TECHILA

TECHILA® DISTRIBUTED COMPUTING ENGINE

BUNDLE GUIDE

28 NOVEMBER 2016

Disclaimer

Techila Technologies Ltd. disclaims any and all warranties, express, implied or statutory regarding this document or the use of thereof by you to the full extent permitted by law. Without limiting the generality of the foregoing, this document provided by Techila Technologies Ltd. in connection therewith are provided “as-is” and without warranties of any kind, including, without limitation, any warranties of performance or implied warranties of merchantability, fitness for a particular purpose, title and noninfringement. Further, Techila Technologies Ltd. does not make, and has not made, any presentation or warranty that the document is accurate, complete, reliable, current, error-free, or free from harmful information.

Limitation of Liability

In no event shall Techila Technologies Ltd. or any of its respective directors, officers, employees, or agents, be liable to you or any other person or entity, under any theory, including without limitation negligence, for damages of any kind arising from or related to the application of this document or any information, content, or materials in or accessible through this document, including, but not limited to, direct, indirect, actual, incidental, punitive, special or consequential damages, lost income, revenue or profits, lost or damaged data, or other commercial or economic loss, that result from your use of, or inability to use, this document, even if any of those persons or entities have been advised of the possibility of such damages or such damages are foreseeable.

Use of this document and copyright

Table of contents

1. Introduction

This document is intended for Techila End-Users and Administrators and contains instructions on how to use the Techila Bundle Creator tool and the Techila Command Line Interface (CLI) `'createBundle'` command to create Bundles. The primary focus in this document will be on programming language-specific Runtime Bundles, which are required to perform computations on Workers.

Chapter 2 contains information on Bundle properties, which need to be defined when creating a Runtime Bundle. Among other things, these properties will define the Bundle exports and platforms that will determine which Projects will automatically import the Bundle and which platforms the Bundle will be made available for.

Chapter 3 contains an overview of the Techila Bundle Creator tool, which provides a graphical interface which can be used to create Runtime Bundles. The Bundle Creator tool also comes with premade templates for several Runtime Bundles that reduce the amount of manual input required. Depending on the template, the 'Guess' button can be used automatically set the values of several required parameters.

Chapter 4 contains an overview of the `'createBundle'` CLI command, which provides a command line interface for creating Bundles. Parameters of the `'createBundle'` command are also explained, which can be used to define Bundle properties such as the platforms the Bundle will be made available for.

Chapter 5 contains examples that illustrate how to create Runtime Bundles for different programming languages, including MATLAB, Perl, R and Python. Examples are provided for the Techila Bundle Creator tool and for the CLI `'createBundle'` command.

2. Bundle properties

This Chapter contains a description on the Bundle properties that typically need to be defined when using the CLI **'createbundle'** command or when using the Bundle Creator tool. These properties include:

- Name of the Bundle
- Bundle resource
- Bundle exports
- Bundle platforms
- Expiration time on Workers
- Description of the Bundle
- Path to which the files will be extracted on the Workers
- Files and/or folders that will be included in the bundle

Note! When creating bundles with CLI **'createBundle'** command, it is advisable to perform the initial Bundle creation attempts using the test-mode (enabled with the parameter **'test=true'**), which enables commands to be simulated without creating the Bundle.

2.1.1. Bundle name

All Bundles that are transferred to the Techila Server must have a name. This name must be unique, meaning that no two Bundles can have the same name. If you attempt to create a Bundle that does not have a unique name, the Bundle creation process will fail. It is also advisable to choose an informative name, as the name will be displayed in the Techila Web Interface.

For example, when creating a MATLAB Runtime Bundle the name should contain the MATLAB Compiler Runtime version and the operating system and platform for which the Bundle will be created for. If for example the MATLAB MCR version is 7.14 (MATLAB 2010b) and the Bundle is created for 64-bit Windows Workers, the name should resemble the name illustrated below:

```
"Techila Worker MATLAB v714 Windows amd64"
```

When creating R Runtime Bundles, the naming convention is similar. The name would contain the R version and the platforms the Bundle will be made available for. For example, when creating a Runtime Bundle for R 2.12.1 Windows 64-bit Workers, the name should resemble the name illustrated below:

```
"Techila Worker GNU R v2121 Windows amd64"
```

Module and Data Bundles should always contain the **{user}** notation, which will make the End-Users alias to be included in the Bundle name. This can help ensure that all Bundles have a unique name.

2.1.2. Bundle resource

One role of the Bundle resource parameter is to differentiate Runtime Bundles between different programming languages. The following table lists available resources for different programming languages:

PROGRAMMING LANGUAGE	RESOURCE
GNU R	r
MATLAB	matlab
Python	python
Perl	perl

These resources are included in the Runtime Bundle templates available in the Bundle Creator tool and can also be defined when using the CLI interface.

If you are creating a Bundle for your own personal use (e.g. a Module or a Data Bundle), there are no strict guidelines on how you should define the value of the resource parameter. Good practice is to include some information on the contents of the Bundle. For example, when creating a Perl Module Bundle, the resource parameter could contain the name of the Perl Module.

2.1.3. Bundle exports

Bundle exports define what the Bundle will export. This value can be used to import the Bundle into a computational Project.

MATLAB, R, Perl and Python use version specific Runtime Bundles. For example, if an End-User creates a computational Project from MATLAB 2010b, the compiled computational code will require a MATLAB Runtime Bundle created from the same MATLAB Compiler Runtime version used in MATLAB 2010b (version 7.14).

Examples on how to define the exports parameter for Runtime Bundles configured can be found in Chapter 5.

When creating a Bundle for your own personal use (e.g. a Module or a Data Bundle), the exports parameter should contain the {user} notation. This will include the End-Users alias (defined in the `'techila_settings.ini'` file) to the Bundle exports.

2.1.4. Bundle platforms

Bundle platforms determine the operating systems and processor architectures the Bundle will be made available for. A Bundle can for example be made available for 32-bit Windows Workers or the Bundle can be made available for all operating system and processor architectures.

Bundle platforms should always be defined according to the role and type of the Bundle. For example, when creating a MATLAB Runtime Bundle, which can only be used on specific operating systems, Bundle platforms need to be defined so that the Runtime Bundle will only be made available for those operating systems.

On the other hand, when creating a Data Bundle which contains generic, non-platform specific data files, the Bundle platforms can be defined so that the Bundle will be offered to all operating systems and processor architectures.

2.1.5. Bundle expiration

The Bundle expiration parameter determines how long the Bundle is stored on Workers after the last time it has been used. If a Bundle remains unused longer than the expiration period, the Bundle will be deleted during the next scheduled clean-up operation, which typically occurs once an hour.

The Bundle expiration should always be defined according to the type and role of the Bundle that will be created. General guidelines for expiration periods are listed below:

- Runtime Bundles: 365d
- Module Bundles: 30d
- Other Bundles: 15d

Please note that the `'d'` notation means that the number represents days. Other available values include `'s'` (seconds), `'m'` (minutes) and `'h'` (hours). If no notations are used, the number will be interpreted as milliseconds.

2.1.6. Bundle description

All Bundles should be given an informative description about the content and role of the Bundle. For example, when creating a MATLAB Runtime Bundle, the description should contain the MATLAB Compiler Runtime version and the operating system the Bundle is available for. If for example the MATLAB MCR version is 7.14 and the Bundle is created for Windows Workers, the name should resemble the name illustrated below:

```
"Techila Worker MATLAB v714 Windows"
```

2.1.7. Bundle path

The Bundle path is determined by the values of **`'trimpath'`** and **`'prependpath'`** parameters, which are available in both the graphical Techila Bundle Creator tool and in CLI **`'createBundle'`** command.
Modifications to the Bundle path are often required to ensure that the files in the Bundle will be extracted to the correct directory on the Techila Worker.

When entering values to the **`trimpath`** parameter, some characters need to be escaped with a backslash **`'\'`**. These characters include:

- Backslash '\'
- Parentheses '(' and ')'
- Brackets '[' and ']'

2.1.8. Bundle files

Bundle files define the content of the Bundle. When creating Runtime Bundles, the Bundle must contain the files and/or folders that are listed in Chapter 5.

2.1.9. File permissions

File Permissions can be used to define permissions of certain key files that are located in MATLAB Runtime Bundles that are created for Linux based Workers. The necessary file permissions will be automatically set when using the Runtime Bundle templates in the Techila Bundle Creator tool. Information on how to define the necessary permission using the CLI can be found in Chapter 4.1.18.

3. Techila Bundle Creator tool

The Techila Bundle Creator tool (`'bundlecreator.jar'`) is included in the Techila SDK and is located in the `'lib'` directory.

3.1. Launching the Techila Bundle Creator tool

To launch the Techila Bundle Creator tool, navigate to the `'lib'` directory in the Techila SDK and double click on the `'bundlecreator.jar'` icon. The Techila Bundle Creator tool can also be launched from the command line using the command:

```
java -jar bundlecreator.jar
```

The Bundle Creator Tool contains several tabs, which are explained in the following Chapters.

3.2. General tab

The 'General' tab is automatically displayed when the application is launched. This tab can be used to select an appropriate Template, e.g. when creating a Runtime Bundle. This tab also contains fields for defining the Bundle Name (required field) and the platforms the Bundle will be offered to. The 'General' tab is shown in Figure 1 below.

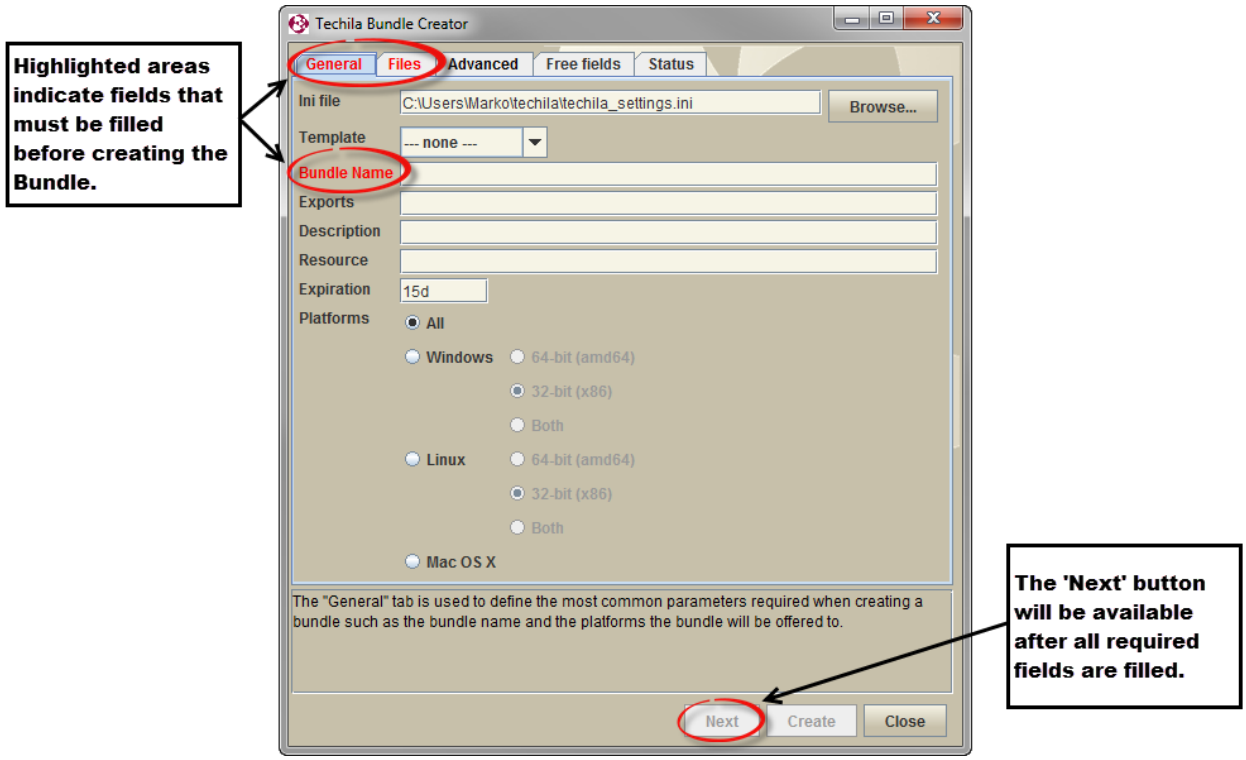

Figure 1. The 'General' tab. The 'Bundle Name' field is highlighted in red, meaning a value must be specified. The 'Next' button will become available after all required fields on the current tab (highlighted in red) have been given values.

The fields in the 'General' tab are explained in the following Chapters.

3.3. Ini file

The 'Ini file' field displays the location of the **'techila_settings.ini'** file that contains the settings that will be used when creating and transferring the Bundle to the Techila Server. By default, the **'techila_settings.ini'** file in the **'techila'** directory will be automatically used. If you wish to use the settings in a different settings file, click the **Browse** button and select the desired **'techila_settings.ini'** file.

3.3.1. Template

The 'Template' dropdown menu contains a list of predefined templates that can be used as the base when creating a new Bundle. These include templates for Perl, Python, MATLAB and R Runtime Bundles. The Runtime Bundle templates will automatically fill in the following fields:

- Bundle Name
- Exports
- Description
- Resource
- Expiration

When creating Runtime Bundles, selecting the correct template will reduce the amount of manual input required.

There are also Module Bundle templates for a selection of programming languages. The **'--none--'** template can be used in situations where you do no not wish any of the fields to be automatically filled, e.g. when creating a Data bundle.

3.3.2. Bundle Name

The 'Bundle Name' field displays the name of the Bundle that will be created. If any of the Runtime Bundle templates is chosen, the Bundle Name field will be automatically updated with a template specific prefix. Clicking on the radio buttons in the Platforms field will also automatically update the Bundle Name field.

The Bundle Name field will also always have a free text field, which can be used to define the version of your runtime components when creating a Runtime Bundle. Examples on how this free text field should be modified can be found in the Chapter 5.

3.3.3. Exports

The 'Exports' field defines the Bundle exports. Choosing any of the Runtime Bundle templates will automatically update this field with the correct value. If you plan on creating a Module Bundle or a Data Bundle, the field is not updated automatically. In these situations, you can define the value of the exports parameter simply by typing it to the text field.

3.3.4. Description

The 'Description' field displays the Bundle description, which will be visible in the Techila Web Interface. The value of the description field will be automatically generated based on the value of the Bundle Name field.

3.3.5. Resource

The 'Resource' field defines the resource that the Bundle will offer. This field will be automatically filled with a correct value when any of Runtime Bundle templates is chosen. If you plan on creating a Module or Data Bundle and wish to specify a resource for the Bundle, simply type the desired resource value to the text field.

3.3.6. Expiration

Defines how long the Bundle will be stored on Workers since the last time it has been used. Choosing a template will automatically define an expiration period for the Bundle according to the list shown below:

- 365 days for Runtime Bundles
- 30 days for Module Bundles
- 15 days for other bundles

If you wish to modify the expiration values, simply enter the desired expiration time in the text field. Please note that when creating a Runtime Bundle, which will be used in several Projects over a long time period, it is advisable to use a reasonably long expiration time.

3.3.7. Platforms

The radio buttons in the Platforms section can be used choose which operating system and platforms the Bundle will be offered to. As mentioned, the Bundle Name field updated based on what operating systems are selected. The effect of the radio buttons is explained below:

- **All** specifies that the Bundle will be offered to all operating systems and platforms.
- **Windows** specifies that the Bundle will be offered to Workers with a Windows operating system.
  Linux specifies that the Bundle will be offered to Workers with a Linux operating system.

The 32-bit and 64-bit radio buttons can be used to further specify which processor architectures the Bundle should be offered to. Clicking the 32-bit radio button will make the Bundle only available for 32-bit Workers. Respectively, selecting the 64-bit radio button will make the Bundle only available for 64-bit Workers. If no selections are made, the Bundle will be offered to 32-bit Workers.

3.3.8. Files tab

The 'Files' tab can be used to specify which files that will be included in the Bundle. The Files tab is shown below in Figure 2.

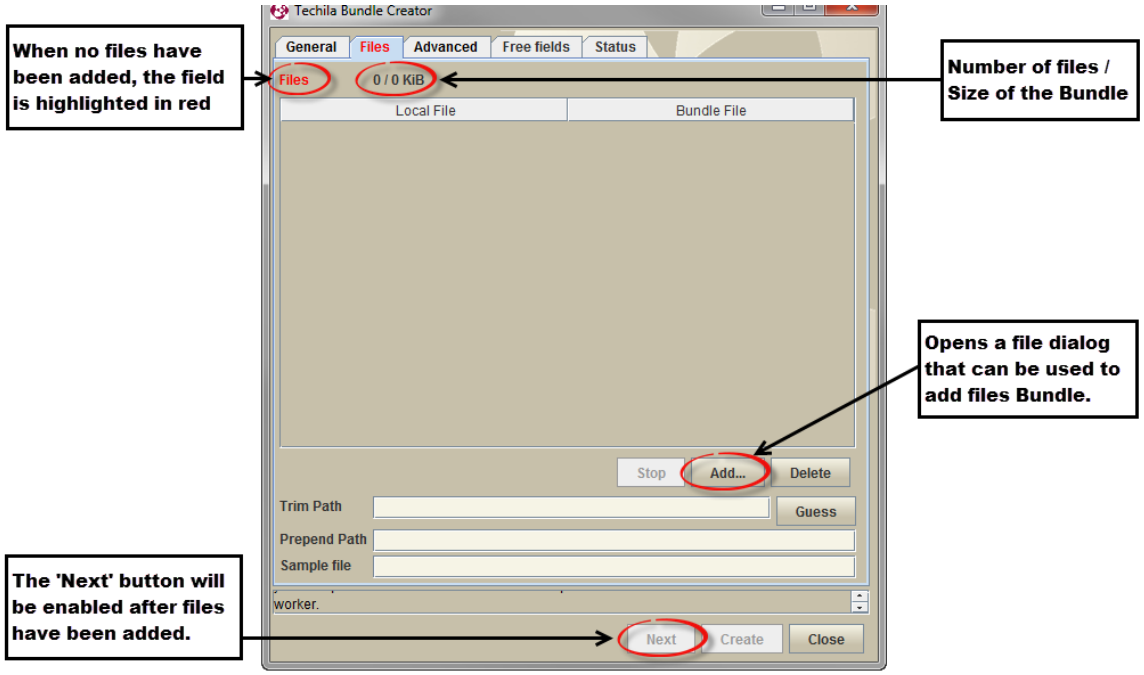

Figure 2. The 'Files' tab. Files can be added by using the 'Add…' button, which opens a file dialog window. After files have been added, the 'Next' button will be enabled. The number of files and the size of the Bundle can be seen at the top of the tab.

Files can be added by clicking the **Add…** button and using the file dialog window to select the folders and/or files that should be placed in the Bundle. Several folders can be added by pressing the Ctrl-key while selecting the folders. The **Stop** button can be used to stop the file adding process.

Files can be removed from the Bundle by selecting the files in the Files section and clicking the **Delete** button. Several files can be selected simultaneously by pressing either the Ctrl or Shift key while selecting files.

A list of required files that need to be included in the programming language specific Runtime Bundles can be found in the corresponding Chapter for each programming language in Chapter 5.

3.3.9. Guess button

The **Guess** button attempts to correctly set the values for the 'Prepend Path', 'Trim Path', 'Sample file', 'Natives' and other required fields. The **Guess** button is enabled or disabled based on what template is selected on the 'General' tab. The Guess button is enabled for the following templates:

- MATLAB Runtime
- Perl Runtime
- R Runtime

- Python Runtime

If the Guess button is not enabled for the template you are using, you will have to enter the values for the following parameters manually: 'Prepend Path', 'Trim Path' and 'Sample file' .

3.3.10. Trim Path

The 'Trim Path' field can be used to modify the location of files on the Techila Worker by removing (unnecessary) paths. To trim the path, simply enter the path that should be trimmed from the path of the file. The path will be updated in a continuous manner as the value of the Trim path field is updated. It also possible to use regexp notations when trimming the path. Please see Chapter 2.1.7 for a list of characters that need to be escaped.

3.3.11. Prepend Path

The 'Prepend Path' field can be used to modify the location of files on the Techila Worker by adding a prefix to the path. To modify the path, simply enter the path that should be prepended to the path of the file. The path will be updated in a continuous manner as the value of the Prepend path field is updated.

3.3.12. Sample file

A sample file is chosen automatically after files are added to the Bundle. If you wish to select a different sample files, simply click on the desired file in the Files section. Clicking the Guess button will also automatically choose a sample file when using one of the Runtime Bundle templates listed below:

- MATLAB Runtime
- Perl Runtime
- R Runtime
- Python Runtime

3.4. Advanced tab

The fields in the 'Advanced' tab can be used to specify several advanced parameters for the Bundle. Please note that typically the fields in the 'Advanced' tab do not require any manual configuration unless otherwise stated.

3.4.1. Natives

The 'Natives' field displays the operating systems and processor architectures for which the Bundle will be offered to. Selections made in the Platforms section will automatically be updated to the Natives field.

3.5. Environment

The 'Environment' field can be used to define environment parameters for the Bundle. This field will be automatically filled when creating a Bundle using the Templates listed below:

- Perl Module
- R Module

3.5.1. File Permissions

The 'File Permissions' field can be used to define permissions for files when creating Bundles for Linux Techila Workers. This field will be automatically filled when creating a Bundle using the Template / Platform combinations listed below:

- MATLAB Runtime / Linux

3.5.2. External Resources

The 'External Resources' field can be used to define external resources for the Bundle.

3.5.3. Category

The 'Category' field can be used to specify a category for the Bundle. This field is typically only used in Bundles created by developers at Techila Technologies. With Bundles created by Techila End-Users and Administrators, this field will be typically left empty.

3.5.4. Create Locally

This option can be used to create the Bundle locally. When enabled, the text field can be used to define the name of the '.jar' file and the location where it will be stored. By default locally created Bundles will be stored in the current working directory.

3.6. Free fields tab

The 'Free fields' tab can be used to specify additional Bundle parameters. New empty fields can be added with the + button. Fields can be deleted by selecting a field (or several fields) and either clicking the - button or the Delete key on your keyboard.

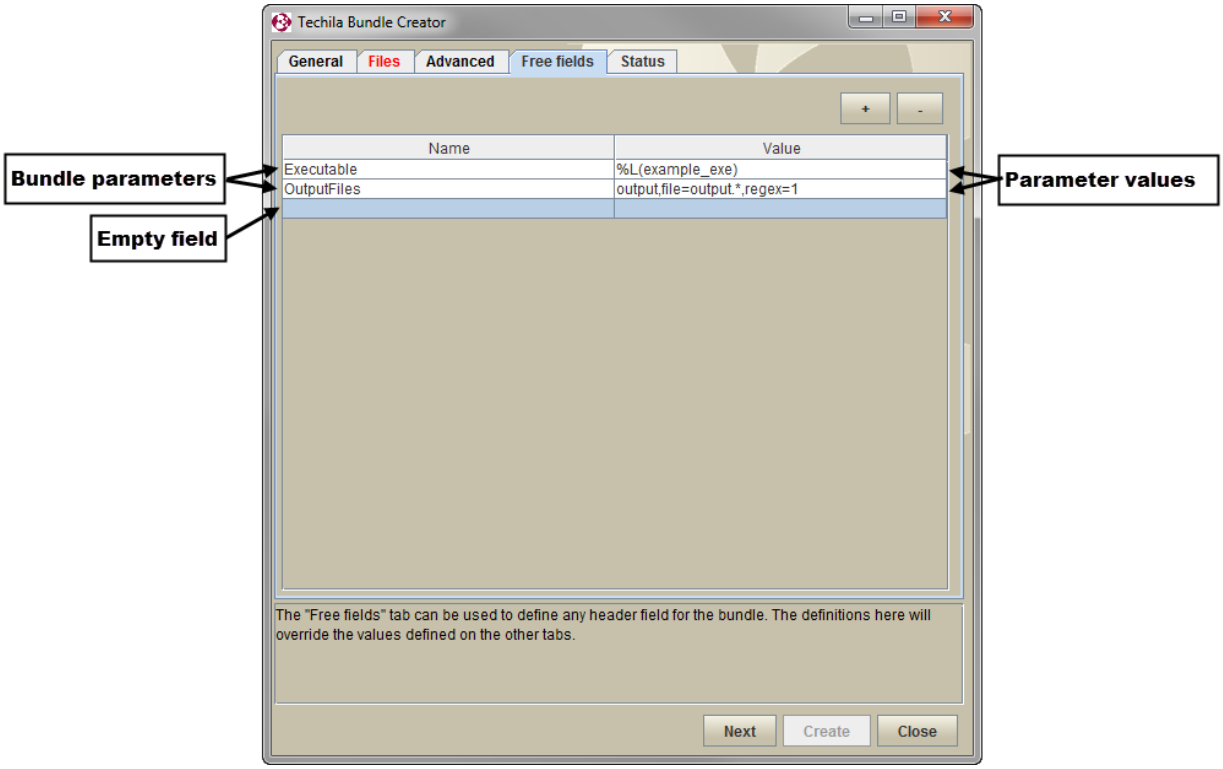

Figure 3. The Free Fields tab in the Techila Bundle Creator. The screenshot contains three fields, where two of the fields contain values and one field has been left empty. By default, all new fields will be empty.

3.7. Status tab

The 'Status' tab will be automatically displayed when the **Create** button is clicked on any of the tabs. This tab will contain information on the Bundle creation process. The 'Create', 'Upload' and 'Approve status bars will display the progress of relevant steps in the Bundle creation process.

- The 'Create' progress bar will be active when the Bundle file is being created and signed locally on your computer.
- The 'Upload' progress bar will be active when the Bundle is being transferred from your computer to the Techila Server.
- The 'Approve' progress bar will be active when the signature of the Bundle is being verified on the Techila Server.

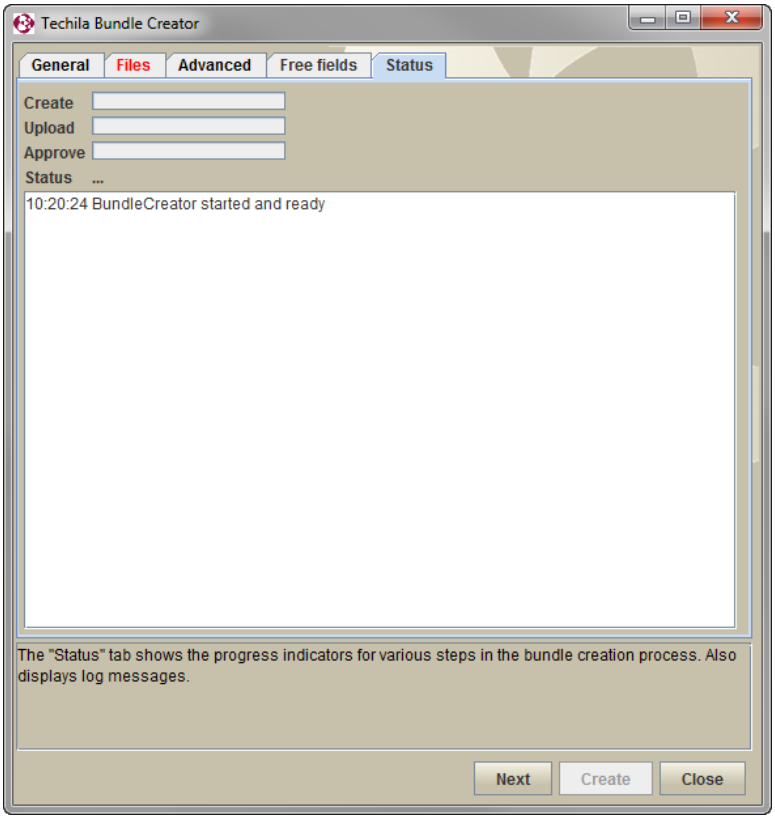

Figure 4. The 'Status' window will automatically be displayed when the 'Create' button is clicked on any of the tabs.

4. CLI createBundle command

The Techila CLI interface (the **'techila.jar'** application in the **'lib'** directory) includes the **'createBundle'** command which can be used to create Bundles using the command line interface. Accessing the CLI interface will display the internal help file, which contains information on the parameters of the **'createBundle'** command. You can display the internal help using command:

```
java -jar <full path>\techila.jar
```

A screenshot of the internal help is included below in Figure 5.

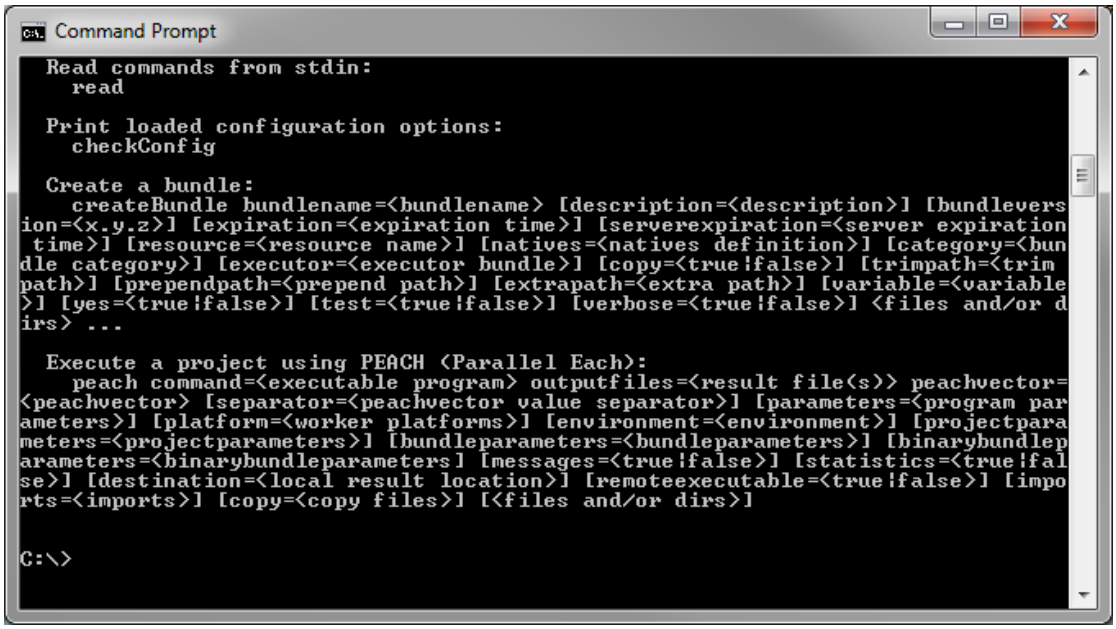

Figure 5. Parameters of the createBundle command are listed in the internal help.

Parameters enclosed in '<>' are mandatory parameters, meaning they must be defined when creating a Bundle. Parameters enclosed in brackets '[]' are optional parameters, meaning a value does not have to be defined when executing the **'createBundle'** command.

4.1. CLI createBundle parameters

This Chapter contains a description of the parameters of the CLI **'createBundle'** command.

4.1.1. bundlename

The **'bundlename'** parameter is a mandatory parameter, which used to define the name of the Bundle. For example, the following syntax would define that the name of the Bundle to be **"Techila Worker MATLAB v714 Windows x86".**

```
bundlename="Techila Worker MATLAB v714 Windows x86"
```

4.1.2. description

The **'description'** parameter can used to define a description for the Bundle. For example, the following syntax would define that the description of the Bundle would be: **"Techila Worker MATLAB v714 Windows".**

```
description="Techila Worker MATLAB v714 Windows"
```

4.1.3. bundleversion

The **'bundleversion'** parameter can be used to define a version for the Bundle. For example, following syntax would define that the version of the bundle is 0.0.1

```
version=0.0.1
```

Typically this parameter does not have to be defined.

4.1.4. expiration

The **'expiration'** parameter can be used to define the Bundle expiration period. This will determine how long an unused Bundle will be stored on the Workers before it will be removed. For example, the following syntax would define a 15 day expiration period for the Bundle.

```
expiration=15d
```

4.1.5. serverexpiration

The **'serverexpiration'** parameter can be used to the defined the minimum time how long an unused Bundle will be stored on the Techila Server before it will be removed. Note that Bundles will only be removed if the Techila Server has been configured to remove expired Bundles. For example, the following syntax would define a 15 day server expiration period for the Bundle.

```
serverexpiration=15d
```

4.1.6. resource

The **'resoure'** parameter is used to define the Bundle resource. For example, the following syntax would define **'matlab'** as the Bundle resource, indicating that the Bundle is a MATLAB Runtime Bundle.

```
resource=matlab
```

4.1.7. natives

The Bundle platforms are defined with the **'natives'** parameter, which is a list of elements defining the operating systems and processor architectures. Each list element contains the path of a sample file, target operating system and target processor architecture. Each target platform is entered separately.

One list element of the **'native'** parameter is defined as follows:

```
natives="<path of the sample file>;osname=<operating system>;processor=<arch>"
```

Available values for the **'osname'** parameter are:

- WindowsXP
- Windows2000
- Windows Vista
- Windows 7
- Windows 2003
- Windows Server 2008
- Linux

Available values for the **'processor'** parameter are:

- amd64 (Windows and Linux 64-bit)

In addition to the operating system and processor architectures, the sample file must be defined. The file can be chosen quite freely, the only prerequisite is that the file will be included in the Bundle. This file is used to determine the path of the files on Workers with different platforms.

For example, when creating a Perl Runtime Bundle for 64-bit Linux Workers, the **'natives'** parameter would be defined as shown below:

```
natives="usr/bin/perl;osname=Linux;processor=amd64"
```

4.1.8. category

The **'category'** parameter can be used to define a category of the Bundle. Typically this parameter does not need to be defined.

4.1.9. executor

The **'executor'** parameter can be used to define the export of the Bundle that contains the executable file. Typically when creating Runtime, Module or Data Bundles this parameter does not have to be defined.

4.1.10. copy

The **'copy'** parameter defines if the files in the Bundle will be copied to the temporary working directory on the Techila Worker. To enable files to be copied, use the syntax shown below:

```
copy=true
```

4.1.11. trimpath & prependpath

The **'trimpath'** and **'prependpath'** parameters can be used to modify the path where the files in the Bundle will be extracted on the Techila Worker.

Tip! If you are unsure on what values should be given to the **'trimpath'** and **'prependpath'** parameters, it is advisable to perform the first tests without any parameters that modify the path. This will provide information on what modifications to the path are required. When performing initial tests, enable the test mode with the following parameter:

```
test=true
```

4.1.12. extrapath

The **'extrapath'** parameter can be used to define an additional path that will be appended to the path of the files on the Techila Worker. Typically this parameter does not need to be defined.

4.1.13. variable

The **'variable'** parameter provides an alternative method for defining environment variables on Workers. The variable parameter is typically used when creating any of the Bundles listed below:

- Perl Module Bundle
- R Module Bundle

The syntax for defining the value of the **'variable'** parameter is listed below:

For Perl:

```
variable=perl5lib
```

For R:

```
variable=r_libs
```

4.1.14. yes

The **'yes'** parameter can be used to disable the confirmation prompt. This can be useful in situations where you wish to execute the **'createBundle'** command without any manual intervention. To disable the confirmation prompt, use the parameter shown below:

```
yes=true
```

4.1.15. test

The **'test'** parameter can be used to enable the test-mode, which can be used to analyse the output of the **'createBundle'** command. The test-mode can be enabled with the following parameter:

```
test=true
```

4.1.16. verbose

The `'verbose'` parameter can be used to reduce the amount of information displayed when executing the `'createBundle'` command. When verbose mode is disabled, the files that will be included in the Bundle will not be listed when creating the Bundle. To disable the verbose mode, use the following parameter:

```
verbose=false
```

4.1.17. output

When the `'output'` parameter is defined, the Bundle will be created locally with the specified name. This means the Bundle will not be transferred to the Techila Server. For example, the following parameter would create a Bundle locally with the name `'example_bundle.jar'` and store the file in the current working directory.

```
output="example_bundle.jar"
```

4.1.18. File Permissions

File permissions for certain key files can be defined using the `'bundleparameters'` parameter. Defining file permissions is required when creating one of the following Bundles:

- MATLAB Runtime Bundle for Linux (file: matlab_helper)

The general syntax for defining file permissions is shown below:

```
FilePermissions=<path of the file on the Worker>/<file name>:<permissions>
```

For example, the following syntax would define necessary rights for the file `'matlab_helper'` file belonging to a MATLAB Runtime Bundle for 64-bit Linux Workers that is created from the MCR version 7.14:

```
FilePermissions=amd64/Linux/v714/bin/glnxa64/matlab_helper:rwx
```

4.1.19. Adding files and/or directories

Files and/or directories that should be included in the Bundle are listed as the last input arguments when executing the `'createBundle'` command. Several files and/or folders can be defined by separating each entry with a space ( ). For example, the following syntax would include the files (or folders) datafile1 datadir1 from the current working directory:

```
createBundle <other parameters> datafile1 datafile2
```

You can also use regular expressions to define the names of the files and folders that should be included. For example, the following syntax would use the asterisk notation (*) to include all files and folders in the current working directory in the Bundle.

```
createBundle <other parameters> *
```

5. Creating Runtime Bundles

This Chapter contains instructions on how to create Runtime Bundles for MATLAB, Perl, Python and R. Instructions will be provided in the form of examples both for the Techila Bundle Creator tool and for the CLI **'createBundle'** command.

Notes

- The notation **%techila%** is used to represent the commands necessary access the CLI interface. This notation corresponds to the following syntax:

  ```
  java -jar <full path>/techila.jar
  ```

  For more information on how to create an environment variable for accessing the CLI interface, please refer to the document "Techila Distributed Computing Engine with Command Line Interface".

- Before creating Runtime Bundles, please verify the terms of your license of your runtime application.

Prerequisites

- Access to a signed End-User Key is always required when creating a Runtime Bundle.
- Your **'techila_settings.ini'** file must contain correct End-User Key details (alias and keystore parameters) and Techila Server details (hostname and port parameters)

5.1. MATLAB Runtime Bundle

MATLAB Runtime Bundles are created from the files that are generated when the MATLAB Runtime Compiler (MCR) is installed. MCR's are typically included with MATLAB distributions that are equipped with a compiler.

This Chapter contains information on how to create a MATLAB Runtime Bundle. A short description on the installation procedure of the MATLAB Compiler Runtime is also provided.

The table below contains a description on the parameters that can be used as a reference when creating the MATLAB Runtime Bundle.

PARAMETER	VALUE	NOTES
Bundle Name	Techila Worker MATLAB <MCRversion> <OS> <arch>	<MCRversion> is version of MATLAB Compiler Runtime. <OS> is operating system of Techila Worker <arch> is the processor architecture of Techila Worker
Exports	fi.techila.grid.cust.common.library.matlab.v<MCRversion>.client	When using the CLI, the <MCRversion> notation in the exports parameter needs to be replaced MCR version that you will use when creating the Bundle. For example, if the Bundle is created from MCR version 7.14 (MATLAB 2010b), the <MCRversion> notation will be 714. The Runtime Bundle will be imported based on the exports parameter, meaning the exports parameter needs to follow the example shown here.
Description	Techila Worker MATLAB v<MCRversion> Windows	<MCRversion> is version of MATLAB Compiler Runtime.
Resource	matlab	Defines that the Bundle is a MATLAB Runtime Bundle. The resource parameter must be defined exactly as shown here.
Expiration	365d	Defines that the Bundle can remain unused for 365 days before it will be removed from Workers.
Files	All directories and files in MATLAB Compiler Runtime installation directory	All files and folders from the MCR installation directory must be included.
Trim Path	C:\\<MCR installation directory>	The path leading to the MCR installation directory should be trimmed. Note, the Guess button in the Bundle Creator tool will attempt to set this value automatically. Note that this path might be different on your system than in the one illustrated here.
Prepend path	<arch>\\<OS>\\v<MCRversion>\\	Note, the Guess button in the Bundle Creator tool will attempt to set this value automatically. Values for <arch>: -Windows (for Windows) -Linux (for Linux) Values for <OS>: -x86 (32-bit Windows) -i386 (32-bit Linux) -amd64(64-bit Windows)
Sample file	Windows: <arch>\\Windows\\<MCRversion>\\patents.txt Linux: <arch>/Linux/<MCRversion>/patents.txt	After clicking the Guess button in the Techila Bundle Creator tool the path of the sample file should match the ones illustrated. When using the CLI 'createBundle' command, you should use the trimpath and prependpath parameters to modify the path of the sample so it matches the ones illustrated here.

Natives	amd64\Windows\v714\patents.txt;osname=WindowsXP;processor=amd64,amd64\Windows\v714\patents.txt;osname=Windows 2000;processor=amd64,amd64\Windows\v714\patents.txt;osname=Windows Vista;processor=amd64,amd64\Windows\v714\patents.txt;osname=Windows 7;processor=amd64,amd64\Windows\v714\patents.txt;osname=Windows Server 2008;processor=amd64	The natives defines the operating systems and processor architectures the Bundle will be offered to. Note! When using the Techila Bundle Creator tool, selections made in the Platforms section will be automatically update the Natives parameter. When using the CLI, the natives parameter needs to be defined manually. The example on the left illustrates the value of the natives parameter when creating a MATLAB Runtime Bundle for 64-bit Windows Workers.

The MATLAB Compiler Runtime (MCR) versions corresponding to different MATLAB versions can be found on the following MathWorks website: http://www.mathworks.com/support/solutions/en/data/1-4GSNCF/

5.1.1. Installing MATLAB Compiler Runtime for Windows

This Chapter contains a short description on how to install a MATLAB Compiler Runtime. Please note that the installation procedure is described for a Windows computer. If you are installing the MCR on a different operating system, some of the steps might differ from the ones listed here.

Prerequisites

- MATLAB Compiler Runtime (MCR) Installer for the platform the Runtime Bundle will be created for

Notes

- The appearance of pages and dialogs may differ depending on your MATLAB Compiler Runtime version and operating system. Screenshots in this section are for Windows XP and MATLAB Compiler Runtime version 7.9 (MATLAB 2008b)
- MCR Installers for latest MATLAB versions can also be downloaded from the MathWorks website: http://www.mathworks.se/products/compiler/mcr/

Procedure

1. Start the MCRinstaller application. A Choose Setup Language window will be displayed. Select a language and click **OK**

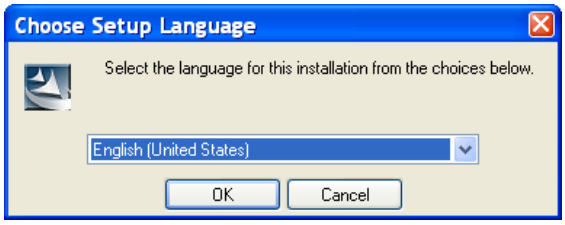

Figure 6. Select the language.

2. A requirement window will be displayed. Install all requirements by clicking **Install**.

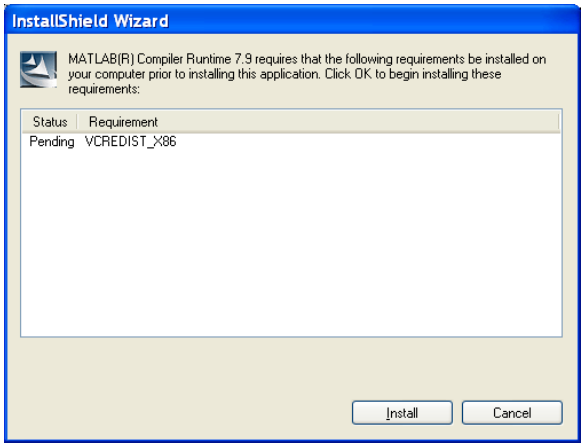

Figure 7. Install all required requirements.

3. After all requirements have been installed, the MATLAB Compiler Runtime Install Shield Wizard will be displayed. Click **Next**.

Figure 8. Click Next to proceed with the installation.

4. A Customer Information window will be displayed. Fill in the required information. Click **Next**.

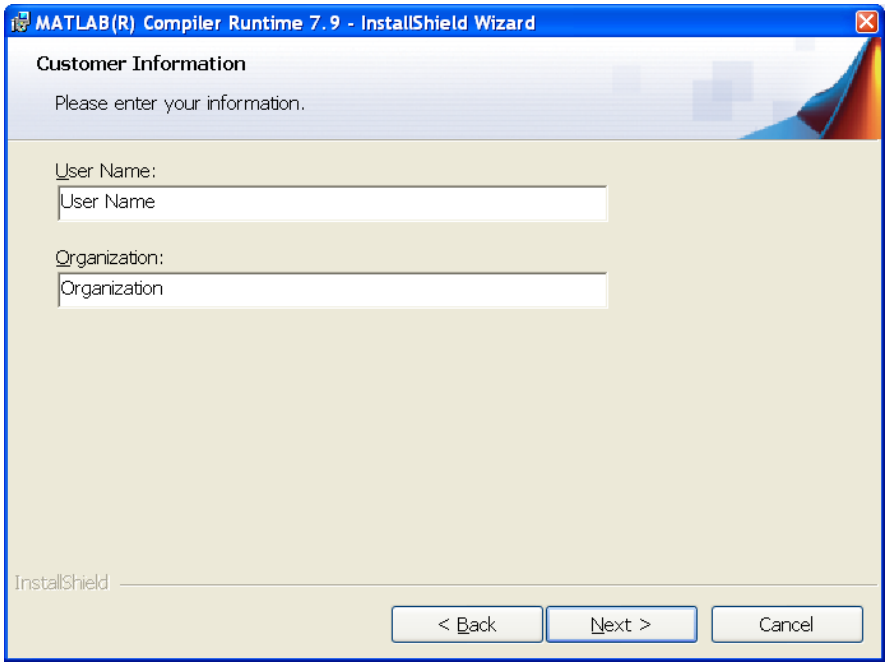

Figure 9. Fill in the required fields.

5. A Destination Folder window will be displayed. Choose where to install the Compiler Runtime. The files should be installed on a local hard disk to speed up the Bundle creation process. Click **Next**.

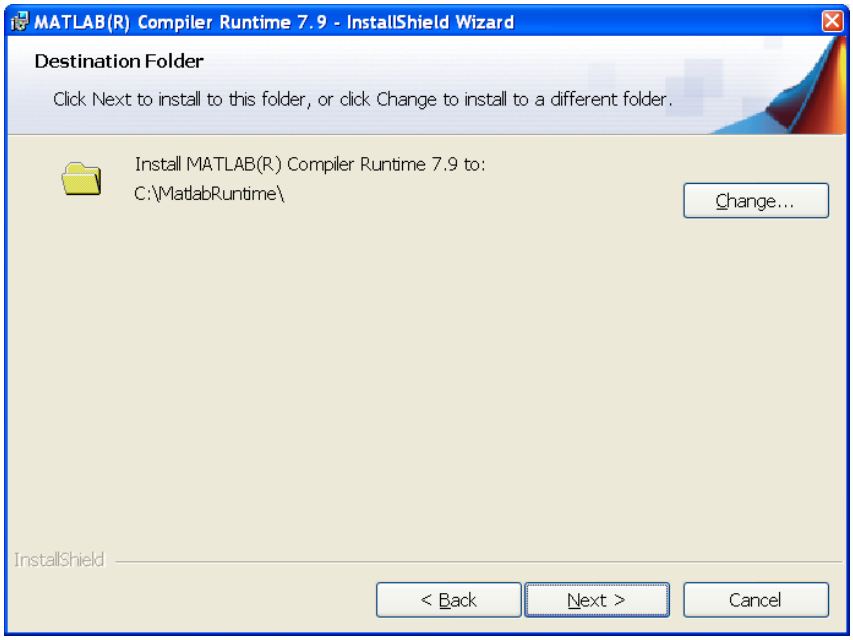

Figure 10. Select a location for the installation.

6. A confirmation window will be displayed. Click **Install** to install the MATLAB Compiler Runtime.

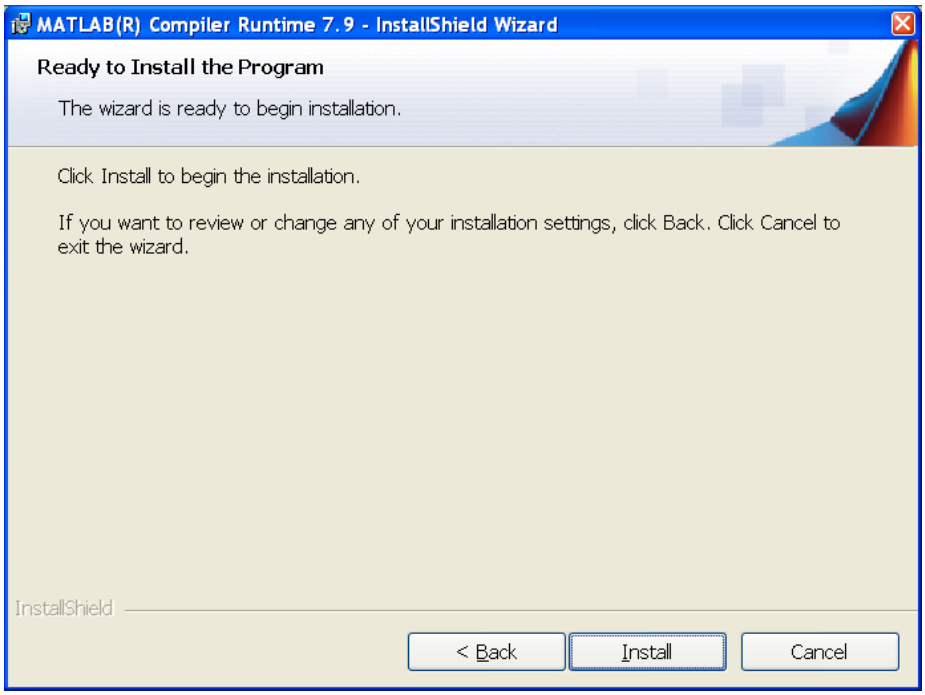

Figure 11. Start the installation process.

7. A window will be opened displaying the progress of the installation.

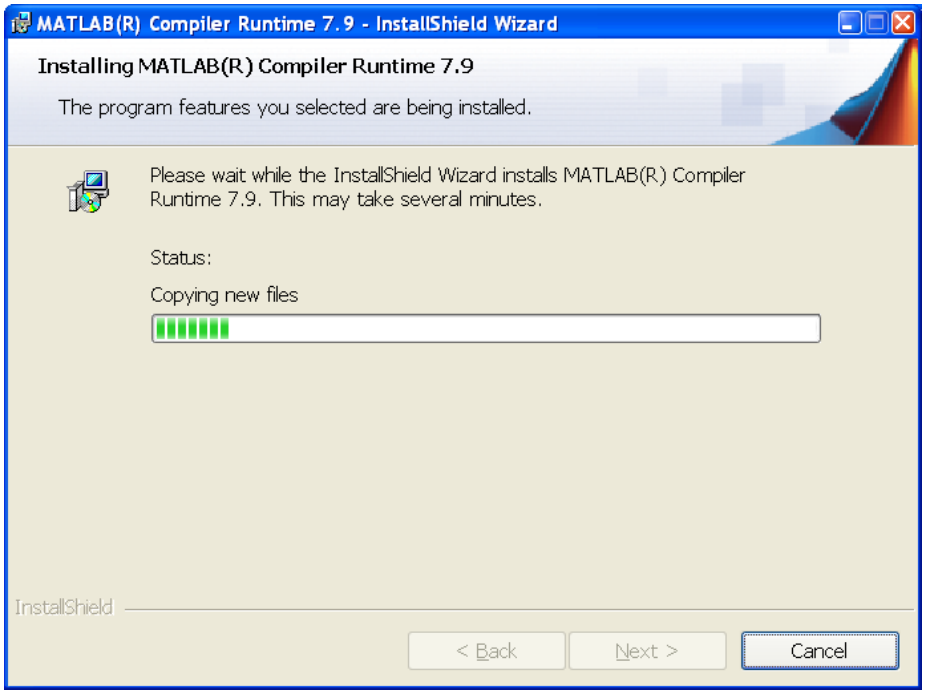

Figure 12. Wait until all files have been installed.

Wait until the installation is complete.

8. When the installation is complete, a new window will be displayed. Click **Finish**.

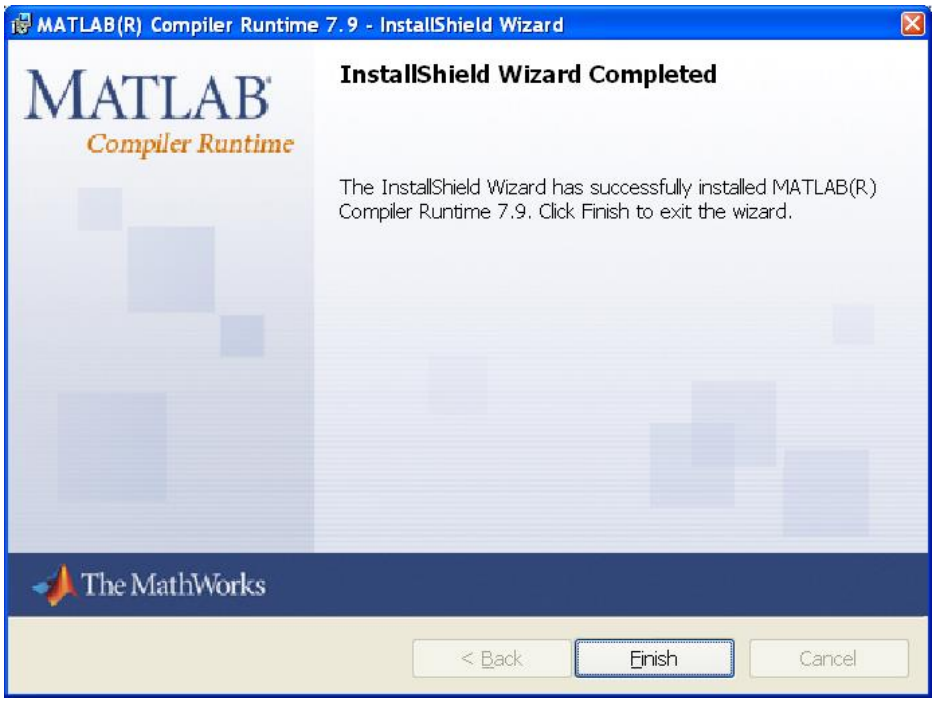

Figure 13. Close the InstallShield Wizard.

5.1.2. Creating a MATLAB Runtime Bundle with the Bundle Creator tool for Windows

This Chapter contains instructions on how to create an MATLAB Runtime Bundle using the Bundle Creator tool on a Windows platform.

Notes

- The MCR version used in this example is v80 which corresponds to the MATLAB 2012b release.

Procedure

1. Launch the Techila Bundle Creator tool by navigating to the **'lib'** directory in the Techila SDK and double clicking on the '**bundlecreator.jar'** icon.

2. Check the location of 'techila_settings.ini' file and correct if needed

3. Choose the **Matlab Runtime** template from the **Template** drop-down menu

4. Modify the free text field in the Bundle Name field to match the MATLAB Compiler Runtime (MCR) version you are creating the Runtime Bundle from. A list of MCR versions used by different MATLAB versions can be found at:

   http://www.mathworks.com/support/solutions/en/data/1-4GSNCF/

5. Choose applicable operating systems and processor architectures in the **Platforms** section. This step determines which Workers will be able to use the Bundle. After this step, the view in the 'General' tab should resemble the one shown below. Click the **Next** button to continue to the 'Files' tab.

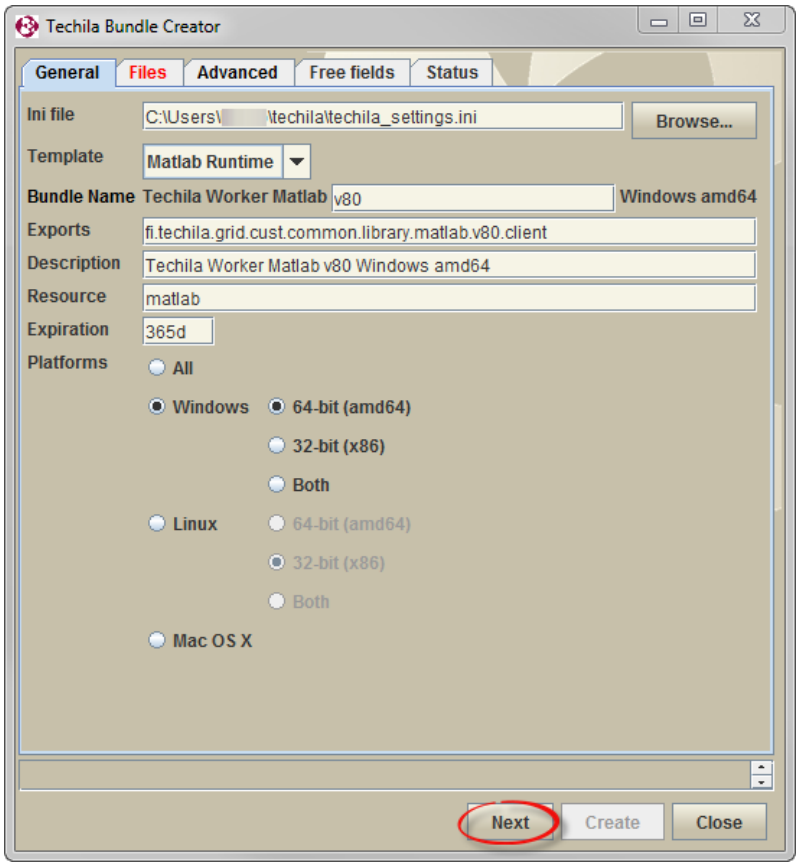

Figure 14.The 'General' tab view after configuring all required fields.

6. On the 'Files' tab, click the **Add..** button and add the following directories from the MCR installation directory:

   - All files and folders in the MCR installation directory

7. Click the **Guess** button. Clicking the button will attempt to correctly set values for the following fields: **Prepend Path**, **Trim Path** and **Sample File**. The Sample File field should contain the following value after clicking the Guess button:

   - <arch>\Windows\<MCRversion>\patents.txt

   After this step, the 'Files' tab should resemble the one shown below.

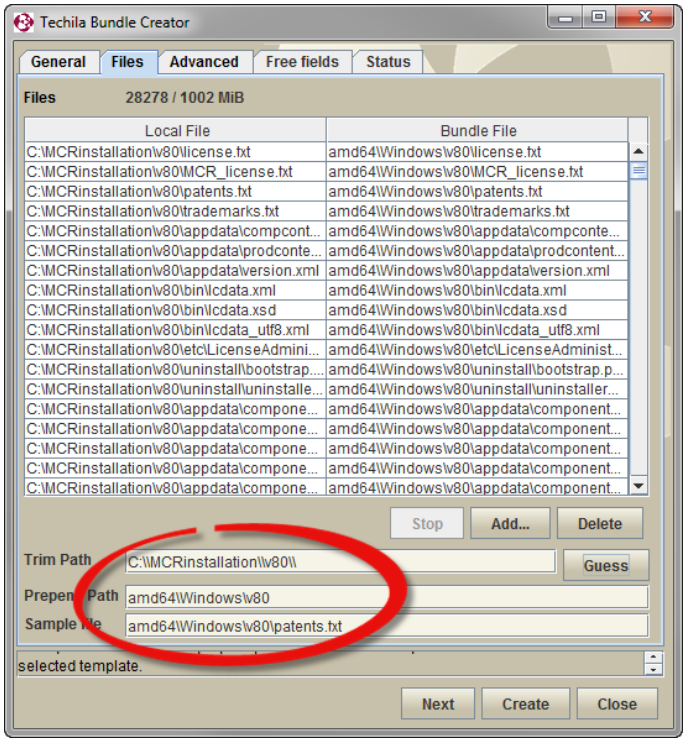

Figure 15. The 'Files' tab after adding the files from the MCR installation directory. Path has been trimmed by using the 'Guess' button

8. Click the **Create** button. After clicking the button, the 'Status' tab will be displayed which will contain information on the Bundle creation process. After successfully creating the Runtime Bundle, the Status tab should resemble the one shown below.

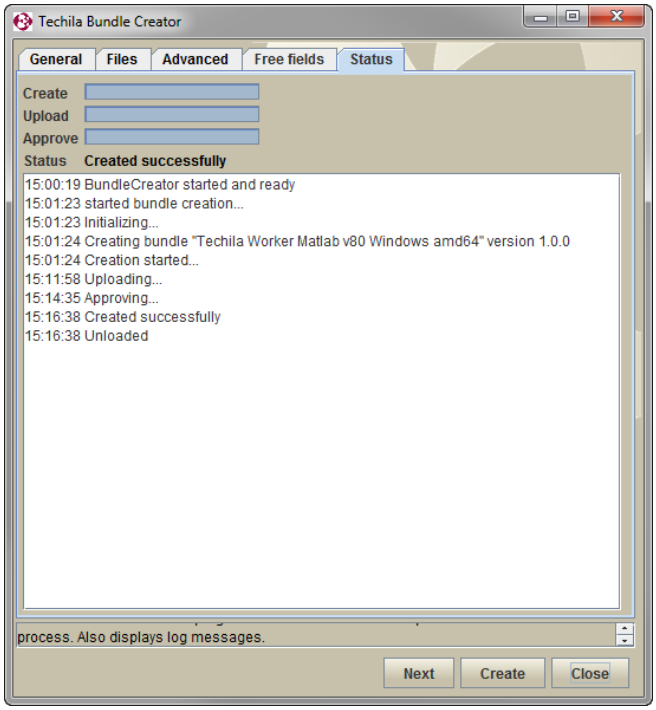

Figure 16. The 'Status' tab after successfully creating a MATLAB Runtime Bundle.

9. Close the Techila Bundle Creator tool window by clicking the **Close** button.

5.1.3. Creating a MATLAB Runtime Bundle with the CLI for Windows

This Chapter describes the procedure for creating a MATLAB Runtime Bundle using the CLI `'createBundle'` command.

Notes

- The MCR version used in the Techila Bundle Creator tool example is v80 which corresponds to the MATLAB 2012b release.

Procedure

1. Open a command prompt and change the current working directory where you have installed the MATLAB Compiler Runtime.

Figure 17. Navigate to the directory containing the MCR installation.

2. This step includes creating the Bundle with the `'createbundle'` command. The files that were extracted in steps 1-8 in Chapter 5.1.1 will be stored in the MATLAB Runtime Bundle and transferred to the Techila Server.

   Create and upload the MATLAB Runtime Bundle with the command:

```
%techila% createbundle bundlename="Techila Worker MATLAB v80"
resource=matlab
exports=fi.techila.grid.cust.common.library.matlab.v80.client
natives="amd64/Windows/v80/patents.txt;osname=WindowsXP;processor=amd64,am
d64/Windows/v80/patents.txt;osname=Windows2000;processor=amd64,amd64/Windo
ws/v80/patents.txt;osname=Windows Vista;processor=amd64,amd64
/Windows/v80/patents.txt;osname=Windows
7;processor=amd64,amd64/Windows/v80/patents.txt;osname=Windows
2003;processor=amd64,amd64/Windows/v80/patents.txt;osname=Windows Server
2008;processor=amd64" expiration=365d yes=true description="Techila Client
MATLAB v80 Windows" trimpath=.\\\\ prependpath=amd64/Windows *
```

Note, depending on what MATLAB version you are using and other system specific parameters, you might need to adjust the values of the following parameters:

- bundlename (update the v80 notation to the MCR version you are using)
- exports (update the v80 notation to match the MCR version you are using)

- natives (replace with a parameter that contains the operating systems and processor architectures you wish to make the bundle available for)
- description (update the v80 notation to the MCR version you are using)

3. After executing the command, the files to be included in the Bundle will be displayed on the page. The Bundle will be created and transferred to the Techila Server. The signature of the bundle will be verified automatically on the Techila Server. Bundles signed with End-User Key, which has a signed chain leading to the Master Administrator Key will be approved.

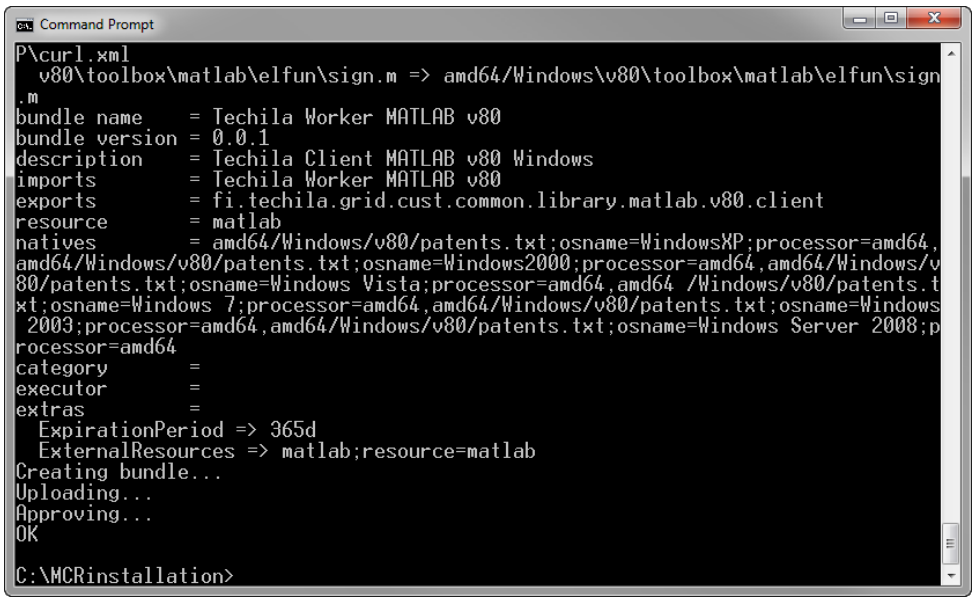

Figure 18. The view after executing the command. After the Bundle has been transferred and approved by the Techila Server a message stating "OK" will be displayed.

The MATLAB Runtime Bundle is now ready for use.

5.1.4. Installing MATLAB Compiler Runtime for Linux

This Chapter contains a short description on how to install a MATLAB Compiler Runtime on a computer with a Linux operating system.

Prerequisites

- MATLAB Compiler Runtime (MCR) Installer for Linux operating system

Notes

- The appearance of pages and dialogs may differ depending on your MATLAB Compiler Runtime version and operating system. Screenshots in this section are for CentOS operating system and MATLAB Compiler Runtime version 7.14 (MATLAB 2010b)
- MCR Installers for latest MATLAB versions can also be downloaded from the MathWorks website at: http://www.mathworks.se/products/compiler/mcr/

Procedure

1. Navigate to the folder containing the MCR installer. In this example, the path to the installer is:

   ```
   /usr/local/MATLAB/R2010b/toolbox/compiler/deploy/glnxa64/MCRInstaller.bin
   ```

   Note! The location of the installer can be displayed by executing the following command in MATLAB:

   ```
   mcrinstaller
   ```

2. Launch the installer using command the following command:

   ```
   ./MCRInstaller.bin
   ```

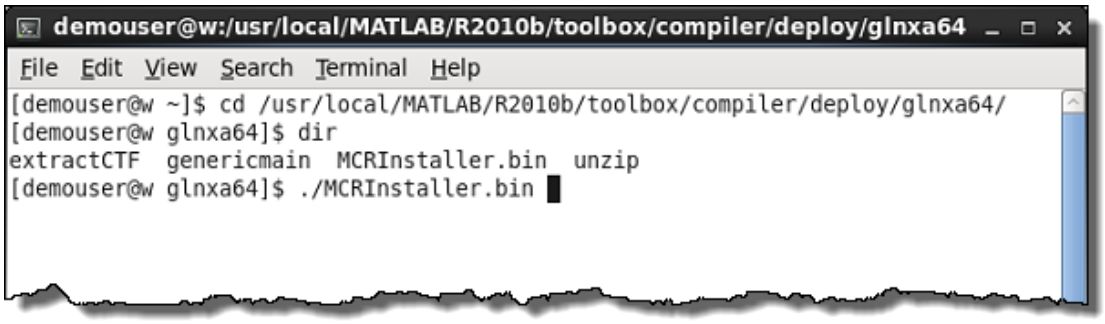

Figure 19. Launching the installer.

3. Click **Next** to continue with the installation procedure.

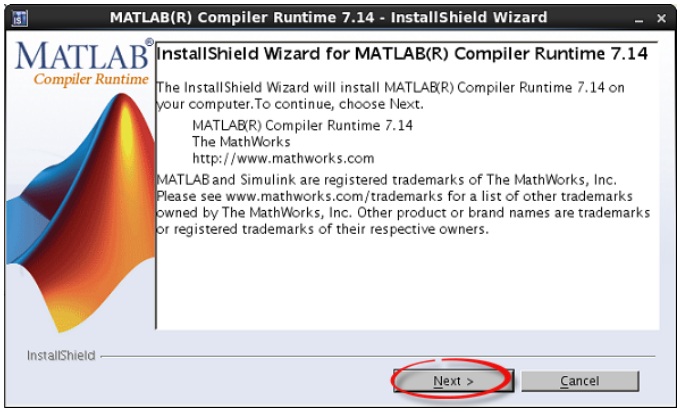

Figure 20. Click next to continue.

4. Specify the location where the MCR will be installed. This can be a temporary directory, as the files will only be required during the Runtime Bundle creation process. Click **Next** to continue.

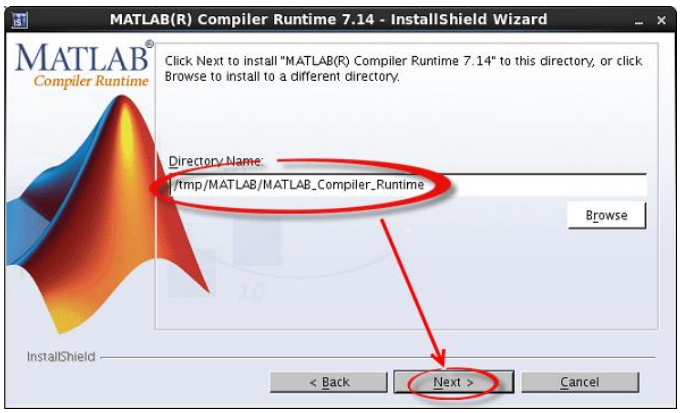

Figure 21. After selecting where to install the MCR, click Next.

5. Click **Install** to begin the installation process.

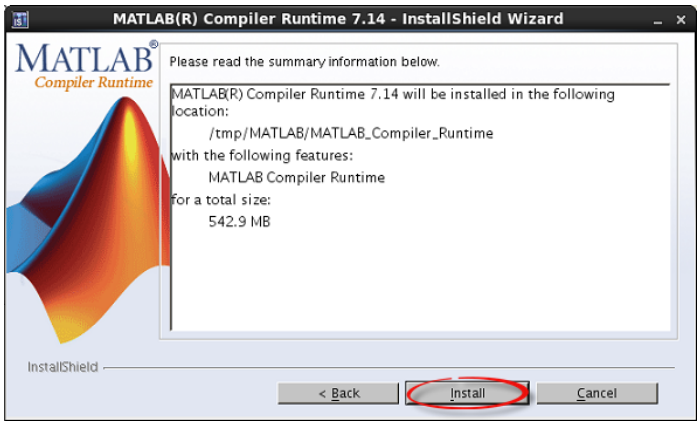

Figure 22. Start the installation process by clicking Install.

6. Wait until the installation process has been completed. Click **Finish** to close the installation wizard.

Figure 23. After completing the installation process, close the installer by clicking Finish.

The MATLAB Compiler Runtime has now been successfully installed on the Linux computer.

5.1.5. Creating a MATLAB Runtime Bundle with the Bundle Creator tool for Linux

This Chapter contains instructions on how to create an MATLAB Runtime Bundle using the Bundle Creator tool on a Linux platform.

Notes

- The MCR version used in this example is v714 which corresponds to the MATLAB 2010b release.
- You will need access to the files that were generated during the MCR installation process.

Procedure

1. Launch the Techila Bundle Creator tool by navigating to the **'lib'** directory in the Techila SDK and double clicking on the '**bundlecreator.jar**' icon.

   You can also launch the application from the command line by navigating to the **'lib'** directory in the Techila SDK and executing the following command: 'java -jar bundlecreator.jar'.

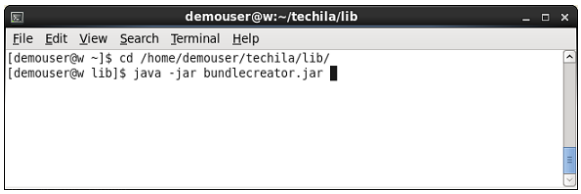

Figure 24. Launching the Techila Bundle Creator using the command line.

2. After launching the Techila Bundle Creator, check the location of 'techila_settings.ini' file and correct if needed.

3. Choose the **Matlab Runtime** template from the **Template** drop-down menu

4. Modify the free text field in the Bundle Name field to match the MATLAB Compiler Runtime (MCR) version you are creating the Runtime Bundle from. A list of MCR versions used by different MATLAB versions can be found at:

   http://www.mathworks.com/support/solutions/en/data/1-4GSNCF/

Figure 25. If necessary, replace the 'v714' with your MCR version. For example, if creating a Bundle that contains runtime components for MATLAB 2013b (MCR 8.2), the value should be set to 'v82'.

5. Choose applicable operating systems and processor architectures in the **Platforms** section. This step determines which Workers will be able to use the Bundle. After this step, the view in the 'General' tab should resemble the one shown below. Click the **Next** button to continue to the 'Files' tab.

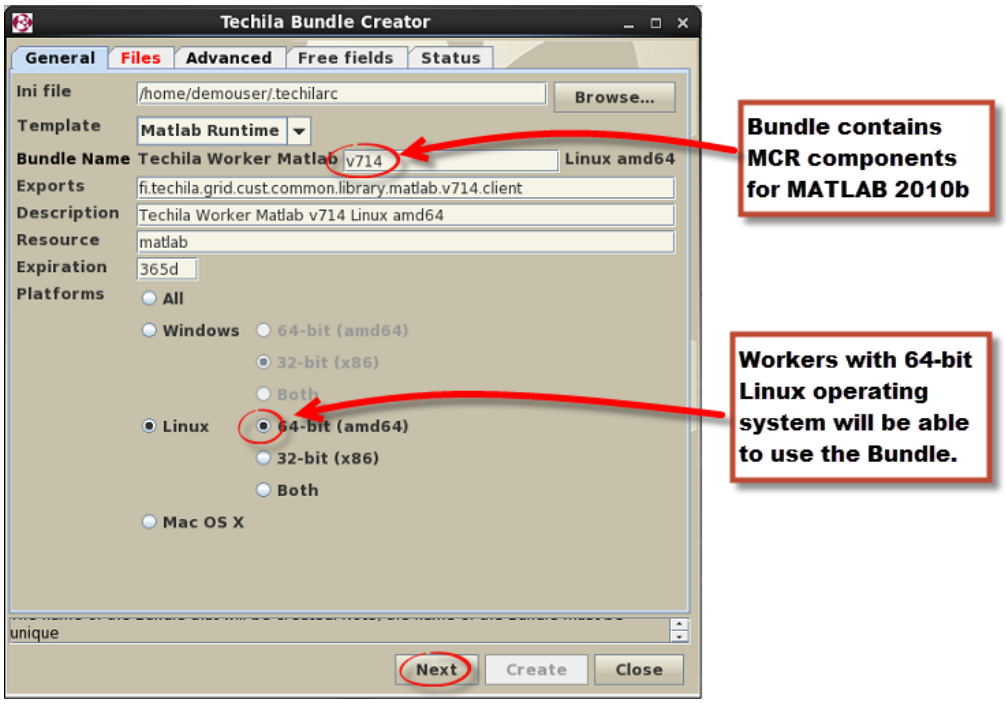

Figure 26. Specify the version and select which Techila Worker platforms can use the Bundle.

10. On the 'Files' tab, click the **Add..** button and add the following directory from the MCR installation directory:

    - v<Your MCR Version>

    In the example below, the directory 'v714' has been selected, which matches MATLAB version 2010b.

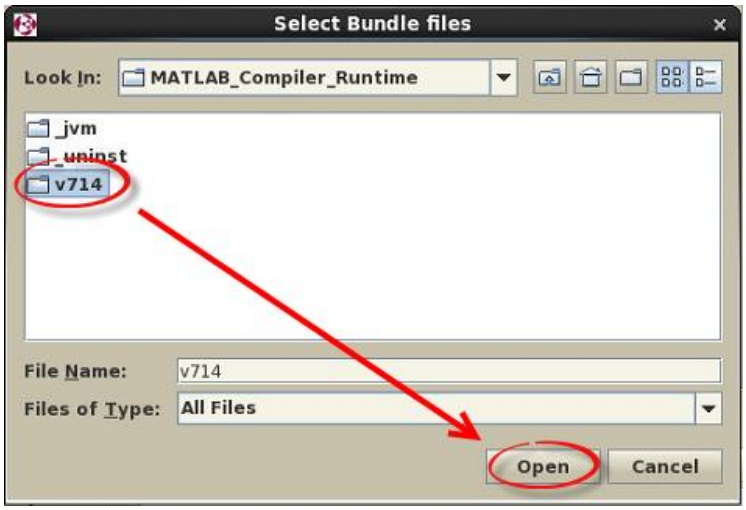

Figure 27. Select the folder containing the MCR installation.

11. After the files have been added, Click the **Guess** button.

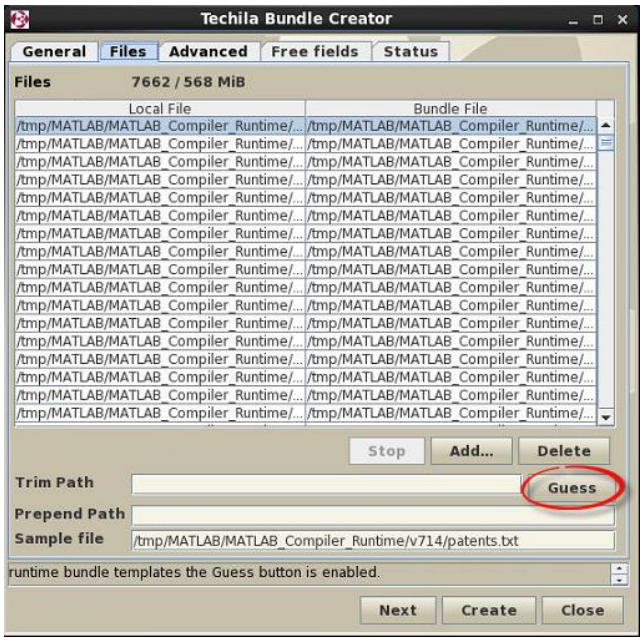

Figure 28. Click the Guess button.

12. After clicking the button, the values of the following fields should be automatically filled: **Prepend Path**, **Trim Path** and **Sample File**. The Sample File field should contain the following value after clicking the Guess button:

    - <arch>\Linux\<MCRversion>\patents.txt

    Click the **Create** button to create the Bundle.

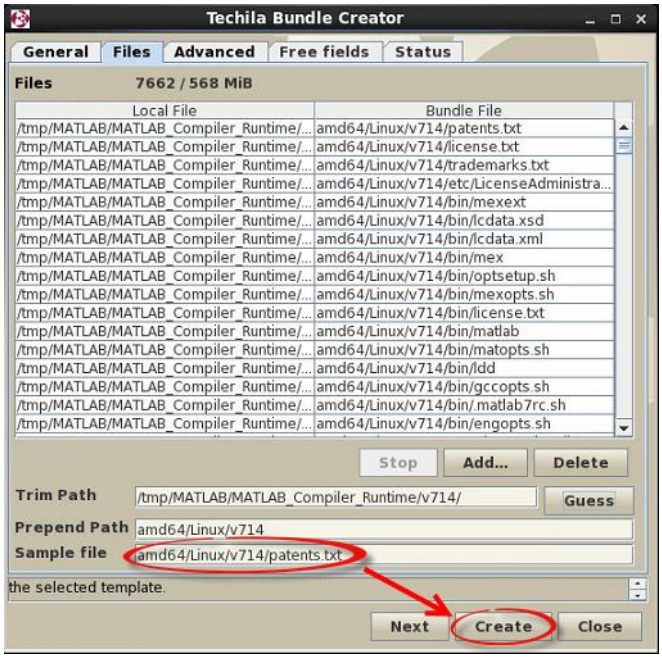

Figure 29. Click the create button to start the Bundle creation process.

13. After the Bundle has been created, the Status page should resemble the one shown below. Click the **Close** button to close the Bundle Creator Tool

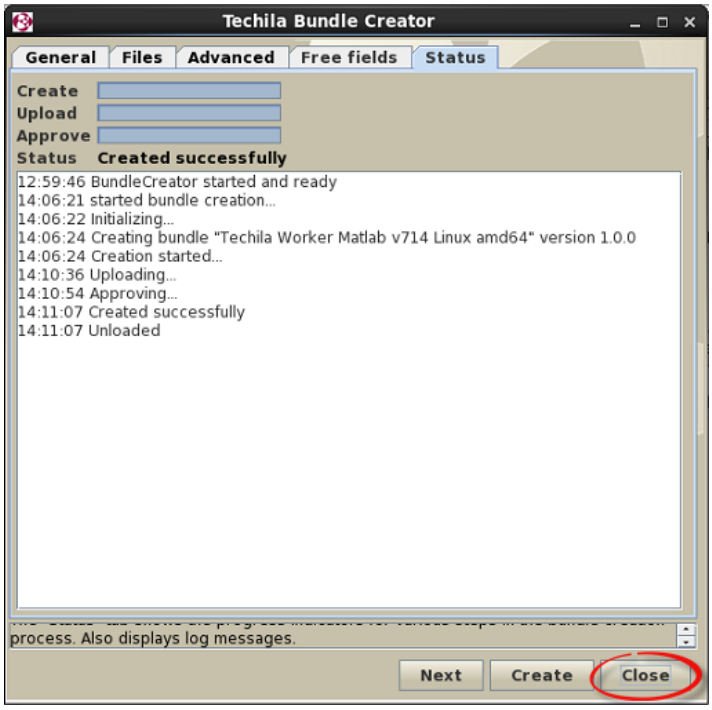

Figure 30. Click the Close button to close the Bundle Creator.

5.2. Perl Runtime Bundle

Perl Runtime Bundles are created from the files that are generated when Perl is installed on a computer. Depending on what Perl installation or operating system you are creating the Bundle from, the process will be slightly different.

This Chapter contains information on how to create a Perl Runtime Bundle.

Prerequisites

- Perl installed on a computer with the same platform as the Workers that will be allowed to use the Bundle.

The table below contains a description on the parameters that can be used as a reference when creating a Perl Runtime Bundle.

PARAMETER	EXAMPLE VALUE	NOTES
Bundle Name	Techila Worker Perl <Ver> <OS> <arch>	<Ver> is the Perl version. E.g. 5.16 <OS> is operating system of Techila Worker <arch> is platform of Techila Worker
Exports	fi.techila.grid.cust.common.library.perl.v<Ver>.client	<Ver> is the Perl version. E.g. 5.16 Sets the Bundle exports so that the Bundle can be imported into Projects created with Perl peach. Note that the exports parameter needs to be defined exactly as shown here, because the Bundle is imported based on the exports parameter.
Description	Techila Worker Perl <Ver>	<Ver> is the Perl version. E.g. 5.16 Informs that the Bundle is a Perl Runtime Bundle
Resource	perl	Defines that the Bundle is a Perl Runtime Bundle. The resource parameter must be defined exactly as shown here.
Expiration	365d	Defines that the can remain unused for 365 days before it will be removed from Workers.

Files/ directories	Windows: Following directories in Perl installation directory. **See note 1** -bin -lib -site -vendor Debian/Ubuntu: The files in the following Perl packages. **See Note 2.** - perl - perl-base - perl-modules - libfreezethaw-perl - libspiffy-perl - libio-all-perl - libgraph-perl - libheap-perl - libxml-writer-perl - libxml-parser-perl - liburi-perl - libwww-perl - libhtml-parser-perl - libhtml-tagset-perl - libhtml-tree-perl	**Note 1:** Ensure that the following Perl module is installed before creating the Runtime Bundle: - FreezeThaw **Note 2.** In order to create a Perl Runtime Bundle for Linux Workers, the files in the listed Perl packages need to be placed in a temporary staging directory. Note that the full paths of the original files must also remain unchanged. The Runtime Bundle can then be created by including the directories in the temporary staging directory.
Trim Path	C:\\<perl installation directory>\\	The path leading to the Perl installation directory (or temporary staging directory) should be trimmed. Note that the value of the trimpath depends on your system settings. Also note, the Guess button in the Bundle Creator tool will attempt to set this value automatically.
Prepend path		Need for the prependpath depends on your system settings. Note, the Guess button in the Bundle Creator tool will attempt to set this value automatically.
Sample file	perl\bin\perl.exe (Windows) usr/bin/perl (Linux)	After clicking the Guess button in the Techila Bundle Creator tool the path of the sample file should match the ones illustrated. When using the CLI 'createBundle' command, you should use the trimpath and prependpath parameters to modify the path of the sample so it matches the ones illustrated here.
Natives	perl.exe;osname=WindowsXP;processor=amd64,perl.exe;osname=Windows 2000;processor=amd64,perl.exe;osname=Windows Vista;processor=amd64,perl.exe;osname=Windows 7;processor=amd64,perl.exe;osname=Windows Server 2008;processor=amd64Windows Server 2008;processor=amd64	The natives defines the operating systems and processor architectures the Bundle will be offered to. Note! When using the Techila Bundle Creator tool, selections made in the Plaforms section will be automatically update the Natives parameter. When using the CLI, the natives parameter needs to be defined manually. The example on the left illustrates the value of the natives parameter when creating a R Runtime Bundle for 64-bit Windows Workers.

5.2.1. Creating a Perl Runtime Bundle with the Bundle Creator tool

This Chapter contains instructions on how to create a Perl Runtime Bundle using the Bundle Creator tool.

Notes

- The Perl version used in this example is a Perl 5.16 64-bit Windows.
- If you are creating Perl Runtime for Linux Workers, ensure that you have created a temporary staging directory and that all necessary files have been placed in the directory.

Procedure

1. Launch the Techila Bundle Creator tool by navigating to the **'lib'** directory in the Techila SDK and double clicking on the **'bundlecreator.jar'** icon.

2. Check the location of 'techila_settings.ini' file and correct if needed

3. Choose the **Perl Runtime** template from the **Template** drop-down menu

4. Modify the free text field in the Bundle Name field to match the Perl version you are using. By default, the free text field will contain the value v516, corresponding to Perl version 5.16.

5. Choose applicable operating systems and processor architectures in the **Platforms** section. This step determines which Workers will be able to use the Bundle. After this step, the 'General' tab should resemble the one shown below in Figure 31. Click the **Next** button to continue to the Files tab.

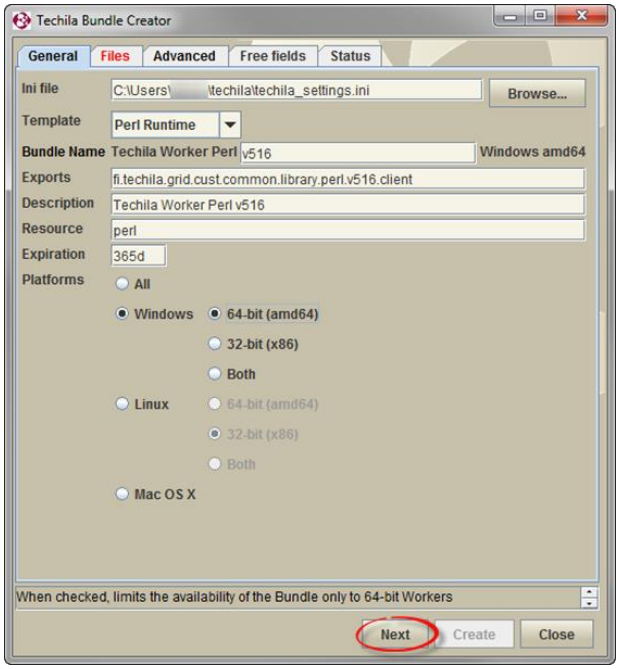

Figure 31. The 'General' tab view after configuring all required fields.

6. On the 'Files' tab, click the **Add..** button to open the file dialog window

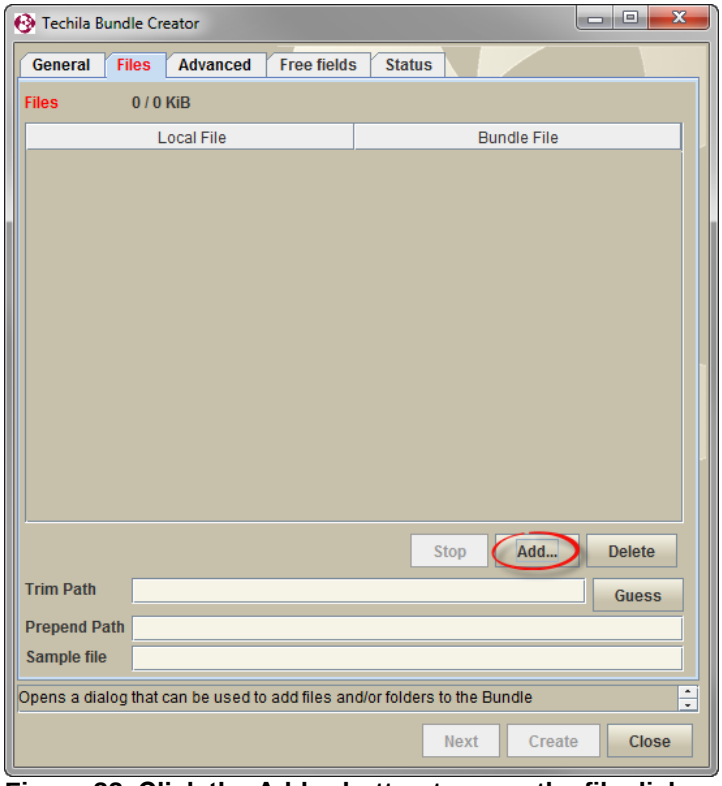

Figure 32. Click the Add… button to open the file dialog window.

7. Using the file dialog window, add the following directories from the Perl installation directory:

- bin
- lib
- site
- vendor

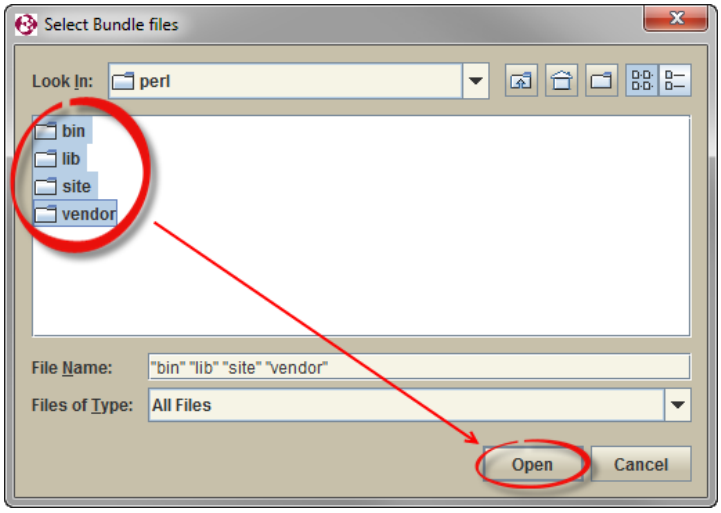

Figure 33. Select required directories and click the Open button.

8. Wait until all files have been added. After all files have been added, click the **Guess** button.

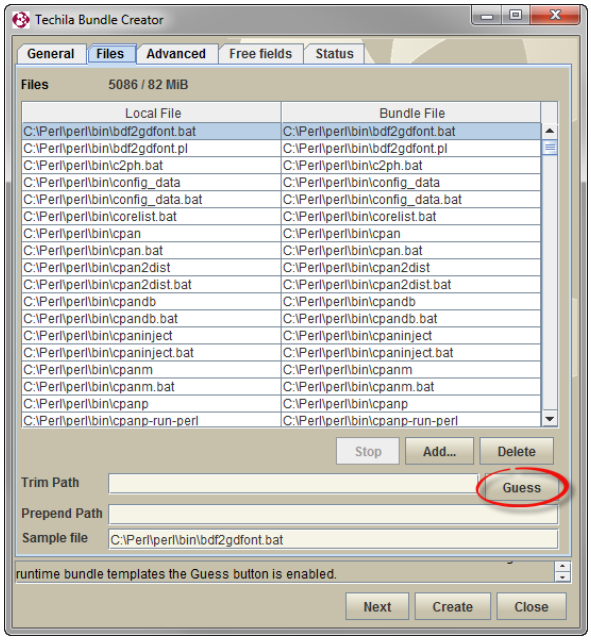

Figure 34. Click the Guess button.

Clicking the button will attempt to correctly set values for the following fields: **Prepend Path**, **Trim Path** and **Sample File**. The Sample File field should contain the following value after clicking the Guess button:

- perl\bin\perl.exe

After this step, the 'Files' tab should resemble the one shown below in Figure 35.

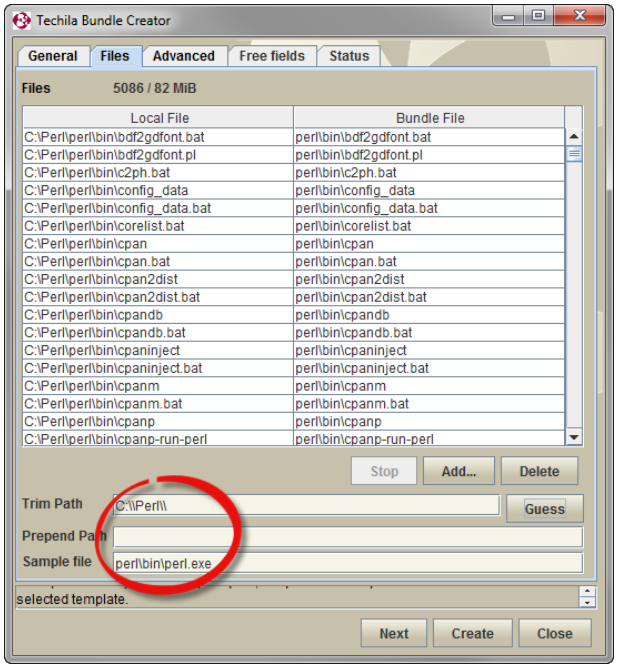

Figure 35. The 'Files' tab after adding files required in the Perl Runtime Bundle. The path has been trimmed by using the 'Guess' button.

9. Click the **Create** button to create the Bundle.

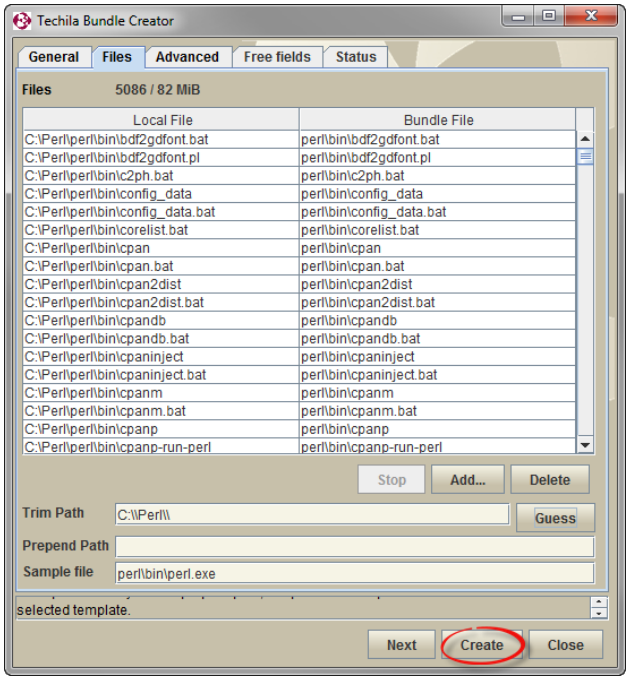

Figure 36. Clicking the Create button will start the Bundle creation process.

10. After clicking the button, the 'Status' tab will be automatically displayed which will contain information on the Bundle creation process. After successfully creating the Runtime Bundle, the Status tab should resemble the one shown below.

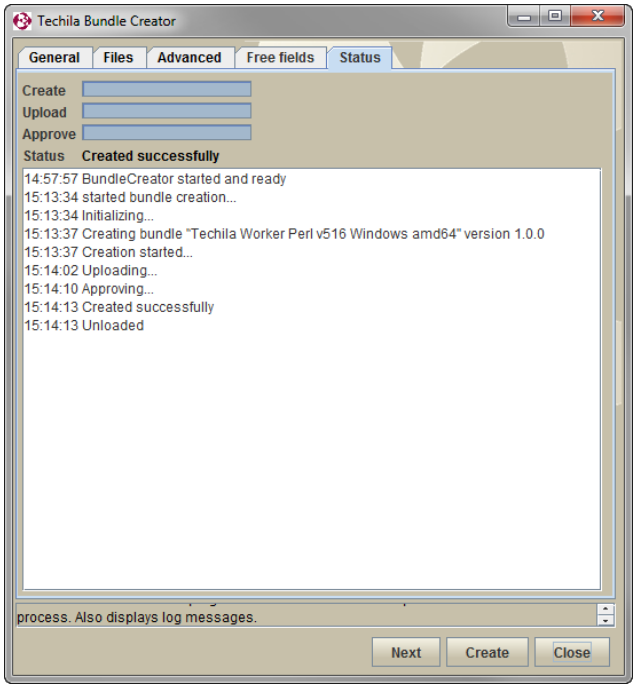

Figure 37. The 'Status' tab after successfully creating a Perl Runtime Bundle.

11. Close the Techila Bundle Creator tool window by clicking the **Close** button. The Perl Runtime Bundle is now ready for use.

5.2.2. Creating a Perl Runtime Bundle with the CLI

This Chapter contains instructions on how to create a Perl Runtime Bundle using the CLI **`'createBundle'`** command.

Notes

- The Perl version used in this CLI example is a Perl 5.16 64-bit Windows.
- If you are creating Perl Runtime for Linux Workers, ensure that you have created a temporary staging directory and that the necessary files have been placed in the directory.

Procedure

1. Open a command prompt and change the current working directory to the Perl installation folder.

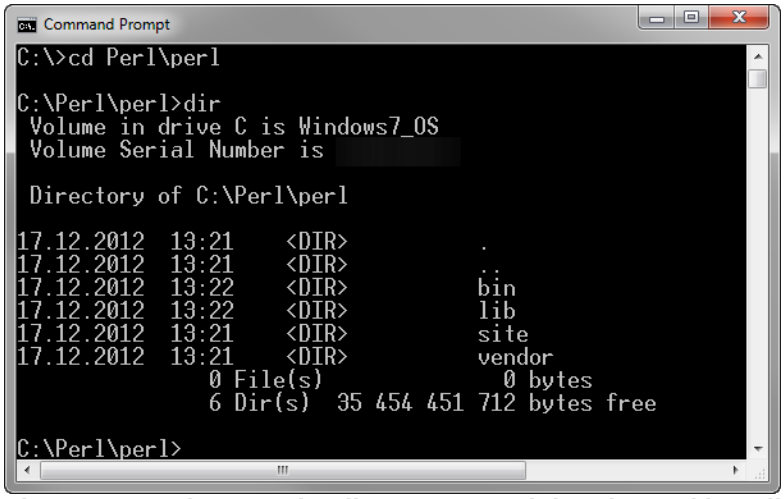

Figure 38. Navigate to the directory containing the Perl installation

2. If you wish to make the Bundle available for 64-bit Workers, create and upload the Perl Runtime Bundle with the command:

```
%techila% createbundle bundlename="Techila Worker Perl v516" resource=perl
exports=fi.techila.grid.cust.common.library.perl.v516.client
natives="perl/bin/perl.exe;osname=WindowsXP;processor=amd64,perl/bin/perl.exe;o
sname=Windows2000;processor=amd64,perl/bin/perl.exe;osname=Windows
Vista;processor=amd64, perl/bin/perl.exe;osname=Windows
7;processor=amd64,perl/bin/perl.exe;osname=Windows
2003;processor=amd64,perl/bin/perl.exe;osname=Windows Server
2008;processor=amd64" expiration=365d yes=true description="Techila Worker Perl
v516" prependpath=perl *
```

Note! Depending on system specific parameters, you might need to adjust the values of the following parameters:

- bundlename (update the v516 notation to match the Perl version you are using)

- natives (replace with a parameter that contains the operating systems and processor architectures you wish to make the bundle available for)
- exports (update the v516 to match the Perl version you are using)
- description(update the v516 to match the Perl version you are using)

After executing the command, the Bundle will be automatically created, uploaded and approved.

Please note that each step might take several minutes. After all steps have been completed, the view should resemble the one shown below.

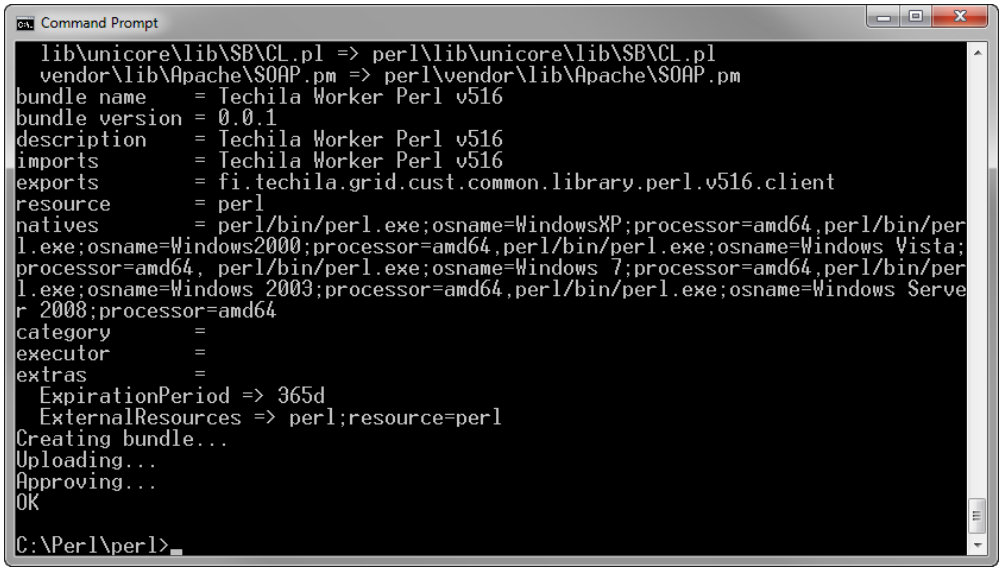

Figure 39. After the Bundle has been created, uploaded and approved an OK message will be displayed.

The Perl Runtime Bundle is now ready for use.

5.3. Python Runtime Bundle

Python Runtime Bundles are created by storing selected files and directories from the Python installation directory in to the Runtime Bundle.

Prerequisites

- Python installed on the same platform the Runtime Bundle is created for
- Access to a signed End-User Key

The table below contains a description on the parameters that can be used as a reference when creating a Python Runtime Bundle.

PARAMETER	VALUE	NOTES
Bundle Name	Techila Worker Python r<version> <OS> <arch>	<version> is the Python version <OS> is the operating system of the Techila Worker <arch> is processor architecture of the Techila Worker.
Exports	fi.techila.grid.cust.common.library.python.r<version>.client	Sets the Bundle exports so that the Bundle can be imported into Projects created with Python peach. <version> should be replaced with the Python version e.g. 273. Note that the exports parameter needs to be defined exactly as shown here, because the Bundle is imported based on the exports parameter.
Description	Techila Worker Python r<version>	Informs that the Bundle is a Python Runtime Bundle. <version> should be replaced with the Python version.
Resource	python	Defines that the Bundle is a Python Runtime Bundle. The resource parameter must be defined exactly as shown here.
Expiration	365d	Defines that the Bundle can remain unused for 365 days before it will be removed from Workers.
Files/ directories	Directories in Python installation directory: - DLLs - Lib - libs Files to be included from the Python installation directory: - python.exe (Windows) - python (Linux) In addition, all files required by Python located under the Windows directory. For example with Python 2.7.3, include file : -python27.dll	The listed files and/or directories will be required in the Bundle.
Trim Path	C:\\ Python27\\	The path leading to the Python installation directory should be trimmed. Note that this path might be different on your system.
Prepend path		No prepend path is required.
Sample file	python.exe (Windows) python (Linux)	The sample file is located in the Python installation directory.

Natives	python.exe;osname=WindowsXP;processor=amd64, python.exe;osname=Windows 2000;processor=amd64, python.exe;osname=Windows Vista;processor=amd64, python.exe;osname=Windows 7;processor=amd64, python.exe;osname=Windows Server 2008;processor=amd64	The 'natives' parameter defines the operating systems and processor architectures the Bundle will be offered to. Note! When using the Techila Bundle Creator tool, selections made in the Plaforms section will be automatically update the Natives parameter. When using the CLI, the natives parameter needs to be defined manually. The example on the left illustrates the value of the natives parameter when creating a Python Runtime Bundle for 64-bit Windows Workers.

5.3.1. Creating a Python Runtime Bundle with the Bundle Creator tool for Windows

This Chapter contains instructions on how to create a Python Runtime Bundle using the Bundle Creator tool for Microsoft Windows operating system.

Notes

- The Python version used in the Techila Bundle Creator tool example is 2.7.3

Procedure

1. Create a temporary staging directory. This directory will be used to temporarily store the files and directories that will be stored in the Bundle. In this example, the path of the temporary staging directory is 'C:\Staging'.

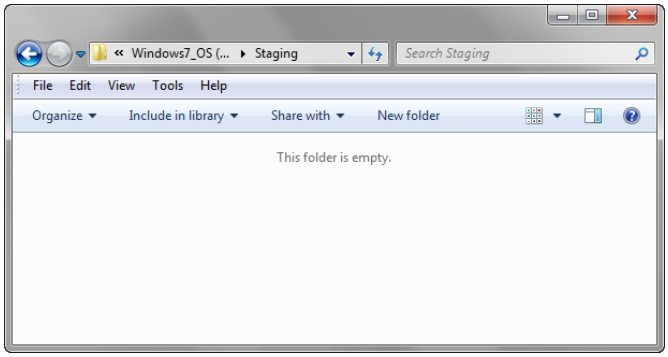

Figure 40. An empty staging directory.

2. Copy the python<version>.dll file from "Windows\system32" directory to the temporary staging directory. In this example, the Bundle is created from Python 2.7.3, meaning the file 'python27.dll' has been be copied.

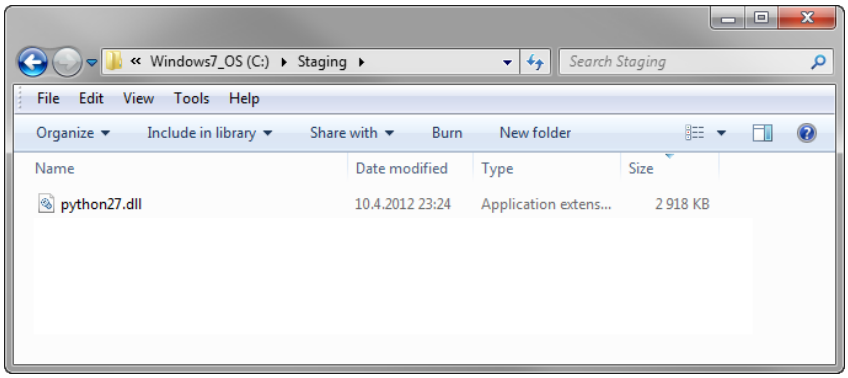

Figure 41. Add additional files required when executing Python on Workers to the staging directory.

3. Copy the contents of your Python installation directory to the temporary staging directory. If you want, you can exclude files/directories that are not required during computation.

   Note! Files/directories that must be included are:

   Directories:

   - DLLs
   - Lib
   - libs

   Files:

   - python.exe
   - python27.dll

   In the example below, only required Python installation files have been copied to the temporary staging directory.

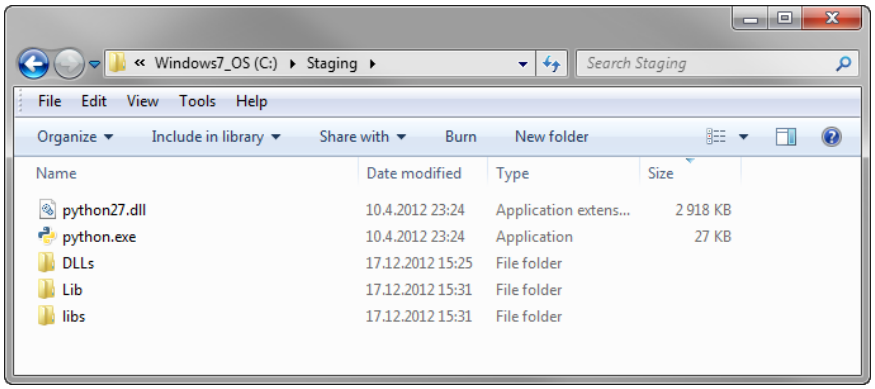

Figure 42. A staging directory after files have been copied from the Python installation directory.

4. Launch the Techila Bundle Creator tool by navigating to the **'lib'** directory in the Techila SDK and double clicking on the **'bundlecreator.jar'** icon.

5. Check the location of 'techila_settings.ini' file and correct if needed

6. Choose the **Python Runtime** template from the **Template** drop-down menu

7. If required, modify the value in the "Bundle Name" field to match your Python version. In this example, the version corresponds to Python 2.7.3

8. Choose applicable operating systems and processor architectures in the **Platforms** section. This step determines which Workers will be able to use the Bundle. With the settings in this example, 64-bit Windows Workers will be allowed to use the Bundle.

9. After the steps described above, the 'General' tab should resemble the one shown below. Click the "**Files**" tab to continue with adding files.

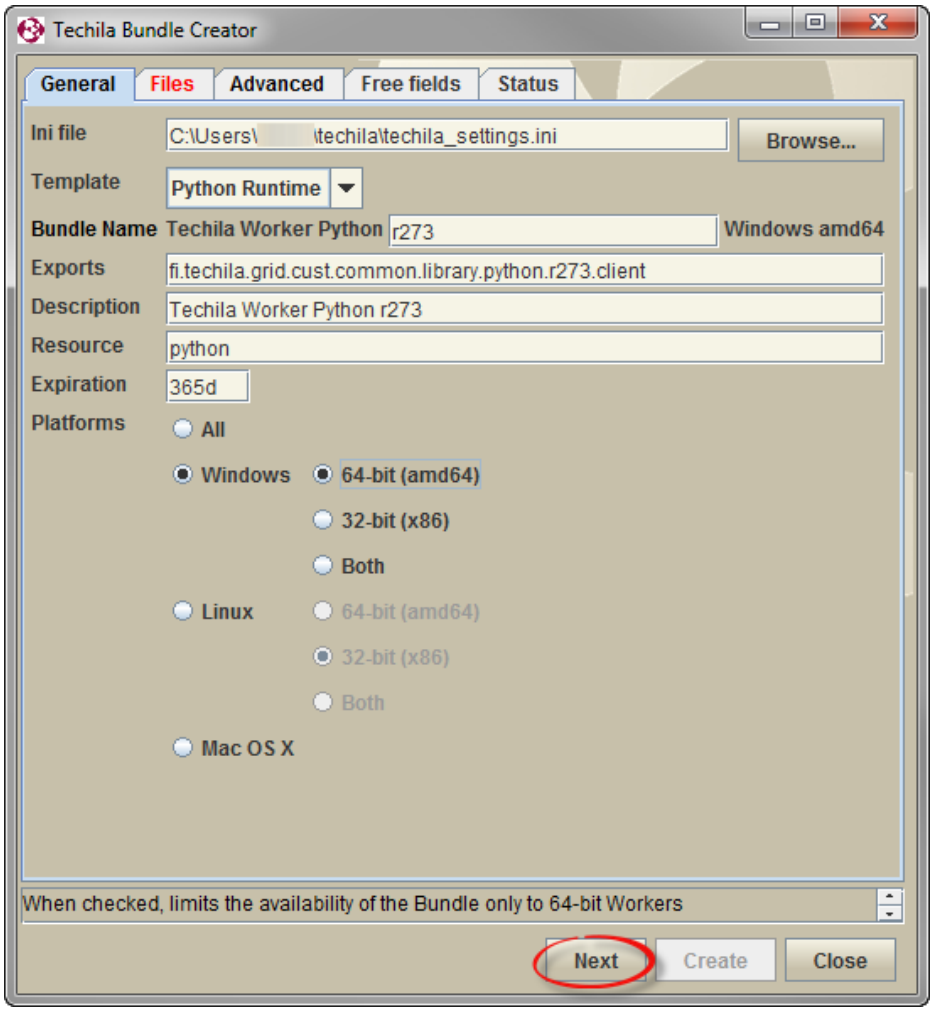

Figure 43. After configuring the "General" tab, continue with adding files.

10. On the 'Files' tab, click the **Add...** button to open the file dialog window.

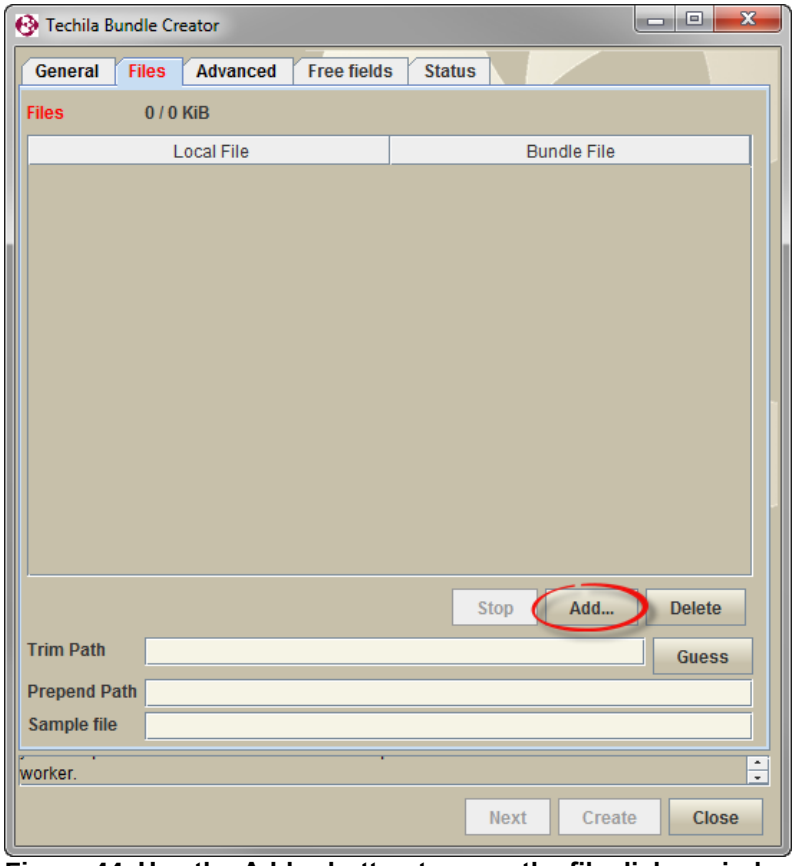

Figure 44. Use the Add... button to open the file dialog window.

11. Using the file dialog window, select the temporary staging directory you created earlier. Click the **Open** button to add the files.

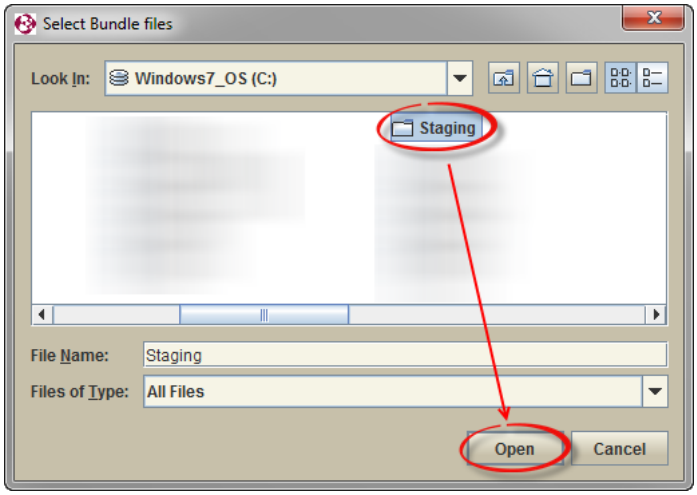

Figure 45. Select and add the temporary staging directory you created earlier.

12. After adding the files, click the **Guess** button to modify the path of the files.

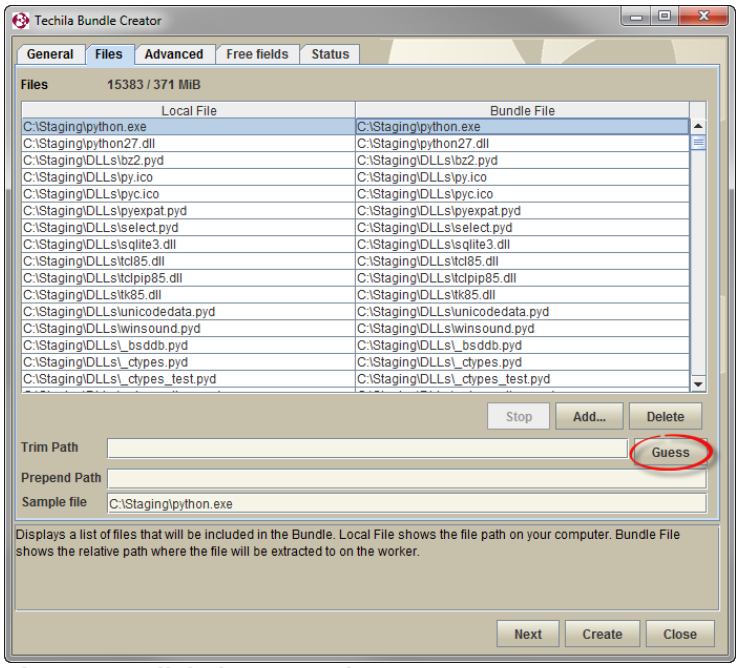

Figure 46. Click the Guess button.

After clicking the Guess button, the **Sample File** field should contain the following value:

- python.exe (if you are creating the Bundle for Windows Workers)
- python (if you are creating the Bundle for Linux Workers)

If the Sample File field does not contain the values described above, modify the path manually using the Trim Path field.

After this step, the "Files" tab should resemble the one shown below.

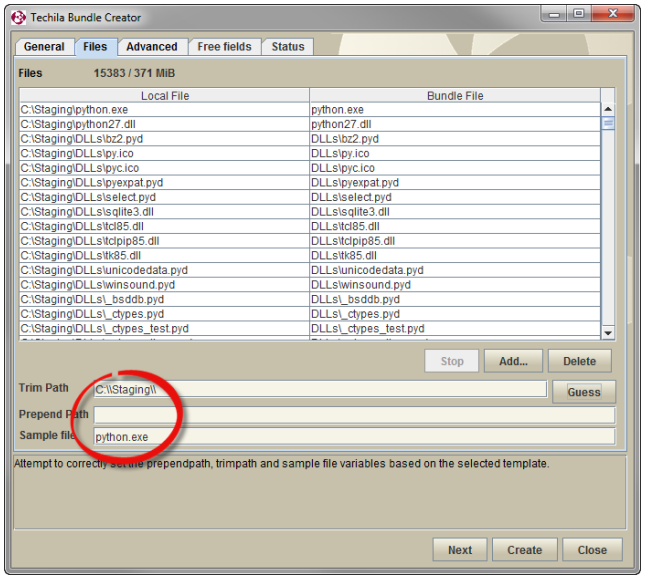

Figure 47. The 'Files' tab after adding files and clicking the Guess button.

13. Click the **Create** button to start the Bundle creation process.

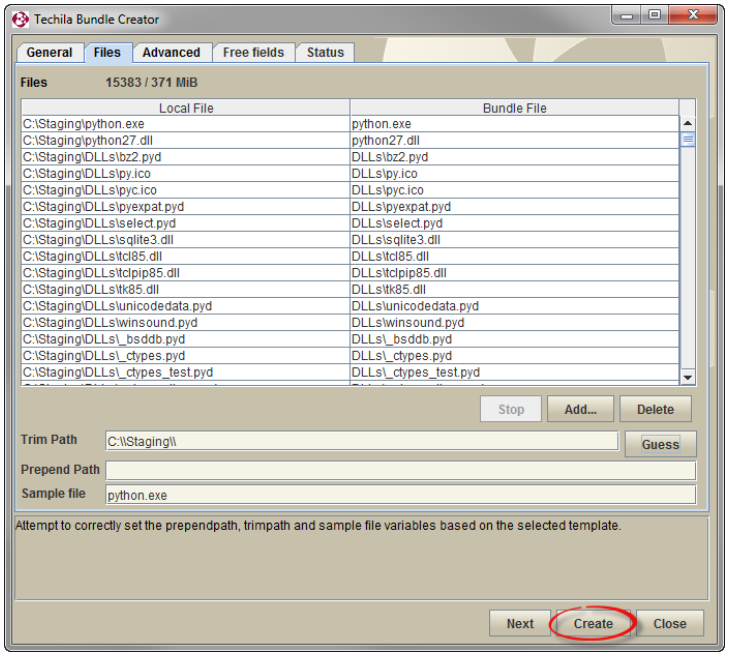

Figure 48. Start the Bundle creation process.

14. The 'Status' tab will now be automatically displayed and will contain information on the Bundle creation process. After successfully creating the Runtime Bundle, the Status tab should resemble the one shown below.

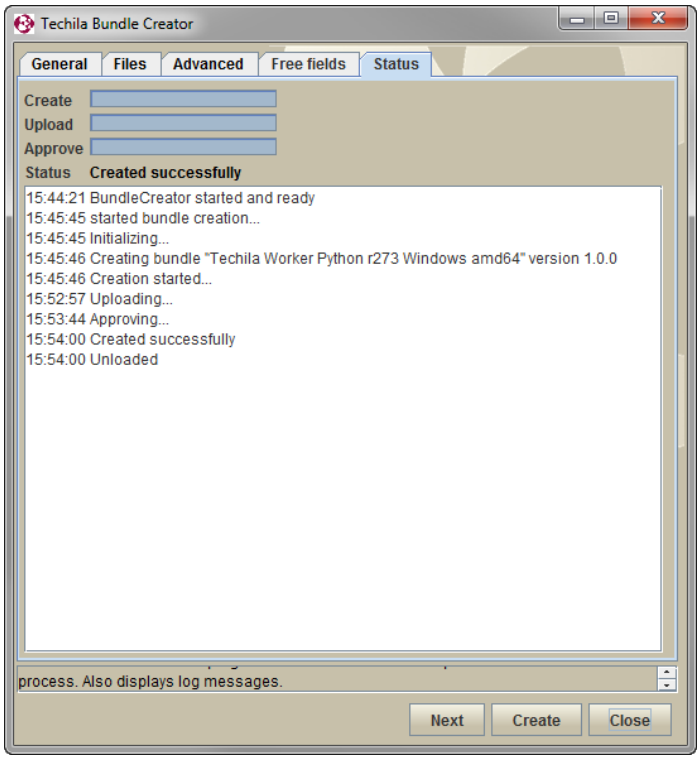

Figure 49. The 'Status' tab after successfully creating a Python Runtime Bundle.

15. Close the Techila Bundle Creator tool window by clicking the **Close** button.

16. Delete the staging directory you created earlier.

5.3.2. Creating a Python Runtime Bundle with the CLI for Windows

This Chapter contains instructions on how to create a Python Runtime Bundle using the `'createBundle'` command included in the Techila Command Line Interface.

Notes

- The Python version used in the CLI example is 2.7.3.

Procedure

1. Create a temporary staging directory for the Bundle files as instructed in Chapter 5.3.1 steps 1-3.
2. Open a command prompt and change your current working directory to the temporary staging directory. In this example, the temporary staging directory used is "C:\Staging".

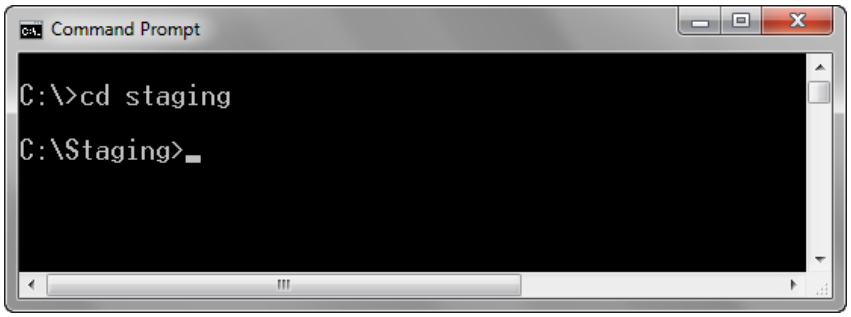

Figure 50. Navigate to the temporary staging directory.

3. Create the Bundle with the following command:

```
%techila% createbundle bundlename="Techila Worker Python r273 Windows 64"
resource=python exports=fi.techila.grid.cust.common.library.python.r273.client
natives="python.exe;osname=WindowsXP;processor=amd64,python.exe;osname=Windows20
00;processor=amd64,python.exe;osname=Windows Vista;processor=amd64,amd64
python.exe;osname=Windows 7;processor=amd64,python.exe;osname=Windows
2003;processor=amd64,python.exe;osname=Windows Server 2008;processor=amd64"
expiration=365d yes=true description="Techila Worker Python 2.73 Windows" *
```

Note that depending on system specific parameters, you might need to adjust the values of the following parameters:

- bundlename (Update to match your Python version and platforms you are using)
- exports (Update the notation 'r273' to match your Python version. If you are using Python 2.7.3, no change is required.)
- natives (Update with a parameter that contains the operating systems and processor architectures you wish to make the bundle available for.)

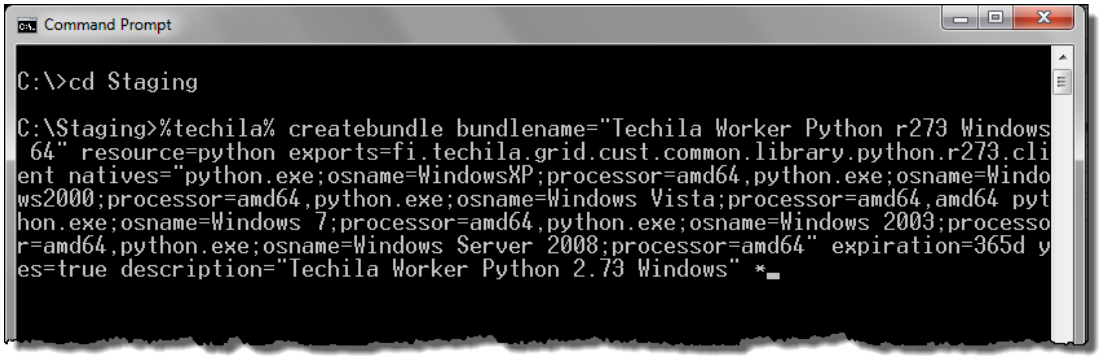

Figure 51. Entering the 'createbundle' command to the command line. Executing the command will also list each file that will be included in the Bundle.

4. After executing the command, the Bundle will be automatically created, uploaded and approved.

   Please note that each step might take several minutes. After all steps have been completed, the view should resemble the one shown below.

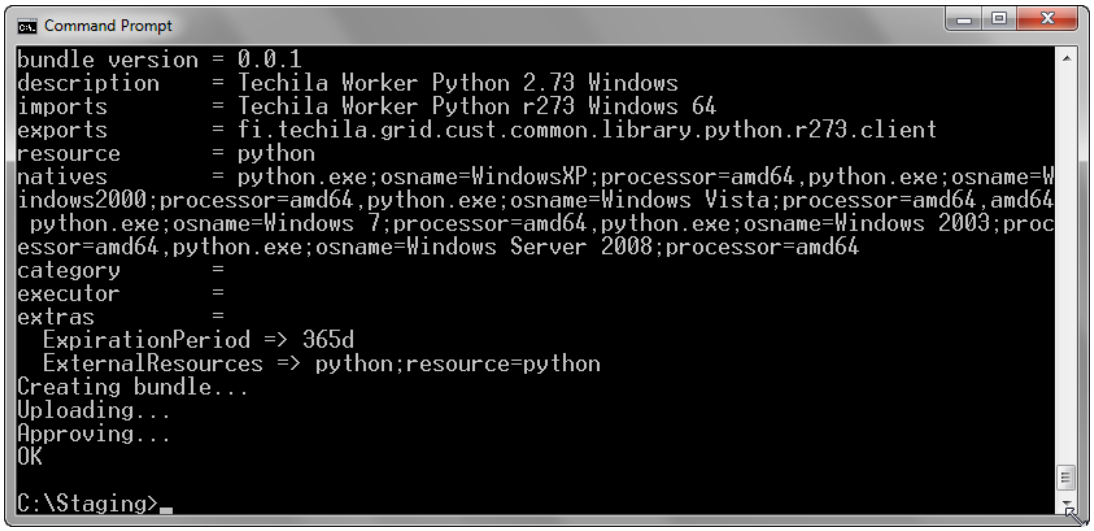

Figure 52. The view after executing the command. After the Bundle has been transferred and approved by the Techila Server a message stating "OK" will be displayed.

The Python Runtime Bundle is now ready for use.

5.3.3. Creating a Python Runtime Bundle with the Bundle Creator tool for Linux

This Chapter contains instructions on how to create a Python Runtime Bundle using the Bundle Creator tool for Linux operating system.

Notes

- The Python version used in the Techila Bundle Creator tool example is 2.6.0

Procedure

1. Create a temporary staging directory. This directory will be used to temporarily store the files and directories that will be stored in the Bundle. In this example, the path of the temporary staging directory is '/tmp/staging'.

```
mkdir /tmp/staging
```

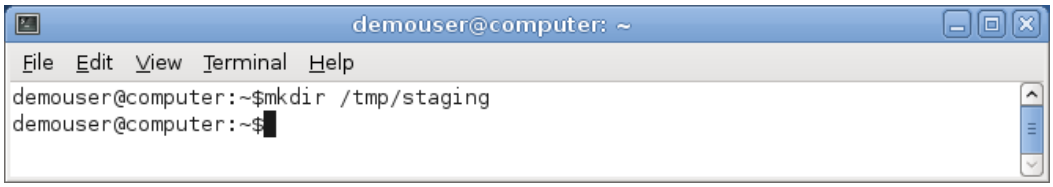

Figure 53. Creating the temporary staging directory.

2. Create a director named 'Lib' in the temporary staging directory.

```
mkdir /tmp/staging/Lib
```

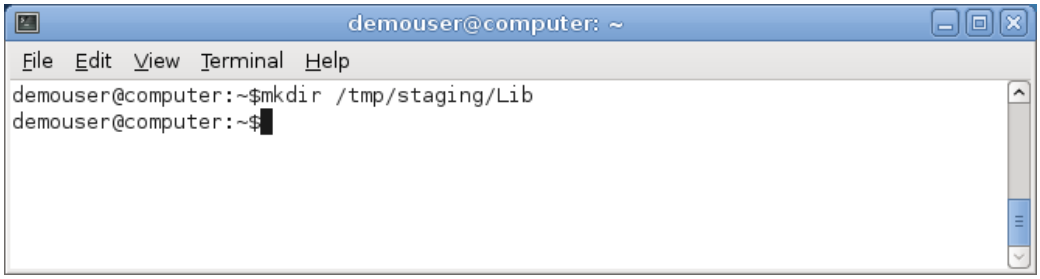

Figure 54. Creating the Lib directory.

3. Copy the contents of the Python installation directory to the 'Lib' directory. In the example below, installation files for Python 2.6 are copied.

```
cp -r /usr/lib/python2.6/* /tmp/staging/Lib/
```

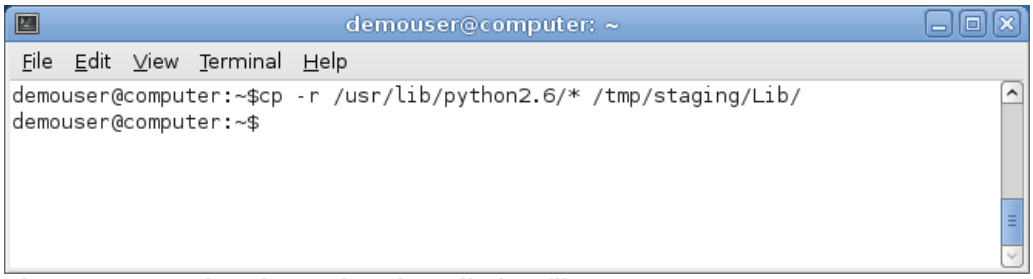

Figure 55. Copying the Python installation files.

4. Create a director named 'Modules' in the temporary staging directory.

```
mkdir /tmp/staging/Modules
```

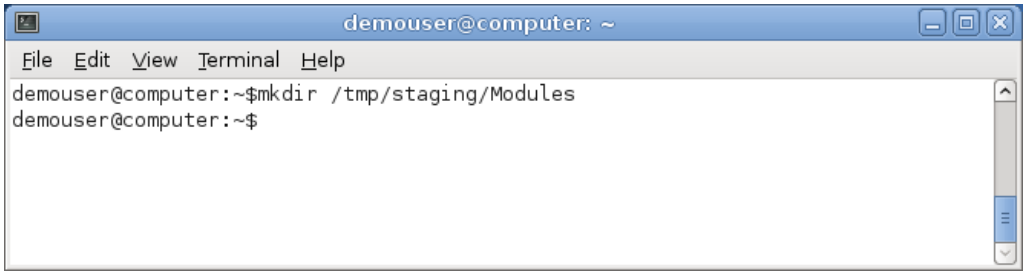

Figure 56. Creating the Modules directory.

5. Copy all files from 'Lib/lib-dynload' to the 'Modules' directory.

```
cp -r /tmp/staging/Lib/lib-dynload/* /tmp/staging/Modules/
```

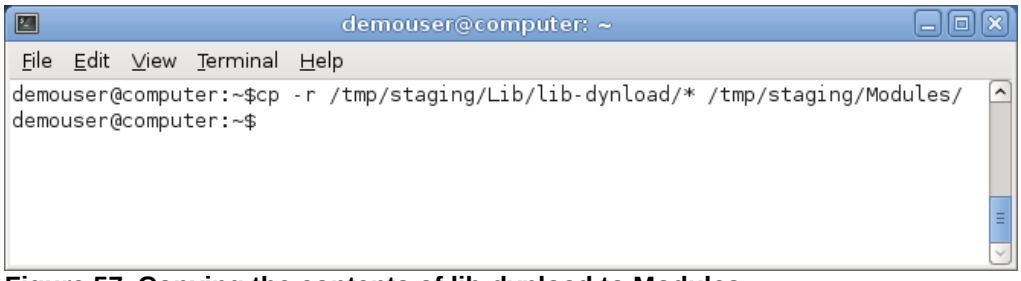

Figure 57. Copying the contents of lib-dynload to Modules

6. Locate and copy the Python shared library to the staging directory. The Python shared library is named based on the python version that is used. For example, when using Python 2.6 the name of the shared library is 'libpython2.6.so'.

   Typically this file is located in '/usr/lib'. The file can also be located by using 'find'. The example below shows how to find the file for Python 2.6.

```
find / -name libpython2.6.so 2>/dev/null
```

After locating the file, copy the file to the staging directory.

```
cp /usr/lib/python2.6/config/libpython2.6.so /tmp/staging
```

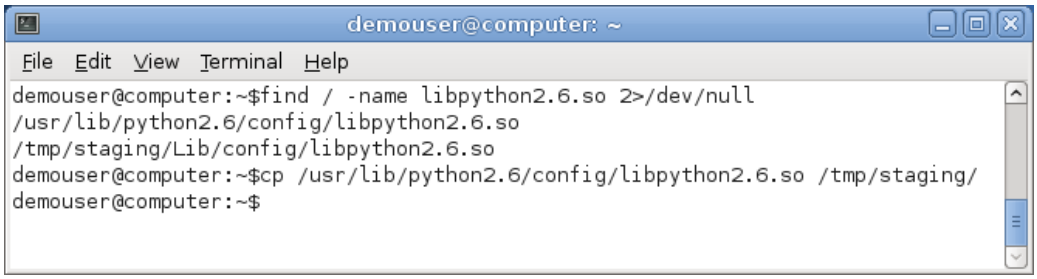

Figure 58. Locating and copying the shared library to the staging directory

7. Copy the Python executable to the staging directory. Typically the python executable is located in '/usr/bin'.

```
cp /usr/bin/python /tmp/staging
```

Figure 59. Copying the python executable to the staging directory.

8. The staging directory should now have the following contents:

   - Lib
   - Modules
   - libpython2.6.so
   - python

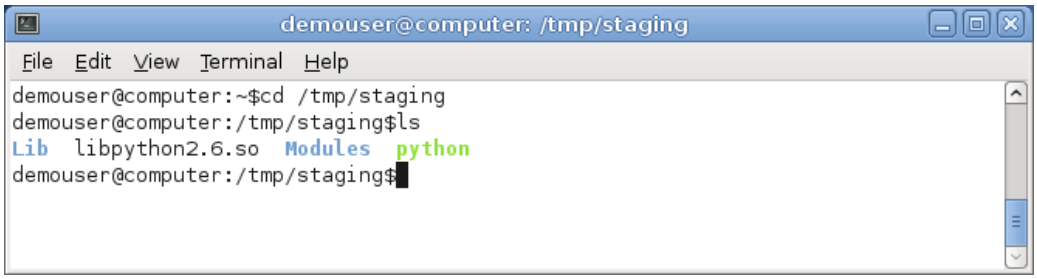

Figure 60. Checking that all required files are in the staging directory.

All required files and directories are now in the staging directory. Next steps describe how to use these files to create a Python Runtime Bundle for workers with a Linux operating system.

9. Launch the Techila Bundle Creator tool by navigating to the **'lib'** directory in the Techila SDK and double clicking on the **'bundlecreator.jar'** icon. The Bundle Creator can be launched from the command line with command:

   ```
   java -jar <full path>/lib/bundlecreator.jar
   ```

   Replace the **`<full path>`** notation with the path leading to the Techila SDK

10. Check the location of 'techila_settings.ini' file and correct if needed

11. Choose the **Python Runtime** template from the **Template** drop-down menu

12. If required, modify the value in the "Bundle Name" field to match your Python version. In this example, the version corresponds to Python 2.6.0

13. Choose applicable operating systems and processor architectures in the **Platforms** section. This step determines which Workers will be able to use the Bundle. With the settings in this example, 64-bit Linux Workers will be able to use the Bundle.

14. After the steps described above, the 'General' tab should resemble the one shown below. Click the "**Files**" tab to continue with adding files.

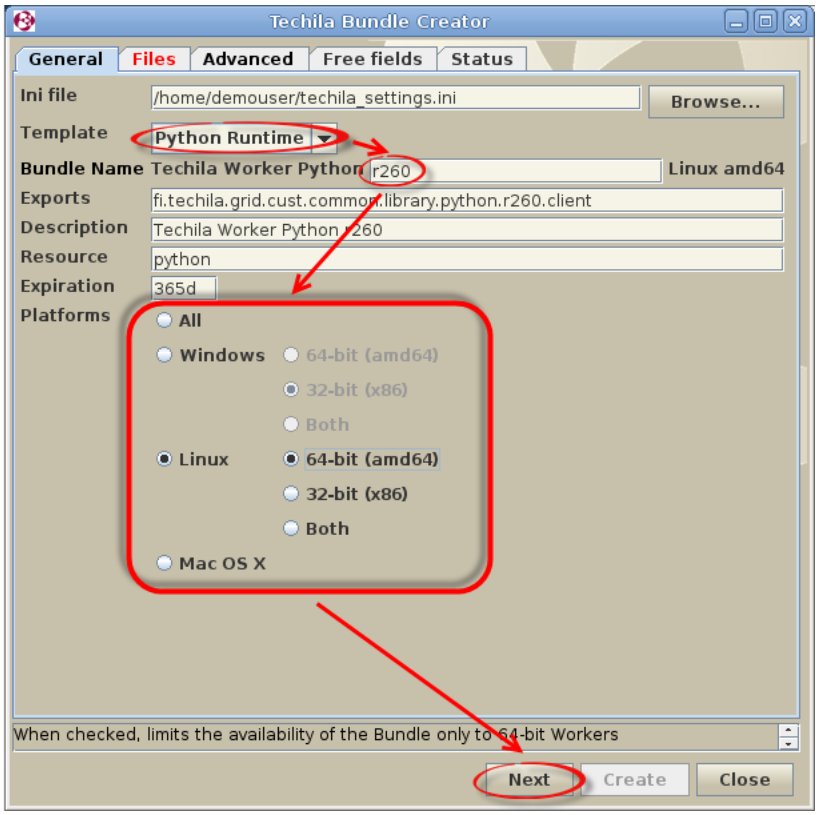

Figure 61. Defining Bundle version and applicable platforms.

17. On the 'Files' tab, click the **Add...** button to open the file dialog window.

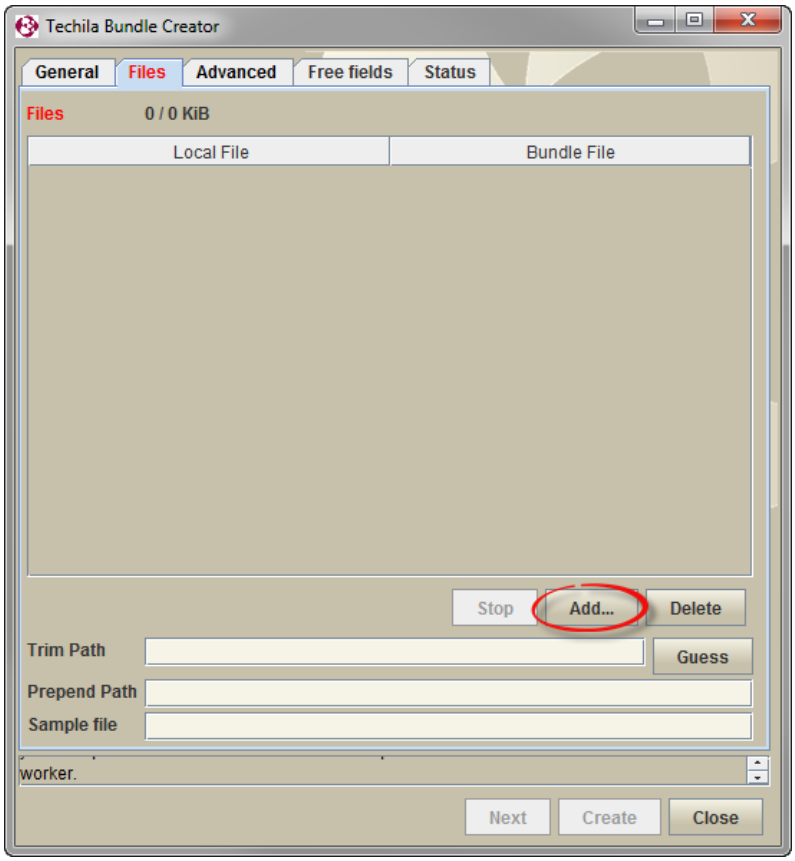

Figure 62. Use the Add… button to open the file dialog window.

18. Using the file dialog window, select the temporary staging directory you created earlier. Click the **Open** button to add the files.

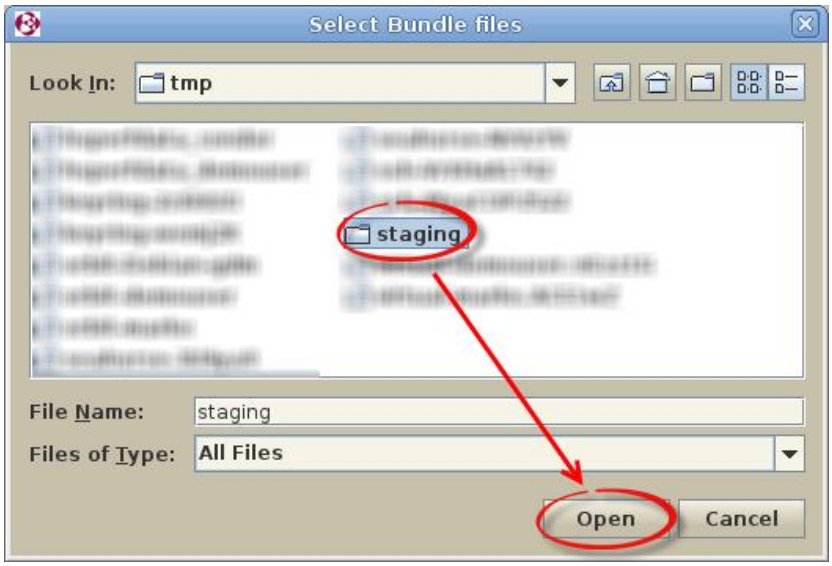

Figure 63. Select and add the temporary staging directory you created earlier.

19. After adding the files, click the **Guess** button to modify the path of the files.

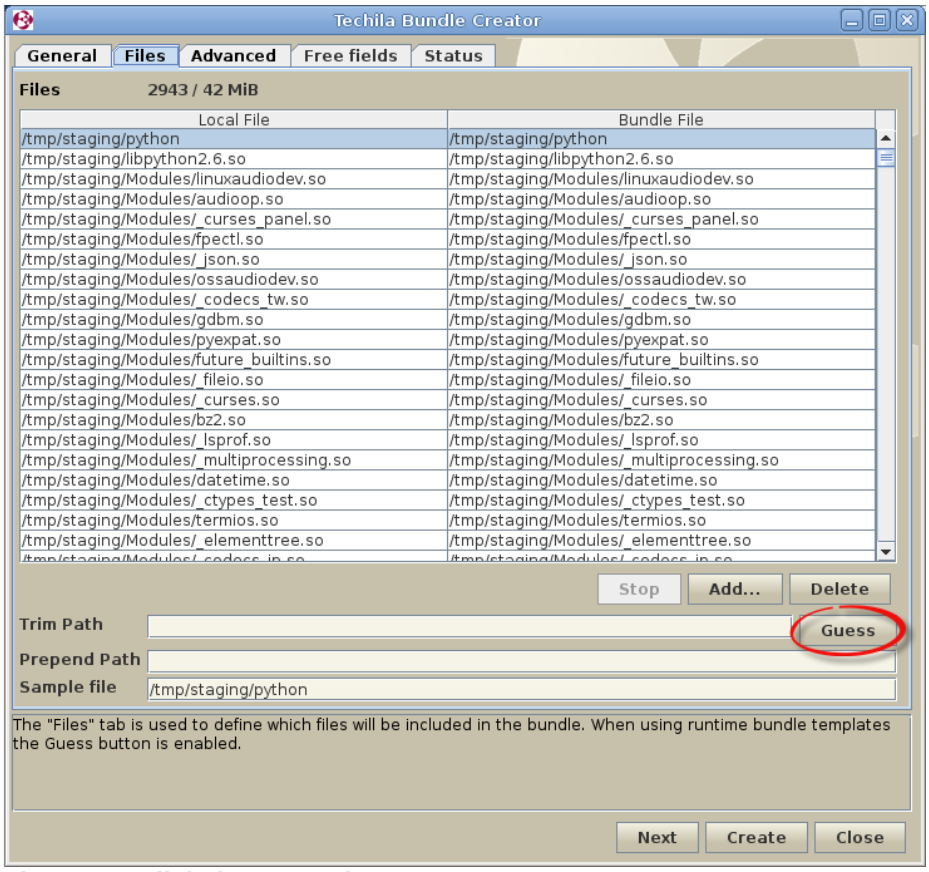

Figure 64. Click the Guess button.

After clicking the Guess button, the **Sample File** field should contain the following value:

- python.exe (if you are creating the Bundle for Windows Workers)
- python (if you are creating the Bundle for Linux Workers)

If the Sample File field does not contain the values described above, modify the path manually using the Trim Path field.

After this step, the "Files" tab should resemble the one shown below.

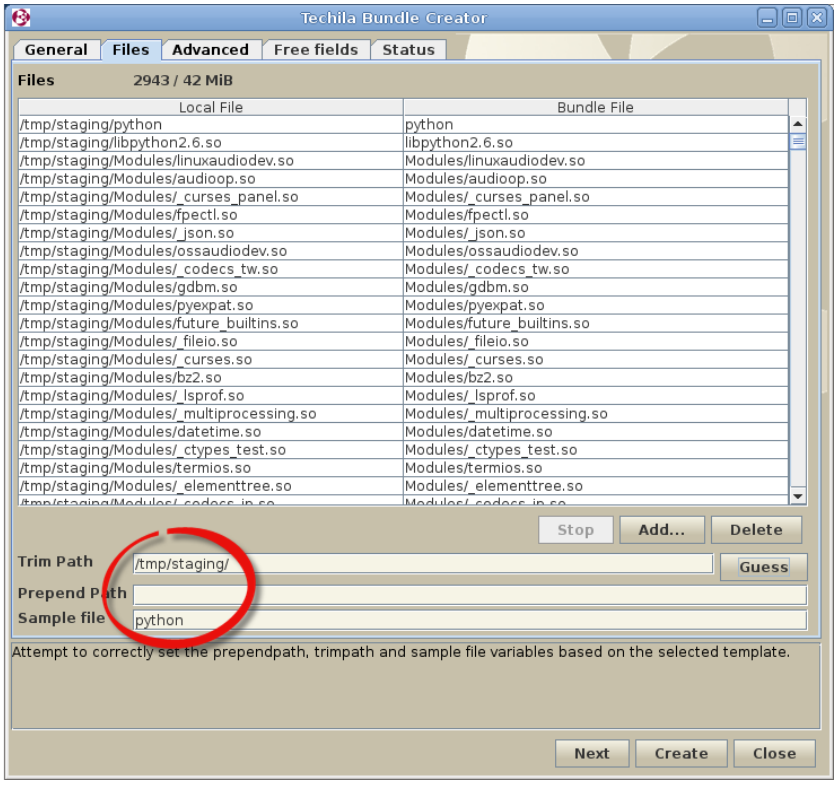

Figure 65. The 'Files' tab after adding files and clicking the Guess button.

20. Click the **Create** button to start the Bundle creation process.

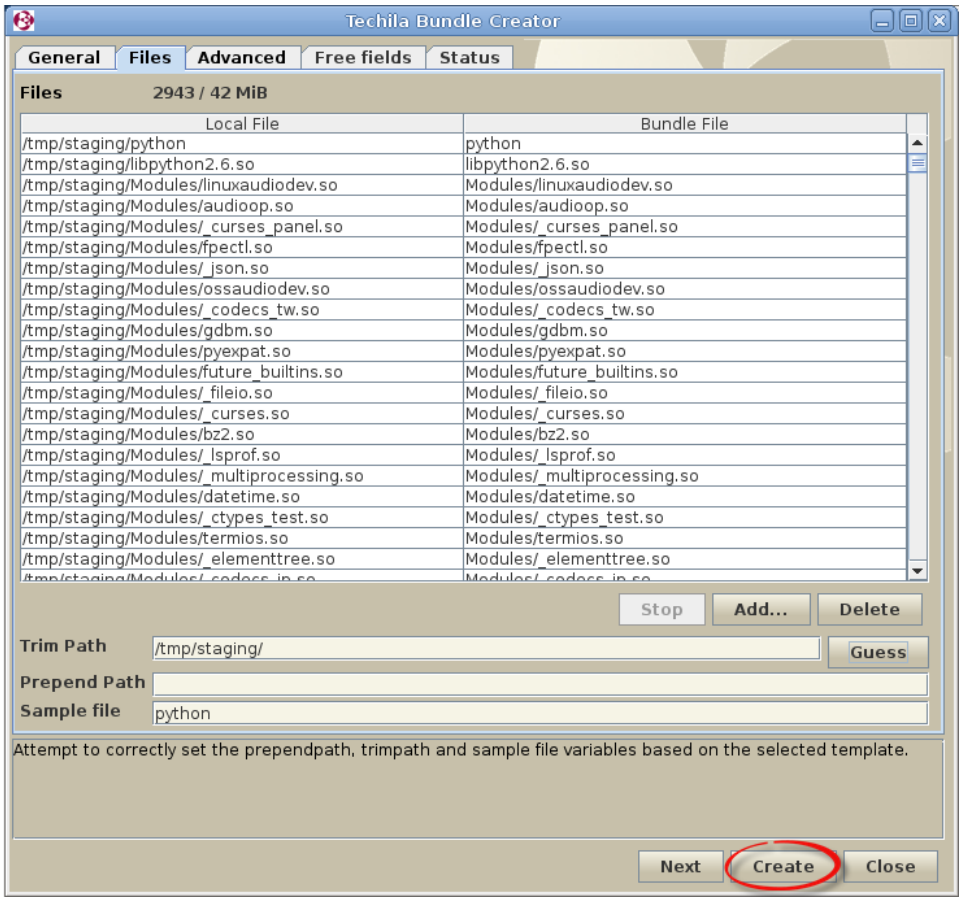

Figure 66. Start the Bundle creation process.

21. The 'Status' tab will now be automatically displayed and will contain information on the Bundle creation process. After successfully creating the Runtime Bundle, the Status tab should resemble the one shown below.

    Close the Techila Bundle Creator tool window by clicking the **Close** button.

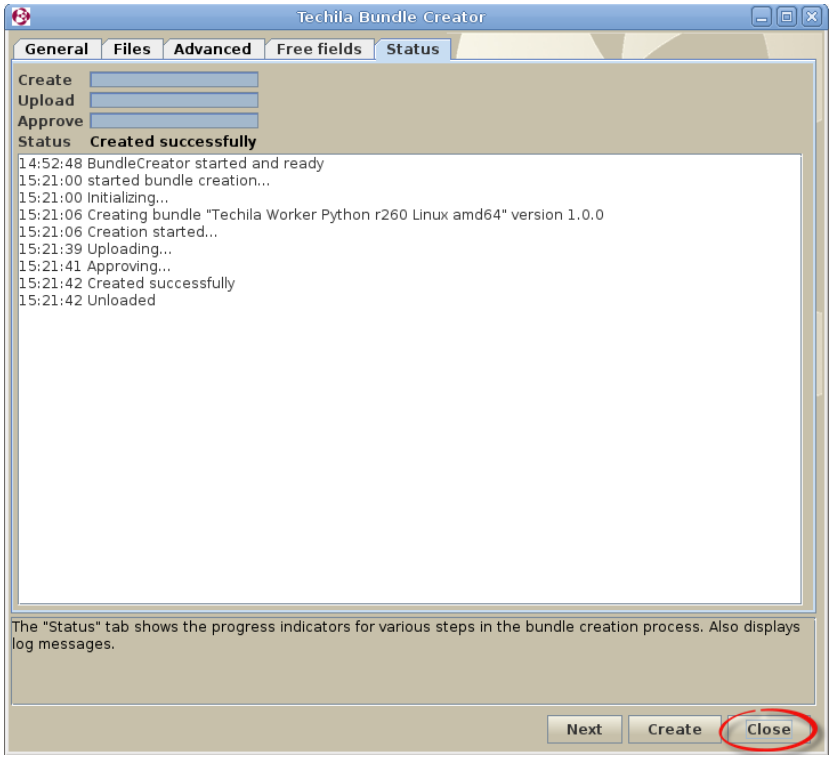

Figure 67. The 'Status' tab after successfully creating a Python Runtime Bundle.

22. Delete the staging directory you created earlier

5.4. R Runtime Bundle

R Runtime Bundles are created by storing the specified files and directories from the R installation directory in to the Runtime Bundle.

Prerequisites

- R installed on the same platform the Runtime Bundle is created for

The table below contains a description on the parameters that can be used as a reference when creating the R Runtime Bundle.

PARAMETER	VALUE	NOTES
Bundle Name	Techila Worker GNU R v<version> <OS> <arch>	<version> is the R version <OS> is operating system of the Techila Worker <arch> is processor architecture of the Techila Worker
Exports	fi.techila.grid.cust.common.library.gnur.v<version>.client	When using the CLI, the <version> notation needs to be replaced R version, used for creating the Bundle. For example, if the Bundle is created from R 2.12.1 files, the <version> notation will be 2121. Note that the exports parameter needs to be defined similarly as shown here (replacing the <version> with the correct value), because the Bundle is imported based on the exports parameter.
Description	Techila Worker GNU R v<version>	<version> should be replaced with the R version
Resource	r	Defines that the Bundle is a R Runtime Bundle. The resource parameter must be defined exactly as shown here.
Expiration	365d	Defines that the can remain unused for 365 days before it will be removed from Workers.
Files/ directories	Directories to be included from R installation directory: bin etc library modules share	The listed files and/or directories will be required.
Trim Path	C:\\Program Files\\R\\R-2.12.1\\	The path leading to the R installation directory should be trimmed. Note, the Guess button in the Bundle Creator tool will attempt to set this value automatically. Note that this path might be different on your system.
Prepend path	r	The path should be prepended with 'r'. Note that the Guess button in the Bundle Creator tool will attempt to set this value automatically.
Sample file	r\bin\R.exe (Windows) r\bin\R (Linux)	After clicking the Guess button in the Techila Bundle Creator tool or when running tests from the CLI, the path of the sample file should match the ones illustrated.

Natives	r\bin\R.exe;osname=WindowsXP;processor=amd64,r\bin\R.exe;osname=Windows 2000;processor=amd64,r\bin\R.exe;osname=Windows Vista;processor=amd64,r\bin\R.exe;osname=Windows 7;processor=amd64,r\bin\R.exe;osname=Windows Server 2008;processor=amd64	The natives parameter defines the operating systems and processor architectures the Bundle will be offered to. Note! When using the Techila Bundle Creator tool, selections made in the Plaforms section will be automatically update the Natives parameter. When using the CLI, the natives parameter needs to be defined manually. The example on the left illustrates the value of the natives parameter when creating a R Runtime Bundle for 64-bit Windows Workers.

5.4.1. Creating a R Runtime Bundle with the Bundle Creator tool

This Chapter contains instructions on how to create an R Runtime Bundle using the Bundle Creator tool.

Notes

- The R version used in this example is 3.1.1
- The R installation contains both 32-bit and 64-bit R components.

Procedure

1. Launch the Techila Bundle Creator tool by navigating to the **'lib'** directory in the Techila SDK and double clicking on the **'bundlecreator.jar'** icon.

2. Check the location of 'techila_settings.ini' file and correct if needed

3. Choose the **R Runtime** template from the **Template** drop-down menu

4. Modify the free text field in the Bundle Name field to match the R version you are creating the Runtime Bundle from (for example version **3.1.1** would be represented with **v311**. **Do not include dots in the notation.**). This value will be automatically updated to the Bundle Exports field.

5. Choose applicable operating systems and processor architectures in the **Platforms** section. This step determines which Workers will be able to use the Bundle. If your R installation contains both 32-bit and 64-bit R components, choose the 'Both' option.

6. Click the **Next** button to continue to the 'Files' tab.

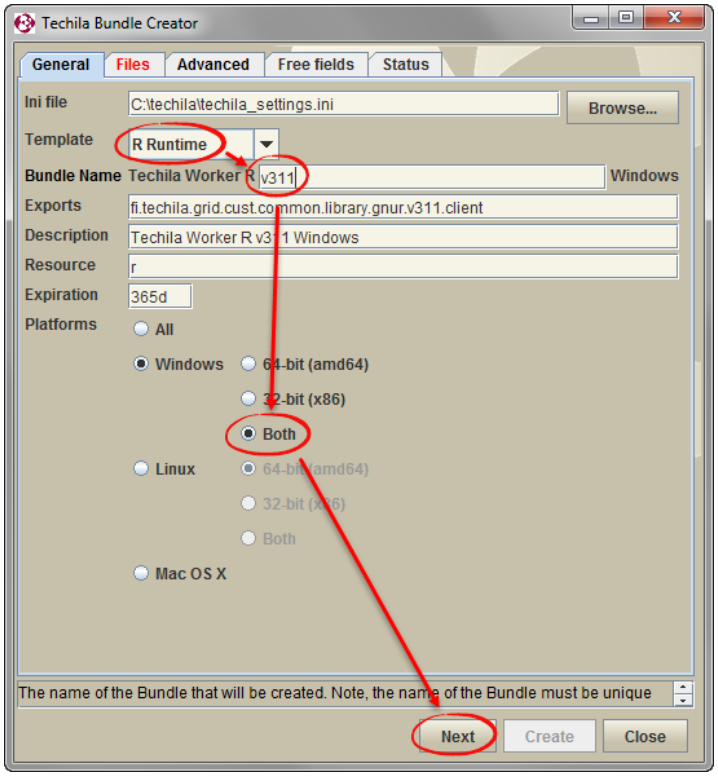

Figure 68. The 'General' tab after selecting the R Runtime template and configuring all required fields. The 'Both' option can be used because the R installation in this example contains both 32-bit and 64-bit R components. If your R installation only has 32-bit or 64-bit components, select the matching platform using the radio button. Note! Do NOT enter dots in the free text field in the "Bundle Name" when specifying the version.

7. On the 'Files' tab, click the **Add..** button to open the file dialog window.

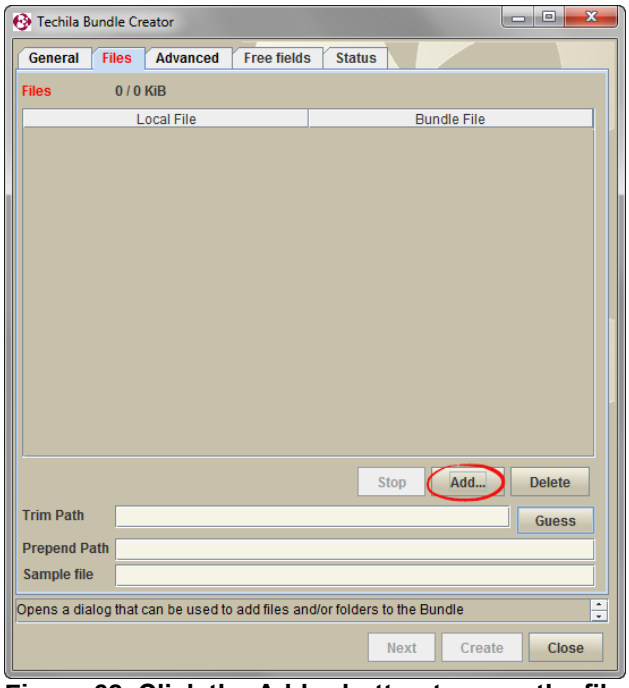

Figure 69. Click the Add… button to open the file dialog window.

8. Using the file dialog window, select the R installation directory and click the **Open** button.

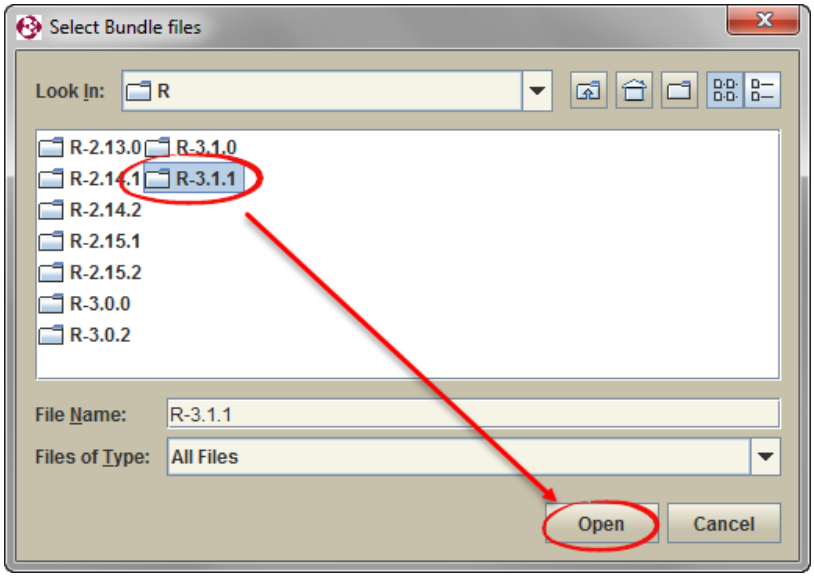

Figure 70. Select and add the required directories.

9. Wait until all files have been added. After the files have been added, click the **Guess** button.

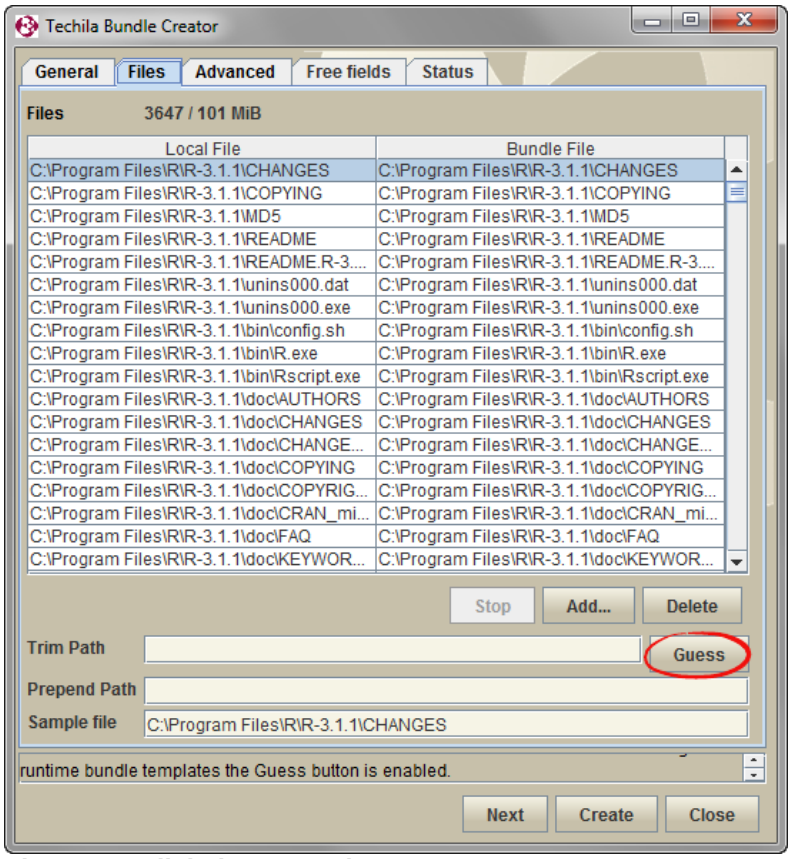

Figure 71. Click the Guess button.

Clicking the button will attempt to correctly set values for the following fields: **Prepend Path**, **Trim Path** and **Sample File**. The Sample File field should contain the following value after clicking the Guess button:

- r\bin\R.exe

After this step, the 'Files' tab should resemble the one shown below.

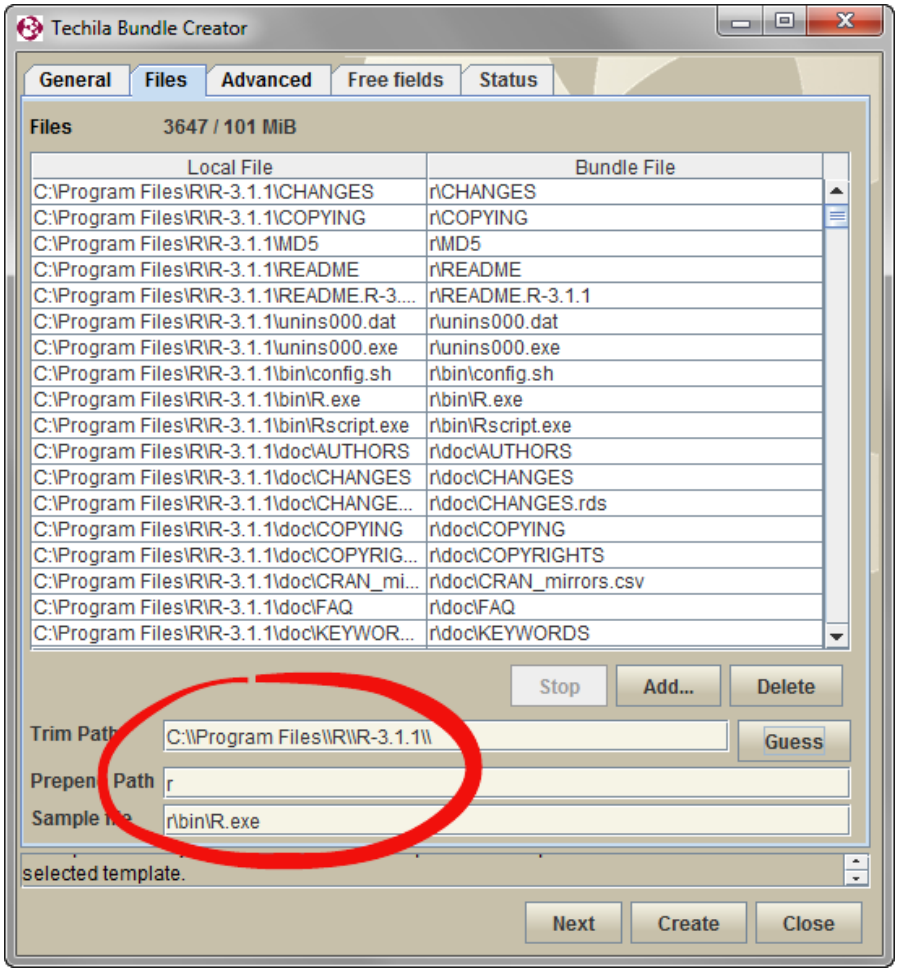

Figure 72. The 'Files' tab after files required in the R Runtime Bundle have been added. The path has been trimmed by using the Guess button.

10. Click the **Create** button to create the Bundle.

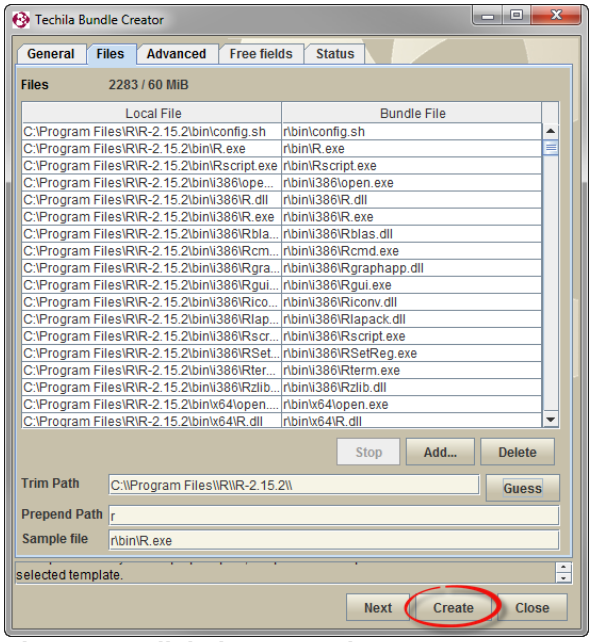

Figure 73. Click the Create button.

11. After clicking the button, the 'Status' tab will be displayed which will contain information on the Bundle creation process. After successfully creating the Runtime Bundle, the Status tab should resemble the one shown below.

    After the Bundle has been created, close the Techila Bundle Creator tool window by clicking the **Close** button.

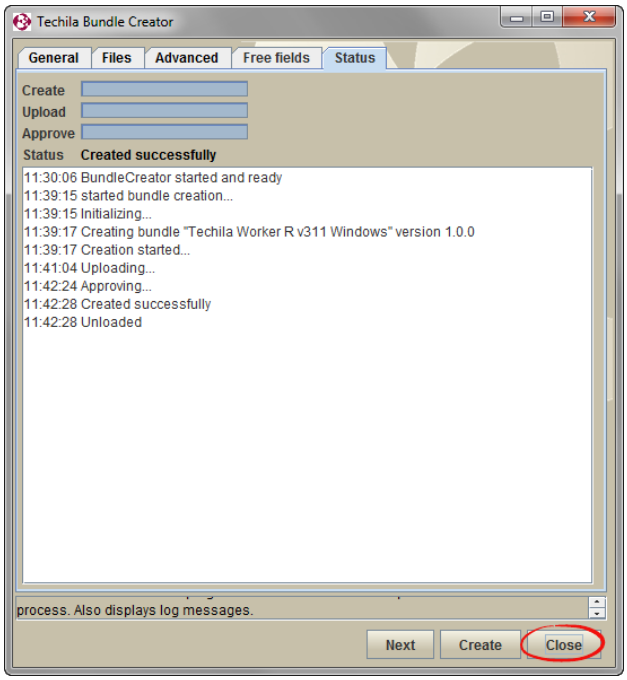

Figure 74. After creating the Bundle, close the Bundle Creator Tool by clicking the Close button.

5.4.2. Creating an R Runtime Bundle with the CLI

This Chapter contains instructions on how to create an R Runtime Bundle using the CLI `'createBundle'` command.

Notes

- The R version used in this example is 3.1.1
- The R installation contains both 32-bit and 64-bit R components.

Procedure

1. Open a command prompt and navigate to the R installation directory

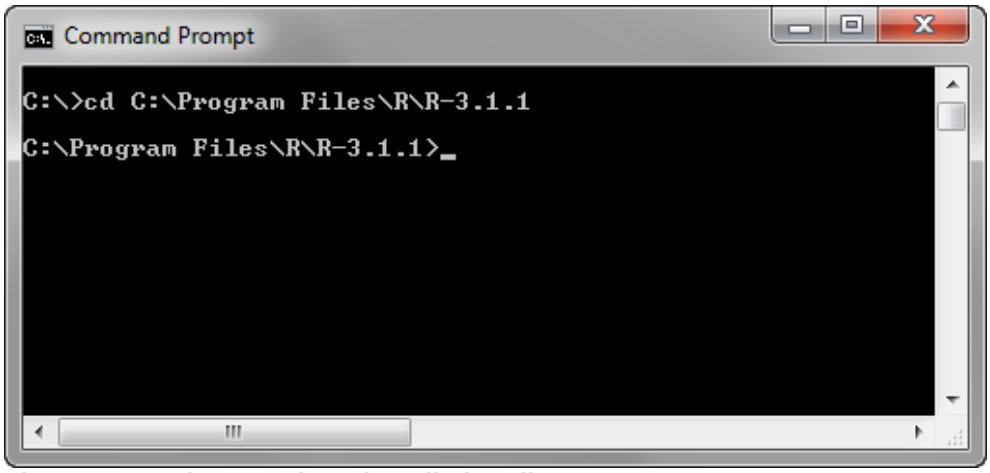

Figure 75. Navigate to the R installation directory.

2. Create the R Runtime Bundle with using the applicable command listed below:

   To create a Bundle with **32-bit and 64-bit** runtime components:

```
%techila% createbundle bundlename="Techila Worker R v311" resource=r
exports=fi.techila.grid.cust.common.library.gnur.v311.client
natives="r/bin/R.exe;osname=WindowsXP,r/bin/R.exe;osname=Windows2000,r/bin/R.
exe;osname=Windows Vista,x86 r/bin/R.exe;osname=Windows
7,r/bin/R.exe;osname=Windows 2003,r/bin/R.exe;osname=Windows Server 2008"
expiration=365d yes=true description="Techila Worker R v311 Windows"
prependpath=r *
```

   To create a Bundle with **only 32-bit** runtime components:

```
%techila% createbundle bundlename="Techila Worker R v311" resource=r
exports=fi.techila.grid.cust.common.library.gnur.v311.client
natives="r/bin/R.exe;osname=WindowsXP,r/bin/R.exe;osname=Windows2000,r/bin/R
.exe;osname=Windows Vista,x86 r/bin/R.exe;osname=Windows
7,r/bin/R.exe;osname=Windows 2003,r/bin/R.exe;osname=Windows Server 2008"
expiration=365d yes=true description="Techila Worker R v311 Windows"
natives="r/bin/R.exe;osname=WindowsXP;processor=x86,r/bin/R.exe;osname=Windo
```

```
ws2000;processor=x86,r/bin/R.exe;osname=Windows Vista;processor=x86,
r/bin/R.exe;osname=Windows 7;processor=x86,r/bin/R.exe;osname=Windows
2003;processor=x86,r/bin/R.exe;osname=Windows Server 2008;processor=x86"
prependpath=r *
```

To create a Bundle with **only 64-bit** runtime components:

```
%techila% createbundle bundlename="Techila Worker R v311" resource=r
exports=fi.techila.grid.cust.common.library.gnur.v311.client
natives="r/bin/R.exe;osname=WindowsXP,r/bin/R.exe;osname=Windows2000,r/bin/R
.exe;osname=Windows Vista,x86 r/bin/R.exe;osname=Windows
7,r/bin/R.exe;osname=Windows 2003,r/bin/R.exe;osname=Windows Server 2008"
expiration=365d yes=true description="Techila Worker R v311 Windows"
natives="r/bin/R.exe;osname=WindowsXP;processor=amd64,r/bin/R.exe;osname=Win
dows2000;processor=amd64,r/bin/R.exe;osname=Windows Vista;processor=amd64,
r/bin/R.exe;osname=Windows 7;processor=amd64,r/bin/R.exe;osname=Windows
2003;processor=amd64,r/bin/R.exe;osname=Windows Server 2008;processor=amd64"
prependpath=r *
```

Note! Depending on what R version you are using and other system specific parameters, you might also need to adjust the values of the following parameters:

- bundlename (update the v311 notation to the R version you are using and platforms you are using)
- exports (update the v311 notation to match the R version you are using)
- description (update the v311 notation to the R version you are using)

Figure 76. Entering the 'createbundle' command to the command line.

3. After executing the command, the Bundle will be automatically created, uploaded and approved.

   Please note that each step might take several minutes. After all steps have been completed, the view should resemble the one shown below. The command prompt will display a list of files that will be included in the R Runtime Bundle.

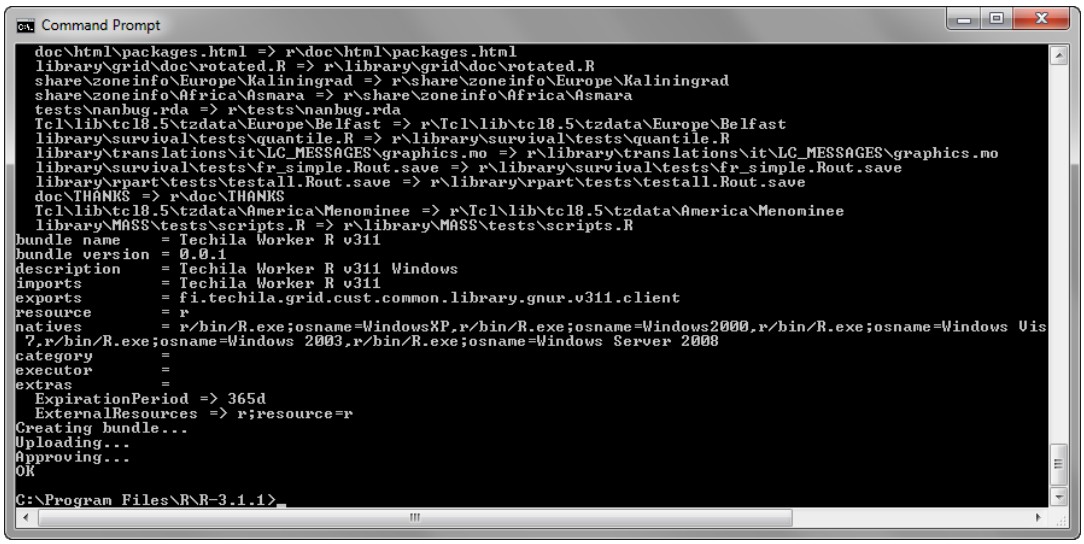

Figure 77. After successfully creating, uploading and approving the Bundle the view should resemble the one shown here.

The R Runtime Bundle is now ready for use

6. Creating Module and Data Bundles with the Bundle Creator tool

This Chapter contains instructions on how to create Module and Data Bundles using the Bundle Creator tool.

Procedure

1. Launch the Techila Bundle Creator tool by navigating to the 'lib' directory in the Techila SDK and double clicking on the **bundlecreator.jar** icon.

2. Check the location of 'techila_settings.ini' file and correct if needed

3. If you are creating a Module Bundle, choose the applicable **Module Bundle template** from the **Template** drop-down menu. If you are creating a Data Bundle, choose the **--none--**template.

4. Enter a descriptive and unique name in the **Bundle Name** field. The {user} notation may be used to include your alias in the Bundle name

5. Enter a descriptive resource name to the **Resource** field.

6. Select applicable operating systems and processor architectures in the **Platforms** section. This step determines which Workers will be able to use the Bundle.

7. Use the **Add..** button on the 'Files' tab to select the files and/or directories that should be placed in the Bundle.

8. Enter the path that should be trimmed from the path of the files to the **Trim Path** field.

9. If required, prepend the path using the **Prepend Path** field.

10. Click the **Create** button. The Bundle will be created and transferred to the Techila Server. The Bundle is now ready for use.

11. Close the Techila Bundle Creator tool window by clicking the **Close** button

7. Appendix 1: Parameters of the createBunde command

The table below contains a list of the parameters for the CLI '`createBundle`' command. Default values are in **bold**.

PARAMETER	REQUIRED	VALUE	DESCRIPTION
bundlename	Yes	<bundle name>	Name of the Bundle. Will be visible in the Techila Web Interface.
description	No	<description>	Description of the Bundle. Will be visible in the Techila Web Interface.
bundleversion	No	<x.y.z>	Bundle version determines the version of the Bundle. When the Techila Worker requests to download a Bundle the Bundle with the largest version number will be sent to the Techila Worker.
expiration	No	<expiration time>	Time the bundle is stored on the Workers after last use. Default is 15d, which stores the Bundle for 15 days.
exports	No	<exports>	Defines the bundle exports.
serverexpiration	No	<server expiration time>	Time the bundle is stored on the Techila Server after last use. By default, bundles on the Server do not expire.
resource	No	<resource name>	The name of the resource the Bundle is offering. Used to differentiate Bundles, e.g. MATLAB Runtime Bundles and Perl Runtime Bundles.
natives	No	<natives>	Defines which platforms the Bundle will be available for. Typically used to differentiate Bundles for Windows and Linux platforms.
category	No	<bundle category>	The Bundle category. Used for example for differentiating Core Bundles and End-User Bundles.
executor	No	<executor bundle>	The name of the Executor Bundle.
copy	No	<true|**false**>	Defines whether the Bundle contents will be copied to the temporary working directory on the Techila Worker.
trimpath	No	<trim path>	Removes the first substring of this string that matches the given regular expression.
prependpath	No	<prepend path>	Defines the path that will be prepended to the Bundle path on the Workers.
extrapath	No	<extra path>	Defines an extrapath
variable	No	<variable>	Defines an environment variable
yes	No	<true|**false**>	Defines whether a confirmation prompt will not be displayed.

test	No	<true|**false**>	Enables the test mode. When enabled, the output of the commands will be displayed. Useful for testing purposes.
verbose	No	<**true**|false>	Enables verbose mode. When enabled, more detailed information will be printed to the console.
extras	Yes	<extras>	Defines extra bundle parameters.
files	Yes	<files and/or dirs>	Last input argument. Defines which files and/or folders will be included in the Bundle.