ionCube Package Foundry
User Guide 4.1

ionCube®

ionCube.com

Table of Contents

1 Introduction

The ionCube Package Foundry is an easy to use, flexible set of tools for creating cross-platform web installers. These installers use a built in, transparent FTP client to upload PHP applications to remote servers, or to install files to a local machine. Although IPF is built around PHP, it is possible to install things such as purely static websites as well.

The latest version of this documentation is always available online

IPF has the following major features:

- **Built-in application configuration tool**

After installation, an optional configuration file can be used to allow the user to set custom values in a text file that could then be read by PHP. In the example project, the configuration file is used to set an image width in a static html page.

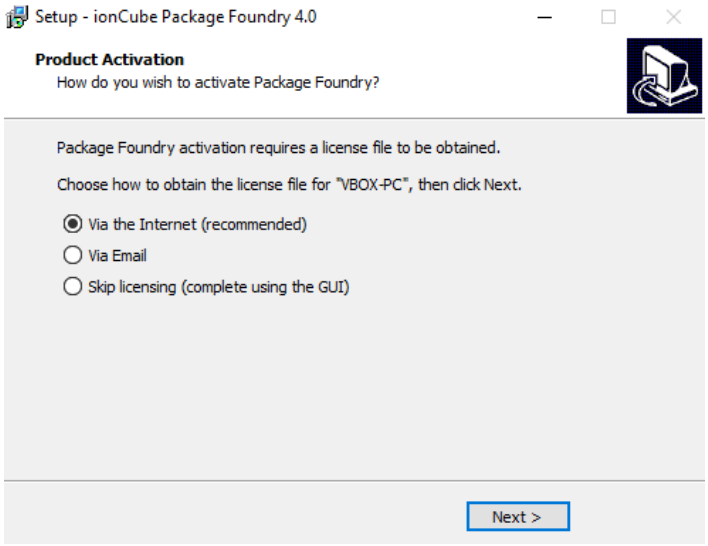
- **Elegant "Browse For FTP Folder" Control**
- **Support for multiple languages**
- **SSL support for sites using HTTPS**
- **Familiar installer GUI**
- **Easy to use GUI front end to IPF**
- **Create UNIX style permission structures**
- **Automatic retrieval and installation of ionCube Loaders (if encoded files used)**
- **Local or remote installation**
- **Detection of server capabilities to help upload files in the most efficient manner**

IPF will detect the capabilities of the server it is uploading to and use that knowledge to be more efficient. For example, if possible IPF will download the ionCube PHP Loader relevant to your environment straight to your remote server. If this is not possible then it will download first to your local machine and then upload to remote.

- **End-User contact details and log file (optionally) sent if problems occur during installation**
- **Option to use multiple parallel FTP connections to greatly increase FTP throughput**
- **Automatically detects whether to use active or passive FTP mode**
- **Support for multiple install targets**
- **Customisable images**
- **Wildcard pattern matching to ignore files when creating a package**
- **Option to set an expiry date at which point the package will refuse to install**
- **Gzip and Zip compression**

1.1 Activation

To help protect against piracy the ionCube Package Foundry uses license files which contain machine-specific data including data related to network cards. A license file `lic.txt` is used as verification. On Linux this file should be located in the root directory of the application, on Windows this file can be found in `C:/Users/Public/Documents/ionCube/Package Foundry/lic.txt`

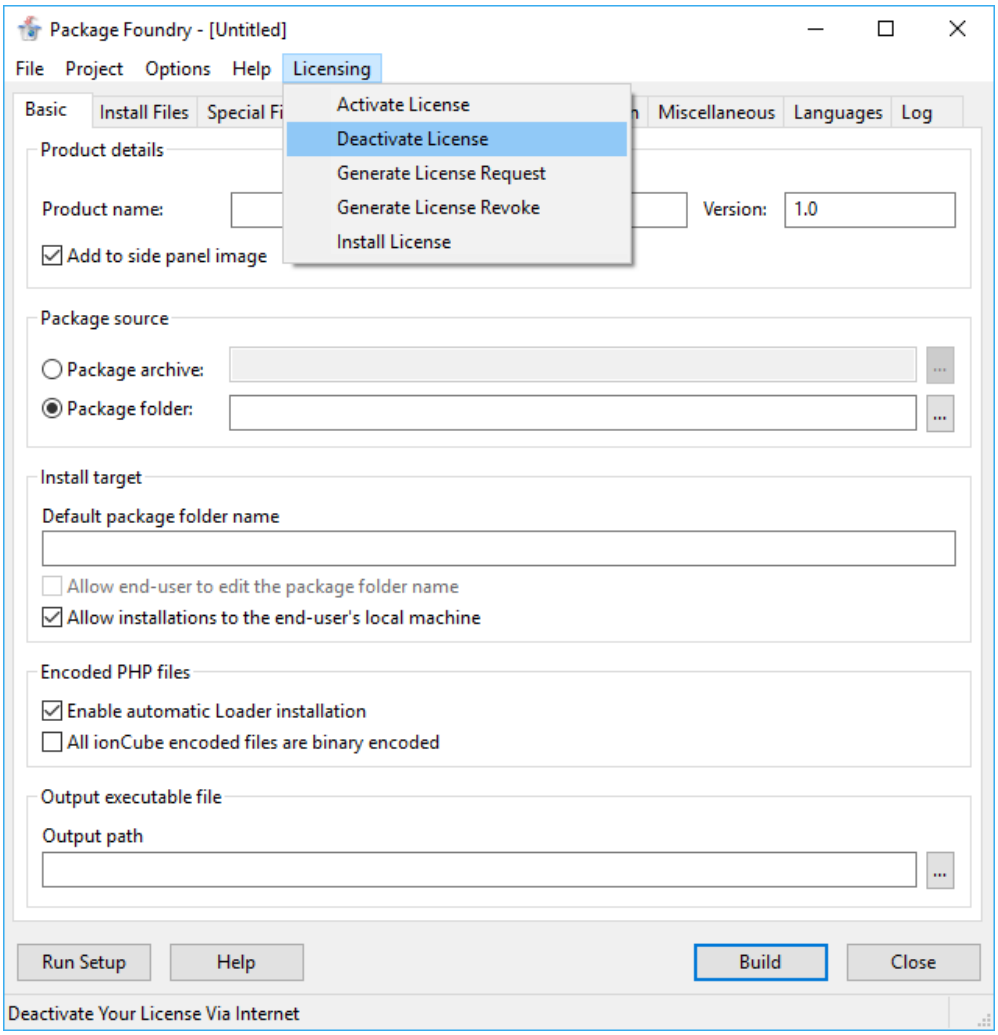


License activation can be done in a number of ways:

During installation the installer will allow you to activate your license. Internet based licensing will automate the process and place the license file in the correct location.

Email based licensing will produce a request file which can then be emailed to licenses@ioncube.com and a license file will be emailed back. This license file can automatically be put in the correct place using the `Licensing->Install License` File option inside the GUI. Using the uninstaller packaged with the application will also revert this process and de-activate your license. Alternatively you can skip licensing at the current time and do it at a later date via either the GUI or the command line tool.

Licensing actions can also be done from the GUI:



The GUI exposes ways to use internet via licensing for activation and de-activation, email based licensing for creating both request and revoke files which can be emailed to licenses@ioncube.com. When receiving a license file from internet based licensing, you can use the `Install License` feature to automatically put it in the correct place.

These methods are also exposed via the command line with -- options. These are:

- `activate`
- `deactivate`
- `gen-license-request`
- `gen-license-revoke`

2 Installing The Example Project

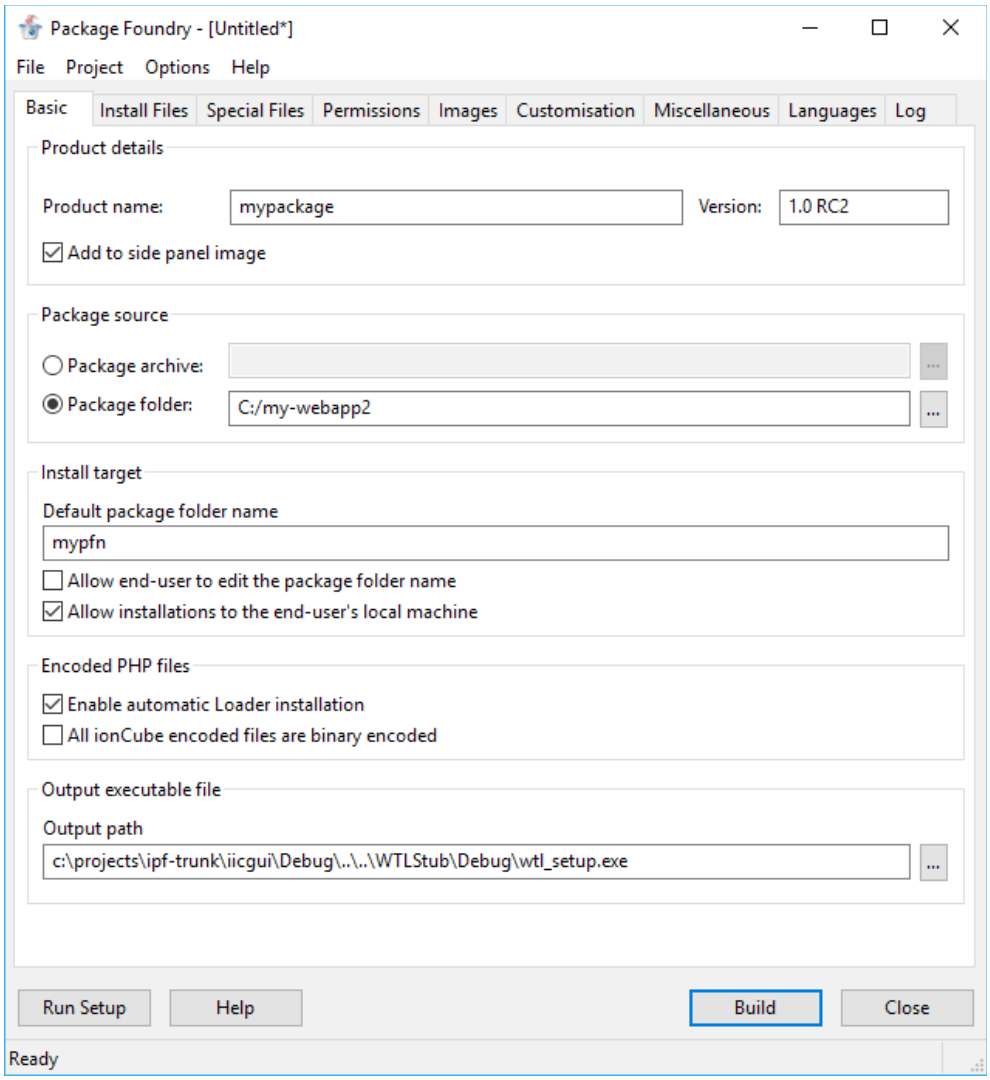
This short guide will lead you through the creation of an example installation package, explaining all the steps taken and various options available throughout in short detail. More detailed explanations of the various components of IPF can be found further in the documentation.

2.1 Package Foundry Scripts

PFS files are used to store configuration options when building a package. While all these options can be added one by one to the IPF command line, PFS files allow an easier to use system to store settings. The one used in this example will be inside the Samples folder called `Example-Project.pfs`. PFS scripts are explained in more detail here

2.2 Using The GUI

Running IPF from it's Windows Start Menu item will bring up the IPF GUI



Firstly, go to **File → Open** and select the `Example-Project.pfs` file mentioned above, this will read all the settings inside the script and set them accordingly in the GUI. The GUI offers a quicker and simpler way to edit PFS scripts without having to worry about typos or errors. Feel free to look around the various tabs in the GUI at this point to get acquainted with it and edit any sections you like.

When you're done making adjustments click the Build button. Internally this will call `ipf.exe` (the command line tool) and pass it all the options selected in the GUI. You should now have an executable file located wherever the GUI has it's `Output Path` option set to under the Basic tab.

For more information on the IPF GUI and it's available options, you can read the IPF section.

2.3 The Setup Executable

The executable created by IPF will be the file you distribute to anyone wanting to install your package. It contains all the files that were selected as the package folder/archive as well as the core program needed to unpack them and perform remote setup.

The installer will follow a path of well defined steps, separated by different pages within the installation wizard:

Any introduction & license files will be displayed to the user, with the license file needing the user to select an “I accept” button to continue.

After initial introduction, Install method and location will be selected. As of version 4.0, IPF supports local installs and FTP/SFTP remote installs. A remote directory browser is included as part of remote install.

The installer will then ask for the HTTP address of the server if available, this is used for querying the necessary data for ionCube Loader transfer, if selected as part of the package options.

A summary of information and connection options are then displayed to the user, giving them a chance to confirm that everything is correct. They can backtrack at this point to correct any invalid options.

Installation is then started with a progress bar displaying overall progress.

After installation has completed, any post-install scripts will be run, the ionCube Loader will be downloaded(if options selected) and then if any configuration files exist, the configuration window will open.

The configuration window allows users to go through configuration files, using an attached code editor with PHP/YAML/JSON syntax highlighting for more complex files. After the user has finished editing the given configuration files, they can commit their changes, overwriting the files on the remote server (or simply saving the file locally).

Running through the setup executable, installing, and configuring your package should give you a good example of a creating a more basic package. As well as our Example-Project, IPF also ships with a PFS script for **SquirrelMail**, a web-based email client. This is a working example of IPF in action and includes all the various configuration/readme files to set up a functional version of SquirrelMail with relative ease.

3 IPF Command-Line Tool

3.1 Basic Usage

While it is recommended to use the IPF GUI for everyday use, you can use the command line IPF tool.

The command line tool is invoked on windows by:

```
ipf.exe --config <path to PFS file>
```

and on Linux by:

```
ipf--config <path to PFS file>
```

For the full version of IPF, if a `lic.txt` file is not in the default licensing location (C://Users/Public/Documents/ionCube/Package Foundry) then it should be specified with `--ipf-license`. `--ipf-license` is only available on Windows versions of IPF

Options can also be given to the command line tool that do not exist in the pfs script with `--<option name>`. In a scenario where there is a conflict between an option in the pfs script and the command-line, the command-line option takes precedence. This allows custom packages to be made "on the fly"

A full list of the available command line options can be viewed on the next page.

3.1.1 ipf vs ipf64

The Linux version of IPF ships with 2 executables, `ipf` and `ipf64`. `ipf64` is built for `x86_64` machines and should be used on 64 bit versions of Linux whereas `ipf` should be used on 32 bit architecture (e.g `i686`). If unsure, you can check your architecture by running "`arch`" in a terminal window.

3.2 Command-Line Options

`-V | --version`	Displays IPF version information.
`-v | --verbose`	Output verbose messages.
`-h | --help`	Display command-line help.
`--config <path>`	Specify an IPF .pfs script file.
`--activate`	Activate a license on your account automatically via the internet
`--deactivate`	Deactivate a license on your account automatically via the internet
`--gen-license-request`	Generate a license request file to be emailed to licenses@ioncube.com
`--gen-license-revoke`	Generate a license revoke file to be emailed to licenses@ioncube.com
`--ipf-license <path>`	Specify the path to the IPF license file. Defaults to `C:/Users/Public/Documents/ionCube/Package Foundry/lic.txt`. This option is only available on Windows
`-o <output .exe>`	This item sets the path of the Package Foundry's output file.
`--package-name <name>`	This item sets the name of the package, as used throughout the package installer.

`--default-package-folder-name <name>`	Default name for the top-level package folder after installation. This can be omitted.
`--package-dir <path>`	Either a folder or a file (see below) must be specified containing the files comprising the application to be installed.
`--package-file <path>`	If the application has already been compressed with a supported compression method as described below, the path of the archive may be specified.
`--package-version <version>`	The product version may be specified with this item. The default value is '1.0'.
`--compression-level <level>`	When using gzip compression to compress a package folder a compression level may be specified from 0 to 9. Level 9 gives the best compression but is slower than level 1. Level 6 is a reasonable value. Level 0 specifies that no compression should be used.
`--license-file <path>`	If this item is used the specified file will be displayed during the installation process.
`--ignore-file <path>`	The ignore file is a line seperated list of wildcard patterns (using * and ?) that can be used to ignore certain files or folders in a package when building it, for example a `.gitignore` file or tests folder.
`--expire-date <date>`	Declares a date at which point opening the executable file will display a message that the package has expired and they need to ask the package distributor for a new one. Accepts ISO 8601 formatting (`YYYY-mm-dd`)
`--readme-file <path>`	If this item is used the use will have the option of displaying the readme file at the end of the installation process.
`--intro-file <path>`	An introduction can be shown after the license agreement (or start page, if the license in not present). This introduction may contain installation instructions or other information. Either the path of a file can specified, or a URL.
`--intro-file-external`	If this option is set, the intro file will be launched externally using the Windows file association. For example, an html file will be launched in the default web browser. If this option is not set then the intro file will be displayed in the installer itself.

`--use-default-permissions`	Specifies that files and folders should be set to the permissions specified by the options `--default-file-mode and --default-folder-mode` instead of given the default file and folder permissions assigned by the server. This will slow down package installation as a chmod command must be issued for each uploaded file and would rarely be required. In a configuration file, use the item `use-server-permissions 0` to enable this functionality.
`--default-file-mode <mode>`	When compressing a package archive with gzip compression this setting specifies the default permission mode to use for those files not specified with `chmod-package`. This option only applies to the Windows version of IPF, as on Linux permissions can be easily set for the package folder using the shell. The default value is 644.
`--default-folder-mode <mode>`	When compressing a package archive with gzip compression this setting specifies the default permission mode to use for those folders not specified with `chmod-package`. This option only applies to the Windows version of IPF, as on Linux permissions can be easily set for the package folder using the shell. The default value is 755.
`--disable-permissions-warning`	The setting of permissions will only work if the chmod command is supported on the target FTP server. If disable-permission-warning is not set, then a warning will be displayed if permissions cannot be set. The files and permissions specified with `chmod-package` will be listed. The warning is only displayed if at least one item is specified with `chmod-package`.
`--disable-local-installs`	If a product will only be installed to remote servers, and never to an end-user's local machine, then this option can be used to remove the 'Installation Type' wizard step. This option would be useful, for example, if the creator of the package was intending to use the package to install to a handful of their own remote servers. If a product is for general release then usually this option should not be used.
`--allow-pfn-edit`	When this option is set to 1 the end-user will be able to edit the package folder name.
`--hide-product-info`	When this option is set to 1 the product name and version will not be displayed over the side panel image.
`--main-page <path>`	This item can be set to the main page of the package. The Configuration Tool will then launch this page in the user's default browser when they click **Launch**.
`--image-side-panel <path>`	This item enables customisation of the side panel image on the first page and last pages of the package installer. The image must have file extension .png, .jpeg, .jpg, .bmp, or .gif, and be a valid image of the corresponding image format.

`--image-motif <path>`	This item enables customizations of the image appearing on the top right of most pages in the package installer wizard. As with the side panel image, this image must be in Portable Network Graphics format.
`--disable-auto-loader-install`	By default, Loaders will be installed automatically if they are not already installed on the target server. Use this option to control whether to enable the automatic installation of Loaders.
`--all-encoded-file-binary`	Use this option to specify whether all files in the application which are encoded with the ionCube Encoder have been encoded in binary mode.
`--install-script-url <path>`	Set this item to the URL of a page that should be launched in the user's default web browser after installation is complete. The URL is relative to the folder the user uploads the application into.
`--post-install-php`	Specifies the relative URL to the post-install php script. This file URL is relative to the installation folder of the package. This is required for a post-install php script to run. (E.g `scripts/post-install.php`)
`--post-install-php-interactive`	This item specifies whether the post-install script requires user-interaction. If so, then the page will be launched in a browser, otherwise it will be executed silently by the Package Installer.
`--run-post-install config`	Sometimes a post-install script should be run at the end of the installation wizard, and sometimes it should be run after configuration files have been saved to the server. Use the value `config` to specify that the post-install script should run after configuration, and any other value to specify that it should run after file upload and any Loader installation.
`--stub-version <version>`	Use this option to customise the version number of the installer executable file.
`--stub-product <product name>`	Use this option to customise the product name in the installer executable file's version info resource.
`--stub-copyright <copyright>`	Use this option to customise the copyright message in the installer executable file's version info resource.
`--stub-description <description>`	Use this option to customise the product description in the installer executable file's version info resource.

`--stub-comments <comments>`	Use this option to customise the comments field in the installer executable file's version info resource.
`--stub-icon <path>`	Use this option to customise the application icon of the installer executable file's version info resource.
`--with <plugin>`	IPF has a modular architecture, and this option can be used to specify which installed plugins to enable. There are currently no officially supported plug-ins. *The old SSL plugin for HTTPS support is now enabled by default*
`--language-pack <locale>`	Language packs can be included with the installer by using this option. End-users whose system locale to an included locale will see messages displayed in their language. See the 'locale' subfolder of the IPF installation folder for the available language packs.
`--default-language <locale>`	If the end-user's current system locale is not available as an included language pack, unlocalised (English) messages will be displayed by the installer. Specify this option to use a different language pack as the default.
`--commmon-documents <path>`	If this option is used then the common documents folder will be searched for unlocalised versions of install files (readme, license etc). Otherwise IPF will search relative to the current .pfs script.
`--localised-documents <path>`	This option sets the root of the localised documents tree. Subfolders are named according to language codes (e.g. en), or language and country codes (e.g. en_GB). Localised versions of install files are then added to the appropriate language subfolder.
`--active-help-url <URL>`	This option sets the target URL for the **Help** button on IPF created installers.

4 Package Foundry Scripts

Package Foundry scripts have the extension .pfs and are used to store settings for the GUI, and to store directives for the IPF command-line tool. Each line in a .pfs file contains a directive for the Package Foundry. Blank lines are ignored, and text on a line after the symbol `#` is a comment and ignored.

Relative paths in the script file are relative to the directory the script file is contained in, unless otherwise stated in the following sections. Absolute paths are also allowed.

All items in the script file have the form

```
item-name arg1, arg2, ...
```

where most items take a single argument but some take a second. Arguments containing spaces need to be double-quoted

PFS file options will be documented in the following pages.

4.1 Mandatory Items

Item: output-file

Usage: `output-file C:\setup.exe`

This item sets the path of the Package Foundry's output file.

Item: package-name

Usage: `package-name "Example Application"`

This item sets the name of the package, as used throughout the package installer.

Item: package-dir

Usage: `package-dir "../my package"`

Either a folder or a file (see below) must be specified containing the files comprising the application to be installed.

Item: package-file

Usage: `package-file package.tar.gz`

If the application has already been compressed with a supported compression method as described below, the path of the archive may be specified. Archive file's use their base folder as the current directory for configuration files and the main index page. More detail is given in 4.2 Basic Settings

4.2 Optional Items

Item: default-package-folder-name

Usage: `default-package-folder-name "WebEdit"`

Default name for the top-level package folder after installation. This can be omitted.

Item: product-version

Usage: `product-version "1.0 RC8"`

The product version may be specified with this item. The default value is '1.0'.

Item: compression-level

Usage: `compression-level 7`

When using gzip compression to compress a package folder a compression level may be specified from 0 to 9. Level 9 gives the best compression but is slower than level 1. Level 6 is a reasonable value. Level 0 specifies that no compression should be used.

Item: license-file

Usage: `license-file license.txt`

If this item is used the specified file will be displayed during the installation process.

Item: ignore-file

Usage: `ignore-file ipf.ignore`

The ignore file is a line separated list of wildcard patterns (using * and ?) that can be used to ignore certain files or folders in a package when building it, for example a .gitignore file or tests folder.

Item: expire-date

Usage: `expire-date 2030-01-27`

Declares a date at which point opening the executable file will display a message that the package has expired and they need to ask the package distributor for a new one. Requires ISO 8601 formatting (`YYYY-mm-dd`).

Item: readme-file

Usage: `readme-file readme.txt`

If this item is used the use will have the option of displaying the readme file at the end of the installation process. This can be a txt, pdf, or HTML file and Windows will open it using the associated program.

Item: intro-file

Usage: `intro-file intro.html`

An introduction can be shown after the license agreement (or start page, if the license in not present). This introduction may contain installation instructions or other information. Either the path of a file can specified, or a URL.

Item: intro-file-external

Usage: `intro-file-external 1`

If this option is set, the intro file will be launched externally using the Windows file association. For example, an html file will be launched in the default web browser. If this option is not set then the intro file will be displayed in the installer itself.

Item: configurable-file

Usage: `configurable-file config.txt`

After installation of a product it is often necessary to edit text files in order to configure the application. Installers created with the Package Foundry can automate this task by downloading, editing and uploading configuration files. You can specify (possibly multiple) such files using the configurable-file item.

Item: chmod-package

Usage: `chmod-package o-w file.php`

If a folder is to be used as the package source, and the compression type is set to `gzip`, permissions can be specified for files or subfolders in the package. These permissions override the permissions of the source file or folder and are used when installing the package either locally or via FTP. This is especially useful on Windows as it enables one to create a package with UNIX-style permissions. Permissions can be specified either as a list as in `g+r, o-w`, or in octal as in `660`.

Item: use-server-permissions

Usage: `use-server-permissions 1`

If the package is a folder rather than an archive this option specifies that file and folder permissions should not be edited once the item is uploaded to the target FTP server. If this option is set to 0 then permissions will be modified according to the default-file-mode and default-folder-mode options as described below. The default value for this option is 1.

Item: default-file-mode

Usage: `default-file-mode 606`

When compressing a package archive with gzip compression this setting specifies the default permission mode to use for those files not specified with chmod-package. This option only applies to the Windows version of IPF, as on Linux permissions can be easily set for the package folder using the shell. The default value is 644.

Item: default-folder-mode

Usage: `default-folder-mode 755`

When compressing a package archive with gzip compression this setting specifies the default permission mode to use for those folders not specified with `chmod-package`. This option only applies to the Windows version of IPF, as on Linux permissions can be easily set for the package folder using the shell. The default value is 755.

Item: disable-permissions-warning

Usage: `disable-permissions-warning 0`

The setting of permissions will only work if the `chmod` command is supported on the target FTP server. If disable-permission-warning is not set, then a warning will be displayed if permissions cannot be set. The files and permissions specified with `chmod-package` will be listed. The warning is only displayed if at least one item is specified with `chmod-package`.

Item: disable-local-installs

Usage: `disable-local-installs 1`

If a product will only be installed to remote servers, and never to an end-user's local machine, then this option can be used to remove the `Installation Type` wizard step. This option would be useful, for example, if the creator of the package was intending to use the package to install to a handful of their own remote servers. If a product is for general release then usually this option should not be used.

Item: allow-pfn-edit

Usage: `allow-pfn-edit 0`

When this option is set to 1 the end-user will be able to edit the package folder name.

Item: hide-product-info

Usage: `hide-product-info 1`

When this option is set to 1 the product name and version will not be displayed over the side panel image.

Item: main-page

Usage: `main-page index.php`

This item can be set to the main page of the package. The Configuration Tool will then launch this page in the user's default browser when they click Launch.

Item: image-side-panel

Usage: `image-side-panel my-panel.png`

This item enables customisation of the side panel image on the first page and last pages of the package installer. The image must have file extension `.png`, `.jpeg`, `.jpg`, `.bmp`, or `.gif`, and be a valid image of the corresponding image format.

Item: image-motif

Usage: `image-motif motif.png`

This item enables customizations of the image appearing on the top right of most pages in the package installer wizard. As with the side panel image, this image must be in Portable Network Graphics format.

Item: auto-install-loaders

Usage: `auto-install-loaders 1`

By default, Loaders will be installed downloaded if they are not already installed on the target server. Use this option to control whether to enable the automatic donwnload of Loaders.

Item: all-encoded-files-binary

Usage: `all-encoded-files-binary 0`

Use this option to specify whether all files in the application which are encoded with the ionCube Encoder have been encoded in binary mode.

Item: post-install-php

Usage: `post-install-php my-app/readme.php`

Set this item to the URL of a page that should be launched in the user's default web browser after installation is complete. The URL is relative to the folder the user uploads the application into.

Item: post-install-php-interactive

Usage: `post-install-php-interactive 1`

This item specifies whether the post-install script requires user-interaction. If so, then the page will be launched in a browser, otherwise it will be executed silently by the Package Installer.

Item: run-post-install

Usage: `run-post-install config`

Sometimes a post-install script should be run at the end of the installation wizard, and sometimes it should be run after configuration files have been saved to the server. Use the value `config` to specify that the post-install script should run after configuration, and any other value to specify that it should run after file upload and any Loader installation.

Item: stub-version

Usage: `stub-version 1.2.0.1`

Use this option to customise the version number of the installer executable file.

Item: stub-product

Usage: `stub-product "My Product"`

Use this option to customise the product name in the installer executable file's version info resource.

Item: stub-copyright

Usage: `stub-copyright "Copyright ionCube 2019"`

Use this option to customise the copyright message in the installer executable file's version info resource.

Item: stub-description

Usage: `stub-description "IPF installer product"`

Use this option to customise the product description in the installer executable file's version info resource.

Item: stub-comments

Usage: `stub-comments "Beta build"`

Use this option to customise the comments field in the installer executable file's version info resource.

Item: stub-icon

Usage: `stub-icon C:\my_icon.ico`

Use this option to customise the application icon of the installer executable file's version info resource.

Item: with

Usage: `with myPlugin`

IPF has a modular architecture, and this option can be used to specify which installed plugins to enable. There are no currently supported official plugins

Item: add-language-pack

Usage: `add-language-pack de`

Language packs can be included with the installer by using this option. End-users whose system locale to an included locale will see messages displayed in their language. See the `locale` subfolder of the IPF installation folder for the available language packs.

Item: default-locale

Usage: `default-locale de`

If the end-user's current system locale is not available as an included language pack, unlocalised (English) messages will be displayed by the installer. Specify this option to use a different language pack as the default.

Item: commmon-documents

Usage: `commmon-documents C:\docs`

If this option is used then the common documents folder will be searched for unlocalised versions of install files (readme, license etc). Otherwise IPF will search relative to the current `.pfs` script.

Item: localised-documents

Usage: `localised-documents C:\locale`

This option sets the root of the localised documents tree. Subfolders are named according to language codes (e.g. en), or language and country codes (e.g. `en_GB`). Localised versions of install files are then added to the appropriate language subfolder.

Item: active-help-url

Usage: `active-help-url http://www.mysite.com/install-help/index.php`

This option sets the target URL for the Help button on IPF created installers.

5 IPF GUI For Windows

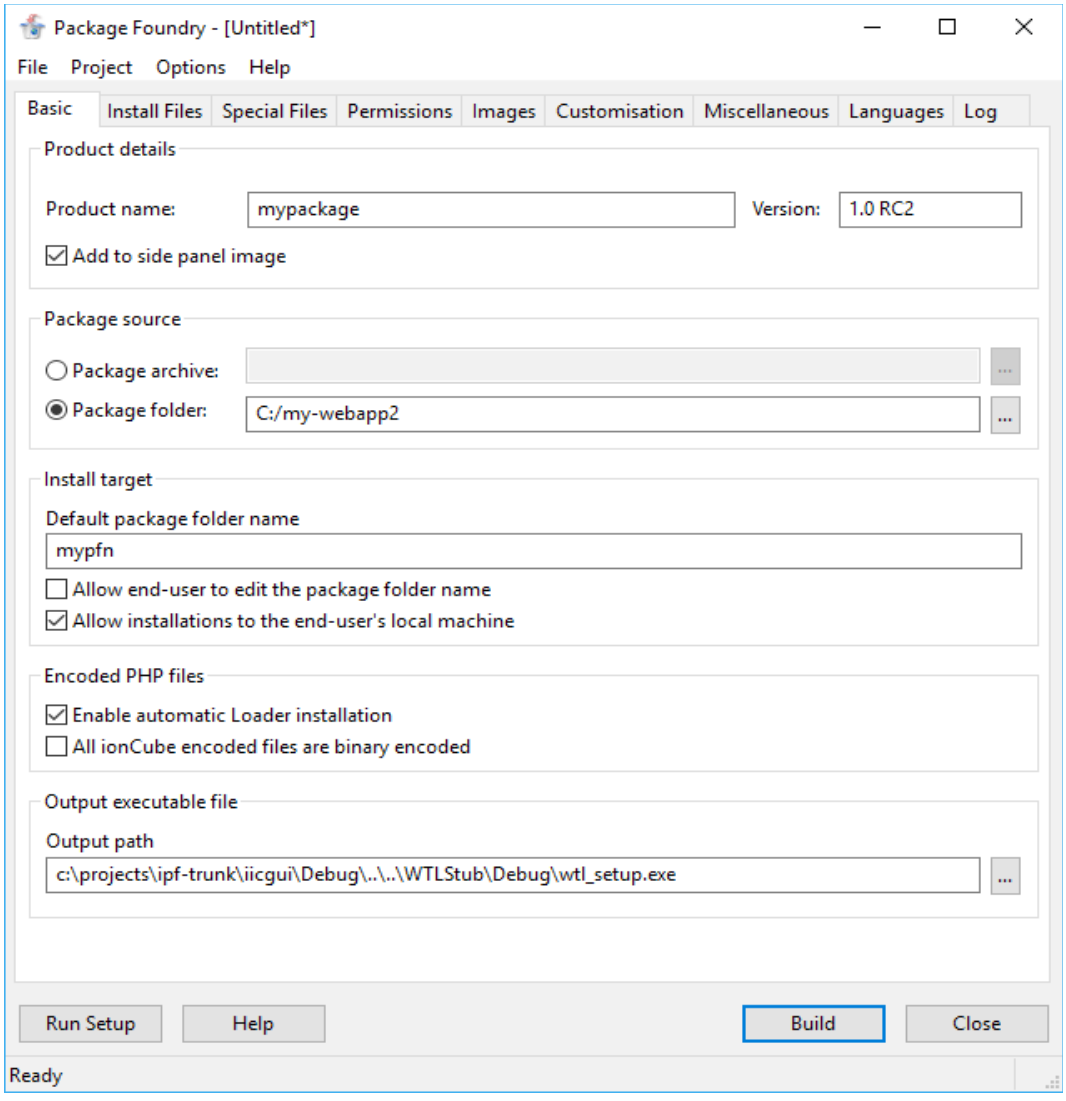
5.1 Introduction

The Windows release of IPF comes with the IPF GUI, a simple and easy to use way to both select and change various options for packages, and also to build and run them. To build a product, click Build or press the F7 key. The Log tab will show detailed output from the IPF command-line tool. Once a build is complete, a dialogue will appear to report success or failure. You can test your package executable at any time by pressing F5 or selecting Run from the Project application menu.

This section will explain the various options page by page inside the GUI.

5.2 Basic Settings

Basic IPF settings are selected on the Basic tab:



5.2.1 Product Details

The Product name and Product Version can be specified and appear on the created executable multiple times throughout the installation process.

5.2.2 Package Source

Package Source is where you select the package you would like to create an installer for, this can either be inside a folder or compressed as an archive:

Choose the “Package archive” option if you have a zip file or tar.gz file which should be used as the package when creating the installer. Usually it is preferable to use a tar.gz since some servers are able to upload these files in compressed form, then extract on the server, which might reduce transfer time. IPF requires the archive to be a single compressed folder (with subfolders and files), although the name of the top level folder will be replaced with the package folder name (see below). For example the structure should be project.zip/myproject/(project files)

If using an archive, related files such as “Main page path” and any configuration files will act as if the current directory was one level in to the archive. E.G in project.zip/myproject/config.txt, the configuration file path should just be “config.txt” as the base directory would be “myproject”. The same applies for Main page path.

The benefits to using a folder are that custom permissions (both file and folder) can only be used with a folder and are not accessible for archive files. On the other hand if the package is provided as an archive, the IPF build process will be faster. The name of the specified folder will be ignored, and replaced with the package folder name (see below).

5.2.3 Install Target

Some applications require their top-level folder to have a specific name, other applications can have a user-defined top-level folder name, and some applications are designed to be located in the 'web root' of a domain. To achieve the first type of package, enter a name in the Default package folder name field, but leave the Allow end-user to edit the package folder name unchecked. Check this box on the other hand if the user should be allowed to modify the name of the top-level folder. Lastly, for applications that should be installed to the web-root, leave the default package folder name blank.

If a product will only be installed to remote servers, and never to an end-user's local machine, then the Allow installations to the end-user's local machine checkbox can be unchecked. This will remove the 'Installation Type' wizard step. This option would be useful, for example, if the creator of the package was intending to use the package to install to a handful of their own remote servers. If a product is for general release then usually this option should not be used.

In order to automatically enable Loaders for ionCube Encoded files, check the Enable automatic Loader installation box. Checks will be performed on the end-user's server to verify whether the server is able to execute encoded files. If Loaders are not currently installed then the installer will attempt to transfer appropriate Loaders to the target server. If this option is selected, the end-user will be prompted to enter the URL of the package on their web server.

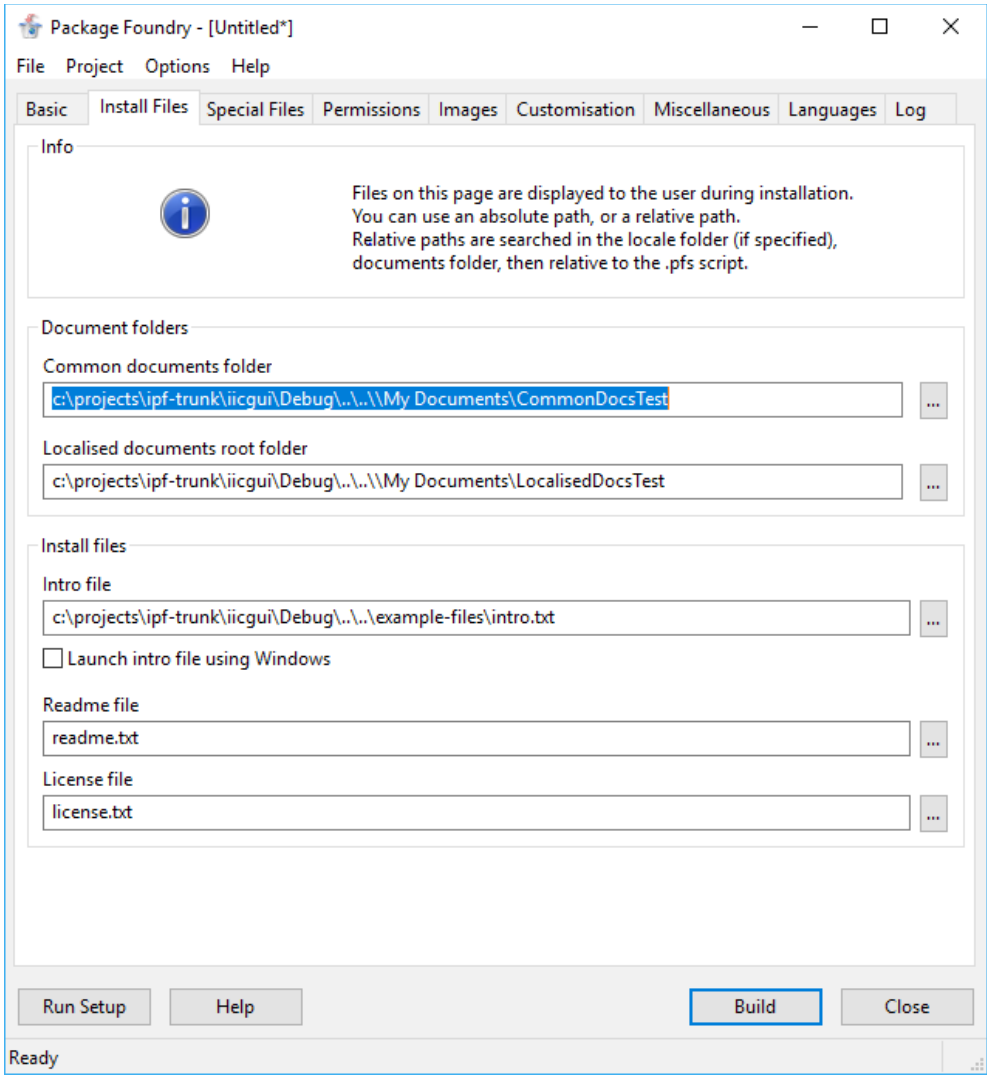
Specify whether any encoded files are encoded in the newer ASCII format, or all are encoded in binary format. In the case of ASCII files, and where Loaders are installed on the target server but are not recent enough, updated Loaders will be downloaded.

5.2.4 Output Executable File

The output file must be a filename ending in .exe, in an existing directory.

5.3 Install Files

Files such as a readme and end-user license agreement can be displayed to the user during the installation process within the wizard interface. We refer to these files as install files. Settings related to these files can be entered on the Install Files tab:



5.3.1 Document Folders

Localised install files are supported. The path to a folder containing localised documents should be entered in the `Localised Documents` root folder in order to use this feature. This folder will be searched during the installation for an appropriately localised version of the document, and if a suitable version isn't found, then the non-localised version will be used.

Put simply, the localised documents folder should contain localised versions of install files, and the common documents folder should contain default, non-localised versions of these documents (used in case an appropriately localised version is not found).

For example, suppose we enter `C:\local` as the localised documents folder. We create a subfolder de, and save a German readme.txt into the de folder. We also save a non-localised version, English in this case, to `C:\docs\readme.txt`, and enter `C:\docs` as the Common Documents Folder. We should enter readme.txt into the Readme File edit box.

In this example, when the installer is run by an end-user whose system is set to the German locale, the readme.txt will be selected from the de folder and displayed to the user. If the user's system was set to Spanish, and no readme.txt was found localised to Spanish, then the file copied from `C:\docs\readme.txt` would be displayed.

It is not necessary to enter a localised documents folder - in that case the files will always be taken from the common documents folder.

If a common documents folder is not used, then relative paths will be resolved with respect to the location of the .pfs script.

5.3.2 Install Files

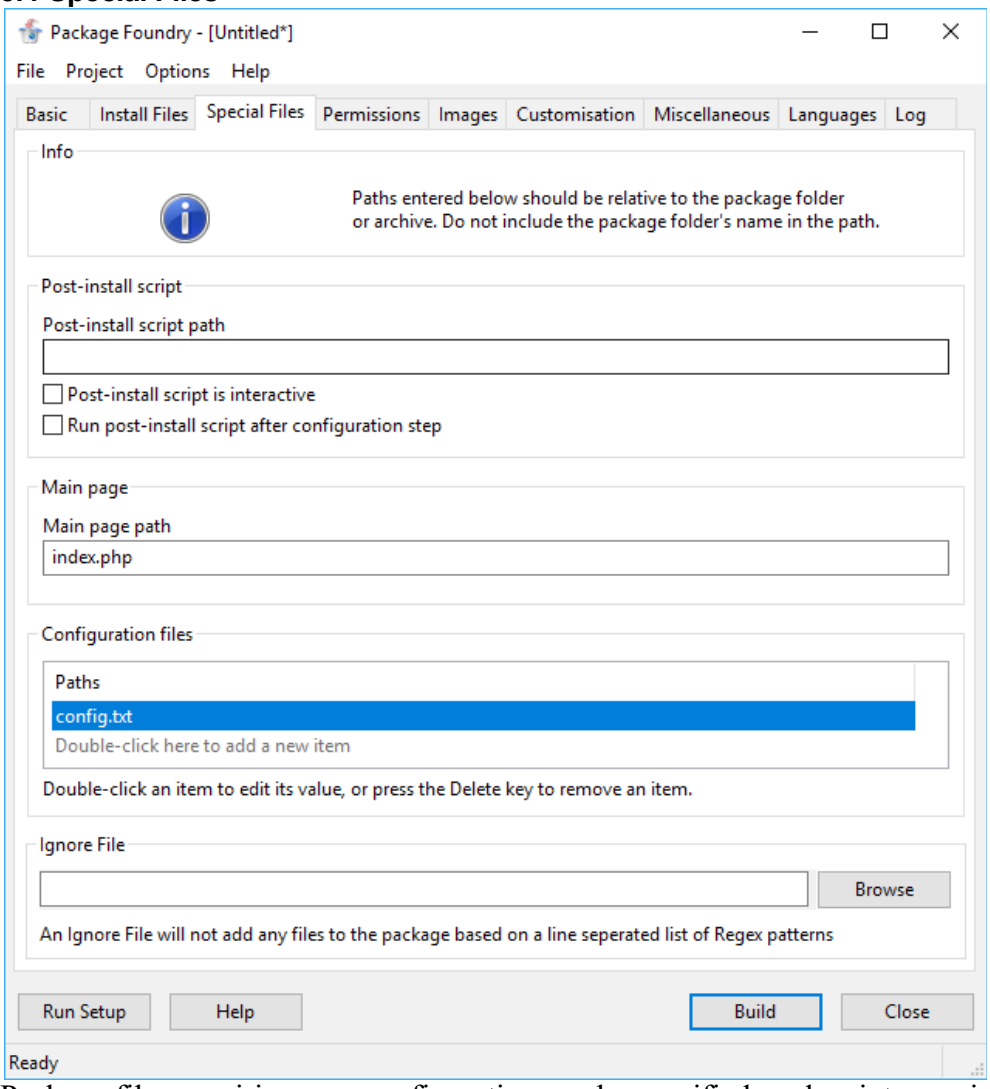
Install files are displayed to the user at certain points during the installation process. All files are optional - if the files are not specified, then the corresponding installer wizard page will be omitted.

The License file is displayed immediately after the start page.

The Intro File is shown immediately after the license agreement. It is typically used to display a welcome message, installation instructions, or other text. If the intro file is a text file it can be displayed internally in the installer wizard. Any file with a Windows file association (for example a html file) can be used if the Launch intro file using Windows option is checked. A URL may be specified instead of the path to a file.

The Readme File is displayed at the end of the installation process.

5.4 Special Files



Package files requiring user configuration can be specified, and scripts nominated to be executed when a package has been installed or configured. These files are entered on the Special Files tab:

5.4.1 Post-Install Script

A post-install script can be triggered at the end of installation. This might be a background PHP script or the main web page that you want the user to see after installation, for example to guide the user through setup of the application with database creation and so on. The script can be any file in the package such as a PHP script, a plain .html file or an image. To launch the script in a web browser window check the Post-install script is interactive option. Otherwise leave this unchecked to execute the script silently. If a non-interactive post-install script does produce output such as error messages, the output will be shown in a dialog and not lost.

5.4.2 Configuration Files

Any files that should be configured by the user can be entered in the list of Configuration files. If configuration files are specified, after installation the end user will be able to configure those files in a text editor window locally using a Configuration Tool. Once configured, the configuration files are automatically updated in the target installation.

If the Run post-install script after configuration step box is checked and there are configuration files, the post-install script will not be run at the end of installation. Instead the script will be executed each time the end-user clicks the Apply button of the Configuration Tool and after the configurable files have been uploaded to the server. This allows any background tasks dependent on configuration files to be run again.

When configuration files are specified, a Main page path may also be given for a script that will be launched in the user's default browser when they click the Launch button of the Configuration Tool after they have completed their configuration changes.

NOTE: the Main page path is used only when configuration files have been specified. If there are no configuration files, the Main page path is ignored and only the post install script will be run.

5.4.3 Ignore File

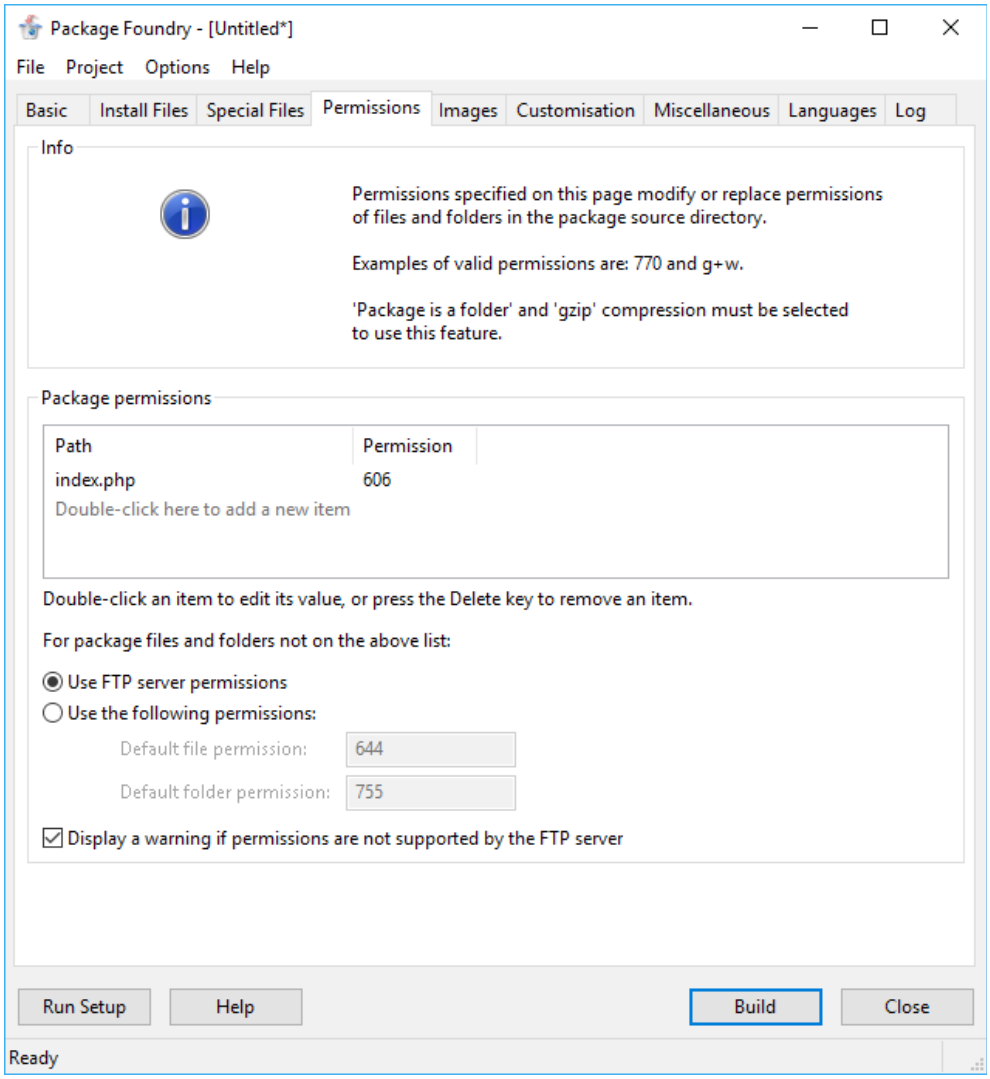
Any specific files or folders can be ignored during packaging. For example you may not want to add a tests folder, a `.gitignore` file or perhaps any files ending in `.temp` to your package. Ignore files have support for wildcard matching (e.g the * and ?) symbols for more complex needs. These 3 scenarios can be fixed using a line separated plain text ignore file ending in `.ignore`. For the example just specified, the ignore file may look like:

```
tests

.gitignore

*.temp
```

5.5 Custom Permissions



When using Windows it can be difficult to set up a package which will have the desired permissions when transferred to the target FTP server. Behaviour with respect to UNIX style permissions can be set on the Permissions tab:

5.5.1 Package Permissions

Individual items within the folder can be given a custom Unix style permission. To add an item, double-click on the grey final item on the list.

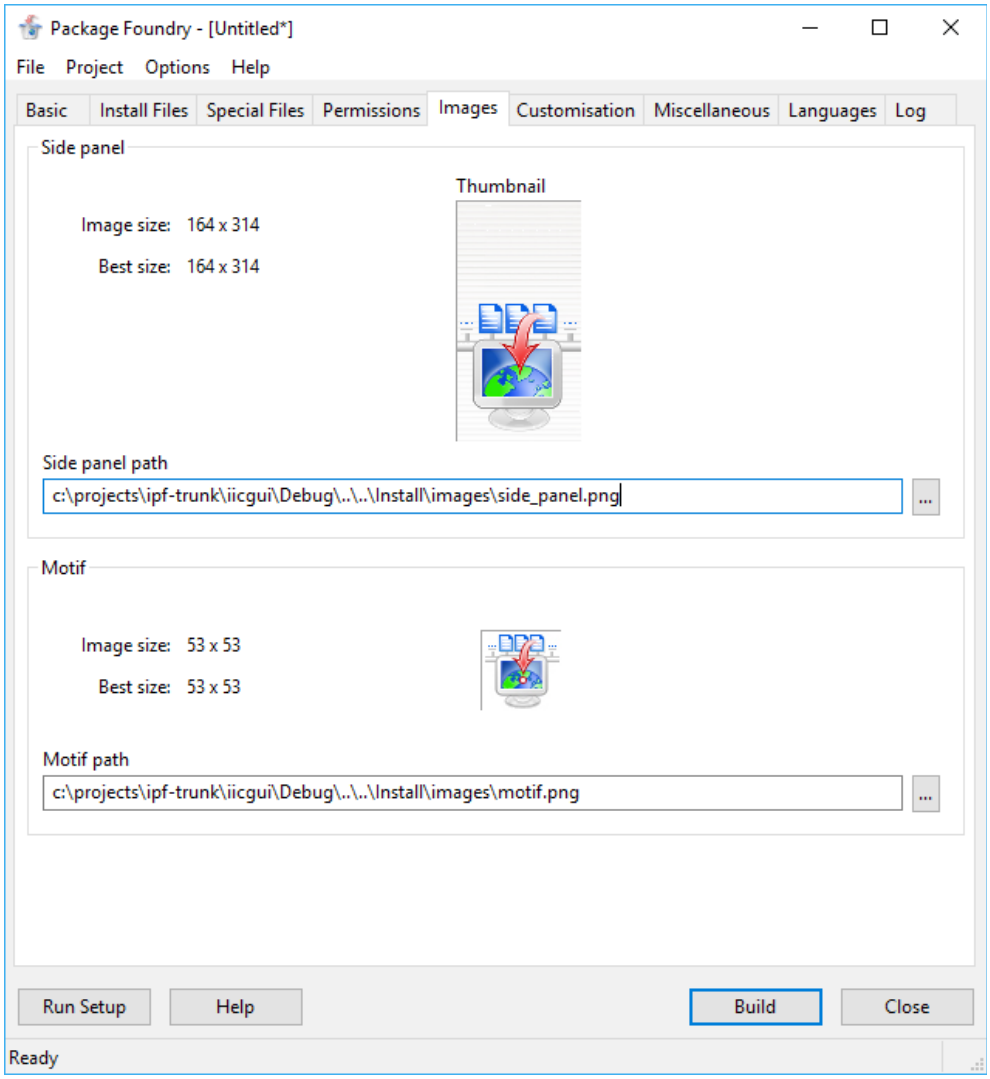
Usually most items in a package should have the same permission. If the `Use FTP server permissions` radio button is selected, then items not on the custom permissions list will receive whatever permission the FTP server uses as its default. Choosing this option is a good idea, as it results in less commands sent over the FTP connection.

If a specific permission is required for most files then this can be set by selecting `Use the following permissions` and entering the desired values. as noted above, this may increase transfer times since a permission will need to be set for every item in the package.

Some FTP servers do not support permission setting. This is true in particular for Windows FTP servers, where UNIX style permissions do not make sense. A warning dialog is displayed by default if the target server does not support permission changes, but the final option on this tab allows the dialog to be disabled.

5.6 Custom Images

Custom images can be assigned on the Images tab:



Custom images must be either `.jpg`, `.jpeg`, `.png`, `.bmp`, or `.gif` and must be a valid image of the corresponding file type. Images can be of any dimension but to look best should fit the following criteria:

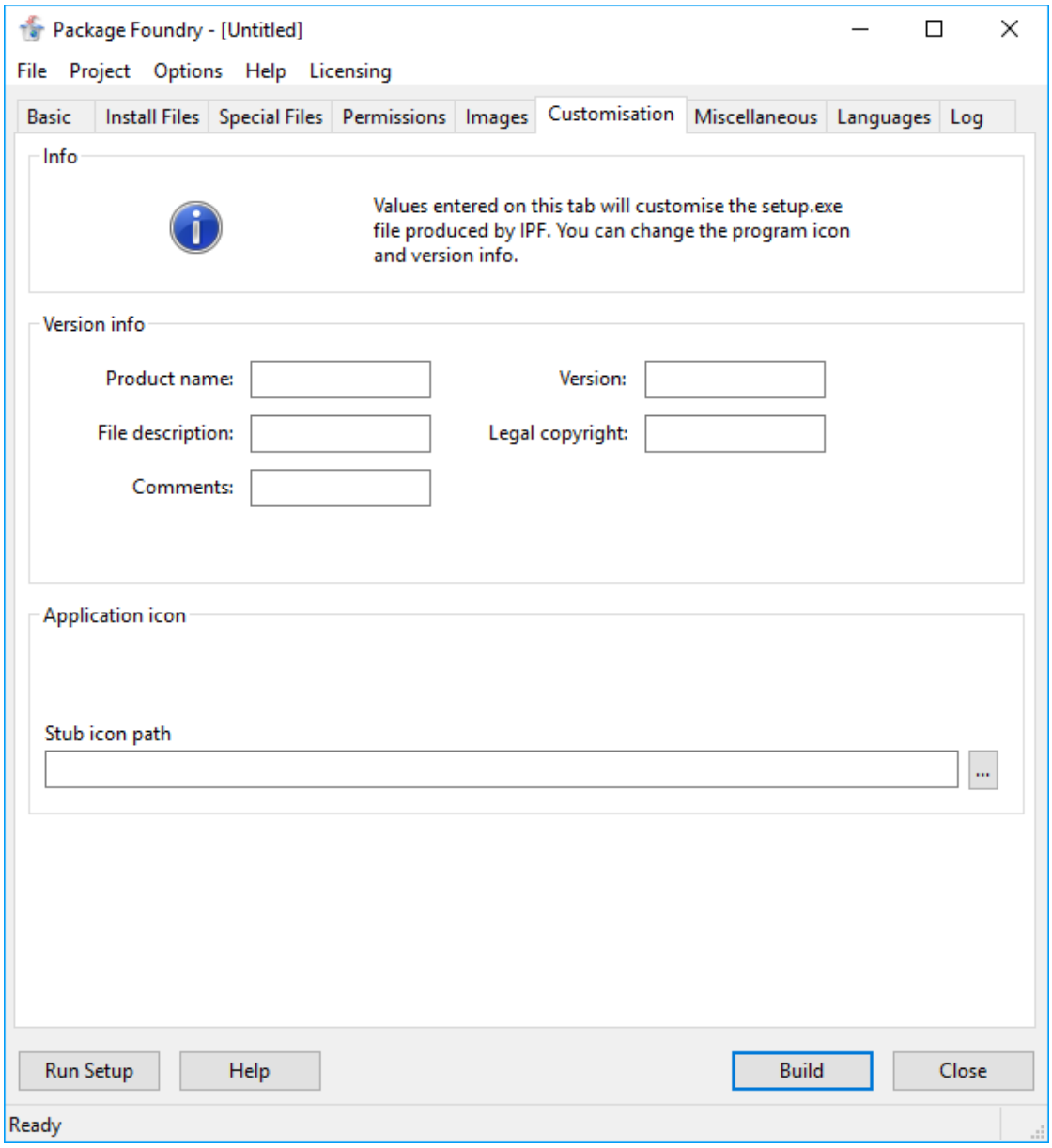
The Side Panel image should be **164x400** pixels.

The Motif image should be **53x53** pixels.

If no custom images are given, the images used will fall back to the defaults included in the IPF download.

5.7 Stub Customisation

The executable file produced by IPF can be customised in several ways. Settings related to this feature are entered on the Customisation tab:



5.7.1 Version Info

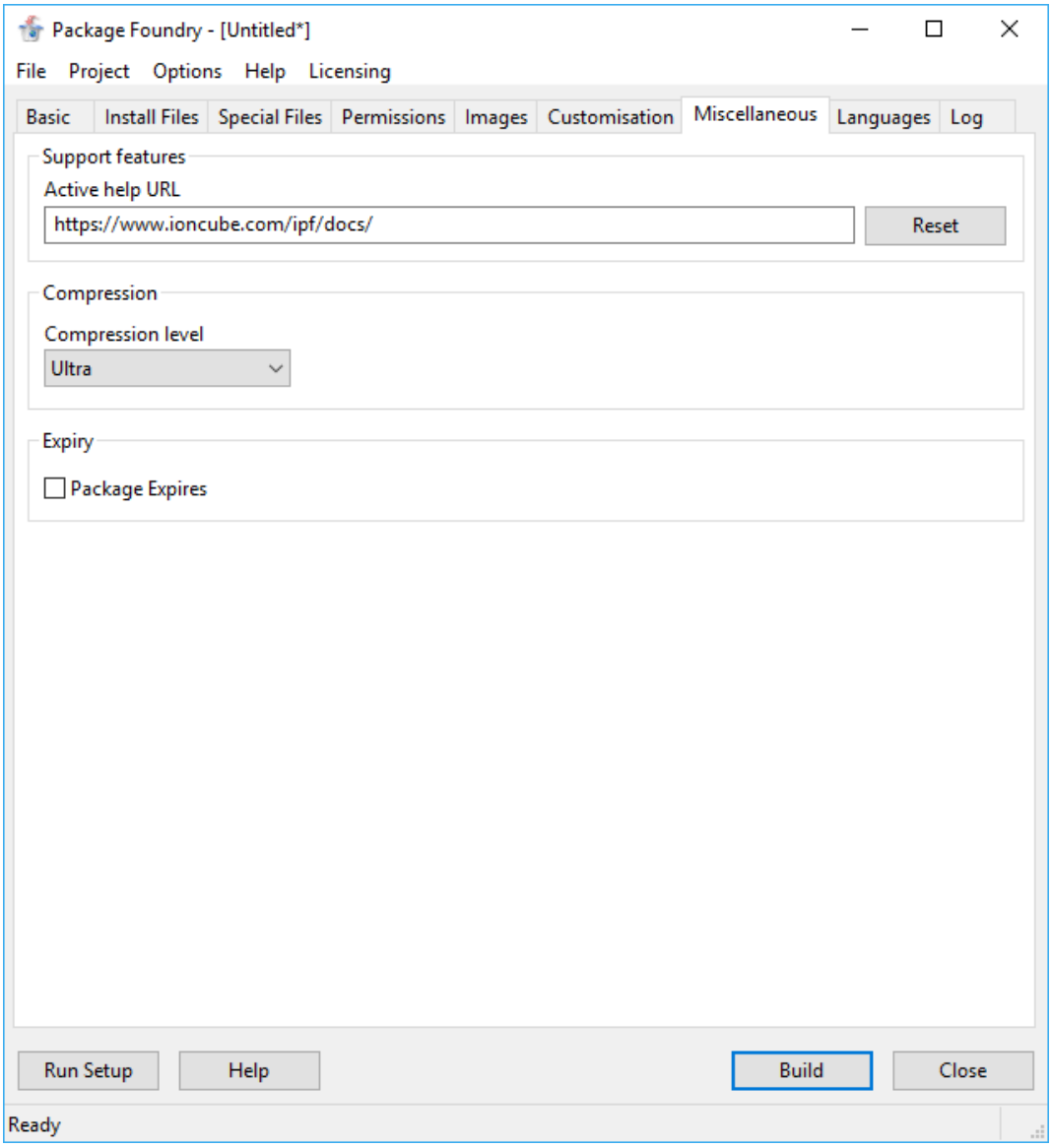
The fields in the Version Info group are used to edit the version info resource of the installer. These settings will change the text displayed in the installer's Explorer tooltip. The text will also be shown if the user views the properties of the installer by right-clicking on the .exe in Explorer.

5.7.2 Application Icon

The installer's application icon can also be customised on this page. Enter the path to a .ico file in the Stub Icon Path field.

5.8 Miscellaneous

The IPF Miscellaneous tab is shown below:



5.8.1 Support Features

When certain errors occur during an installation, for example if ionCube Loaders are required but not available for the target server, a log file may be generated and sent to ionCube.

Installers created with IPF feature a context-sensitive active help system. A Help button on the installer Wizard launches a web browser and navigates to a web site, passing information related to the current state of the installer. By default the script linked with the active help system is `http://help.ioncube.com/ipf/index.php`, but a custom site can be used. This enables the active help system to be branded or customised for a particular product or company. Content can be copied from the default active help site. Please read the active help page in this documentation for details on how to customise the active help system for your product.

5.8.2 Compression

The package executable created can be compressed to multiple different levels. Higher levels of compression will result in a longer build time. This should make little difference to smaller projects but will be noticeable in larger ones.

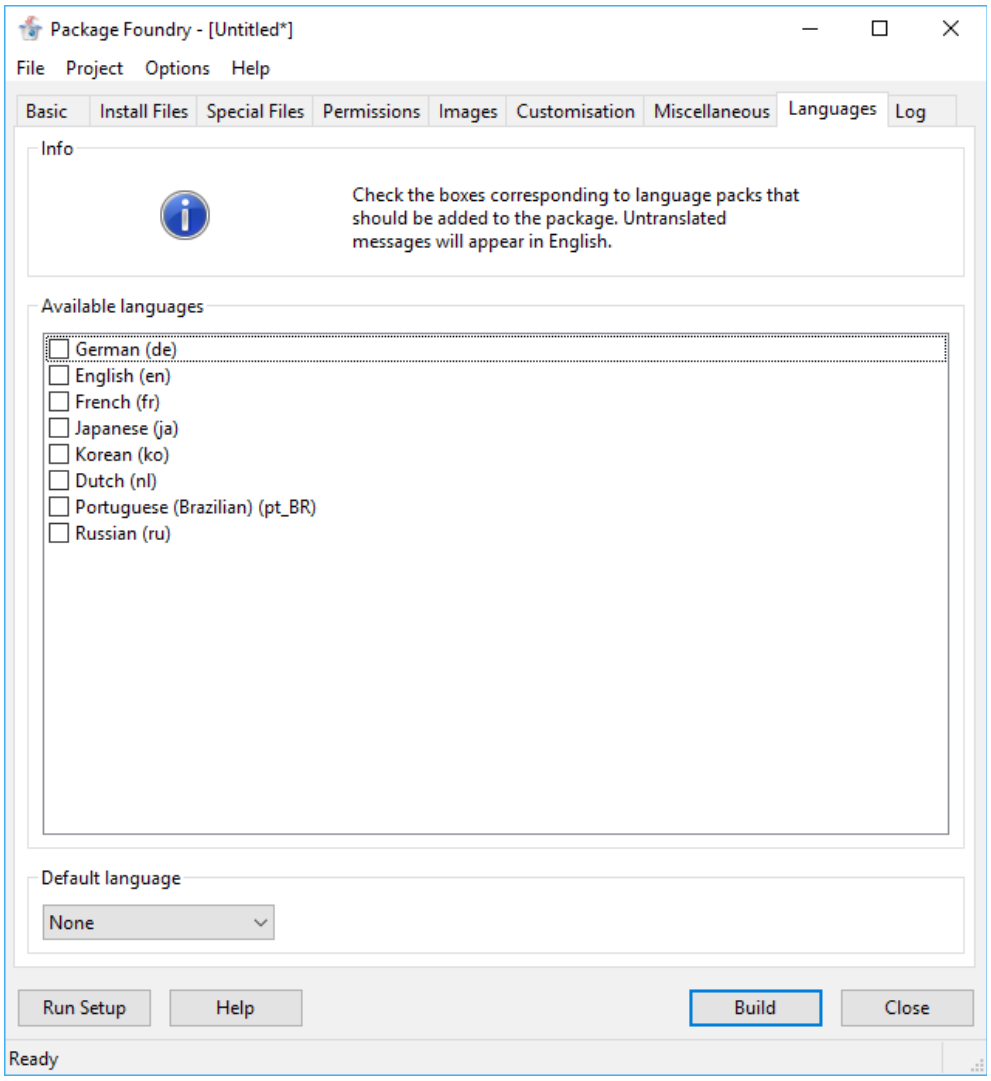
5.8.3 Installer Features (HTTPS)

SSL support is now included by default when before it was an option. This enables sites using HTTPS to be accessed.

5.8.4 Package Expiry

An expiry date can be given to the built package at which point running the executable will bring up an error message that the package has expired and to contact the package distributor for a new one. This is useful if you want to release packages on a temporary basis. Expiry dates should be given in ISO 8601 formatting (`YYYY-mm-dd`).

5.9 Language Packs



The text displayed to the end-user in the installer interface can be localised by using the Language Pack feature. Settings related to this feature are entered on the Languages tab:

Click on the boxes next to the available languages list items to select which languages should be included in the installer. The language which is used in the Installer to display text to the end-user will depend on the current system locale set on the user's system.

Used for if the installer is run by an end-user whose system's current language has not been included with the installer. If no Default Language is set in IPF, then the non-localised text (English) will be displayed. Under some circumstances it may be preferable to display a different language. To do this, change the Default Language option to one of the included languages.

More information related to the use of locales in IPF, and information on how to add new languages, can be found on the Translating Installers page.

5.9.1 Translating Installers

The contents of the Available Languages list in the IPF GUI is generated dynamically by IPF at start-up. The IPF installation folder (typically `C:\Program Files\ionCube Package Foundry` contains a subfolder locale. Subfolders of the locale folder determine the items displayed in the IPF available languages list.

To add a new language to IPF it is necessary to add a folder to the locale subdirectory. The name of this directory must contain a two letter language code - en for English, for example. The list of language codes is set by the ISO 639 standard (it is usual to use lower case, however). The directory can also contain a 2 letter country code suffix. The list of 2 letter country codes is contained in the ISO 3166-1-alpha-2 standard. As a full example, pt_BR would be the correct folder name for Brazilian Portuguese.

Inside the language folder, a subfolder LC_MESSAGES must be created, and inside this subfolder the translated messages file driver.mo must be copied. Producing a driver.mo file is described in the next section.

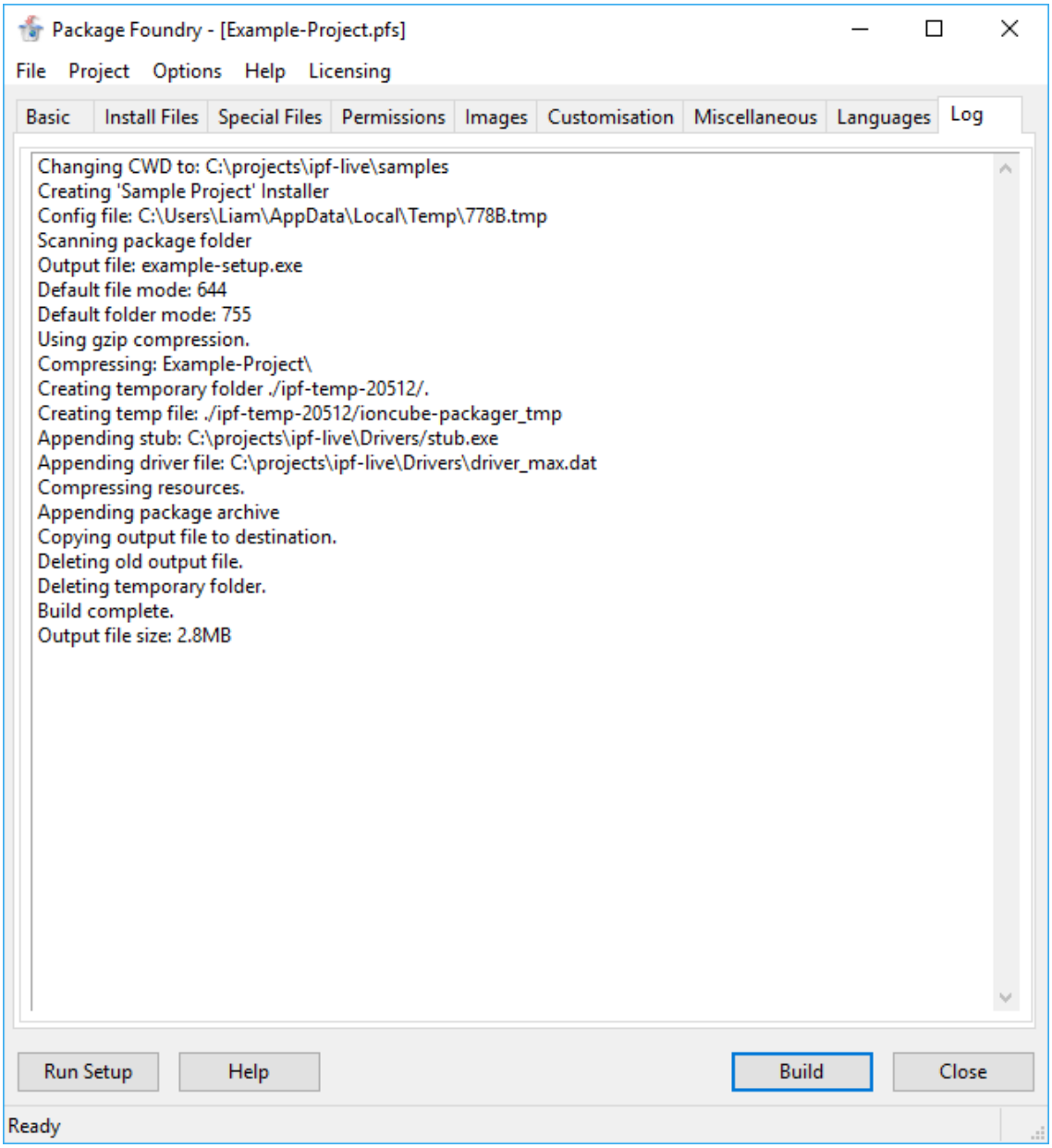
5.9.2 Producing a Language Translation

A language translation is made by editing a file containing the English messages to add translations, and then using a tool to compile the human readable file of translations into a more efficient binary format. By convention, the human readable file has a .po file suffix, and the compiled file has a .mo suffix. For this application, the compiled file must be called driver.mo as mentioned above.

The untranslated messages for the installer are contained in the messages template file driver.pot, located in the folder locale_src of the IPF installation folder. Before editing anything, first copy the driver.pot file to driver.po so that the master POT file does not get changed. The recommended tool to then use for editing the PO file is poEdit. poEdit makes managing the PO file structure easier than a regular text editor, and it can also perform some checks for likely formatting errors in the translation text. poEdit also can automatically produce the MO file for you when you save the PO file.

Other standard tools that may be of interest are the GNU gettext command line tools, including the program msgfmt for converting a PO file to a MO file.

5.10 IPF LOG



The status of the build process can be monitored on the Log tab:

When a build is started, the IPF GUI launches the IPF command-line tool. The command-line tool's output is displayed in the Log tab. If an error occurs during the build process, that error is displayed in a dialog.

6 Active Help API

The Help button on installers created with IPF provides web-based context sensitive help. By default the URL associated to the Help button is `http://help.ioncube.com/ipf/index.php`, but any web page can be used by setting the `active-help-url` option.

Several GET parameters are used to control the content displayed to the user. The possible parameters are listed below.

6.1 page

The page parameter is set according to the page on which the user clicks the Help button. Possible values for this parameter are:

- `intro`
- `license`
- `introfile`
- `localremote`
- `loc`
- `ftp`
- `http`
- `review`
- `actions`
- `done`
- `configure`

6.2 version

This parameter is set to the version of IPF used to create the installer, for example 4.0.0.

6.3 type

This parameter can take values 'local' or 'remote', according to whether the user is performing a local or remote install.

6.4 locale

This parameter is the language and country code of the system locale on the user's machine, for example `en_GB`.

6.5 introfile

This parameter is 1 or 0 according to whether the installer has an intro file or not

6.6 license

This parameter is 1 or 0 according to whether the installer has a license file or not.

6.7 ftp_host

If the user clicked Help on the HTTP settings page then this parameter will be set to the FTP host name which the user entered on the FTP settings page.

6.8 ftp_dir

If the user clicked Help on the HTTP settings page then this parameter will be set to the FTP install directory which the user entered on the FTP settings page.

7 Using htaccess files

Sometimes a package may contain an .htaccess file in its top level folder. This may be used to limit web access to specific subfolders of the application. The use of such a file may cause problems during installation, since IPF will create its own temporary folder which it will need to access via the web.

The working directory used by IPF will have the name `ipf-work-dir-X` where X will be replaced by a timestamp. This directory will be deleted at the end of the installation. If a `.htaccess` file is used it will be necessary to allow web access to folders with names of the form `ipf-work-dir-X`. It may also be advisable to allow access to the `ioncube` folder. In case Loaders cannot be installed, web access will be needed to `ioncube/loader-wizard.php`. This script helps users manually install Loaders.