Manifest Generator
User instructions

Purpose and description

Manifest Generator is used to generate manifest.xml files that are needed for Tekla Structures Extension Packages (TSEP). You can fill all needed data and have a fully functional manifest file, or you can leave some or all fields empty and manually edit the manifest file as needed.
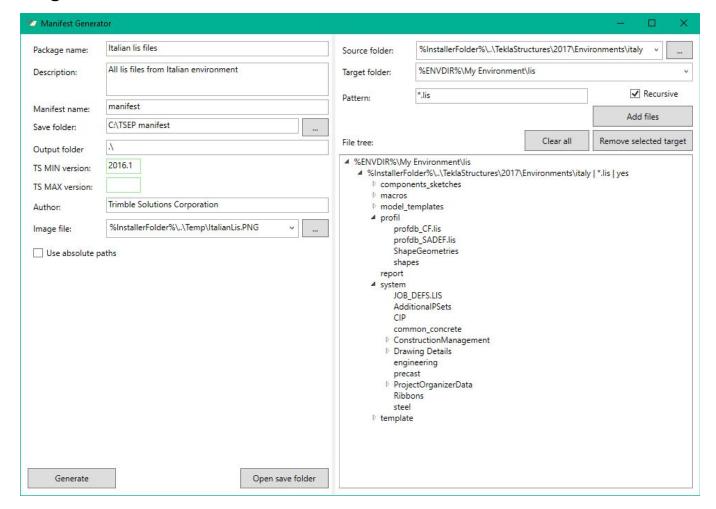
System requirements

.NET Framework version: 4.5

Installation

Copy the executable and Defaults.ini file to any suitable location and run the tool separately

Usage

Workflow:

1. Fill in the common information

   ○ Package name: used as both name and id. For id the special characters and spaces are automatically converted to underscores ('_')

   ○ Description: free text describing the package

   ○ Manifest name: name of the manifest file. The .xml file extension is added automatically

   ○ Save folder: the folder where the manifest file will be created

      ■ Changing save folder will empty the image file and source folder fields if "Use absolute paths" is selected. Also prompts if file tree should be emptied, since the files might not be found anymore

   ○ Output folder: path where package builder will place the .tsep file relative to manifest file location. ".\" or empty means same folder where manifest file is located, "..\" means the folder one level higher in hierarchy. Can contain longer path definitions as well, e.g. "..\output\"

   ○ TS MIN/MAX version: version number in any of formats 2016.1, 2016i, 2017, 2017.0. MAX version is only needed when package installation to newer versions need to be limited, so it can be left empty.

   ○ Author: Author or manufacturer of the package

   ○ Image file: location of a .png image file which will be used as the package thumbnail

      ■ Using the shortcut %InstallerFolder% will point to the Save folder, available from the drop down menu

   ○ Use absolute paths: use absolute paths for image file and source folder. If unselected, paths are relative to save folder, if possible, using the %InstallerFolder% shortcut. Relative paths are needed when the package is built by e.g. TeamCity.

2. Select source files and target folders

   ○ Source folder: select which folder and its contents will be added to the package

      ■ Using the shortcut %InstallerFolder% will point to the Save folder, available from the drop down menu

   ○ Target folder: select suitable shortcut from the dropdown menu and complete it as needed, e.g. %ENVDIR%\ → %ENVDIR%\common to place package contents to common environment folder.

   ○ Pattern: files matching the pattern will be included in the package. Normal Windows wildcards asterisk (*) and question mark (?) can be used. "*.*" means all files, "?eam.txt" means files that end with "eam.txt" but can start with any single character, like beam.txt and seam.txt. Single filenames are possible as well

   ○ Recursive: if checked, will go through all the subfolders as well. Must be off for single files (done automatically)

3. Press "Add files" button to add a new node to File tree. May take awhile if there are lots of files

4. Remove entire target-source pairs by pressing "Remove selected target" button if needed, or empty the whole list by pressing "Clear all" button.

5. Repeat steps 2 – 4 as needed

6. Press "Generate" to have the manifest file created to the save folder

Defaults.ini

Default values in Defaults.ini file are grouped by keywords inside brackets. Under each key word is the default value definition. `[target folders]` defines several target folder shortcuts but for other values only the last one is used, so only one value is useful.
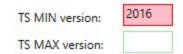
File contents used in the screenshots:

```
[author]
Trimble Solutions Corporation

[target folders]
%ENVDIR%\
%DefaultExtensionsDir%\
%Desktop%\
%ENVDIR%\..\bitmaps\
%ENVDIR%\common\system\
%usimpEnvFolder%\system\
%usmetEnvFolder%\system\
%franceEnvFolder%\system\

[output folder]
.\

[manifest name]
manifest

[manifest folder]
C:\TSEP manifest
```
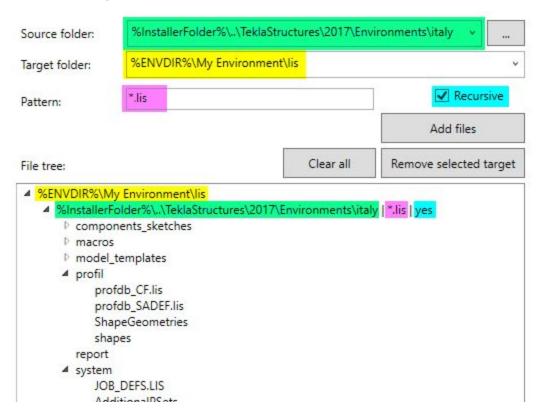
Common information

Version numbers are checked automatically. If the number field is red, then the version number is incorrect. The smallest allowed version is 2016i or 2016.1 which mean the same version. The biggest allowed version is 2099i or 2099.1. In the manifest file only .1 and .0 style is allowed, and e.g. 2017 will automatically be converted to 2017.0 in file creation.

2016 is invalid version number, so it is highlighted. 2016i (2016.1) is the smallest valid version
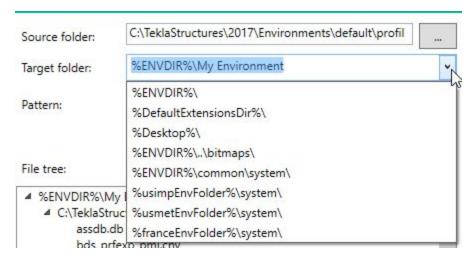
Source and target folders

In the file tree the highest level shows the target folder. Under that is the source folder, pattern and recursive state. Under source is the list of files and subfolders that will be included in the package.

Target folder shortcuts are read from file Defaults.ini under section `[target folders]`. %ENVDIR% points to the folder where all environments are installed, %DefaultExtensionsDir% points to extensions folder and %Desktop% to Windows desktop. In addition there is possibility to use %<environment name>EnvFolder% shortcuts that point directly to defined environment, e.g. %franceEnvFolder% points to the French environments folder, which is same as %ENVDIR%\france. These shortcuts are case sensitive (the text between % signs), but the subfolder names are not.

Good to know

If you need to use relative paths for source and image files (e.g. for TeamCity usage), you can use the %InstallerFolder% shortcut to point to the location where the manifest file will be saved (Save folder). The shortcut is available in the dropdown menus. Also, if "use absolute paths" is not selected, using the dialogs to select image file and source folder will convert to use the %InstallerFolder% shortcut if possible.

Changing the save folder will empty the fields when "use absolute paths" is not selected, since %InstallerFolder% will point to different location. For file list there will be a question if user wants to empty it as well.

The file list doesn't necessarily represent the contents of the created package. It all depends on when and on which computer the package will eventually be created. It is possible to add folders and network drives that don't exist, but then there will be a warning about it. User can add the missing folder or drive to the file list and manifest file, which can be helpful if packages are built on a different computer environment, e.g. TeamCity.