## Introduction

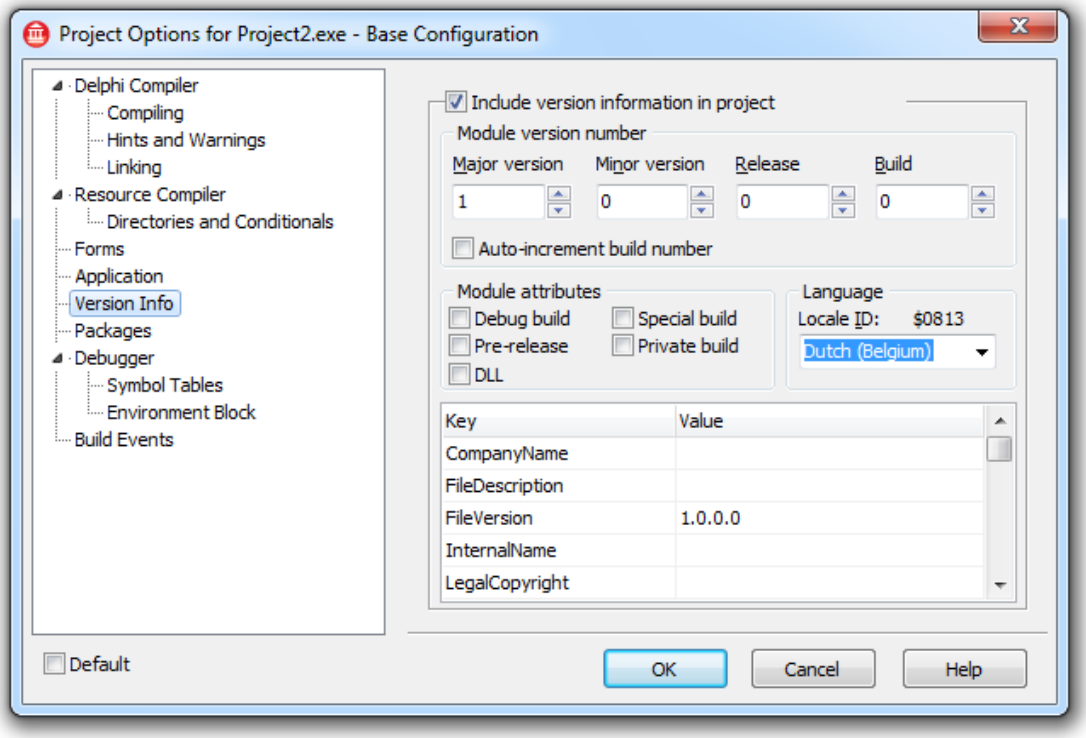
In most modern desktop software applications, the ability to have seamless automatic updates of the application itself or application data is a prerequisite. We want that this process is as transparent as possible for the user. Typically that means that applications figures out at startup whether an update for the application or for the data is available. There are some fundamental problems though with this ideal approach. First of all, every since Windows existed, it is not possible to overwrite an executable code file that is running in memory. So, this restriction affects for example EXE or DLL files. Secondly, with the introduction of Windows Vista, the "Program Files" folder has become restricted territory. It requires elevation to admin rights to be able to write into this folder. At TMS software, we have developed a component TwebUpdate and a companion application UpdateBuilder that make it very simple to add auto updating capabilities for applications or application data. TWebUpdate conveniently solves the fundamental restrictions imposed by Windows for self-updating applications in the "Program Files" folder.

## TWebUpdate basic features

TMS TWebUpdate has a wide range of capabilities to offer a really flexible solution for automatic updates. It can retrieve updates from a HTTP server, a FTP server or via network file transfer for internal business applications for example. It can perform a test whether a new version is available based on a version info resource, a date, file checksum or custom versioning. It also supports CAB compressed files to reduce file sizes for network transfer. Optionally, it also offers a wizard to guide an end user to various steps of the update process including showing "What's new" information or an updated "EULA".

## Getting started with TWebUpdate & UpdateBuilder

This step by step walk-through shows how easy it is to add auto-updating capabilities for an application. Start with dropping TWebUpdate on your application's main form. As in this example, the update will be based on the version number stored in the version info resource, go to Project Options, select to include Version Info and specify the version number as 1.0.0.0.



In the form's OnCreate event, TWebUpdate will check if a new version is available with following code:

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  WebUpdate1.URL := 'http://www.tmssoftware.net/public/project1.inf';
  if WebUpdate1.NewVersionAvailable then
  begin
    ShowMessage('A new version of this application is available');
  end;
end;
```

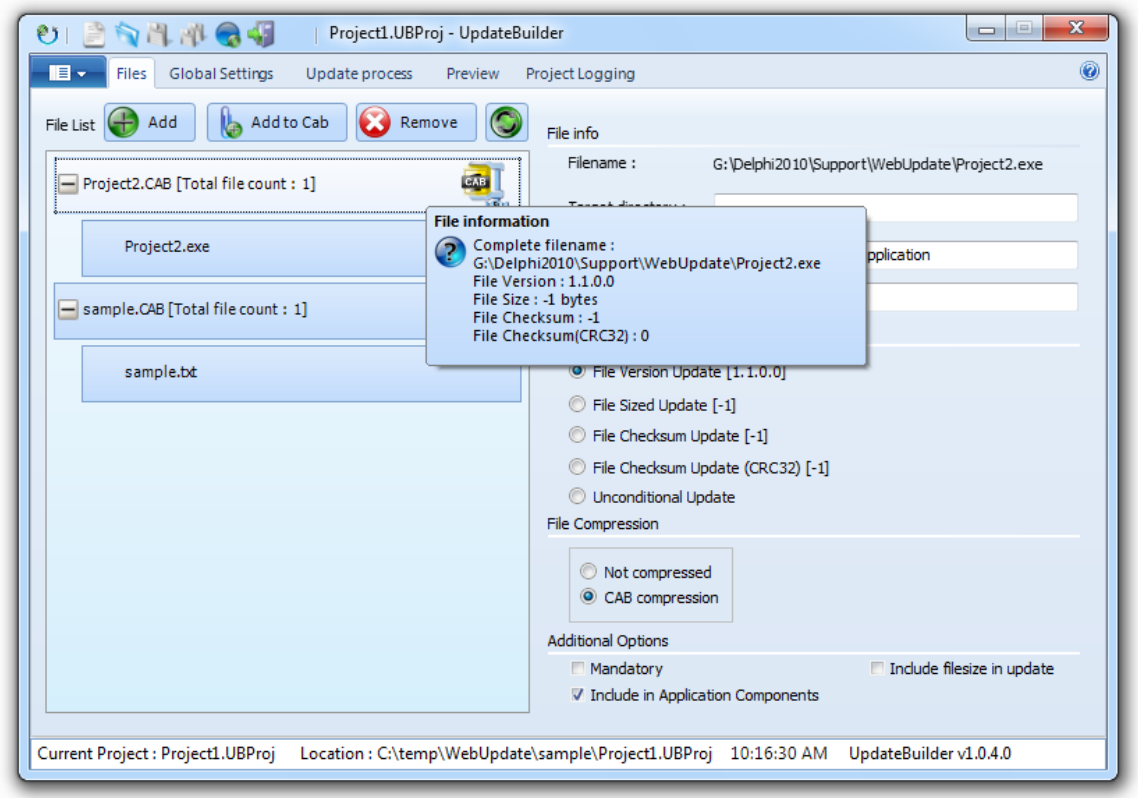
First we set the URL where TWebUpdate gets the update instructions via the default HTTP protocol and the function NewVersionAvailable returns true when a newer version is available on the server.

To do the actual update process, we could use the code:

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  WebUpdate1.URL := 'http://www.tmssoftware.net/public/project1.inf';
  WebUpdate1.DoUpdate(true);
end;
```

The TWebUpdate.DoUpdate() method performs the full update process. By default, when a new update is available, TWebUpdate will prompt to continue downloading the update, will perform the download and then prompt to close & restart the application's new version.

To create and upload the update itself and the update control file (the .INF file), UpdateBuilder was used.




First step here is to add in the Files list the executable, in this case PROJECT1.EXE. UpdateBuilder prompts to ask if this is the main executable to update. Respond by selecting "Yes". For this file, select "File Version Update" under "File Update" and select "CAB compression" under "File Compression" in the properties. Note that UpdateBuilder has already automatically extracted the version information from the executable file. When you will open the project again at a later time, UpdateBuilder will always automatically extract the latest

version info. Next, specify under "Global Settings" in the Upload tab the FTP account information of the server where to upload the file. To finish making the update available, go to the "Update Process" tab and select "Build project". You can verify the generated .INF file under "Preview" and for a basic update process, this is:

```
[update]
newversion=1.0.0.0
localversion=Project2.exe
[files]
count=1
[file1]
url=http://www.tmssoftware.net/public/Project2.CAB
newversion=1.0.0.0
localversion=Project2.exe
compressed=0
[application]
appupdate=1
appname=project2.exe
appcomps=Project2.CAB
```

### Using a wizard to guide the user through the update process

TMS TWebUpdate comes with an optional ready to use wizard component to help the user through the steps of updating an application.

To start using the wizard, drop TWebUpdateWizard on the form. By default, the wizard is in English. If you need another language, select any of the WebUpdateWizard language files, for example TWebUpdateWizardDutch. Assign this language component to WebUpdateWizard.Language. Also assign the instance of TWebUpdate to WebUpdateWizard.WebUpdate. The wizard is started with following code:

```
begin
  WebUpdate1.URL := 'http://www.tmssoftware.net/public/project1.inf';
  WebUpdateWizard1.Execute(true);
end;
```

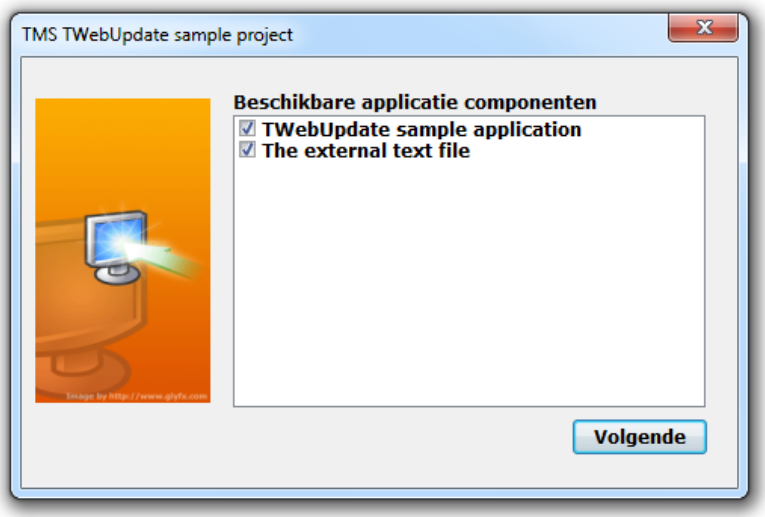
### Logging

For debugging or trouble-shooting purposes, TWebUpdate can automatically log all steps during an update process. The log file is created by setting WebUpdate.Logging := true. The default location where WebUpdate.log is generated is in the "\My Documents" folder. To change the location and/or filename of the generated log file, change this with the property WebUpdate.LogFileName: string.

### Putting it all together

As you could see, the basic steps to create a self updating application are really simple. Let's move to creating a more real-life situation with creating an update process that shows "What's new" info and can update both data and application.

To demonstrate this, a simple Delphi application is used with a memo control that displays a TXT file. This TXT file is an external file shipped with the application. We want that TWebUpdate can update the executable and/or the TXT file.




Start UpdateBuilder and add both the executable and the updated text file. As the text file has no version info resource, we select under "File Update" properties "File Checksum Update (CRC32)". This means that a CRC32 checksum of the local file will be compared with the new version and when different, it is considered an update. We also specify a description under "File Info" for both text file and executable. This description will be shown to the user in the wizard to inform what updated parts are available. We select that both files are delivered in CAB compressed format and are application components. This means that if new versions of any of these two files are available, the application will be restarted. Under "Global Settings", "Commands", we also specify the "What's new file".

As the update should be done whenever any of these two files are new, we set under "Global Settings" the criteria for update as date based. This means that TWebUpdate will always compare the date of the update with the last update date stored in the registry. The location where this is stored in the registry is set under WebUpdate.LastURLEntry.

The code used to start this update with the wizard from a button is:

```
begin
  WebUpdate1.LastURLEntry.Key := 'tmssoftware/WebUpdate';
  WebUpdate1.LastURLEntry.Section := 'UpdateDate';
  WebUpdate1.LastURLEntry.Save := true;
  WebUpdate1.URL := 'http://www.tmssoftware.net/public/project1.inf';
  WebUpdate1.Logging := true;
  WebUpdate1.LogFileName := GetCurrentDir + '\webupdate.log';
  WebUpdateWizard1.Execute(true);
end;
```

## Conclusion

We hope this brief overview of TWebUpdate made clear that the component takes all complexity out of the self updating process. The TWebUpdate component has many more advanced capabilities like custom version verification, URL redirection, using patch files with 3rd party binary patch tools, downloading updates to different target folders and much more ... TWebUpdate comes with a developer guide that explains all these extra features. We hope you can make good use of it to make the delivery of updates to your customers a lot more efficient and trouble-free. Free trial download of TMS TWebUpdate & TMS UpdateBuilder can be found at: http://www.tmssoftware.com/site/wupdate.asp