TMS TWebUpdate DEVELOPERS GUIDE

Oct 2023

Web: https://www.tmssoftware.com
Email : info@tmssoftware.com

Table of contents

Welcome

Welcome to the TWebupdate Developer's Guide, created by tmssoftware.com.
At tmssoftware.com, we strive to produce world class software components that enable developers to produce quality software for the most demanding of environments.
Our innovative component suites are designed to be extensible, easy to use and design time rich.
We provide full source code to enable seamless integration of our components with our customers' projects.

TWebUpdate availability

WebUpdate is available as VCL component for Win32/Win64 application development.

VCL versions:
TWebUpdate is available for Delphi 7,2007,2009,2010,XE,XE2,XE3,XE4,XE5,XE6,XE7,XE8,10,10.1,10.2, 10.3, 10.4 and C++Builder 2007,2009,2010,XE,XE2,XE3,XE4,XE5,XE6,XE7,XE8,10,10.1,10.2, 10.3, 10.4.

TWebUpdate use

The TMS TWebUpdate component is designed to make distribution of application updates easier. TWebUpdate can handle network file transfer, FTP or HTTP/HTTPS based distribution methods. Formats for distributing updates can be LZ compressed files, CAB files, ZIP files (supported from XE2), patch difference files and normal files. The application updates can consist of one or multiple executable files as well as other application related data files. A high degree of customization is possible in the setup of the update process. For easy of use, a WebUpdate Wizard is included as well as an UpdateBuilder that creates update process control files.

TWebUpdate organisation

The application update process happens in several steps. Step 1 is obtaining the control file (.INF file) from a network, FTP or HTTP/HTTPS based location. The next step is the processing of the .INF file by verifying new files, new actions, new versions and if necessary downloading the files. Update files that are not part of the executable running process (.EXE, .DLL's) can be immediately extracted to the proper location. If necessary, in the last step, the application is closed, the new EXE and/or DLL files are extracted and the application new version is automatically restarted.
This update process can run automatically or steered programmatically through the TWebUpdate component. As an alternative, the TWebUpdateWizard can be used that is a user-friendly front-end for guiding the user through the update process.
For internet based updates, TWebUpdate is based on the Microsoft WinInet API, available in Win 95 OSRB, Win 98, Win 98SE, Win ME, Windows 2000, Windows XP, Windows 2003, Windows Vista, Windows 7 and Windows 2008.

TWebUpdate setup and methods

Setting the update distribution type

The main selection for TWebUpdate is the distribution method: network, FTP or HTTP/HTTPS based. This is set with the property TWebUpdate.UpdateType : httpUpdate, ftpUpdate, fileUpdate.

In case of FTP and HTTP update distributions, an Internet connection is required. For machines connecting through dial-up, TWebUpdate can automatically try to establish an internet connection. This is controlled by TWebUpdate.UpdateConnect. Various scenarios are possible: not trying to connect, try to connect with prompt, try to silently connect and try to establish a connection with and without automatic hangup.

TWebUpdate can prompt for each file to download, prompt once for first file or download all files silently. The TWebUpdate.UpdateUpdate property controls this.

Setting the update distribution location for network file based updates

For network file based updates, this is set with the URL property, ie:

\\networkserver\sharename\controlfile.inf

Setting the update distribution location for HTTP based updates

For HTTP based updates, this is also set with the URL property, ie:

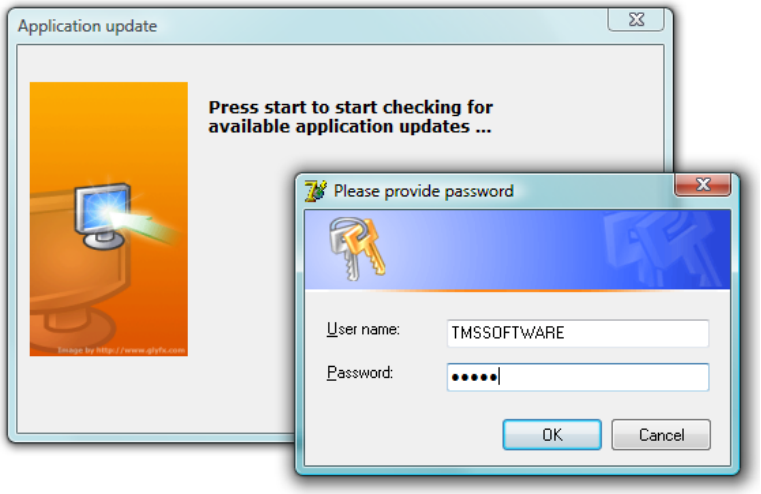
http://www.myserver.com/myupdatefolder/controlfile.inf

Note that no proxy information should be set by default. TWebUpdate uses by default the proxy information as setup in Internet Explorer. If the update is located on a password protected website (via HTTPS), the username and password must be set with:

UserID: website login username
Password: website login password

If username & password cannot be set to a fixed value, TWebUpdate can prompt for this.

Setting TWebUpdate.Authentication to waAlways will cause the TWebUpdate to prompt for username and password. If TWebUpdate.Authentication is set to waAuto, TWebUpdate will only prompt if it finds a password protected website.



Two additional options HTTPKeepAliveAuthentication and ProxyUserID, ProxyUserPassword can be set when required.

Setting the update distribution location for FTP based updates

For FTP based updates, the update control file name is set with the URL property but in addition, the FTP server specifications and login details must be set with:

Host: sets the FTP server name
Port: sets the FTP server port
UserID: FTP login username
Password: FTP login password
FTPDirectory: folder on FTP server where update is located
FTPPassive: true when PASV FTP mode needs to be used

Controlling the update process

Once TWebUpdate is setup for the correct update distribution type, several methods and events are available for starting and controlling the update process.
The full update process is started by

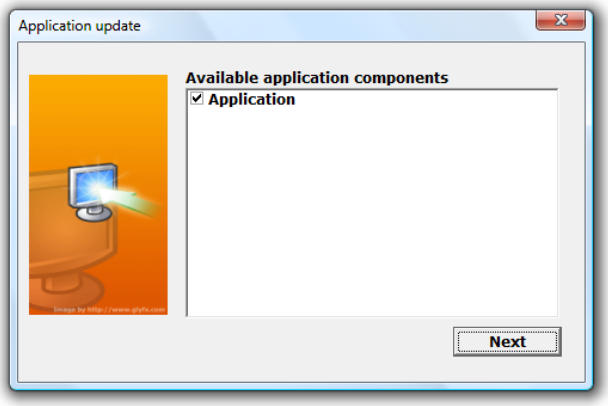
TWebUpdate.DoUpdate; (this performs the update in the main application thread)
or
TWebUpdate.DoThreadUpdate; (this performs the update in a separate thread)

If only a check is required if a new version is available, this can be done with the function:

TWebUpdate.NewVersionAvailable: Boolean;

Using the TWebUpdateWizard

The TWebUpdateWizard provides a user-friendly front-end to guide the user through the update process. To use the TWebUpdateWizard, drop the component on the form, connect the TWebUpdate component to its WebUpdate property and if required, connect a TWebUpdateWizardLanguage component to its language property. Start the wizard by calling TWebUpdateWizard.Execute;



Events during the update process

During the update process, various events are triggered that can be used to steer and monitor the update.

OnBeforeFileDownload(Sender: TObject; FileIdx: Integer; FileDescription: String; var URL: String);

Event triggered before a file is downloaded. The event returns the URL of the file and allows that the URL is dynamically changed through this event. FileIdx is the index of the file to be downloaded, FileDescription is the description found in the update control file. URL is the location found in the control file and can be dynamically changed.

OnBeforeURLToFile(hfile:hinternet;url,tgt:string; var uncompress,handled,success:Boolean);

Event triggered when a valid WININET handle to the download was obtained and that allows to replace the WININET file downloader that is included in TWebUpdate. Set the handled var parameter to true if the download was handled at application level.

OnFileNameFromURL(Sender: TObject; URL: String; var FName: String);

Event triggered to allow dynamically changing a download filename to another filename. If the URL would be http://www.myserver.com/myfile.cab, the download will automatically be named myfile.cab. My using the OnFileNameFromURL event, this can be dynamically renamed to another filename.

OnFileProgress(Sender: TObject; FileName: String; Pos, Size: Integer);

Event triggered for file download progress indication. Pos holds the position of the download in a file of size set by the Size parameter.

OnFileDownloaded(Sender: TObject; FileName: String);

Event triggered when a file has finished downloading.

OnFileVersionCheck(Sender: TObject; NewVersion, LocalVersion: String; var IsNew: Boolean);

Event triggered for custom version checking for each available file in the update. NewVersion holds the new version information (retrieved from the .INF control file customversion= parameter) and LocalVersion the reference (in most cases FileName) from where the local version information can be extracted. The IsNew parameter indicates whether the update is a newer version or not.

Note that the property WebUpdate.VersionCheck can affect this. With the standard setting, VersionCheck = vcUpdateOnly, this means that a file will only be downloaded when there is already an existing version on the machine and the version check determines the web update has a new version.
When VersionCheck = vcAlways, the web update will consider the version newer when there is not yet a local file present and download it and when it is already present on the machine, a regular version check will determine whether to download & install the file or not.

OnProgress(Sender: TObject; Action: String);

Event triggered for each step during the web update. Action indicates what is happening.

OnProgressCancel(Sender: TObject; var Cancel: Boolean);

Event triggered during update to allow cancelling updates while busy. By setting the Cancel parameter to true, the update process is stopped.

OnStatus(Sender: TObject; StatusStr: String; StatusCode, ErrCode: Integer);

Event for status messages during web update. StatusStr shows the status as text, StatusCode returns the reference number of the status and ErrCode returns the type of the error (if an error happened)

Following constants are defined for this:

Constants used for status

```
WebUpdateSuccess         = 0;
WebUpdateAccessError     = 1;
WebUpdateNotFound        = 2;
WebUpdateInformation     = 3;
WebUpdateNoNewVersion    = 4;
WebUpdateNewVersion      = 5;
WebUpdateHTTPStatus      = 6;
WebUpdateHTMLDialog      = 7;
WebUpdateCABError        = 8;
WebUpdateSpawnFail       = 9;
WebUpdateWrongSource     = 10;
WebUpdateSignatureError  = 11;
WebUpdateWhatsNew        = 12;
WebUpdateEUL             = 13;
WebUpdateWhatsnewCancel  = 14;
WebUpdateEULACancel      = 15;
WebUpdatePostConnectFail= 16;
WebUpdatePostPostFail    = 17;
WebUpdateExecAndWait     = 18;
WebUpdateUndefined       = $FF;
```

Constants for errors

```
ErrControlFileNotFound  = 0;
ErrUpdateFileNotFound   = 1;
ErrUpdateFileZeroLen    = 2;
ErrUpdateTargetEqual    = 3;
ErrUpdateSignatureError = 4;
ErrConnectError         = 5;
ErrUndefined            = $FF;
```

OnThreadUpdateDone(Sender: TObject);

Event triggered when the threaded update process is finished

OnAppRestart(Sender: TObject; var Allow: Boolean);

Event triggered before restarting the downloaded new version. Set Allow to false, if restarting the application is not allowed.

OnAppDoClose(Sender: TObject);

Event triggered to allow the application to execute its own close method, otherwise, TWebUpdate will force an application close itself. Note that when OnAppDoClose is assigned and the application is not closed from this event, the update will not automatically restart.

OnCustomValidate(Sender: TObject; Msg, Param: String; var Allow: Boolean);

Event triggered for custom validation of parameters from the update control file. This can be used for example for custom licensing control and disallowing updates to happen for products where the free update period has expired. This event is only triggered when custom validation information is available in the control file.

OnCustomProcess(Sender: TObject; Msg, Param: String);

Event triggered when custom update processing info is present in the control file.

OnGetFileList(Sender: TObject; List: TStringList);

Event triggered to allow user to select from multiple updated application components. The list contains the descriptions of the application components available. By removing descriptions from the list, the corresponding files will not be downloaded in the update process.

OnConvertPrefix(Sender: TObject; var Path: String);

Event triggered to allow user to convert directory prefixes into directory on the system. A directory prefix is the use of {PREFIX} in file paths where PREFIX is dynamically replaced by machine dependent paths. Default supported prefixes are

{WIN} : Windows folder
{SYS} : Windows System32 folder
{SYSWOW64} : Windows SysWOW64 folder
{PF} : Program Files folder
{TMP} : Temporary files folder
{APP} : Application folder
{DOC} : My documents folder

To support additional prefixes, the OnConvertPrefix event can be used.

OnDownloadedEULA(Sender: TObject; Text: TStrings; var Res: Integer);

Event triggered when EULA file found to allow customizing of display of EULA. The EULA is returned in Text and the result of accepting or not accepting the EULA is set in Res. Setting Res to mrOK accepts the EULA.

OnDownloadedWhatsNew(Sender: TObject; Text: TStrings; var Res: Integer);

Event triggered when What's New file found to allow customizing of display of What's New file. Set Res to mrOk to let WebUpdate continue the processing.

OnSuccess(Sender: TObject);

Event triggered when the update process has finished successfully.

OnSetAppParams(Sender: TObject; var AppParams: String);

Event triggered to set the application parameters with which the new version is restarted.

OnProcessPostResult(Sender: TObject; var AllowPostResult: Boolean);

When TWebUpdate posts information to the HTTP server, the result of this POST is stored in TWebUpdate.PostUpdateInfo.PostResult. After this, the event OnProcessPostResult is triggered that can be used check this result and depending on this result, allow the update process to continue or not by setting AllowPostResult to true or false.

TWebUpdate control file

The update control file is an .INI organized file to control the update. Following sections and keywords can be used:

The general update section

1) date based updates:

```
[update]
date=16/2/2007
```

This date is compared with the date of the last update (stored in the registry at a location as specified in the LastURLentry property) If the date is later than the date found in the registry, the update continues.
Notice: the data format can be specified with the DateFormat/DateSeparator property of TWebUpdate to allow both US and Euro dates.

Optionally, a time can be specified as well.

```
[update]
date=16/2/2007
time=14:00
```

When no time is specified, 0:00h is assumed. The time format can be customized with the TWebUpdate TimeFormat/TimeSeparator properties.

2) version based updates:

```
[update]
newversion=1,1,0,0 OR
newsize= OR
newchecksum=
localversion=application.exe
```

The new version (major,minor,release,build) or new filesize or new file checksum (integer number) is compared with the version info, filesize or file checksum of the application as specified with the localversion keyword. If the newversion is newer than the version found in the .EXE (or .DLL) file or

the filesize or file checksum is different, the update continues. Note that a custom compare is also possible when using the OnFileVersionCheck event.

The [update] section can hold the optional key

signature=

This signature value is compared against the TWebUpdate.Signature property when TWebUpdate.SignatureCheck is true. This can be used to check the integrity of the control file. When an incorrect signature is encountered, the OnStatus update is triggered with error parameter ErrUpdateSignatureError.

Actions to do for the update

These actions are specified in the [action] section

```
[action]
updateURL=http://yourserver/update.inf
msg=Automatic web update
query=This is the update version 1.5.0.0 Continue ?
showURL=http://www.yourserver.com/updates/doc.htm
htmldlg=http://www.yourserver.com/updates/dialog.htm
runbefore=
runafter=
```

In the action section following keywords are supported :
- updateURL : will update the URL of the update control file
- msg : simple message displayed during the update
- query : query string allowing the user to cancel the update
- showURL : shows the URL in the default browser
- htmldlg : shows the specified html file at the URL as dialog
- runbefore : run any app. before the update starts
- runafter : run any app. after the updated files are transferred

What's new and EULA options

It is possible to add a reference to a file containing information about what is new in the new version of the application and a file holding the license agreement. This is done in the Whatsnew and EULA sections:

```
[WhatsNew]
file=location of file

[EULA]
file=location of file
```

Example:

```
[WhatsNew]
file=http://www.tmssoftware.com/update/whatsnew.txt
```

Through the TWebUpdate.LanguageID, it is possible to provide different What's new and EULA files for different languages. By setting TWebUpdate.LanguageID to a language specifier, the sections looked for are SectionName + LanguageID.

Suppose a What's new file is available in English and French. The language ID for English is choosen as EN and for French as FR. As such, in the control file, 2 sections are available:

[WhatsNewEN]
file=http://www.tmssoftware.com/update/whatsnew_english.txt

[WhatsNewFR]
file=http://www.tmssoftware.com/update/whatsnew_french.txt

Before calling TWebUpdate.DoUpdate, the French version of the software would set LanguageID to FR and would download the What's new file whatsnew_french.txt while the English version would set the LanguageID to EN and download whatsnew_english.txt

TWebUpdate can display the "What's new" or EULA information in its own dialog box. This dialog box can display text files (.TXT) as well as rich text (.RTF) files. When TWebUpdate is run via the wizard, it can also display this "What's new" or EULA text or rich text in the wizard.

Number of files to update

The number of new files are defined in the sections [files] with the count keyword

[files]
count=2

Details of files to update

[file%] (where % is number of the file starting at 1)
url=URL (HTTP), filename (FTP) or UNC (file based)
newversion=1,0,0,0 **OR**
newsize= **OR**
newchecksum= OR
date=
time=
localversion=application.exe
targetdir=path
descr=your description of the file
compressed=1
filesize=123456
mandatory=0 or 1
hidden=0 or 1

If no "newversion" keyword is present, the update file is always downloaded. If the "newversion", "newsize" or "newchecksum" keyword is used, this new version or new file size or new file checksum is compared with the version info, file size or file checksum (integer number) of the local file defined with the "localversion" keyword.
Finally also a date (and optionally time) comparison can be done against the timestamp of the local file that the new version isi compared against.
The path can be any path and can also contain prefixes : {APP}, {WIN}, {PF} or {TMP}.
{APP} = directory where the current application is running
{WIN} = Windows directory on machine
{PF} = Program Files directory on machine

{TMP} = Temp. directory on machine

If a file must be installed into a subdirectory of the current application running the update, specify {APP}\SUBDIR. If I file must be installed in a subdirectory of the Windows directory use {WIN}\SUBDIR or in the temporary directory with {TMP}\SUBDIR.
Custom prefixes can also be used and these need to be translated to real directories on the system using the OnConvertPrefix event. The 'Descr' details are only used when the OnGetFileList event is used. The OnGetFileList event can be used to query the user for all available updates based on a description rather than the file name. If compressed is 1, this means that the file should be decompressed using the lzexpand or CAB decompress algorithm, if compressed is 2, it means that the file should be decompressed using the ZIP decompress algorithm during program execution, otherwise it is copied. As an alternative, TWebUpdate also automatically recognizes the .CAB/.ZIP extension and if the property ExtractCAB/ExtractZIP is true, these CAB/ZIP files are extracted as well during the update process. Make sure though that no application components (ie .EXE and .DLL files) are decompressed in this step! As application components are still in memory during this phase, these cannot be overwritten. Application components need to be decompressed after program execution and these are specified in the section [Application] which is discussed further.

The FileSize information can be optionally set. This is used for TWebUpdate to calculate the total file size used during progress indication.

Mandatory can be optionally set to 0 or 1. 1 means that the file is mandatory for the update process. When TWebUpdate is used with the TWebUpdatWizard, this file will be displayed in the list of application components but it will not be possible to uncheck it. 0 is the default value.

Hidden can be optionally set to 0 or 1. 1 means that the file is not displayed in the list of files to download in the TWebUpdateWizard but will be downloaded anyway. 0 is the default value.

Updating application components

To overwrite used .EXE and .DLL files of the running application with the updated files, other file with different names than the ones in use must be used. For the files in use, these can be compressed with the standard Microsoft LZ compression program (COMPRESS -r <file>) When updating, the running application is closed, the files are expanded and the running application (update version) is restarted. The appupdate=1 keyword indicates if application used components need to be updated. The application that must be restarted after the update
is indicated with the appname keyword. Finally, the appcomps keyword indicates a series of LZ compressed files that will be expanded during the update process:

```
[application]
appparam=optional commandline parameters for application restart
appupdate=1
appname=inet.exe
appcomps=inet.ex_ other.dl_ onemore.dl_
silentrestart=0 OR 1
```

As an alternative CAB/ZIP files are also supported giving the extra flexibility to have long filesnames for application EXE and DLLs files and multiple files per CAB/ZIP. Keep in mind that ZIP is only supported from XE2. Instead of specifying the LZ compressed filenames, just add the CAB/ZIP file names in the appcomps keyword, ie.

```
[application]
appparam=optional commandline parameters for application restart
appupdate=1
```

```
appname=CABFileUpdateApp.exe
appcomps=newversion.cab

[application]
appparam=optional commandline parameters for application restart
appupdate=1
appname=ZIPFileUpdateApp.exe
appcomps=newversion.zip
```

If no compression is wanted, the application filenames can be extended with _NEW suffix. After download and application shutdown, the files will be renamed without their _NEW suffix, replacing the existing older files.

```
[application]
appparam=optional commandline parameters for application restart
appupdate=1
appname=CABFileUpdateApp.exe
appcomps=CABFileUpdateApp.exe_NEW
```

In case the download of the new application is a setup application that needs to be executed, this can be done with specifying the downloaded setup application as application component. Application components with the .EXE file extension are always treated as an application component that needs to be executed.

```
[application]
appparam=optional commandline parameters for application restart
appupdate=1
appname=MyApp.exe
appcomps=MyAppSetup.exe
```

Finally, advanced control of the update process after the application has been closed is possible with specifying a new .INF file for appcomps, ie:

```
[application]
appparam=optional commandline parameters for application restart
appupdate=1
appname=CABFileUpdateApp.exe
appcomps=extraprocess.inf
```

This .INF file is either also downloaded during the update process or available in the application directory. This .INF file structure is simple and contains a list of update files to process as:

```
[FILES]
COUNT=3
FILE1=mysysdlls.cab
TARGET1={WIN}\System
FILE2=myapp.ex_
TARGET2={APP}
FILE3=mynewdb.db_NEW
TARGET3={APP}
```

After the application has been closed, this extra .INF file will be processed and mysysdlls.cab will be extracted to \Windows\System, myapp.ex_ will be expanded to myapp.exe in the application directory and mynewdb.db_NEW will be renamed to mynewdb.db.

When the flag for silentrestart is set to 1, no message dialog is shown before the application is closed and the new version is started. When this flag is not in the .INF file or zero, the default message confirmation dialog for closing the application is shown.

Additional custom validation options

Custom validation can be done based on entries in the control file as well. This is done through the OnCustomValidate event which passes the values found in the following section of the control file :

```
[custom]
validatemsg=
validateparam=
```

This could for example be used to query for a valid password or license to obtain the update. The update will continue if the customvalidate parameter is true upon return.

Other than validation, custom processing of parameters can be done as well through the OnCustomProcess event in which following info from the control file is passed :

```
[custom]
processmsg=
processparam=
```

TWebUpdate control file examples

This is a date-based network file based update control file with an instruction to use a file difference patcher (like the ASTA patch utility) for updating the running application:

```
[update]
date=2/4/2007

[action]
msg=This is a file based update

[files]
count=1

[file1]
url=\\bmw\newapp\patchsample.pat

[application]
appupdate=1
```

```
appname=patchsample.exe
appcomps=patchsample.pat
```

This is a date-based HTTP based update control file with an instruction to use a file difference patcher (like the ASTA patch utility) for updating a database:

```
[update]
date=2/4/2007

[action]
msg=This is a file based update

[files]
count=1

[file1]
url=http://www.tmssoftware.com/newapp/cars.pat
localversion=cars.mdb

[application]
appupdate=0
```

This is a version based HTTP based update control file with LZ-compressed updated EXE and a few actions:

```
[update]
newversion=1,1,0,0
localversion=sampapp.exe

[action]
msg=Found updated sample app.
query=Proceed with automatic update ?
htmldlg=http://www.tmssoftware.com/test.htm

[files]
count=2

[file1]
url=http://www.tmssoftware.com/SAMPAPP.EX_
descr=Application

[file2]
url=http://www.tmssoftware.com/SAMPAPP.BMP
descr=Application logo

[application]
appupdate=1
appname=sampapp.exe
appcomps=sampapp.ex_
```

This is a version based HTTP based update control file with CAB-compressed updated EXE, updated applications DLL's and update database files processed after closing the application:

```
[update]
newversion=1,1,0,0
localversion=myapp.exe

[action]
msg=Found updated application.

[files]
count=4

[file1]
url=http://www.tmssoftware.com/MYAPP.CAB
descr=Application
targetdir={APP}

[file2]
url=http://www.tmssoftware.com/MYAPPDLL.CAB
descr=Application dll's
targetdir={WIN}\System32

[file3]
url=http://www.tmssoftware.com/MYAPPDB.CAB
descr=Application Database
targetdir={APP}\DB

[file4]
url=http://www.tmssoftware.com/process.INF
descr=Extra
mandatory=1
hidden=1

[application]
appupdate=1
appname=my.exe
appcomps=process.inf
```

with process.INF:

```
[FILES]
COUNT=3
FILE1=MYAPP.CAB
FILE2=MYAPPDLL.CAB
FILE3=MYAPPDB.CAB
```

TWebUpdate debugging

In case something is not working as desired, it is often convenient to check what steps TWebUpdate has executed. This can be traced by setting TWebUpdate.Logging = true. During execution,

TWebUpdate will create a log file of all steps performed in the file WUPDATE.LOG (default filename or can be changed with TWebUpdate.LogFileName)
If no path is specified, the log file will by default be created in the "My documents" folder. The log file can contain following error codes:

ERROR_FTP_DROPPED 12111
The FTP operation was not completed because the session was aborted.
ERROR_FTP_NO_PASSIVE_MODE 12112
Passive mode is not available on the server.
ERROR_FTP_TRANSFER_IN_PROGRESS 12110
The requested operation cannot be made on the FTP session handle because an operation is already in progress.
ERROR_GOPHER_ATTRIBUTE_NOT_FOUND 12137
The requested attribute could not be located.
ERROR_GOPHER_DATA_ERROR 12132
An error was detected while receiving data from the Gopher server.
ERROR_GOPHER_END_OF_DATA 12133
The end of the data has been reached.
ERROR_GOPHER_INCORRECT_LOCATOR_TYPE 12135
The type of the locator is not correct for this operation.
ERROR_GOPHER_INVALID_LOCATOR 12134
The supplied locator is not valid.
ERROR_GOPHER_NOT_FILE 12131
The request must be made for a file locator.
ERROR_GOPHER_NOT_GOPHER_PLUS 12136
The requested operation can be made only against a Gopher+ server, or with a locator that specifies a Gopher+ operation.
ERROR_GOPHER_PROTOCOL_ERROR 12130
An error was detected while parsing data returned from the Gopher server.
ERROR_GOPHER_UNKNOWN_LOCATOR 12138
The locator type is unknown.
ERROR_HTTP_COOKIE_DECLINED 12162
The HTTP cookie was declined by the server.
ERROR_HTTP_COOKIE_NEEDS_CONFIRMATION 12161
The HTTP cookie requires confirmation.
ERROR_HTTP_DOWNLEVEL_SERVER 12151
The server did not return any headers.
ERROR_HTTP_HEADER_ALREADY_EXISTS 12155
The header could not be added because it already exists.
ERROR_HTTP_HEADER_NOT_FOUND 12150
The requested header could not be located.
ERROR_HTTP_INVALID_HEADER 12153
The supplied header is invalid.

ERROR_HTTP_INVALID_QUERY_REQUEST 12154
The request made to HttpQueryInfo is invalid.
ERROR_HTTP_INVALID_SERVER_RESPONSE 12152
The server response could not be parsed.
ERROR_HTTP_NOT_REDIRECTED 12160
The HTTP request was not redirected.
ERROR_HTTP_REDIRECT_FAILED 12156
The redirection failed because either the scheme changed (for example, HTTP to FTP) or all attempts made to redirect failed (default is five attempts).
ERROR_HTTP_REDIRECT_NEEDS_CONFIRMATION 12168
The redirection requires user confirmation.
ERROR_INTERNET_ASYNC_THREAD_FAILED 12047
The application could not start an asynchronous thread.
ERROR_INTERNET_BAD_AUTO_PROXY_SCRIPT 12166
There was an error in the automatic proxy configuration script.
ERROR_INTERNET_BAD_OPTION_LENGTH 12010
The length of an option supplied to InternetQueryOption or InternetSetOption is incorrect for the type of option specified.
ERROR_INTERNET_BAD_REGISTRY_PARAMETER 12022
A required registry value was located but is an incorrect type or has an invalid value.
ERROR_INTERNET_CANNOT_CONNECT 12029
The attempt to connect to the server failed.
ERROR_INTERNET_CHG_POST_IS_NON_SECURE 12042
The application is posting and attempting to change multiple lines of text on a server that is not secure.
ERROR_INTERNET_CLIENT_AUTH_CERT_NEEDED 12044
The server is requesting client authentication.
ERROR_INTERNET_CLIENT_AUTH_NOT_SETUP 12046
Client authorization is not set up on this computer.
ERROR_INTERNET_CONNECTION_ABORTED 12030
The connection with the server has been terminated.
ERROR_INTERNET_CONNECTION_RESET 12031
The connection with the server has been reset.
ERROR_INTERNET_DECODING_FAILED 12175
WinINet failed to perform content decoding on the response. For more information, see the Content Encoding topic.
ERROR_INTERNET_DIALOG_PENDING 12049
Another thread has a password dialog box in progress.
ERROR_INTERNET_DISCONNECTED 12163
The Internet connection has been lost.
ERROR_INTERNET_EXTENDED_ERROR 12003
An extended error was returned from the server. This is typically a string or buffer containing a verbose error message. Call InternetGetLastResponseInfo to retrieve the error text.

ERROR_INTERNET_FAILED_DUETOSECURITYCHECK 12171
The function failed due to a security check.
ERROR_INTERNET_FORCE_RETRY 12032
The function needs to redo the request.
ERROR_INTERNET_FORTEZZA_LOGIN_NEEDED 12054
The requested resource requires Fortezza authentication.
ERROR_INTERNET_HANDLE_EXISTS 12036
The request failed because the handle already exists.
ERROR_INTERNET_HTTP_TO_HTTPS_ON_REDIR 12039
The application is moving from a non-SSL to an SSL connection because of a redirect.
ERROR_INTERNET_HTTPS_HTTP_SUBMIT_REDIR 12052
The data being submitted to an SSL connection is being redirected to a non-SSL connection.
ERROR_INTERNET_HTTPS_TO_HTTP_ON_REDIR 12040
The application is moving from an SSL to an non-SSL connection because of a redirect.
ERROR_INTERNET_INCORRECT_FORMAT 12027
The format of the request is invalid.
ERROR_INTERNET_INCORRECT_HANDLE_STATE 12019
The requested operation cannot be carried out because the handle supplied is not in the correct state.
ERROR_INTERNET_INCORRECT_HANDLE_TYPE 12018
The type of handle supplied is incorrect for this operation.
ERROR_INTERNET_INCORRECT_PASSWORD 12014
The request to connect and log on to an FTP server could not be completed because the supplied password is incorrect.
ERROR_INTERNET_INCORRECT_USER_NAME 12013
The request to connect and log on to an FTP server could not be completed because the supplied user name is incorrect.
ERROR_INTERNET_INSERT_CDROM 12053
The request requires a CD-ROM to be inserted in the CD-ROM drive to locate the resource requested.
ERROR_INTERNET_INTERNAL_ERROR 12004
An internal error has occurred.
ERROR_INTERNET_INVALID_CA 12045
The function is unfamiliar with the Certificate Authority that generated the server's certificate.
ERROR_INTERNET_INVALID_OPERATION 12016
The requested operation is invalid.
ERROR_INTERNET_INVALID_OPTION 12009
A request to InternetQueryOption or InternetSetOption specified an invalid option value.
ERROR_INTERNET_INVALID_PROXY_REQUEST 12033
The request to the proxy was invalid.
ERROR_INTERNET_INVALID_URL 12005
The URL is invalid.
ERROR_INTERNET_ITEM_NOT_FOUND 12028
The requested item could not be located.

ERROR_INTERNET_LOGIN_FAILURE 12015
The request to connect and log on to an FTP server failed.
ERROR_INTERNET_LOGIN_FAILURE_DISPLAY_ENTITY_BODY 12174
The MS-Logoff digest header has been returned from the Web site. This header specifically instructs the digest package to purge credentials for the associated realm. This error will only be returned if INTERNET_ERROR_MASK_LOGIN_FAILURE_DISPLAY_ENTITY_BODY has been set.
ERROR_INTERNET_MIXED_SECURITY 12041
The content is not entirely secure. Some of the content being viewed may have come from unsecured servers.
ERROR_INTERNET_NAME_NOT_RESOLVED 12007
The server name could not be resolved.
ERROR_INTERNET_NEED_MSN_SSPI_PKG 12173
Not currently implemented.
ERROR_INTERNET_NEED_UI 12034
A user interface or other blocking operation has been requested.
ERROR_INTERNET_NO_CALLBACK 12025
An asynchronous request could not be made because a callback function has not been set.
ERROR_INTERNET_NO_CONTEXT 12024
An asynchronous request could not be made because a zero context value was supplied.
ERROR_INTERNET_NO_DIRECT_ACCESS 12023
Direct network access cannot be made at this time.
ERROR_INTERNET_NOT_INITIALIZED 12172
Initialization of the WinINet API has not occurred. Indicates that a higher-level function, such as InternetOpen, has not been called yet.
ERROR_INTERNET_NOT_PROXY_REQUEST 12020
The request cannot be made via a proxy.
ERROR_INTERNET_OPERATION_CANCELLED 12017
The operation was canceled, usually because the handle on which the request was operating was closed before the operation completed.
ERROR_INTERNET_OPTION_NOT_SETTABLE 12011
The requested option cannot be set, only queried.
ERROR_INTERNET_OUT_OF_HANDLES 12001
No more handles could be generated at this time.
ERROR_INTERNET_POST_IS_NON_SECURE 12043
The application is posting data to a server that is not secure.
ERROR_INTERNET_PROTOCOL_NOT_FOUND 12008
The requested protocol could not be located.
ERROR_INTERNET_PROXY_SERVER_UNREACHABLE 12165
The designated proxy server cannot be reached.
ERROR_INTERNET_REDIRECT_SCHEME_CHANGE 12048
The function could not handle the redirection, because the scheme changed (for example, HTTP to FTP).

ERROR_INTERNET_REGISTRY_VALUE_NOT_FOUND 12021
A required registry value could not be located.
ERROR_INTERNET_REQUEST_PENDING 12026
The required operation could not be completed because one or more requests are pending.
ERROR_INTERNET_RETRY_DIALOG 12050
The dialog box should be retried.
ERROR_INTERNET_SEC_CERT_CN_INVALID 12038
SSL certificate common name (host name field) is incorrect—for example, if you entered www.server.com and the common name on the certificate says www.different.com.
ERROR_INTERNET_SEC_CERT_DATE_INVALID 12037
SSL certificate date that was received from the server is bad. The certificate is expired.
ERROR_INTERNET_SEC_CERT_ERRORS 12055
The SSL certificate contains errors.
ERROR_INTERNET_SEC_CERT_NO_REV 12056
ERROR_INTERNET_SEC_CERT_REV_FAILED 12057
ERROR_INTERNET_SEC_CERT_REVOKED 12170
SSL certificate was revoked.
ERROR_INTERNET_SEC_INVALID_CERT 12169
SSL certificate is invalid.
ERROR_INTERNET_SECURITY_CHANNEL_ERROR 12157
The application experienced an internal error loading the SSL libraries.
ERROR_INTERNET_SERVER_UNREACHABLE 12164
The Web site or server indicated is unreachable.
ERROR_INTERNET_SHUTDOWN 12012
WinINet support is being shut down or unloaded.
ERROR_INTERNET_TCPIP_NOT_INSTALLED 12159
The required protocol stack is not loaded and the application cannot start WinSock.
ERROR_INTERNET_TIMEOUT 12002
The request has timed out.
ERROR_INTERNET_UNABLE_TO_CACHE_FILE 12158
The function was unable to cache the file.
ERROR_INTERNET_UNABLE_TO_DOWNLOAD_SCRIPT 12167
The automatic proxy configuration script could not be downloaded. The INTERNET_FLAG_MUST_CACHE_REQUEST flag was set.
ERROR_INTERNET_UNRECOGNIZED_SCHEME 12006
The URL scheme could not be recognized, or is not supported.
ERROR_INVALID_HANDLE
The handle that was passed to the API has been either invalidated or closed.

Using TWebUpdate with C++Builder

In order to use TWebUpdate in C++Builder applications, add following lines to the main project CPP file:

```
#pragma link “wininet.lib”
#pragma link “w32inet.lib”
```

Using TMS UpdateBuilder

TMS UpdateBuilder is an application that makes building update control files easier. It will also automatically upload files to a server and it can generate new control files automatically based on versions of application files by simply running the TMS UpdateBuilder.
TMS UpdateBuilder can be downloaded from http://www.tmssoftware.com/site/wupdate.asp

The use of UpdateBuilder is explained in a separate guide.

Using TWebUpdateWizard

Using TWebUpdate with the TWebUpdateWizard is straightforward. Drop the TWebUpdate and TWebUpdateWizard on the form, setup TWebUpdate and assign TWebUpdate to the TWebUpdateWizard.WebUpdate property. The wizard can be started by calling TWebUpdateWizard.Execute.

Additional options for the TWebUpdateWizard are:

AutoRun: when true, does not require the user to step through each step
AutoStart: when true, the user does not have to start the update process, it starts automatically
Billboard: sets the left image for the update wizard dialog
BillboardCenter: Boolean: When true, the image is centered
BillboardStretch: Boolean: When true, the billboard image is stretched
BillboardTop: integer: sets the top position of the billboard image
BillboardLeft: integer: sets the left position of the billboard image
BillboardWidth: integer: sets the width of the billboard image
BillboardHeight: integer: sets the height of the billboard image
BorderStyle: sets the border style for the update wizard dialog
Caption: sets the caption text for the update wizard dialog
Font: sets the font for the update wizard dialog
Language: sets the language for the update wizard dialog. By default, the language is English.
Position: sets the screen position of the update wizard dialog

Writing custom TWebUpdateWizard translation components

It is very simple to create a translated version of the TWebUpdateWizard by providing a TWebUpdateWizardLanguage component. To write a custom version, create a class that descends from TWebUpdateWizardLanguage, override the constructor and provided the translated texts.

As an example, this is the source code for the dutch translated version:

```
constructor TWebUpdateWizardDutch.Create(AOwner: TComponent);
begin
  inherited;
  Welcome := 'Druk start om te beginnen met controleren voor applicatie
updates ...';
  StartButton := 'Start';
  NextButton := 'Volgende';
  ExitButton := 'Verlaten';
  CancelButton := 'Annuleren';
  RestartButton := 'Herstarten';
  GetUpdateButton := 'Update';
  NewVersionFound := 'Nieuwe version gevonden';
  NewVersion := 'Nieuwe versie';
  NoNewVersionAvail := 'Geen nieuwe versie beschikbaar.';
  NewVersionAvail := 'Nieuwe versie beschikbaar.';
  CurrentVersion := 'Huidige versie';
  NoFilesFound := 'Geen bestanden gevonden voor update';
  NoUpdateOnServer := 'geen update gevonden op server ...';
  CannotConnect := 'Er kan geen verbinding met de update server tot stand
gebracht worden of';
  WhatsNew := 'Nieuw in update';
  License := 'Licentie overeenkomst';
  AcceptLicense := 'Ik aanvaard';
  NotAcceptLicense := 'Ik aanvaard niet';
  ComponentsAvail := 'Beschikbare applicatie componenten';
  DownloadingFiles := 'Downloaden bestanden';
  CurrentProgress := 'Vooruitgang huidig bestand';
  TotalProgress := 'Totale vooruitgang';
  UpdateComplete := 'Update volledig ...';
  RestartInfo := 'Druk Herstarten om de nieuwe versie te starten.';
  WhatsNewPopup := 'Bekijken met kladblok';
  LicensePopup := 'Bekijken met kladblok';
end;
```