TRUE | FALSE`

These control access to the database and application server respectively. Additionally transfer to the database will only work if you are running against an Oracle 9.0.1 server or higher.

3.4 Access Control

This section contains the following:

- [Defining Read and Write Directives](#)
- [The Work Area](#)

3.4.1 Defining Read and Write Directives

In the case of file transfer to and from the database, the database package used (WEBUTIL_DB) runs with Invokers rights. This means that the code can only upload to database records when the running Form can see that database schema.

In the case of access to the application server disk, there is an additional switch `transfer.appsrv.accessControl` to define if the application can read from anywhere on the application server or from a controlled set of directories. If this switch is set to `TRUE` then WebUtil will use the directories listed in the `transfer.appsrv.read.<n>` entries and the `transfer.appsrv.write.<n>` entries. In both cases the sequence of values must start with 1 (e.g. `transfer.appsrv.read.1=d:\temp`) and be contiguous.

Access control is based on simple string matching, so you should make your specifications as precise as possible, e.g. an entry of `"transfer.appsrv.read.1=d:\"` would allow WebUtil to read any file on the D disk in any subdirectory. An entry of `d:\temp` will restrict it to any file in the `d:\temp` directory or any subdirectory.

You can define up to 50 read and write locations.

So for example the directives required to allow WebUtil applications to read any directory on the server D drive but only write to the `d:\work\upload` and `d:\work\log` directories would be:

- `transfer.appsrv.enabled=true`
- `transfer.appsrv.accessControl=true`
- `transfer.appsrv.read.1=d:\`
- `transfer.appsrv.write.1=d:\work\upload`
- `transfer.appsrv.write.2=d:\work\log`

3.4.2 The Work Area

WebUtil also allows the administrator to define a work area root using the parameter `transfer.appsrv.workAreaRoot`. The work area is a temporary directory on the application server which it is safe for applications to read and write. WebUtil provides an API (`WebUtil_File_Transfer.GetWorkArea()`) which will return a private directory for each connected user under this work area root. This private directory allows multiple users to upload files with the same name to the application server without overwriting files belonging to other users. Each user has a private directory under the work area, which WebUtil can create for them automatically if required.

The `Client_Image` package used for client side `Read_Image_File` and `Write_Image_File` need the work area root to be defined.

The work area is always assumed to be readable and writable and need not be explicitly nominated in the `transfer.appsrv.read` and `write` lists.

3.5 Translatable Strings

The strings in this section of the configuration file can exist in multiple languages. There is an English (EN). version of the message (e.g. `EN.msg.errordialog.title=WebUtil Error`), and optionally versions in other languages prefixed with the correct language code. See [Chapter 8, "Customizing WebUtil"](#) for more information about providing message translations.

3.6 Untranslatable Strings

All message strings within WebUtil are either contained in a Java MessageBundle in the `webutil.jar` file, or in the `webutil.cfg`. During startup, WebUtil detects the browser language settings and if any errors are raised, WebUtil attempts to use the correct language to display the error.

However, some messages may be raised before the browser language is known and so may not be auto-translated. See [Chapter 8.1, "Customizing Client-Side Error Messages"](#) for more information about providing message translations.

Using WebUtil in Your Applications

The following sections contain information about using WebUtil in your Forms applications:

- [Adding The Required Objects](#)
- [Seeing the PJC at Design Time](#)
- [Adding WEBUTIL Code](#)

4.1 Adding The Required Objects

WebUtil has been made as simple as possible to use within your applications. For each form that needs to use WebUtil functions, two actions need to be carried out.

1. Attach the `webutil.pll` library. This library contains all of the APIs available through WebUtil and is the only way of calling the WebUtil functions. Attempting to call the WebUtil PJC's directly will result in errors.
2. Subclass the WebUtil object group. `webutil.olb` contains a WebUtil object group, which you can simply drag into your Form. This object group contains all of the PJC's that implement the WebUtil functionality, along with a Canvas and Window to hold them and the alert used for popup errors. Normally the WebUtil window is hidden and so its presence will not interfere with your user interface.

Note: The order of these operations is important. If you subclass the object group before attaching the library then the triggers within the object group will be marked as uncompiled and WebUtil calls will fail with a WUC-15 error unless you do a Recompile All (Control-shift-K) on your module.

Once the WebUtil objects have been attached to your Forms you can start to use the APIs exposed by the PLL and are detailed in [Chapter 5, "Functions in WebUtil"](#). There is no need to carry out any initialization steps in your code, as the utility automatically initializes itself.

Note that the WebUtil block that contains the PJC is normally hidden but if your application navigates using the `NEXT_BLOCK` and `PREVIOUS_BLOCK` Built-ins, your users may inadvertently navigate to the block. You should check your navigation logic to ensure that this does not happen.

The WebUtil block itself also serves as an “About Screen” for the utility where it lists the version number of the PLL and the versions of each PJC that is used to implement the various functions. To view this information at runtime, call the `SHOW_WEBUTIL_INFORMATION` function.

4.2 Seeing the PJC at Design Time

At design time there is no need to interact with or change the properties of any of the objects in the WebUtil object group. However, you can view the WebUtil canvas to view the PJC versions that are in use. By default, bringing up the WebUtil canvas in the Layout Editor will result in errors because Forms cannot find the `webutil.jar` file to instantiate the beans. To make this jar file available you should edit your `FORMS90_BUILDER_CLASSPATH` in the environment or in the registry and include the physical location of the `webutil.jar` file e.g. `c:\webutil\lib\webutil.jar` and the physical location of `f90all.jar`.

Note: The total length of the `FORMS90_BUILDER_CLASSPATH` cannot exceed 512 characters.

4.3 Adding WEBUTIL Code

Once the WebUtil library has been attached to your form you can start to add calls to the various PL/SQL APIs defined by the utility. However, there is an important restriction in the use of WebUtil functions: WebUtil can only start to communicate with the client once the Form has instantiated the WebUtil PJC. This means that you cannot call WebUtil functions before the Forms UI is rendered. This would include triggers such as `PRE-FORM`, `WHEN-NEW-FORM-INSTANCE` and `WHEN-NEW-BLOCK-INSTANCE` for the first block in the Form.

Likewise you cannot call functions after the UI has been destroyed for instance in `POST-FORM`.

If you do need to use WebUtil at Forms startup, it is recommended that you create a timer with a short duration (say 1ms) in your `When-New-Form-Instance` trigger, then make the WebUtil call in a `When-Timer-Expired` trigger.

4.4 Using OLE Commands

Some OLE commands work fine in a client/server environment, as well as in Web deployed applications (on the server), but when you use WebUtil to enable client side functions, they do not work.

For example, the following code will work by running OLE as client/server, in a 3-tier environment, or on the application server:

```
DECLARE
  app  OLE2.OBJ_TYPE;
  args OLE2.LIST_TYPE;
BEGIN
  -- create a new document
  app := OLE2.CREATE_OBJ('Word.Basic');
  OLE2.INVOKE(app, 'FileNew');
  -- insert data into new document from long item
  args := OLE2.CREATE_ARGLIST;
  OLE2.ADD_ARG(args, :long_item);
  OLE2.INVOKE(app, 'Insert', args);
  OLE2.DESTROY_ARGLIST(args);
  -- save document as example.tmp
  args := OLE2.CREATE_ARGLIST;
  OLE2.ADD_ARG(args, 'c:\temp\example.tmp');
  OLE2.INVOKE(app, 'FileSaveAs', args);
  OLE2.DESTROY_ARGLIST(args);
  -- load example.tmp into ole item in form
  FORMS_OLE.ACTIVATE_SERVER('msw_obj');
  FORMS_OLE.INITIALIZE_CONTAINER('msw_obj', 'c:\temp\example.tmp');
  FORMS_OLE.CLOSE_SERVER('msw_obj');
  -- clear data from example.tmp
  OLE2.INVOKE(app, 'EditSelectAll');
  OLE2.INVOKE(app, 'EditClear');
  -- save empty example.tmp
  args := OLE2.CREATE_ARGLIST;
  OLE2.ADD_ARG(args, 'c:\temp\example.tmp');
  OLE2.INVOKE(app, 'FileSaveAs', args);
  OLE2.DESTROY_ARGLIST(args);
  -- close example.tmp
  args := OLE2.CREATE_ARGLIST;
  OLE2.ADD_ARG(args, 2);
  OLE2.INVOKE(app, 'FileClose', args);
  OLE2.DESTROY_ARGLIST(args);
  -- exit MSWord
  OLE2.RELEASE_OBJ(app);
END;
```

However, when you change this code to use CLIENT_OLE2 (using search-and-replace) it doesn't work.

The following example is code which *will* convert to CLIENT_OLE2:

```
DECLARE
  app      CLIENT_OLE2.OBJ_TYPE;
  docs     CLIENT_OLE2.OBJ_TYPE;
  doc      CLIENT_OLE2.OBJ_TYPE;
  selection CLIENT_OLE2.OBJ_TYPE;
  args     CLIENT_OLE2.LIST_TYPE;
BEGIN
  -- create a new document
  app := CLIENT_OLE2.CREATE_OBJ('Word.Application');
  CLIENT_OLE2.SET_PROPERTY(app, 'Visible', 1);

  docs := CLIENT_OLE2.GET_OBJ_PROPERTY(app, 'Documents');
  doc  := CLIENT_OLE2.INVOKE_OBJ(docs, 'add');

  selection := CLIENT_OLE2.GET_OBJ_PROPERTY(app, 'Selection');

  -- insert data into new document from long item
  CLIENT_OLE2.SET_PROPERTY(selection, 'Text', 'this is a test message');

  -- save document as example.tmp
  args := CLIENT_OLE2.CREATE_ARGLIST;
  CLIENT_OLE2.ADD_ARG(args, 'c:\example.doc');
  CLIENT_OLE2.INVOKE(doc, 'SaveAs', args);
  CLIENT_OLE2.DESTROY_ARGLIST(args);

  -- close example.tmp
  args := CLIENT_OLE2.CREATE_ARGLIST;
  CLIENT_OLE2.ADD_ARG(args, 0);
  CLIENT_OLE2.INVOKE(doc, 'Close', args);
  CLIENT_OLE2.DESTROY_ARGLIST(args);

  CLIENT_OLE2.RELEASE_OBJ(selection);
  CLIENT_OLE2.RELEASE_OBJ(doc);
  CLIENT_OLE2.RELEASE_OBJ(docs);

  -- exit MSWord
  CLIENT_OLE2.INVOKE(app, 'Quit');
END;
```

If you are running client/server Forms you *may* be using code like

```
app := OLE2.CREATE_OBJ('Word.Basic');
```

or

```
app := OLE2.CREATE_OBJ('Word.Application');
```

Both of these code examples initialize an OLE interaction with Microsoft Word. The difference between the two examples is that Microsoft changed the interface between their Office95 and their Office97 applications. Webutil has been tested against the recent (and current) Office97 interface (Word.Application).

So, if you are using the Office97 interface then doing a search and replaced to change OLE2 to CLIENT_OLE2 will probably work. If you are using the old (obsoleted by Microsoft) interface then doing this search and replace may not work.

Functions in WebUtil

This chapter details the coding APIs available within WebUtil. WebUtil contains several classes of APIs:

- [Client Server Parity APIs](#)
- [Ported D2KWUTIL Functions](#)
- [WebUtil Public Functions](#)
- [WebUtil_Host](#)
- [Utility Functions](#)

5.1 Client Server Parity APIs

One of the key functions of WebUtil is to provide a way of re-directing calls commonly used in client server to operate on the Client Browser machine rather than on the application server machine. These functions and packages are all prefixed with “Client_”, and make for simple conversion of client server code to operate on the client browser machine with a simple search and replace operation.

- `Client_Get_File_name`
- `Client_Host`
- `Client_Image` (Includes `Read_Image_File` and `Write_Image_File`)

Note: The `Client_Image` package contains implementations of the Read and Write image file built-ins which work between a Forms image item and the client browser machine hard-disk. In order for this feature to work you must enable File transfer to the application server and set up an appropriate work area – see [Chapter 3.3.1, "Enabling Transfer"](#).

- `Client_OLE2` (Win32 Clients only)
- `Client_Text_IO`
- `Client_Tool_Env`

Each of these packages or program units is designed to work as closely to the basic version of the code as possible. This includes raising the same exceptions and errors. For documentation purposes treat these program units just like their built-in equivalents. Note however, that there may be certain differences in implementation and you should test your code thoroughly to ensure that they behave as expected.

Many of the client server parity features in WebUtil are implemented using the main WebUtil public functions that often will offer more power and flexibility than the “Client_” version.

5.1.1 Running HOST Commands

If you have a host command of the form:

```
HOST ('DEL c:\temp.txt')
```

This will work in both client/server and Web deployed environments, but in web deployed environments the command runs on the middle tier.

If you change this to:

```
CLIENT_HOST('DEL c:\temp.txt')
```

This does not work. You must use:

```
CLIENT_HOST('cmd /c DEL c:\temp.txt');
```

You must ensure that the command will run from the **Start | Run** box on Windows.

5.2 Ported D2KWUTIL Functions

D2KWUTIL is another add-on package used extensively by client-server applications running on Windows. WebUtil contains client side ports of a limited number of D2KWUTIL functions, specifically those involved with environment information on the client machine, such as the functions to read and write the Windows Registry.

Only the `Win_API_Environment` package has been ported, along with some packages that it is dependant upon.

The APIs remain identical to the basic D2KWUTIL APIs, but again the prefix of “Client_” needs to be used. Errors in Client D2KWUTIL code can be referenced using the `Client_Win_API` package, just like in the client server version of the utility.

The following functions are available when running the browser on a Win32 machine only:

- Win_API_Environment Functions
- Create_Registry_Key
- Delete_Registry_Key
- Get_Computer_Name
- Get_Environment_String
- Get_Net_Connection
- Get_Temp_Directory
- Get_Windows_Directory
- Get_Windows_Username
- Get_working_Directory
- Read_INI_File
- Read_Registry
- Write_INI_file
- Write_Registry
- Write_RegistryEx

Note: These Client_Win_API functions are a direct port of the D2KWUTIL functions and are implemented using the WebUtil_C_API package to call into the D2KWUTIL DLL. The same information or function may be available in a more efficient manner elsewhere in WebUtil e.g. WebUtil_ClientInfo

5.3 WebUtil Public Functions

The core of WebUtil is a set of packages prefixed with "WebUtil_" These provide a set of APIs covering"

- Client server Parity APIs
- Ported D2KWUTIL functions
- WebUtil Public Functions
- Utility Functions
- Internal APIs
- Information about the client machine
- A low level API to call C functions on the Client (Win32 clients only), this can be used to interface with the Windows APIs and other 3rd party code.
- File manipulation functions for files and directories on the client
- File transfer services to the client, database and application server file system
- Execute operating system commands on the client
- Recover from Forms session time-outs and other failures

5.3.1 WebUtil_ClientInfo

WebUtil_ClientInfo contains functions to obtain information about the client machine:

Function	Purpose
Get_Date_Time	Returns a PL/SQL date containing the date and time on the client machine.
Get_file_Separator	Returns the character used on the client machine as the file separator e.g. "\" on Windows.
Get_Host_Name	Gets the name of the client machine.
Get_IP_Address	Gets the IP address of the client machine as a string.
Get_Java_Version	Returns the Version of the JVM that is running the Forms Applet. This may be useful if you have your own JavaBeans in the form that depend on a certain Java version.
Get_Language	Returns the language code of the client e.g. en for English, de for German.
Get_Operating_System	Returns the O/S name of the O/S that the browser is running under.
Get_Path_Separator	Returns the character used to separate directory locations on Paths e.g. ";" on Windows.
Get_System_Property	Allows you to get any Java System Property.
Get_Time_Zone	Returns the time zone that the client machine is running in.
Get_User_Name	Returns the name of the user logged into the client machine.

5.3.2 WebUtil_C_API

C_API is a comprehensive API, which allows you to call into C libraries on the client. At present the API only supports Win32 Clients.

You must register your function that you wish to call, this requires the name of the library file (e.g. DLL or .EXE) that contains the function and the name of the function itself. Once declared, the reference to the function is held in a specialist type WebUtil_C_API.FunctionHandle.

If the function takes parameters then you will need to declare a parameter list (of type WebUtil_C_API.ParameterList) to which you add each parameter to the function in turn, providing an indicator of the datatype, a value if required, a maximum size (for char* variables) and an indicator saying if this parameter is IN, OUT or IN OUT.

Functions are invoked passing the handle of the registered function and the handle of the parameter list that defines its parameters.

5.3.3 WebUtil_File

WebUtil_File contains APIs to manipulate files and directories on the client and to display file selection dialogs. The package contains a new type FILE_LIST, which is a PL/SQL table (array) used to pass back multiple file names. This array can be accessed

using standard PL/SQL table techniques. e.g. Use `<file_list>.COUNT` to return the number of elements and `<file_list>(n)` to retrieve a specific numbered element.

Function	Purpose
<code>Copy_File</code>	Copies a file. Returns a Boolean to indicate success.
<code>Delete_File</code>	Deletes a file. Returns a Boolean to indicate success.
<code>Rename_File</code>	Renames a file. Returns a Boolean to indicate success.
<code>Create_Directory</code>	Creates the named directory if it does not exist. Any intermediate directories will also be created. Returns a Boolean to indicate success.
<code>Directory_Root_List</code>	Returns a <code>FILE_LIST</code> containing the directory roots on the client system. On a Windows PC these would correspond to the drives on the machine.
<code>Directory_List</code>	Returns a <code>FILE_LIST</code> containing all the files and subdirectories in a particular directory. There is an optional <code>Return_Files_Only</code> argument to restrict the returned list to just files with no directories.
<code>Directory_Filtered_List</code>	Like <code>Directory_List</code> this will return a list of files in a directory but you are able to filter using the '*' and '?' wildcards e.g. *.FMB to restrict the list.
<code>File_Exists</code>	Returns a Boolean value indicating if the named file exists on the client.
<code>File_Is_Directory</code>	Returns a Boolean value indicating if the file name supplied is actually a directory on the client.
<code>File_Is_Hidden</code>	Returns a Boolean value indicating if the file has its hidden attribute set.
<code>File_Is_Readable</code>	Returns a Boolean value indicating if the file can be read.
<code>File_Is_Writable</code>	Returns a Boolean value indicating if the file can be written to.
<code>File_Size</code>	Returns the size of the file in bytes.
<code>Directory_Selection_Dialog</code>	Displays a Directory selection dialog. The initial directory and the title for the dialog can be defined. The selected directory is returned.
<code>File_Selection_Dialog</code>	Allows the definition of a File Save or Open dialog with a configurable file filter. Returns a single file selection. The file filter can use ? and * as wildcards and is in the format <label> <filterString> . Multiple filters can be specified, e.g. ' Gif Files *.gif JPEG Files *.jpg '.
<code>File_Multi_Selection_Dialog</code>	As <code>File_Selection_Dialog</code> except that the user can select multiple files and these are returned in a <code>FILE_LIST</code> .

Function	Purpose
File_Open_Dialog	A convenience method that creates a File Open Dialog.
File_Save_Dialog	A convenience method that creates a File Save Dialog.
Get_file_Separator	Returns the character used on the client machine as the file separator e.g. "\" on Windows.
Get_Path_Separator	Returns the character used to separate directory locations on Paths e.g. ";" on Windows.

5.3.4 The File transfer package

WebUtil_File_Transfer provides APIs for transferring files to and from the client browser machine. All of the transfer functions provide the ability to display a progress bar as the transfer takes place, to re-assure the end user if the connection is slow or the transfer is large.

Transferring data directly to a BLOB column in the database require that you are running Oracle Server 9.0.1 or above and that you have installed the WEBUTIL_DB package and granted execute privilege to the current user. You will also need a public synonym of WEBUTIL_DB for the package.

Transferring data to or from the application server disk involves using some Java on the middle tier and so your .env file should be suitable configured with the correct Classpath information and Path / Library load path information to call Java from Forms PL/SQL.

Function	Purpose
URL_To_Client	Transfers a file from a specified URL to the client machine. The file size can be specified as a checksum to see that the transfer succeeded. If it is not supplied the check will not be carried out. The function returns true if the download succeeded.
URL_To_Client_With_Progress	A version of URL_To_Client that provides a progress bar as the download happens. You can supply a title, subtitle and message to display in the dialog as the download takes place.
Client_To_DB	Uploads a named file on the client into a BLOB column in the database. You must define the name of the target table, the name of the column, and a where clause that defines one and only one row in that table. This upload is carried out using the current database connection and schema privileges. The process can also be made asynchronous, in which case you can define the name of a trigger which WebUtil will execute as soon as the upload is complete. Note: Asynchronous mode offers better re-drawing of the Forms screen while the transfer takes place, but will not offer the ability to carry out significant work while the transfer is taking place. The webutil.cfg parameter transfer.database.enabled must be set to true to use this function.

Function	Purpose
Client_To_DB_With_Progress	As Client_To_DB but with a progress bar, which again can have various labels defined for it.
DB_To_Client	Downloads a file from a BLOB column in the database to the client machine. The table name, column name and a where clause all have to be specified to identify the binary object to download. The current forms session and schema is used for the download. The webutil.cfg parameter transfer.database.enabled must be set to true to use this function.
DB_To_Client_With_Progress	As DB_To_Client but with a progress bar.
Client_To_AS	Uploads a named file from the client onto the application server disk. The process can also be made asynchronous, in which case you can define the name of a trigger that WebUtil will execute as soon as the upload is complete. Note: Asynchronous mode offers better re-drawing of the Forms screen while the transfer takes place, but will not offer the ability to carry out significant work whilst the transfer is taking place. The webutil.cfg parameter transfer.appsrv.enabled must be set to true to use this function, and if the transfer.appsrv.accessControl parameter is also set to true, the upload target directory must be listed in the transfer.appsrv.write list. It is sensible to call the WebUtil_File_Transfer_IsASWritable function with the target filename before starting the upload to ensure that you will be able to write to this location on the application server.
Client_To_AS_With_Progress	As Client_To_As but with a progress bar.
AS_To_Client	Transfers a file from the application server to the client. The webutil.cfg parameter transfer.appsrv.enabled must be set to true to use this function, and if the transfer.appsrv.accessControl parameter is also set to true, the upload target directory must be listed in the transfer.appsrv.read. list. It is sensible to call the WebUtil_File_Transfer_IsASReadable function with the target filename before starting the download to ensure that you will be able to read from this location on the application server.
AS_To_Client_With_Progress	As AS_To_Client with a progress bar.
In_Progress	If an Asynchronous update is in progress this function will return true. This should be used before starting another transfer off as only one transfer can take place at once.
Asynchronous_Upload_Success	Returns a Boolean indicating if an Asynchronous load succeeded. This should be called from the callback trigger that is executed when an Asynchronous upload has finished.

Function	Purpose
Get_Work_Area	Returns a work area directory on the application server that is private to this user. It can take a Boolean argument (defaulting to true) that indicates that WebUtil should create the private directory if it does not already exist. The private directory is created under the directory specified in transfer.appsrv.workAreaRoot (in the webutil.cfg). The directory that is created is constructed from the IP address and username of the client machine - so will be unique between end users, but multiple applications running on the same client machine will share the directory.
Is_AS_Writable	Given a file name this function returns true if the current rules defined in the webutil.cfg file will allow the file to be written.
Is_AS_Readable	Given a file name this function returns true if the current rules defined in the webutil.cfg file will allow the file to be read.

5.4 WebUtil_Host

Webutil_Host provides a series of routines for executing commands on the browser machine. Importantly, this package contains two types. The first is a `PROCESS_ID` which is used to record a reference to a client side process, which can be used to obtain the output of that process or to terminate it.

The second type is OUTPUT_ARRAY, which is a PL/SQL table of VARCHAR2 holding the output from a client side command.

Function	Purpose
Host	Runs the specified command on the client – optionally this can return the return code. This call is blocking.
Blocking	Runs the specified command on the client – optionally this can return a process id if you need to examine the return code or output. This call is blocking.
NonBlocking	Runs the specified command on the client – optionally this can return a process id if you need to terminate the process. This call is non-blocking.
NonBlocking_With_Callback	Runs the specified command on the client, you also supply a trigger, which WebUtil will execute once the client process has completed. This call is non-blocking.
Terminate_Process	Given a process handle of a process this function will kill the process on the client.
Get_Return_Code	Given a process handle of a completed process this function will return the return code of the process as an integer.
Get_Standard_Output	Given a process handle of a completed process this function will return an OUTPUT_ARRAY containing any output that was sent to standard output by the client process.
Get_Standard_Error	Given a process handle of a completed process this function will return an OUTPUT_ARRAY containing any output that was sent to standard error by the client process.
Release_Process	Given a process handle this frees up the resources associated with the process.
Get_Callback_Process	When a client command has been called from NonBlocking_With_Callback, the callback trigger executes, this function can be used to get the Process Id of the finished process. This can then be used to obtain the output or return code.
Id_Null	Tests if a process id is null.
Equals	Tests if two process ids represent the same process.

5.4.1 WebUtil_Session

WebUtil_Session provides a function for reacting to an interruption to the Forms session. It provides a way of defining a URL that the browser should re-direct to if the Form ends or crashes. This is commonly required when the Forms server times out and you don't want the user to be left looking at a blank applet area in the browser.

Note that the session monitor will not detect that the Forms server has timed out until the user attempts to use the Form in some way. At that point they will be directed.

Function	Purpose
Enable_Redirect_On_TimeOut	Enables the time-out monitor and allows you to specify a URL that the browser should re-direct to when the situation occurs.
Disable_Redirect_On_TimeOut	Switches off the monitor - if you do not call this function before issuing EXIT_FORM then the browser will be still re-directed even though it's a "normal" exit.

5.4.2 WebUtil_Browser

WebUtil_Browser provides some calls for use when the Form that is using WebUtil is running within a browser window (see also WebUtil_SeparateFrame).

Function	Purpose
BrowserMessage	Sends a message to the browser message line (as opposed to the Forms status line)
GetAppletParameter	Provides access to Applet parameters defined in your formsweb.cfg. For instance the lookAndFeel parameter.
GetAppletSize	Returns the width and height of the applet area in the browser, in pixels, as a string in the format "width,height"
ShowMenuBar	Allows you to hide the Forms Menu all together. Note: to work the following must be true: ■ The menu property of the form must be null ■ Logo should be set to no in the formsweb.cfg
ShowStatusBar	Allows you to hide or display the Forms status bar - much like the setting of the Console window at design time, except that the this procedure allows you to do it dynamically at runtime. Note: The Status bar will only display if the console window was initially assigned at design time.

5.4.3 WebUtil_SeparateFrame

WebUtil_SeparateFrame provides some calls for use when the Form that is using WebUtil is running outside of the browser window, using the parameter `separateFrame=true`.

Function	Purpose
IsSeparateFrame	Returns a Boolean indicating that you are running in separate frame mode or not
GetSeparateFrameSize	Returns the width and height of the frame, in pixels, as a string in the format "width,height"
ShowMenuBar	Allows you to hide the Forms Menu all together. Note: to work the following must be true: The menu property of the form must be null. Logo should be set to no in the formsweb.cfg
ShowStatusBar	Allows you to hide or display the Forms status bar - much like the setting of the Console window at design time, except that the this procedure allows you to do it dynamically at runtime. Note: The Status bar will only display if the console window was initially assigned at design time.
SetTitle	Set the title on the MDI frame
SetIcon	Set the icon used in the control box of the MDI frame and in the Task bar etc. The icon specified can be in GIF or JPG format and will be searched for in the same way as icons for buttons within the Form.
AllowResize	Allows you to disable (or re-enable) the resizing of the MDI frame
CenterMDI	Auto-center the MDI frame for you on the display. Comes in two versions, one of which will allow you to nudge up or down to fit in with the task bar etc.

5.4.4 WebUtil_Core

Most functions in WebUtil_Core are private to WebUtil. Some functions, however, can be called directly.

Function	Purpose
IsError	Use this function to check if the last WebUtil call that you made succeeded.
ErrorCode	Use this function to return the last WebUtil error code.
ErrorText	Use this function to return the text of the last WebUtil error.

5.5 Utility Functions

These functions are those within WebUtil which do not perform Client Side integration, but which do provide useful functionality, which you can freely re-use within your own applications.

- DelimStr
- Show_WebUtil_Information
- WebUtil_Util

5.5.1 DelimStr

DelimStr is a package which provides a set of calls for interacting with delimited strings such as comma delimited strings. It provides APIs to treat the delimited string as an array, allowing the programmer to get and set members of the array by index number, and handling the storage of multiple data types within it.

5.5.2 Show_WebUtil_Information

This procedure causes the hidden WebUtil window to display in your Form, providing you with the versions of all the WebUtil components. The procedure can optionally be called with a Boolean argument. Passing a false to this parameter will cause the information window to hide again.

5.5.3 WebUtil_Util

This package contains a single function BoolToStr() for converting a Boolean value to text. By default the strings "true" or "false" will be returned depending on the value of the Boolean, however, the true and false strings can be overridden to reflect other values, for instance "Yes and "No".

5.5.4 Internal APIs

Certain packages within WebUtil are in place purely to help with the requirements of the public APIs. These packages should not be called directly:

- Java_Appserv_Reader
- Java_Appserv_Writer
- Java_Exception
- Java_File
- Java_System

Tracing and Diagnostics

In this chapter the tracing and diagnostic facilities of WebUtil are discussed. These cover three areas:

- [Connection Logging](#)
- [Error Handling](#)
- [Diagnostic Tracing](#)

6.1 Connection Logging

If administrators are interested in knowing detailed information about the users and machines that are connecting to their application, WebUtil provides a connection logging mechanism which is separate from the other tracing and logging within the utility.

Connection logging is enabled by setting the flag `logging.connections` in `webutil.cfg` to true.

Connection Logging gets information about the connected user directly from the client machine, including the IP address and O/S username of the user. Note that because the IP address is obtained from the browser machine, it will be the real IP address of the Client rather than the IP address of a firewall, or router, which the client is behind. The form that the user is connected to will also be recorded.

6.2 Error Handling

In some circumstances WebUtil may raise an error, this may be due to the misuse of an API or some internal problem. The errors that can be raised are detailed in Appendix C – WebUtil Error Reference. Error logging is controlled by the `formsweb.cfg` parameter `WebUtilErrorMode`. This can be set to one of four values:

- Console (the default)
- Alert
- Server
- All

Errors are always displayed on the Java console, these switches force the errors to be reported elsewhere as well.

Setting the parameter to Alert shows the error in a pop up alert in the browser. Setting it to Server streams the error back to the server log file. In order for this to work, the `webutil.cfg` settings `logging.file` and `logging.enabled` must be defined.

Setting the parameter to All both shows the error in an alert on the client and in the server log file.

6.3 Diagnostic Tracing

The final diagnostic feature of WebUtil is the trace facility. This can be used to see the various calls that are made to the client and detail about them.

Tracing is enabled using the `WebUtilLogging` parameter in the `formswb.cfg` file. This can be switched to one of four values:

- Off (the default)
- Console
- Server
- All

By default logging messages are suppressed. Much like errors they can be directed elsewhere using these settings. In order for server side logging to work, the `webutil.cfg` `logging.file` must be defined, the `logging.enabled` parameter set to true and the `logging.erroronly` parameter set to false.

Finally you can control the detail of the logging that is carried out using the `formswb.cfg` parameter `WebUtilLoggingDetail`. This can be set to two detail levels:

- Normal (the default)
- Detailed

Performance Tuning Considerations

WebUtil as a utility is designed primarily to provide extra client side functionality. This functionality comes at a cost. The utility has to exchange messages between the application server and the client, resulting in traffic over the network. Additionally the process of marshalling and passing data across the network will take time.

Taking for instance the case of OLE integration; calling OLE on the application server or the equivalent OLE integration in client server will be much faster than making remote calls to OLE on the client.

However, it is anticipated that the kinds of applications that require WebUtil functionality will primarily be aimed at intranet use over known network conditions of latency and bandwidth.

7.1 Optimizing your use of WebUtil

The following suggestions should help you to reduce the network and other costs inherent in using the utility.

1. Only WebUtil Enable Forms that actually need the functionality. Each form that is WebUtil enabled will generate a certain amount of network traffic and memory usage simply to instantiate the utility, even if you don't use any WebUtil functionality.
2. Disable logging and error recording on the server except for during testing and debugging. Active logging will slow down the client considerably and generate a lot of network traffic.
3. Think carefully about object re-use. Whenever you obtain an object handle from the client, such as a `Client_OLE2.OBJ_TYPE` or a `WebUtil_C_API.FunctionHandle`, this represents one or more Java objects created and held

by the Forms client. Creating these objects is expensive and they should be re-used when possible, rather than being destroyed and re-created.

4. Use functions in an efficient manner. For instance if you need to read a text file on the client machine, it will probably be more efficient to transfer the file to the users's work area using `WebUtil_File_Transfer` and then read it with normal `TEXT_IO`, rather than reading it line by line across the network
5. Once Client side objects have been finished with you should call the relevant destroy or release method, to free up the client side resources. If you do not, they will be automatically cleaned up when the form is closed, but before then they will be consuming memory resources on the client. If you persist a resource so that it can be referenced in another form you must take care to explicitly clean it up.
6. Be aware there may be multiple ways to obtain the same information or execute a function. This applies particularly to `D2KWUTIL` functions which are often duplicated by more efficient `WebUtil` functions.
7. The file transfer functions `Client_To_AS` and `AS_To_Client` use Java calls from the Forms engine on the middle tier. There is a time penalty for the very first call as the JVM is started and the extra memory overhead of running the JVM. Upload and download from the database does not use Java on the middle tier and is more efficient.
8. The most efficient file download facility is the `URL_To_Client` function. This should be used in preference to `AS_To_Client` if possible, but does require that the source directory is mapped as a virtual directory to the web server.
9. If the transfer from the client to the database is done in a forms session, a commit is required after file transfer (to the database) before exiting the Form to save the changes. A transfer succeeds only if the transfer is done in the same session. Otherwise, the next time you start that Form, `WebUtil` raises error "WUT-111: Database LOB is un-initialised wu_test_table.BLOB - where ID=1".

Customizing WebUtil

The various `formsweb.cfg` and `webutil.cfg` parameters that have already been discussed can be used to control many aspects of WebUtil's operations. You can also customize or translate the various error messages and strings used by WebUtil if you require.

WebUtil error messages and strings are stored in two places.

8.1 Customizing Client-Side Error Messages

Those used in the client Java code are held in a message bundle (`MessageBundle.class`) in the `oracle.forms.webutil.common` package. You may translate the existing `MessageBundle.java` file by creating a new class with a country code suffix e.g. `MessageBundle_fr.java` for a French translation and compiling it and `-re-jarring` the WebUtil classes. In this case you will also have to re-sign the Jar file.

8.2 Customizing Server-Side Error Messages

Server side error messages are all held in the `webutil.cfg` file. Each translatable string is prefixed with its country code. For example `EN.msg.error.error007` for the English translation of an error message string and `DE.msg.error.error007` for the German version. Additional translations can be added into the text file as required, or the existing messages altered.

A

Appendix A – Runtime Install Checklist

Use the following checklist to help you deploy WebUtil to a production system.

Task	Reference	Done
1. Check pre-requisites	Chapter 2.1, "WebUtil Prerequisites"	
2. Unzip / Un-Tar WebUtil distribution	Chapter 2.2.1, "The Webutil ZIP/TAR File" . Install Location: _____	
3. Create or choose a database schema to own the WEBUTIL_ DB package. This schema needs Create Procedure privilege and access to certain DBMS_ and UTL_ packages	Chapter 2.3, "Configuring your Database for WebUtil" Schema: _____	
4. Run the create_webutil_db.sql script from the /forms directory using the schema created in [3]	Chapter 2.3, "Configuring your Database for WebUtil"	
5. Create a public Synonym WEBUTIL_ DB for the package	Chapter 2.3, "Configuring your Database for WebUtil"	

Task	Reference	Done
6. Grant execute on the WEBUTIL_DB synonym to any schemas that need to do database upload / download from the schema that owns the WEBUTIL_DB package (see [3])	Chapter 2.3, "Configuring your Database for WebUtil"	
7. Define the virtual directory "/webutil" pointing at the /lib directory of the WebUtil install directory (see [2])	Chapter 2.4, "Configuring your HTTP Server for WebUtil"	
8. For each application that needs to be WebUtil enabled: ■ In the formsweb.cfg config entry for the application add the entry:webUtilArchive=/forms90/webutil/webutil.jar In the formsweb.cfg config entry for the application define the locations of the WebUtil template .htm files. Define entries for: ■ baseHTMLjinitiator ■ BaseHTMLjpi ■ baseHTML ■ These should point at the relevant templates in the /server directory of your install location (see [2])	Chapter 2.5, "WebUtil Entries in the formsweb.cfg File"	

Task	Reference	Done
9. For the environment files (envFile=) used by each configuration in the formsweb.cfg that uses WebUtil ■ Ensure that the FORMS90_PATH includes the /forms directory of the WebUtil install location (see [2]). ■ Add a new environment value WEBUTIL_CONFIG=. This should point at the file /server/webutil.cfg in the WebUtil install location (see [2]) ■ Include the full path and name of /lib/webutil.jar (see [2]) in the CLASSPATH= setting. ■ Include the full path and name of the Java rt.jar in the CLASSPATH= setting. rt.jar is the Java runtime jar file and can be found in the %ORACLE_HOME%/jdk/jre/lib directory or in any other Java 1.3 or 1.4 install. ■ (Windows Only) If not defined in your base O/S path, the java runtime /bin/classic directory (e.g. %ORACLE_HOME%/jdk/jre/bin/classic) should be added to a PATH= setting in the .env file so that WebUtil can find the jvm.dll file (note this path should also include directories required to actually run Forms itself) ■ (UNIX only) Ensure that the LD_LIBRARY_PATH (or equivalent) includes the location of the libjava.so file	Chapter 2.6, "WebUtil Entries in the Environment File"	

Task	Reference	Done
10. Configure your webutil.cfg file ■ Configure your tracing / logging options ■ Define custom system libraries for download if required (Win32 only) ■ Configure file transfer permissions (if required) ■ Define the work area	Chapter 3.2.1, "Logging Options" Chapter 3.2.2, "Installation Options" Chapter 3.3, "File Upload and Download Options"	
11. Add the WebUtil /forms directory (see [2]) to your environment or Registry FORMS90_PATH on your development machine(s)	Chapter 4, "Using WebUtil in Your Applications"	
12. (Optional) Add the path (see [2]) and name of /lib/webutil.jar to your FORMS90_BUILDER_CLASSPATH in your development machine environment or Registry.	Chapter 4.2, "Seeing the PJC at Design Time"	
13. For each Form that needs WebUtil functionality ■ Attach the webutil.pll file ■ Subclass the WebUtil Object group from webutil.olb ■ Ensure that the navigation logic in your Form bypasses the WEBUTIL block ■ When running WebUtil applications from the builder, change your Edit à Preferences à Runtime Settings in the Menu to ensure that the Application Server URL setting specifies a configuration in the formsweb.cfg that is WebUtil "enabled".	Chapter 4.1, "Adding The Required Objects"	

Appendix B – WebUtil Error Reference

This section contains a complete list of all WebUtil error messages, their meanings, and where appropriate, their resolution. Errors are prefixed with a three letter suffix which indicates which JavaBean has raised the error, for instance WUT for errors from the file transfer beans. Errors prefixed with WUC are common errors that are raised by the WebUtil Javabeen superclass.

All of the error messages can be changed by editing the appropriate message bundle (see [Chapter 8, "Customizing WebUtil"](#) for more information).

Errors fall into two categories – User errors which are generally going to be caused by misuse of the supplied APIs or some other factor out of WebUtils control such as a hard disk filling up. And Internal errors which are thrown when something serious has happened within the utility code itself. In all cases, when an error is received it is a useful technique to switch WebUtil logging on to gather as much information as possible about the circumstances leading up to the problem. See [Chapter 6, "Tracing and Diagnostics"](#) for more information.

Error	Description
WUC-1	Component has no parent frame Type: Internal Description: WebUtil cannot find the owning Frame for the Forms application. This frame is used as the parent for modal dialogs such as Alerts Resolution: WebUtil is designed to run within the context of the Forms Java Applet. This error should never occur if run in that way.

Error	Description
WUC-2	Invalid logmode supplied. Type: User Description: An invalid logging mode was requested. Resolution: The following logging modes are valid: OFF ON CONSOLE SERVER ALL
WUC-3	Invalid log detail level supplied. Type: User Description: An invalid detail level was specified for logging. Resolution: The following detail levels are valid: NORMAL DETAILED.
WUC-4	Invalid errormode supplied. Type: User Description: An invalid error mode was requested Resolution: The following error modes are valid: CONSOLE ALERT SERVER ALL
WUC-5	No WebUtil Configuration File specified. Type: User Description: WebUtil cannot find its configuration file. This file is required to set up features like logging Resolution: The location of the webutil config file is determined in two ways either using the environment variable WEBUTIL_CONFIG on the application server or by setting the parameter WebUtilConfig in your formsweb.cfg file. Set in either way, the value should point to a valid file on the application server.
WUC-6	Unable to read WebUtil Configuration File. Type: User Description: The WebUtil configuration file that was specified cannot be read. This file is required to set up features like logging Resolution: The location of the webutil config file is determined in two ways either using the environment variable WEBUTIL_CONFIG on the application server or by setting the parameter WebUtilConfig in your formsweb.cfg file. Set in either way, the value should point to a valid file on the application server.
WUC-7	Unable to write to WebUtil Logging File. Type: User Description: The logging file that was specified in the WebUtil configuration file cannot be written to. Resolution: Ensure that the logging file location exists and if the file already exists it is not read-only
WUC-8	Server side logging was requested but no log file was specified - logging is disabled. Type: User Description: The logging.file entry in the WebUtil config file is missing or empty so the correct logging file cannot be identified. Resolution: Set the logging.file entry in your WebUtil configuration file

Error	Description
WUC-9	Unable to obtain LocalHost info from InetAddress. Type: Internal Description: WebUtil could not get information about the host machine from Java. Resolution: Ensure the JAR file containing WebUtil is signed.
WUC-10	Could not obtain Host IP Address. Type: Internal Description: WebUtil could not get the IP Address of the host machine from Java. Resolution: Ensure the JAR file containing WebUtil is signed.
WUC-11	WebUtil must be signed in order to work. Type: Internal Description: WebUtil could not access secure information from the applet it needs to be signed to allow it to work. Resolution: Ensure the JAR file containing WebUtil is signed.
WUC-12	Object Cache Error: Object is not the expected {0} type. Type: User Description: An object stored in the WebUtil client side Object Cache cannot be cast to the expected type. Resolution: Check your code - do no hardcode object references in your application and only use the correct type of object handle for the operation. For instance do not pass an OLE object handle to the text_io functions.
WUC-13	Object Cache Error: No object Cache present. Type: User Description: WebUtil tried to read from the Object cache but it does not exist yet. Resolution: You should only submit an object handle to WebUtil that was created by WebUtil
WUC-14	Object Cache Error: Specified object handle {0} not found in the cache. Type: User Description: WebUtil tried to read an object from the object cache but no object was found at that position. Resolution: Check that you have not already destroyed the object before this call.
WUC-15	Your Form must contain the following Bean for this function to be available. Type: User Description: WebUtil uses Bean areas to carry out client side functionality and these must be present. Resolution: Drag the WEBUTIL object group into your Form.
WUC-16	When setting a client side property a null value was supplied. Type: User Description: A call to set a property in webutil.properties was made but no property string was sent. Resolution: The property string should be in the format propname=value

Error	Description
WUC-17	Information was missing from the supplied file download request {0}. Type: User Description: The request asking for a file download was not formed correctly Resolution: Only use the supplied PL/SQL APIs to call WebUtil functions
WUC-18	Invalid server side file specification {0}. Type: User Description: The requested server side file URL cannot be read. Resolution: Check the URL
WUC-19	Unable to write to local file {0}. Type: User Description: The attempt to make a local copy of a server side file failed. Resolution: Check that there is space on the client side disk and that the user has permissions to the specified directory
WUC-20	The stated size of the source file {0} does not match that of the downloaded file {1}. Type: User Description: WebUtil has downloaded from an URL and the file that it has created is not the same size as the Size stated in the download command for the original file. This may be because the original file was incorrectly specified. Resolution: Check the downloaded file in a text editor for more information and/or check there is space on the target device.
WUC-21	The downloaded file {0} was not created. Type: User Description: The attempt to make a local copy of a server side file failed. Resolution: Check that there is space on the client side disk and that the user has permissions to the specified directory.
WUC-22	The global object reference was incorrect ({0}). Type: User Description: An invalid object reference was passed to the client. Resolution: Only persist a valid reference
WUC-25	URL cannot be null. Type: User Description: An empty URL was passed to the client. Resolution: enter a URL in the Source field.
WUT-100	Bad file information string: {0}. Type: User Description: WebUtil was asked to upload or download a file, but incorrect information was supplied. Resolution: Supply a valid file name.

Error	Description
WUT-101	File to download has an invalid size: {0}. Type: User Description: The file size of a download file was defined as null, a negative value or some other non integer value. Resolution: Only use the WebUtil APIs to download files.
WUT-102	Input Buffer Empty. Type: User Description: Zero bytes were read from a file during upload. Resolution: Check the client file is not empty.
WUT-103	File Transfer already in progress cannot upload {0}. Type: User Description: WebUtil is already in the middle of processing a file and cannot start on another simultaneous transfer. Resolution: Finish one file transfer before starting the next.
WUT-104	File Transfer already in progress cannot download {0}. Type: User Description: WebUtil is already in the middle of processing a file and cannot start on another simultaneous transfer. Resolution: Finish one file transfer before starting the next
WUT-105	Unable to open file {0} for writing. Type: User Description: WebUtil was trying to download a file but could not open the file on to client to write to. Resolution: Check that you have privileges to write to the specified file and there is enough disk space.
WUT-106	Unable to open file {0} for reading. Type: User Description: WebUtil was trying to upload a file but could not open the file on to client to read. Resolution: Check that you have privileges to read from the specified file.
WUT-107	Decode buffer is empty. Type: Internal Description: A data buffer sent from PL/SQL was empty. Resolution: Contact Oracle.
WUT-108	Error writing to file {0}. Exception: {1} Type: User Description: WebUtil encountered an error trying to write to the specified download file. Resolution: Check that there is enough disk space and that no other process has locked the file.

Error	Description
WUT-109	Error reading from file {0}. Exception: {1} Type: User Description: WebUtil encountered an error trying to read from the specified upload file. Resolution: Check that the file exists and can be accessed.
WUT-110	Database LOB is of zero length. Type: User Description: An attempt was made to download zero bytes from a LOB to the client. Resolution: Ensure that there is data to download.
WUT-111	Database LOB is un-initialized. Type: User Description: An attempt was made to download from a BLOB column that is null. Resolution: Ensure that there is data to download. A commit is required after file transfer (to the database) before exiting the Form to save the changes. A transfer succeeds only if the transfer is done in the same session.
WUT-112	Invalid open mode for BLOB. Value should be W or R. Type: User Description: An invalid open mode was used. Resolution: Use a valid mode.
WUT-113	Too many rows match the supplied where clause. Type: User Description: A call to openBlob() should pass a where clause sufficient to identify one and only one row. Resolution: Correct Where Clause.
WUT-114	SQL Error. Type: User Description: The combination of table name, column name and where clause passed to OpenBlob() did not generate a valid SQL statement. Resolution: See SQL error for details.
WUT-115	Checksum Error. Type: Internal Description: When closing a blob, the size of the input and output files did not match. Resolution: Check for database errors.
WUT-116	Transfer already in progress. Type: User Description: A transfer is already taking place. Only one transfer can be carried out at a time. Resolution: Check that one transfer is complete before starting the next.

Error	Description
WUT-117	Application Server file name cannot be null. Type: User Description: The name of a file to read or write on the application server must be supplied. Resolution: Supply a file name.
WUT-118	Application Server file does not exist or is of zero length. Type: User Description: The file on the application server must exist. Resolution: Supply the name of a valid file.
WUT-119	Error reading data from Application Server file. Type: User Description: The file in the application server could not be read for some reason. Resolution: Consult the exception error information
WUT-120	Zero bytes read from Application Server file. Type: User Description: The file the application server could not be read for some reason. Resolution: Check the file has not been corrupted or removed.
WUT-121	This file transfer has been forbidden by the Administrator. Type: User Description: The Administrator has forbidden file transfers in the webutil.cfg file. Resolution: If the restriction is not deliberate, update your webutil.cfg file with the correct permissions.
WUT-122	Java Functions are not available on the Application Server. Type: User Description: Forms services in this configuration is unable to run mid tier Java code. Resolution: Edit the .env file for this configuration to set the correct path and classpath information to run Java.
WUT-122	The webutil.jar file cannot be found on the Application Server Classpath - Some file transfer functions will not work. Type: User Description: The webutil.jar file contains routines to handle file transfer to and from the ApplicationServer. Without access to this jar file these functions cannot work. Resolution: Correct your classpath in the .env file for this configuration
WUT-123	Unable to open file for writing on the application server. Type: User Description: An exception was raised whilst trying to open a server side file for writing during upload. Resolution: Check the webutil log for more information about the cause of the error

Error	Description
WUT-124	Unable write data on the application server. Type: User Description: An exception was raised whilst trying to write to a server side file during upload. Resolution: Check the webutil log for more information about the cause of the error.
WUT-125	Checksum error closing file on application server. Type: User Description: When an upload to the application server completed the file size was not the value expected. Resolution: Retry the upload operation and check for sufficient disk space / access on the middle tier
WUT-126	Error closing application server file. Type: User Description: An exception was raised whilst trying to close a server side file during upload. Resolution: Check the webutil log for more information about the cause of the error
WUT-127	Unable to create work area. Type: User Description: Webutil failed trying to create a temporary work area. Resolution: Check disk access permissions.
WUT-128	Exception creating work area. Type: User Description: An exception was raised while trying to create a temporary work area. Resolution: Check the WebUtil log for more information about the cause of the error.
WUT-129	Error in reading client file <client_file_name>. Type: User Description: Before the client image is shown in the image item holder, the image is uploaded to a temp directory in the application server. Due to a bug in client_to_AS, an empty file is created if the client file does not exist or not readable. This exception occurs if the client file is not readable. Resolution: Check that the file exists and is readable (has correct permissions and is not corrupt).
WUT-130	Client file name cannot be null. Type: User Description: Occurs when the client file name for file upload or download is null. Resolution: Supply the name of a valid file.

Error	Description
WUT-131	Invalid client file name. Cannot create client file. Type: User Description: occurs when client file cannot be created either due to invalid parent directory or client Java process (or its owner/ user) has no write permission to the parent directory or file. Resolution: Check that you have write privileges to the directory and the specified file.
WUT-132	Invalid Application Server file name. Cannot create application server file. Type: user Description:Occurs when application server file cannot be created either due to invalid parent directory or application server process (or its owner/ user) has no write permission to theparent directory or file. Resolution: Check that you have write privileges to the directory and the specified file.
WUF-200	Unable to open file {0} for read; Exception: {1}. Type: User Description: WebUtil was unable to open the specified file to read it. Resolution: Check that the specified file exists and is not currently locked by another process.
WUF-201	Unable to open file {0} for writing (mode={1}); Exception: {2}. Type: User Description: WebUtil was unable to open the specified file to write to. Resolution: Check that the specified file exists and is not currently locked by another process.
WUF-202	Unable to write to file; Exception: {0}. Type: User Description: WebUtil was unable to write to the specified file. Resolution: Check that the file or media is not read only and the the media has sufficient space.
WUF-203	Unable to read from file; Exception: {0}. Type: User Description: WebUtil was unable to read a line from the specified file. Resolution: Check that the file media has not been removed or the file closed before reading.
WUF-204	Unable to close file; Exception: {0}. Type: User Description: WebUtil was unable to close the specified file. Resolution: Check that the file media has not been removed or the file is already closed.

Error	Description
WUF-205	Invalid value {{0}} supplied for number of Newlines. Type: User Description: The second argument to NEW_LINE should be a positive integer. Resolution: Correct the code that called the function.
WUF-206	Invalid value {{0}} supplied as a File Handle. Type: User Description: The supplied file handle was not a positive integer. Resolution: Ensure that you populate your file handles by opening a file before reading or writing.
WUF-207	Null value supplied as a File Handle. Type: User Description: The supplied file handle was not a positive integer. Resolution: Ensure that you populate your file handles by opening a file before reading or writing.
WUF-208	Too many arguments have been supplied for a call to CLIENT_TEXT_IO.PUTF. Type: User Description: The PUTF function only supports up to 5 substitution variables. Resolution: Only call putf through the supplied PL/SQL interface.
WUF-209	Copy file failed. Type: User Description: After the copy operation the source and target files are not of the same size. Resolution: Check disk space or read permissions on the client machine.
WUI-302	Could not obtain Host Name. Type: Internal Description: WebUtil ClientInfo could not get the name of the host machine from Java. Resolution: Ensure the JAR file containing WebUtil is signed.
WUI-303	Null value supplied as the name of a system property. Type: User Description: When requesting a system property a null value was supplied as its name. Resolution: Request a valid Java System property such as user.name.
WUI-304	An invalid Value was supplied for the name of a system property. Type: User Description: When requesting a system property an invalid property was requested. Resolution: Request a valid Java System property such as user.name. Ensure that the property is in lower case.

Error	Description
WUI-305	Unable to get the user.name property from Java System. Type: Internal Description: Due to security restrictions the Bean was unable to get the username for the current user. Resolution: Ensure the JAR file containing WebUtil is signed.
WUI-306	Security violation getting property. Type: Internal Description: Due to security restrictions the Bean was unable to get the requested property. Resolution: Ensure the JAR file containing WebUtil is signed.
WUH-400	Error reading console InputStream. Exception: {0}. Type: Internal Description: WebUtil was unable to capture the console output generated by a host command. Resolution: Internal Error - no resolution.
WUH-401	Invalid value ({0}) supplied as a Process ID. Type: User Description: The process ID was not a positive integer. Resolution: Make sure that you correctly obtain a Process ID from WebUtil by running a host command before trying to use it.
WUH-402	Null value supplied as a Process ID. Type: User Description: The process ID supplied was null. Resolution: Make sure that you correctly obtain a Process ID from WebUtil by running a host command before trying to use it.
WUH-403	Invalid value ({0}) supplied as an execution mode. Type: User Description: The execution mode was not an integer between 0 and 2 inclusive. Resolution: Only call WebUtil functions through the supplied APIs
WUH-404	Null value supplied for execution mode. Type: User Description: A null value was supplied for the execution mode which should have been 0,1 or 2. Resolution: Only call WebUtil functions through the supplied APIs
WUH-405	Null command supplied. Type: User Description: You tried to execute an empty command string. Resolution: Only call WebUtil functions through the supplied APIs.

Error	Description
WUH-406	Error joining HostCommand Thread; Exception\n{0}. Type: Internal Description: WebUtil was unable to suspend the main PJC Thread while the host command was running. Resolution: Try running the command in callback mode.
WUH-407	OutputSink is null. Type: Internal Description: Could not get the output sink object from the HostCommand. Resolution: This can only happen if the HostCommand object has not yet been executed. Only use the supplied APIs to call WebUtil Functions.
WUH-408	The WebUtil Client code sent back a bad Process ID - Please report this error. Type: Internal Description: A Command completed and dispatched an event to the Form, however, the process handle with the event is incorrect. Resolution: Contact Oracle.
WUH-408	The WebUtil Client code sent back a bad Process ID - Please report this error. Type: Internal Description: A Command completed and dispatched an event to the Form, however, the process handle with the event is incorrect. Resolution: Contact Oracle
WUS-500	Invalid URL supplied for session time out redirect {0}; Exception: {1}. Type: User Description: A URL supplied as the target for a redirect in case of time out was invalid. Resolution: Correct the URL string.
WUO-700	Unable to create the OLE Server {0}; Exception: {1}. Type: User Description: WebUtil was unable to create an instance of the specified OLE Server Application. Resolution: Check that the OLE Server Application works standalone and the name has been specified correctly.
WUO-701	Unable to release the OLE Object Handle; Exception: {0}. Type: User Description: WebUtil was unable to release the specified OLE Object Handle. Resolution: Check that the correct OLE Object Handle has been specified.

Error	Description
WUO-702	Unable to destroy Argument List; Exception: {0}. Type: User Description: WebUtil was unable to destroy the specified Argument List. Resolution: Check that the correct Argument List Handle has been specified.
WUO-703	Unable to add Argument; Exception: {0}. Type: User Description: WebUtil was unable to add the specified Argument to the Argument List. Resolution: Check that the correct Object Handle has been specified for OLE Object type arguments.
WUO-704	Unable to add Argument; Exception: {0}. Type: User Description: WebUtil was unable to add the specified Argument to the Argument List. Resolution: Check that the Argument has been specified correctly.
WUO-705	Unable to invoke Method: {0}; Exception: {1}. Type: User Description: WebUtil was unable to invoke the specified Method. Resolution: Check that the Method and Arguments have been specified correctly
WUO-706	Unable to invoke Method: {0}; Exception: {1}. Type: User Description: WebUtil was unable to invoke the specified Method. Resolution: Check that the Method and Arguments have been specified correctly.
WUO-707	Unable to invoke Method: {0}; Exception: {1}. Type: User Description: WebUtil was unable to invoke the specified Method. Resolution: Check that the Method and Arguments have been specified correctly.
WUO-708	Unable to get Property: {0}; Exception: {1}. Type: User Description: WebUtil was unable to get the value of the specified Property. Resolution: Check that the Property has been specified correctly.
WUO-709	Unable to get Property: {0}; Exception: {1}. Type: User Description: WebUtil was unable to get the value of the specified Property. Resolution: Check that the Property has been specified correctly.

Error	Description
WUO-710	Unable to get Property: {0}; Exception: {1}. Type: User Description: WebUtil was unable to get the value of the specified Property. Resolution: Check that the Property has been specified correctly.
WUO-711	Unable to get Property: {0}; Exception: {1}. Type: User Description: WebUtil was unable to get the value of the specified Property. Resolution: Check that the Property has been specified correctly.
WUO-712	Unable to set Property: {0}; Exception: {1}. Type: User Description: WebUtil was unable to set the specified Property to the specified value. Resolution: Check that the Property and value have been specified correctly.
WUO-713	Unable to set Property: {0}; Exception: {1}. Type: User Description: WebUtil was unable to set the specified Property to the specified value. Resolution: Check that the Property and value have been specified correctly.
WUO-714	Unable to get the last OLE Error details; Exception: {1}. Type: Internal Description: WebUtil was unable to get the details of the last OLE Error. Resolution: Contact Oracle.
WUO-715	Invalid value: {0} specified for OLE Object or Argument List Handle. Type: User Description: The specified OLE Object or Argument List Handle was not a positive integer. Resolution: Check that the OLE Object or Argument List Handle has been specified correctly.
WUO-716	Null value specified for OLE Object or Argument List Handle. Type: User Description: The specified OLE Object or Argument List Handle was not a positive integer. Resolution: Check that the OLE Object or Argument List Handle has been specified correctly.
WUL-901	Could not create an argument list in the Object Cache. Type: Internal Description: WebUtil was unable to create an argument list for passing to a C function. Resolution: Contact Oracle.

Error	Description
WUL-902	Could not remove argument list from Object Cache. Type: User Description: WebUtil was unable to delete an argument list. Resolution: Ensure that you have not already Destroyed this argument list.
WUL-903	Argument could not be added to argument list. Type: User Description: WebUtil was unable to add an argument to an argument list. Resolution: This may be because the argument is NULL - ensure some value is passed.
WUL-904	Supplied handle was not numeric. Type: User Description: An invalid reference to a Function or argument list was passed to WebUtil. Resolution: Only use the supplied APIs to call WebUtil Functions.
WUL-905	Supplied handle was NULL Type: User Description: An invalid reference to a Function or argument list was passed to WebUtil Resolution: Make sure that you create an Argument list or register a function before using a reference to it
WUL-906	Argument could not be re-bound in argument list. Type: User Description: The re-binding of a value to an existing parameter failed. Resolution: Make sure that the value you are re-binding to the parameter is of the correct type.
WUL-907	Fetching argument value from the argument list failed. Type: User Description: WebUtil was unable to get the value of the specified parameter. Resolution: Make sure that the parameter was defined as OUT or IN OUT.
WUL-908	Function could not be registered. Type: Internal Description: WebUtil was unable to create the C Function Object. Resolution: Consult the exception for more information.
WUL-909	Function could not be deregistered. Type: Internal Description: WebUtil was unable to create the C Function Object. Resolution: Consult the exception for more information.

Error	Description
WUL-910	Invoking the function failed. Type: Internal Description: WebUtil was unable to invoke the C Function. Resolution: Consult the exception for more information.
WUL-911	Unsupported return type {0} for C DLL function. Type: User Description: The return type requested was not valid. Resolution: Only use the WebUtil APIs to call this function.
WUL-912	c_char_ptr should have underlying Java instance of CMalloc. Type: Internal Description: Internal Error Resolution: Contact Oracle.
WUL-913	Underlying C DLL function overwrote Java CMalloc object fence post. Type: User Description: The C function that you have called has written more data to one or more of the parameters defined as OUT or IN OUT than the declared length of that variable. Resolution: Consult the API of the C function being called and make sure that your PL/SQL variables and max length definitions are of sufficient size.
WUL-914	Attempting to read unsupported IN/INOUT type parameter. Type: User Description: The parameter is not of a valid type. Resolution: Only use the WebUtil APIs to call this function. Ensure that you pass a valid constant for the parameter type.
WUL-915	Supplied value of bound variable is longer than supplied maximum length. Type: User Description: The maximum length defined for a parameter should be long enough to hold its value. Resolution: Correct the maximum length defined in the code.
WUL-916	Unsupported parameter type for C DLL function. Type: User Description: The parameter is not of a valid type. Resolution: Only use the WebUtil APIs to call this function. Ensure that you pass a valid constant for the parameter type.
WUL-917	Generic c_ptr cannot be used as OUT or INOUT parameter. Only as IN parameter or return value. Type: User Description: The C pointer type cannot be used as an argument type for OUT arguments. Resolution: This is a restriction

Error	Description
WUL-918	Portability error using JNI: C_PTR_LEN does not match underlying host C integral type. Type: User Description: The C_API functions in WebUtil are only supported when calling 32 bit Windows clients. Resolution: These functions are only available on 32 bit windows clients.
