



TMcraft

Shell Tutorial

Basic Development

Original Instructions

This Manual contains information of the Techman Robot product series (hereinafter referred to as the TM AI Cobot). The information contained herein is the property of Techman Robot Inc. (hereinafter referred to as the Corporation). No part of this publication may be reproduced or copied in any way, shape or form without prior authorization from the Corporation. No information contained herein shall be considered an offer or commitment. It may be subject to change without notice. This Manual will be reviewed periodically. The Corporation will not be liable for any error or omission.

 and  logo are registered trademark of TECHMAN ROBOT INC. in Taiwan and other countries and the company reserves the ownership of this manual and its copy and its copyrights.

 **TECHMAN ROBOT INC.**

Table of Contents

Revision History	3
1. Introduction.....	4
2. Prerequisites.....	6
3. TMcraft Shell Program Structure.....	7
3.1 Brief Explanation of TMcraft API.....	7
3.2 Program structure of a TMcraft Shell	8
4. Start Programming a TMcraft Shell.....	10
4.1 System Design	10
4.2 Create a WPF Application Project.....	12
4.3 Source Code.....	14
4.3.1 MainWindow.xaml.....	14
4.3.2 MainWindow.xaml.cs	15
4.4 Debug with TMflow Simulator.....	17
5. Build TMcraft Shell Package.....	19
6. Installation Code and Checksum	22
7. Using TMcraft Shell on TMflow.....	23
8. Dos & Don' ts.....	25

Revision History

Revision	Date	Description
1.0	2024-02-07	Original release

1. Introduction

TMcraft Shell, developed in C#/WPF, is a customized GUI page that overlays TMflow. It allows developers to significantly define the user experience for minimal interaction between users and TMflow. The Shell can act as an application setup wizard, a dashboard, or both. An example application is the development of a Palletizing Operator:

- Wizard:

The Wizard UI in TMcraft Shell enables users to complete a series of settings in a few steps. After confirmation, the program gathers these parameters and generates the relevant script project using the TMcraft API.

- Dashboard:

When the application launches, the robot runs the specified script, and TMcraft Shell displays robot data and project variables on the Dashboard. Users can also benchmark palletizing against Units per Hour, Equipment Efficiency, Defect Counts, and Downtime.

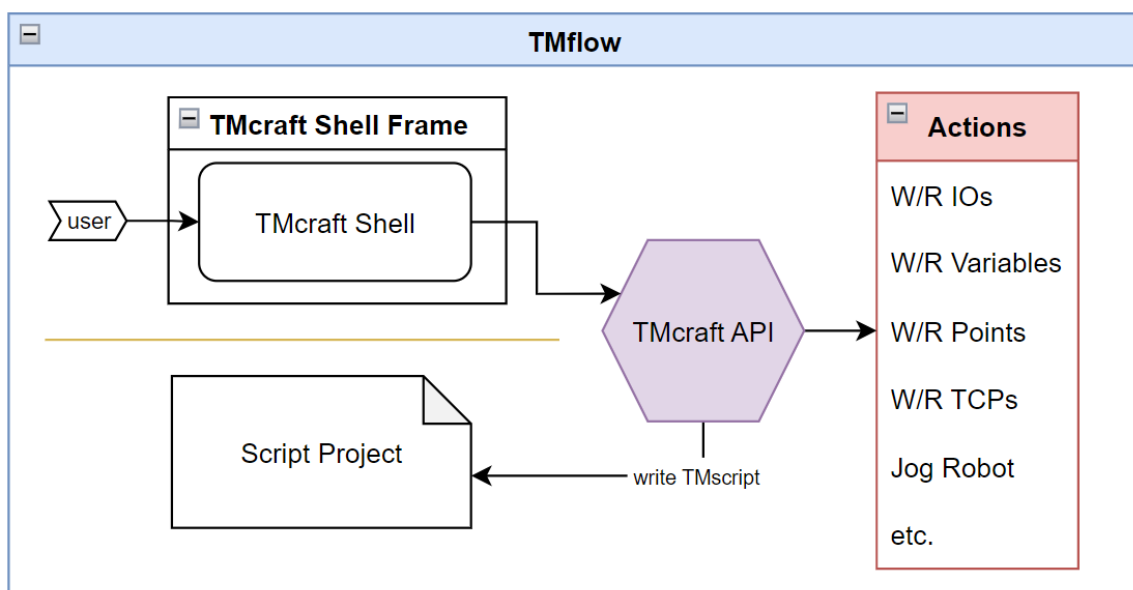


Figure 1: The Architecture of a TMcraft Shell

To determine whether the TMcraft Shell is the right fit to realize concepts of users, consider these questions:

Are users familiar with script programming?

As built with C#, developers need C# skills to develop the TMcraft Shell.

Do users need the integration from other PLC or HMI into the application?

Using TMcraft Shell on the Robot Controller can cut costs by eliminating the need for extra HMI devices. But for complex or big projects, integrating a PLC with its HMI for the GUI might be the better choice.

To understand how to develop a TMcraft Shell, please check the following diagram:

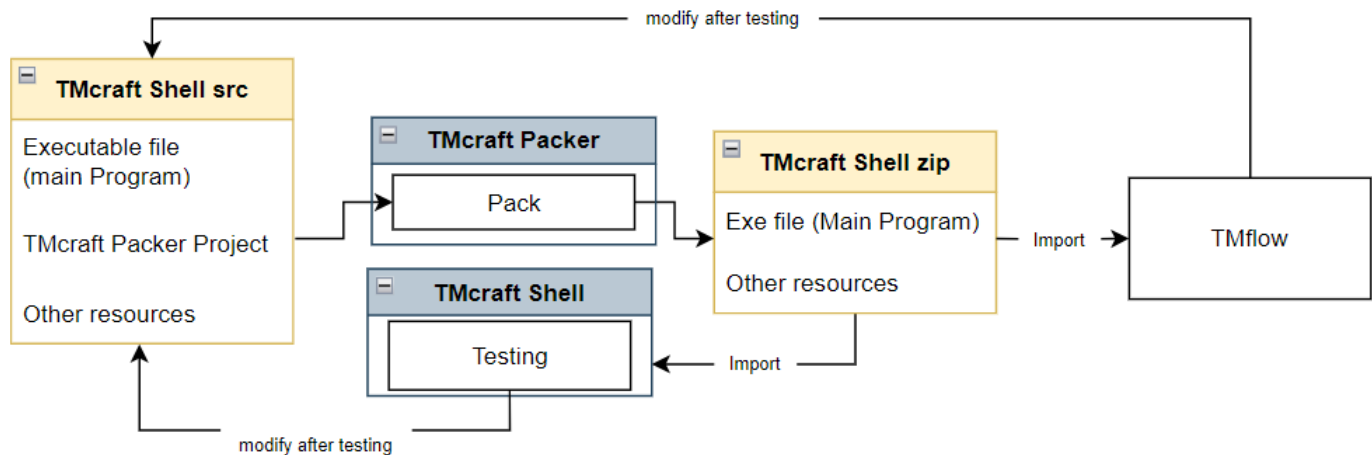


Figure 2: Basic Concept of TMcraft Shell development

Developers create the TMcraft Shell UI as an executable file, a WPF application, packing into a TMcraft Shell zip by TMcraft Packer. Developers can further test it on TMflow Simulator, or directly on a robot. Following multiple iterations of modifications and testing, they can release the TMcraft Shell.

This tutorial organizes the content based on TMcraft.dll version 1.16.1400 from TMflow 2.16.

- Section 2: prerequisites for Developing a TMcraft Shell
- Section 3: the concept and features of the TMcraft API and TMcraft Shell Program
- Section 4: the process of how to develop a simple TMcraft Shell
- Section 5: how to complete building a TMcraft Shell Package
- Section 6: the concept of Unique Identifier
- Section 7: how to import and use the TMcraft Shell on TMflow
- Section 8: the dos and don'ts

2. Prerequisites

To develop TMcraft Shells, developers should be capable of:

- Having Basic knowledge of programming with C#
- Using WPF to build UI
- Using TM script the Programming Language

Software requirements,

- An integrated development environment for C# and WPF (such as Microsoft Visual Studio 2022)
- TMflow 2.16 or above
- .NET 6.0 and .NET SDK
- TMcraft Packer for building and packaging the TMcraft item

3. TMcrafft Shell Program Structure

3.1 Brief Explanation of TMcrafft API

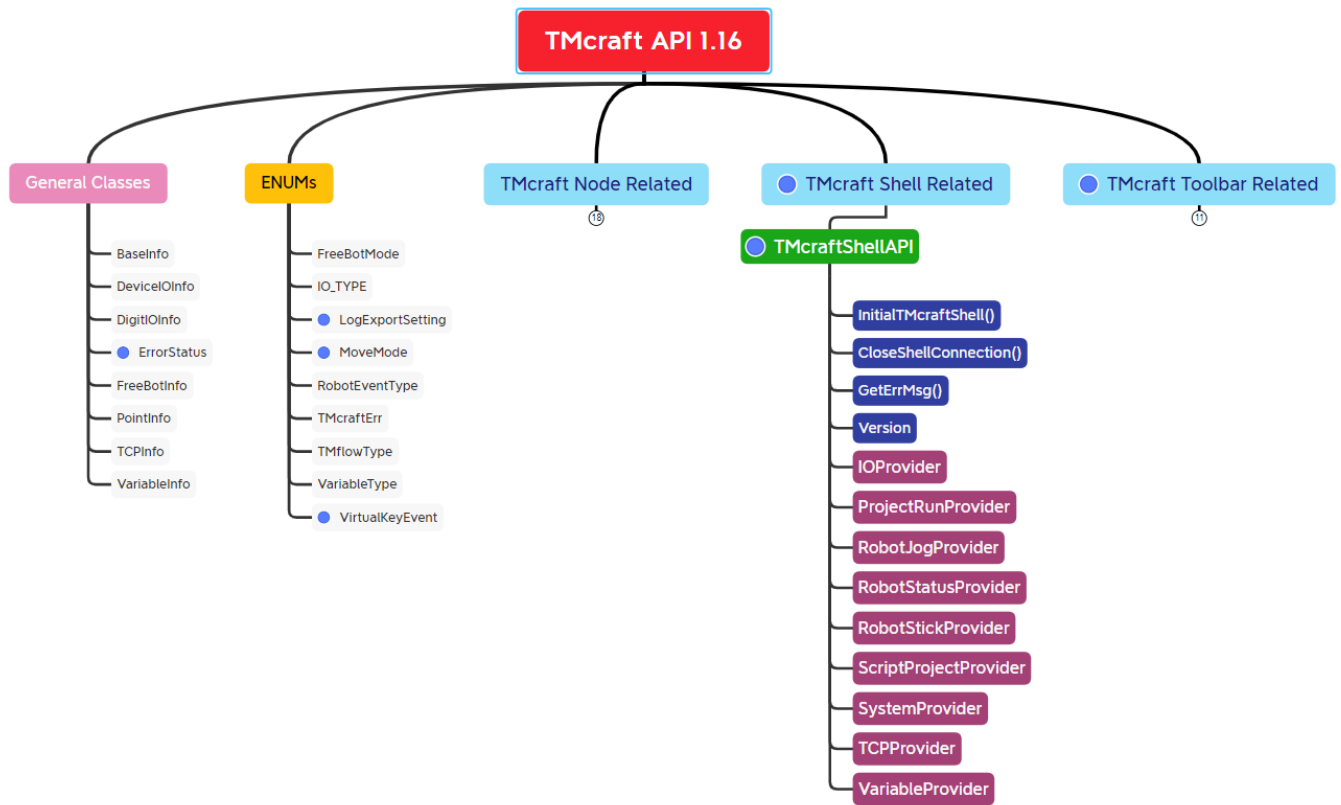


Figure 3: TMcrafft API architecture

TMcrafft API is an interface that connects the TMcrafft Shell with TMflow to get all sorts of robot information and request TMflow to take different actions or generate scripts. In addition, this API is a Dynamic Link Library file (DLL); developers should add it to program dependencies so that the program can use the functions within the API.

TMcrafft.dll consists of classes, enums, and functions for every TMcrafft items. Here are those most related to TMcrafft Shell:

- **TMcrafftShellAPI** has all the functions and provider classes to build TMcrafft Shell. Note that each TMcrafft item comes with its own set of provider classes. Even if some share the same name, their member functions vary.
- **Providers** are a set of classes with functions associated with different interactions with TMflow frequently used among the Programs.

- **General classes** are a collection of classes, such as DeviceIOInfo, TCPInfo, and ErrorStatus, used by different types of Provider functions.
- **Enum**, or enumerators, are value types defined by a set of named constants of the underlying integral numeric type used by provider functions.

Refer the table below to overview the API capabilities across various TMcraft plugins:

Capabilities \ Plugins	Node	Shell	Toolbar
Base (Add/Edit/Delete)	O		
Point (Add/Edit/Delete)	O		
Tool (Add/Edit/Delete)	O	O	O
Digital IO (Read/Write)	O	O	O
Analog IO (Read/Write)	O	O	O
Variables (New/Edit)	O	O	O
Vision Job (Add/Open/Delete)	O		
Jog the robot	O	O	
Freebot (Set/Get)	O	O	O
End Button Event	O	O	O
Get Current Language	O	O	O
Get TMflow Type	O	O	O
TMscript on flow project (Read/Write)	O		
Login/Logout/Get Control		O	
script Project (Add/Edit/Delete)		O	
Robot status (Error, Run, etc.)		O	O
Error Event		O	
Virtual Robot Stick		O	
Export/Import		O	
Variables Runtime Value (Read/Write)		O	Read only

3.2 Program structure of a TMcraft Shell

Refer to the sample code below for the program structure.

```

using TMcraft;

namespace TMcraftSample
{
    public partial class MainWindow : Window
    {
        TMcraftShellAPI ShellUI;

        public MainWindow()
        {
            InitializeComponent();

            private void Window_Loaded(object sender, RoutedEventArgs e)
            {
                ShellUI = new TMcraftShellAPI();
                ShellUI.InitialTMcraftShell(); //connect TMflow
            }

            private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)
            {
                if (ShellUI != null) ShellUI.CloseShellConnection();
            }

            private void Btn_ShowTMflow_Click (object sender, RoutedEventArgs e)
            {
                //example: hide the Shell GUI and show TMflow
                if(ShellUI != null) ShellUI.SystemProvider.ShowTMflow();
            }
        }
    }
}

```

To create the foundation of a TMcraft Shell Program, please remember the key points below:

1. Include TMcraft.dll as a reference. Remember to add the namespace, `using TMcraft`, onto the program.
2. Declare a global object based on the class `TMcraftShellAPI`.
3. When loading the window of the Shell GUI, assign the object (`new TMcraftShellAPI()`), then call the function `InitialTMcraftShell()`.
4. When closing the window of the Shell GUI, be sure to call the function `ShellUI.CloseShellConnection()`.

The rest of the program should be all sorts of event functions that can interact with TMflow through TMcraft functions.

4. Start Programming a TMcraft Shell

Developers are favored to follow the steps to develop a TMcraft Shell.

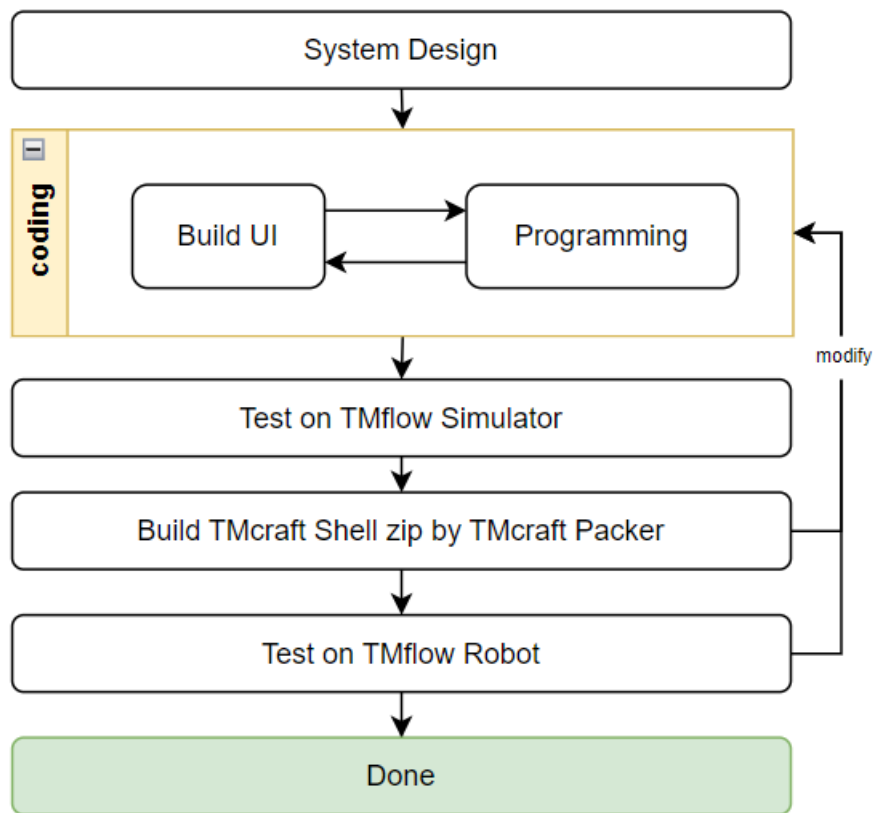


Figure 4: The Recommended Procedure for Developing a TMcraft Shell

From section 4 to section 6, users will learn how to approach these steps with a simple TMcraft Shell, *HelloWorldShell*.

4.1 System Design

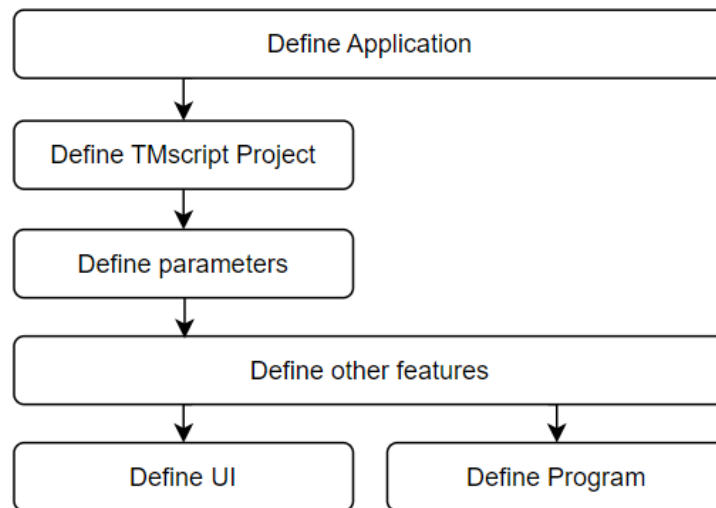


Figure 5: The Recommended Procedure of System Design

The outline of the application:

- Through the Shell GUI, users can generate a script project to display the desired content.
- “Hello, World” will display as a banner of the Display Board.

With such a definition, the relevant TMscript project should go like this:

```

define
{
}
main
{
  Display("Green", "White", "Hello World", [content])
}
closestop
{
}
errorstop
{
}

```

Figure 6: The Script Project Template

As shown, [Content] is the user-decide parameter, and users can use a textbox to define it. On the other hand, there are a few more features required:

- The **Back To TMflow** button: click to hide the Shell GUI and show TMflow.
- The **Login** button: click to login and get control when the robot is in manual mode.
- The **Generate Script** button: click to get the content from the textbox, create/open a script project, and write the associated script into it.

REFER FIGURES BELOW FOR THE UI AND THE PROGRAM ARCHITECTURE.

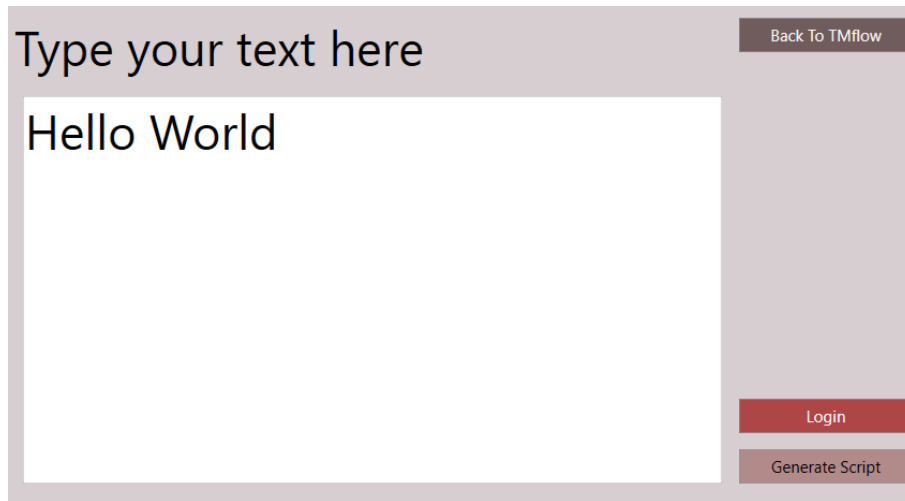


Figure 7: HelloWorldShell UI

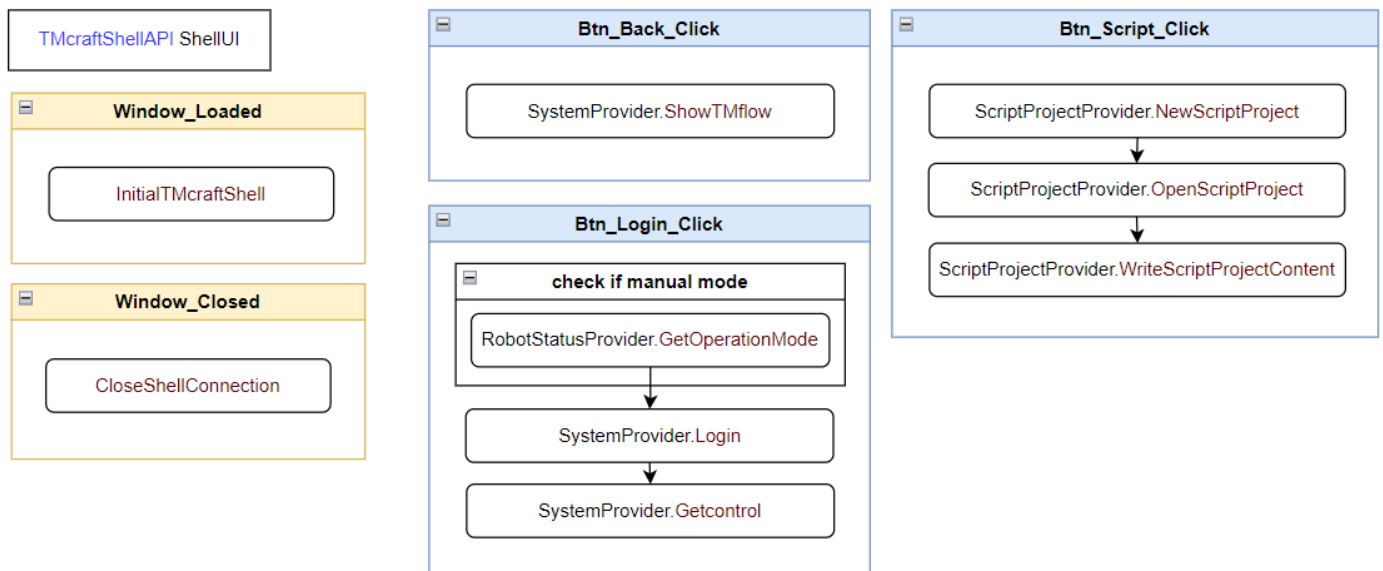


Figure 8: The Program Architecture of HelloWorldShell

4.2 Create a WPF Application Project

This section is written based on the usage of Microsoft Visual Studio 2022. First, users can create a new project with the *WPF Application* template.

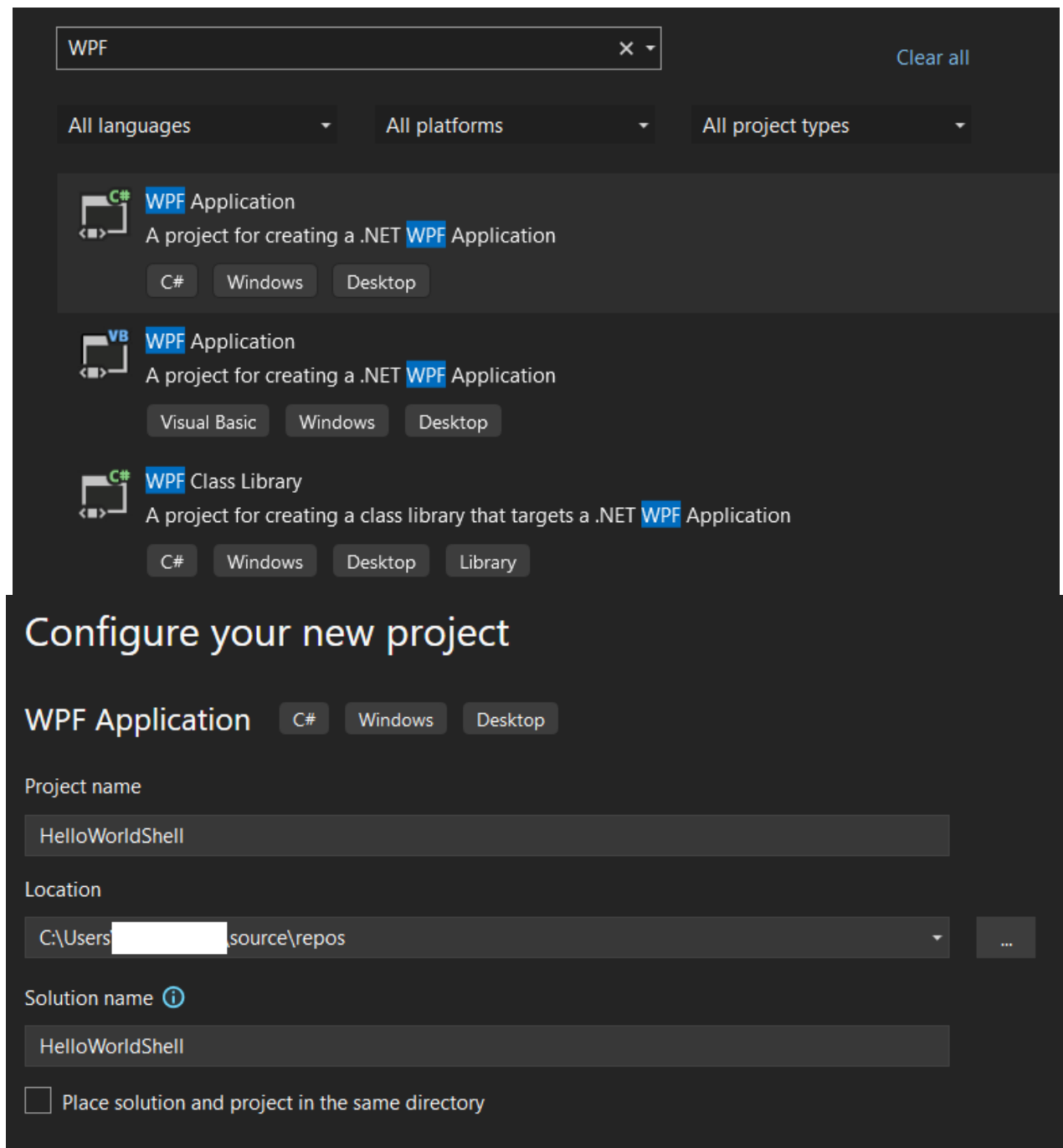


Figure 9: Create a New Project with the WPF Application Template

Remember to set up with .NET 6.0 as the framework of the project.

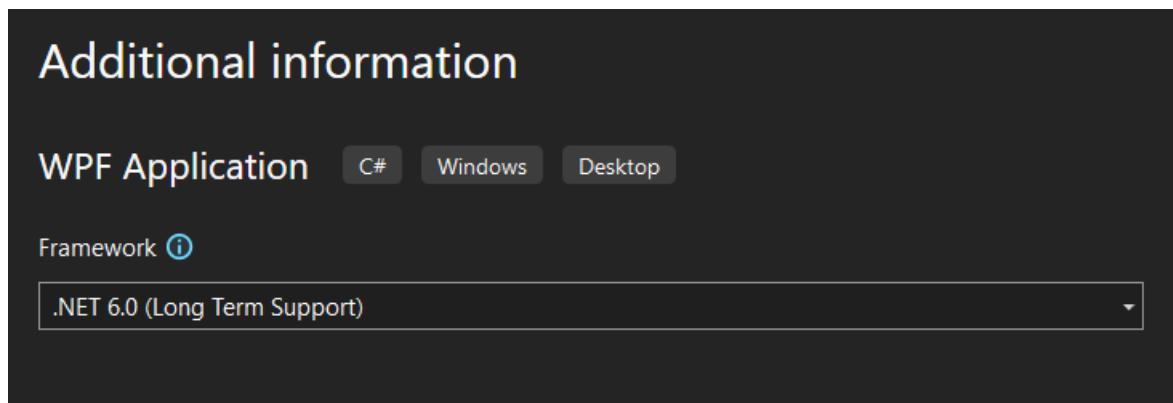


Figure 10: .NET 6.0 Setup as the Framework

Once opened the project in the editor, add the TMcraft.dll as the reference.

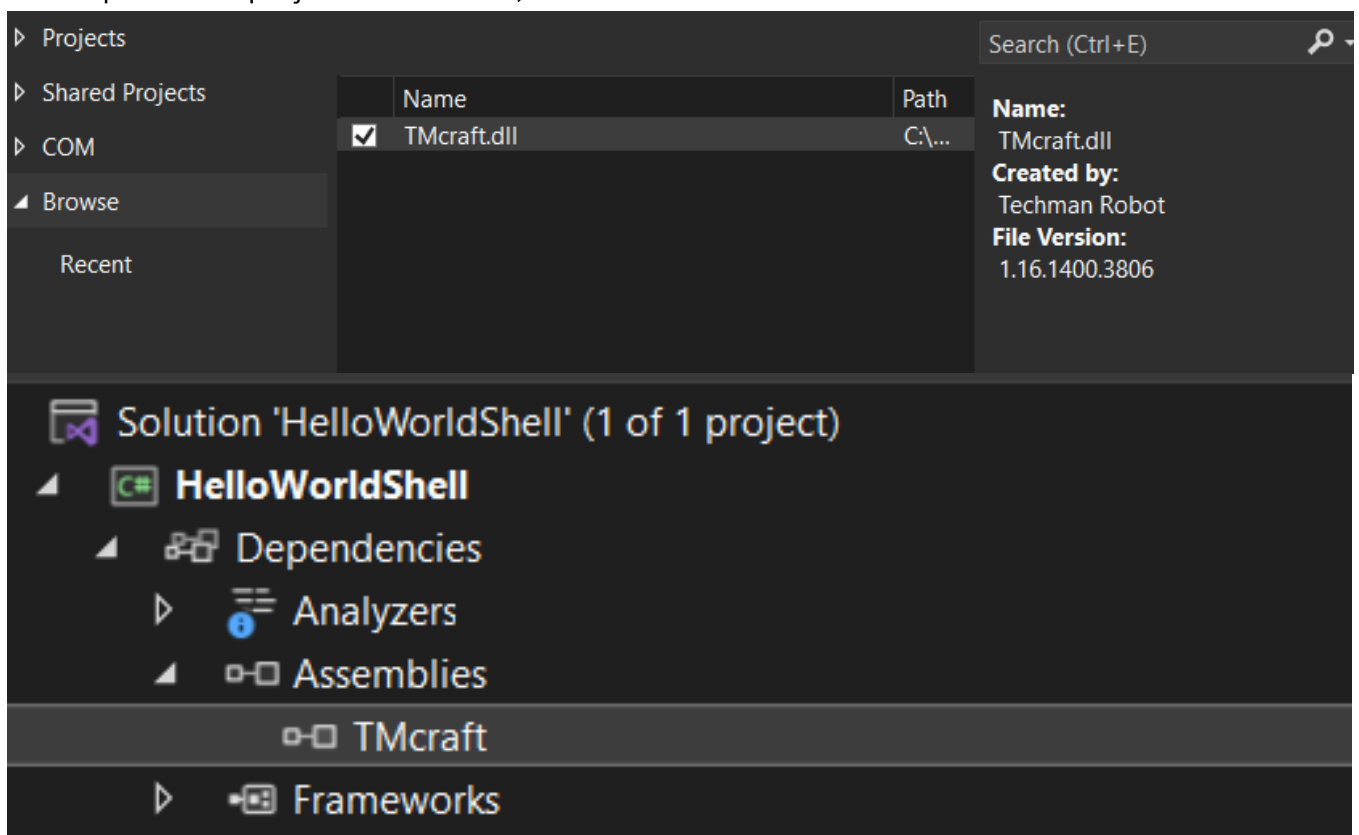


Figure 11: Add TMcraft.dll as a Reference

4.3 Source Code

This section focuses on the key aspects of the HelloWorldShell source code. For the complete source code, please refer to TMcraft Development Kit\Shell\Samples\HelloWorldShell\

4.3.1 MainWindow.xaml

MainWindow.xaml defines the UI; here are a few points necessary:

- **Loaded="Window_Loaded"**: define the UI's loading function by initializing the TMcraft

Shell (`TMcraftShellAPI.InitialTMcraftShell()`) during window loading.

- `Closing="Window_Closing"`: when about to close the window, run the API function `TMcraftShellAPI.CloseShellConnection()`.
- Add click events on each buttons including: `Btn_Back_Click`, `Btn_Login_Click`, `Btn_Script_Click`.

4.3.2 MainWindow.xaml.cs

MainWindow.xaml.cs defines the program of the TMcraft Shell; here are a few points necessary:

- First, add TMcraft.dll onto the program reference.

```
using System.Windows.Shapes;
using System.Xml.XPath;
using TMcraft;

namespace HelloWorldShell
```

- Declare a global `TMcraftShellAPI` object (Shell UI).

```
public partial class MainWindow : Window
{
    TMcraftShellAPI ShellUI;

    0 references
    public MainWindow()
    {
```

- Define the `Window_Loaded` function to initialize the ShellUI and connect TMflow.

```
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    if(ShellUI == null)
    {
        ShellUI = new TMcraftShellAPI();
        ShellUI.InitialTMcraftShell();
    }
}
```

- Define the `Window_Closing` function to close the connection with TMflow.

```
private void Window_Closing(object sender, RoutedEventArgs e)
{
    try
    {
        if(ShellUI != null)
        {
            ShellUI.CloseShellConnection();
        }
    }
}
```

- Define the click event function `Btn_Back_Click`. When triggered, this function should call the API function `SystemProvider.ShowTMflow()`.

- Finally, write the script into the project by calling `ScriptProjectProvider.WriteScriptProjectContent(script)`.

4.4 Debug with TMflow Simulator

Before running (debugging) the program, copy all API-relevant files from the TMcraft Development Kit folder (\\ TMcraft API\1.16.1400) and place them inside the bin folder.

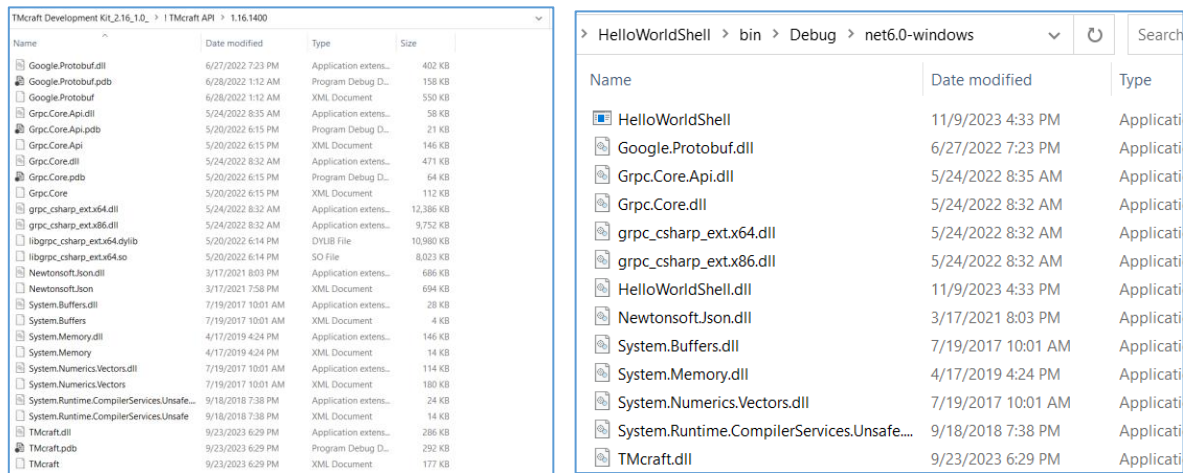


Figure 12: Put All TMcraft API-relevant files to the bin folder

Install and run the TMflow Simulator on the PC. Select a type of virtual robot and proceed with system booting.

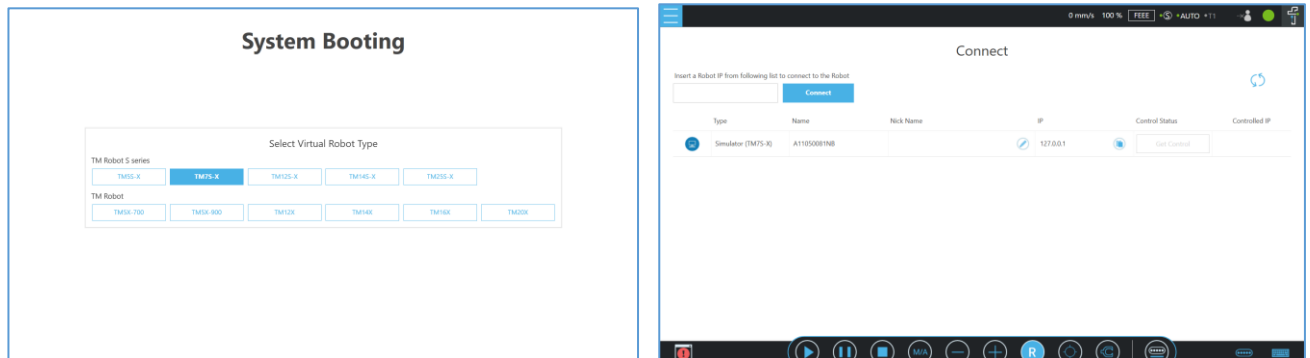


Figure 13: TMflow Simulator

Once the TMflow Simulator is ready to use, users can run the TMcraft Shell Program and start debugging.

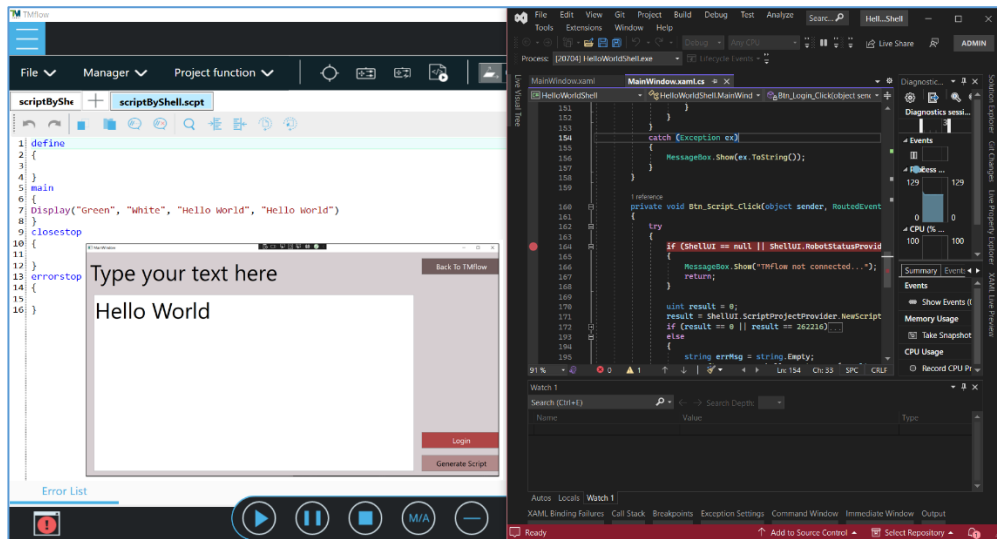


Figure 14: Debug with TMflow Simulator

5. Build TMcraft Shell Package

The previous section described how to build the TMcraft Shell Program. Before using it on TMflow, it must be packaged using the TMcraft Packer.

First, prepare a source folder with the following items.

- The execution file of the Shell UI
- All files from the TMcraft API (1.16 or above) folder from the development kit
- Other resource files, such as:
 - Other reference files used by the Program
 - Files used by the Shell, such as media, documents, etc.

Note**NOTE:**

Suggest putting all files from the `..\bin\Debug` folder into the source folder.

Next, open TMcraft Packer (1.12 or above) and create a project. Select **Shell** as the target TMcraft item and complete the Project Settings. The system saves the project as a `.tmcraft` file at the specified file path in the **Location** field.

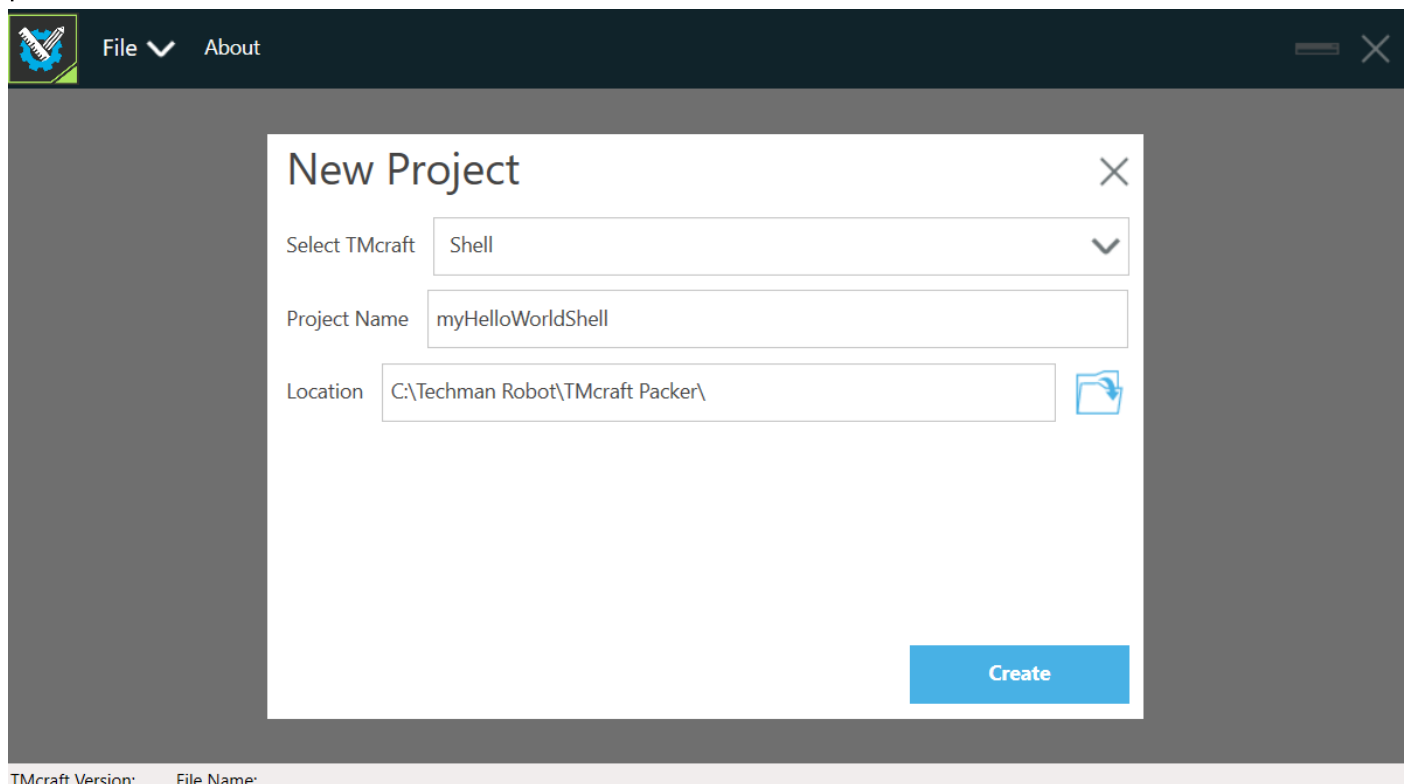


Figure 15: Create a TMcraft Packaging Project

**IMPORTANT:**

Ensure that TMcraft Packer is running in the environment with .NET 6.0 installed.

After opening the project, users can see a form of parameters required to build and package the TMcraft Shell. Refer to the following for explanations.

Parameters	Description	Requisite
Source folder	Path of the source folder	<input type="radio"/>
Target folder	Path of the target folder. For any new projects, the default target folder is ..\TMcraft Packer\USB\TM_Export\[PC Name]\TMcraft\Shell	<input type="radio"/>
Provider	Name of the developer or the company providing this Shell, which is shown on the TMcraft Management Page	
Version	The version of the TMcraft Shell, which is registered onto the exe file and also shown on TMcraft Management Page	<input type="radio"/>
Exe Name	Name of the execution file of the TMcraft Shell Program	<input type="radio"/>
Zip Password	Developers can define the password of the zip file. The password length should be between 6 and 256 characters of the non-case-sensitive Latin alphabet and numbers.	

※ Complete all parameters labeled with a red star in the TMcraft Packer GUI before proceeding.

Figure 16: Set up the Packaging of TMcraft Shell

Once all requirements are ready, click **Generate**. It might take several minutes to package, and users can check the current progress on the Page (Users will see an error message if anything goes wrong). After packaging the TMcraft Shell successfully, users should see the successful status label.

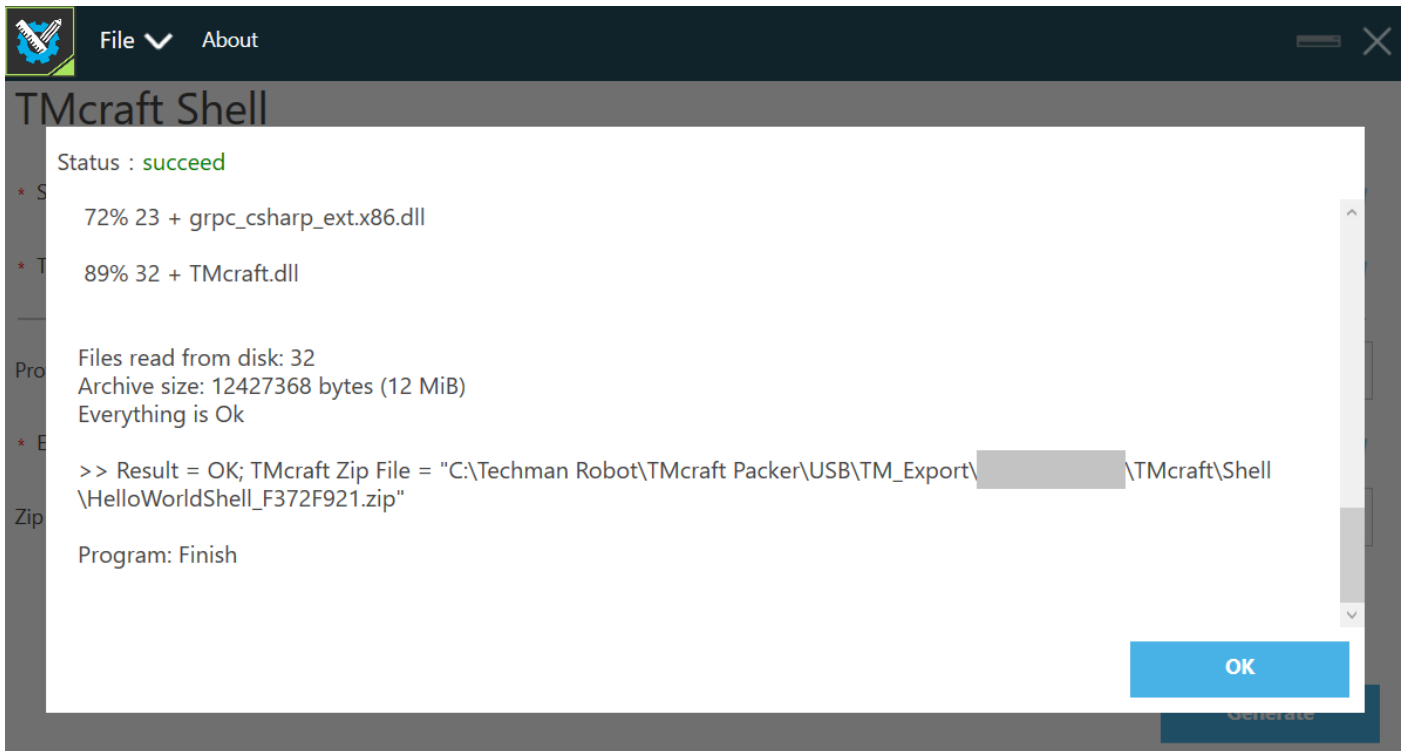


Figure 17: The Message Shown When Packaged Successfully

Finally, users can find the TMcraft Shell zip file named after *[Exe Name]_[Checksum]* in the target folder.

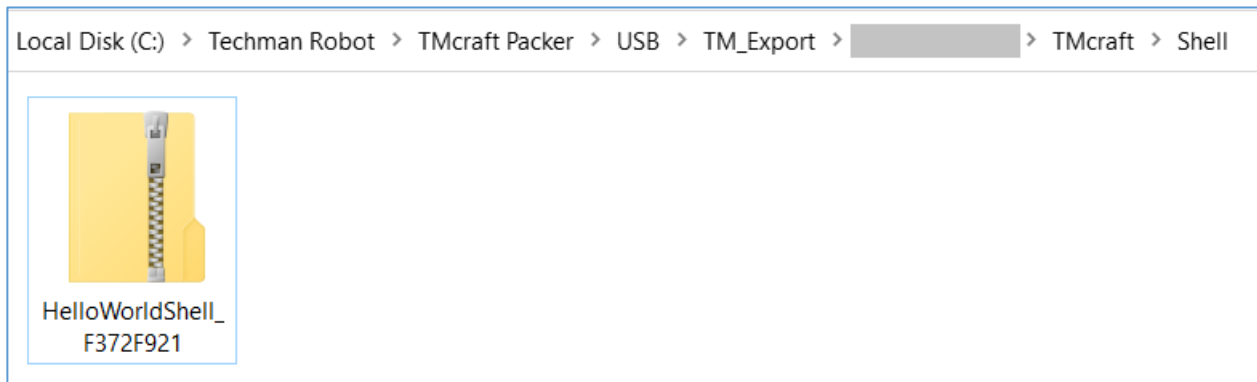


Figure 18: TMcraft Shell zip at Target folder

6. Installation Code and Checksum

To protect the rights and interests of developers, each TMcraft Packer Project has its own encrypted Installation Code (GUID-based). Any TMcraft items generated by this project will share the same Installation Code and be able to replace one another on the same robot. Additionally, items generated from two different TMcraft Packer Projects (without identical Installation Codes) cannot replace each other, even if they share the same name and configuration. Therefore, developers should keep their packer project safe.

On the other hand, TMcraft Packer will also generate a checksum for each TMcraft item based on the binary footprint of the files (exe and dll) within the source folder and the Installation Code. The TMcraft item saves the checksum onto the configuration file. When importing the TMcraft item, TMflow will calculate a checksum by the same method; if these two checksums are not identical, TMflow will block the importing.

Developers can announce the checksum, allowing end users to verify it on the TMcraft Management Page to ensure the product's authenticity.

7. Use TMcraft Shell on TMflow

To import a TMcraft Shell Package to a robot, place the zip file in the following path:

[Storage Drive]\TM_Export\[Folder]\TMcraft\Shell

Plug the storage device into the robot and navigate to **System > Import/Export > Import**. Note that there is a new category: **TMcraft**. Select **TMcraft > Shell**, choose the required TMcraft Shell zip file, and click **Import**.

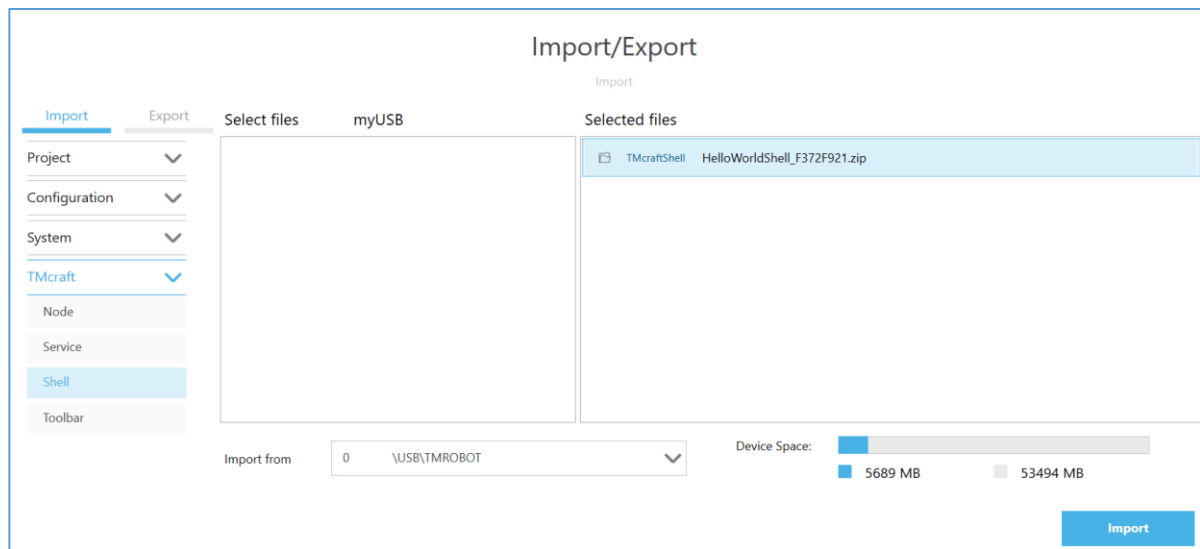
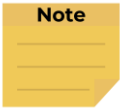


Figure 19: Import TMcraft Shell zip

Next, navigate to **Configuration > TMcraft Management > Shell**. Users can see the imported TMcraft Shell shown in the table with the following information:

- **Name:** the executable file name of the TMcraft Shell
- **Provider:** as defined in Chapter 5
- **Version:** as defined in Chapter 5
- **Checksum:** a number calculated based the binary footprint of the files (exe and dll) in the TMcraft Shell folder. Developers may publish this for identification purpose to their end-users.

There is a checkbox for users to choose if the system returns to TMflow when the TMcraft Shell Program is closed, purposefully or accidentally. Users should assess this option carefully before unchecking it. Once enabled, the TMcraft Shell will execute following a power cycle.



NOTE:
In cases where the checkbox is not selected, ensure an alternative way to access TMflow from the TMcraft Shell. For instance, include a button that triggers the API function [SystemProvider.ShowTMflow](#) to allow users or technical engineers to return to TMflow and disable the Shell UI if needed.

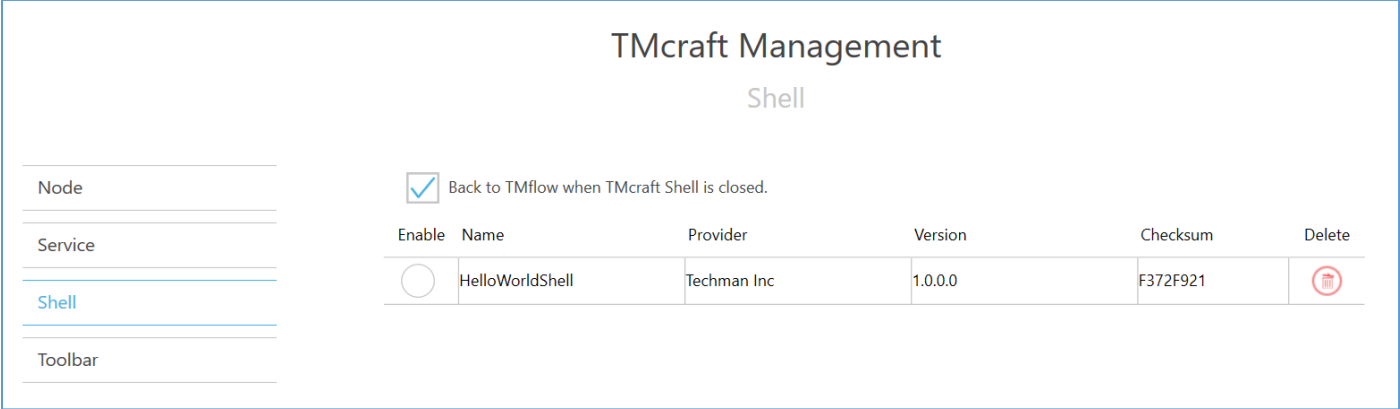


Figure 20: Enable the TMcraft Shell on TMcraft Management

Should the system return from TMcraft Shell to TMflow by calling the API function [SystemProvider.ShowTMflow](#), for instance, an additional button appears in the upper left corner. Clicking this button allows users to return to the TMcraft Shell UI.

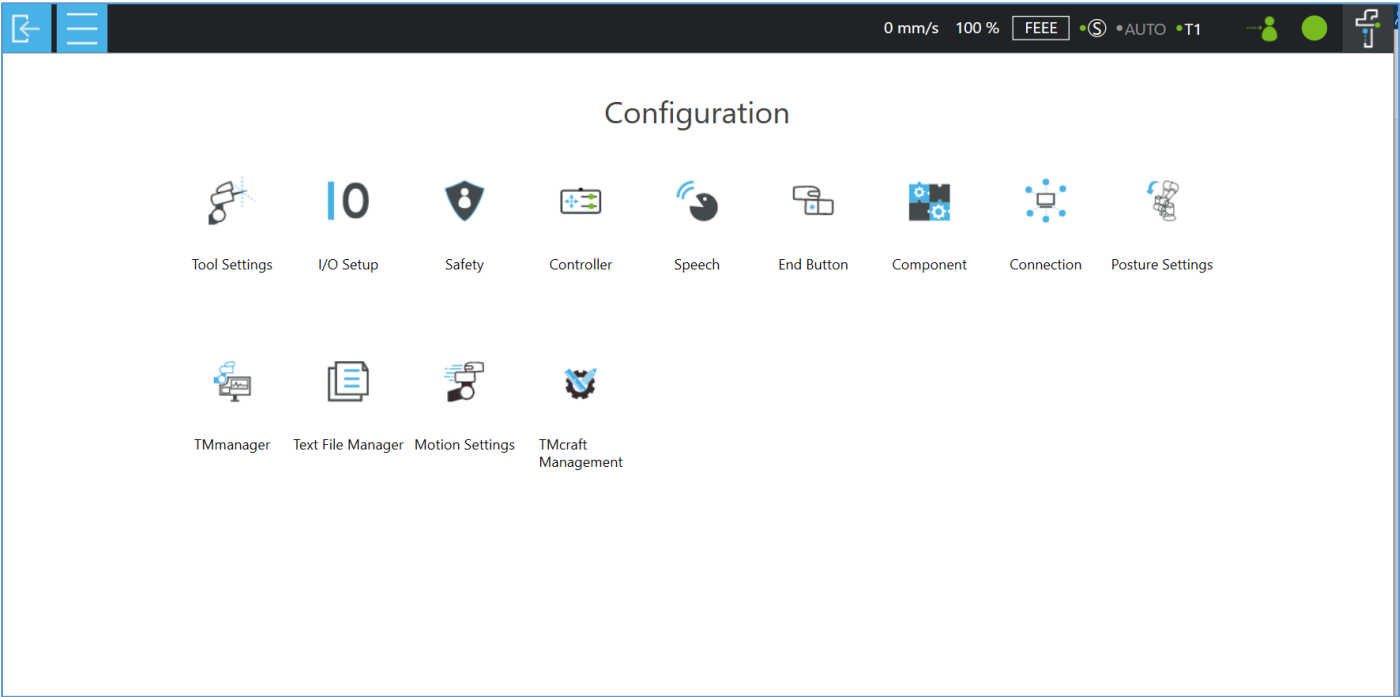


Figure 21: Back to TMcraft Shell by Clicking the Button at Upper Left Upper

8. Dos & Don'ts

- Suggestions on how to use TMcraft API functions:

1. Check if the `TMcraftShellAPI` object is null or not
2. Use a `uint` object to get the result of the API function (replied by TMflow)
3. Check if the result is 0 or not; if not, call `GetErrMsg()` to get the corresponding error message
4. Note that, instead of TMflow error, there might also have error from the API itself; this error is represented as a `TMcraftErr` object and returned by calling `GetErrMsg()`.

- Safety Assessment is necessary when using RobotJogProvider

If a TMcraft Shell utilizes RobotJogProvider functions for motion control, developers are responsible for ensuring a single point of control in compliance with ISO 10218-1.

- About writing files into Control Box through TMcraft items

Robot System will deny the following access from TMcraft executable, including: (1) system shutdown and (2) accessing to drives C and D, except for the following folders:

- D:\...TMflow\TMcraft\
- D:\...TMflow\XmlFiles\
- D:\...TMflow\TextFiles\

- Log

TMcraft Programs should save their log files within the items folder, which allows end-users to export the TMcraft item zip, including the log files, for developer analysis.

- Before packaging with TMcraft Packer:

- ✓ Place all API relevant files from the TMcraft Development Kit into the source folder.
- ✓ Place all files from the bin folder of the TMcraft Project into the source folder
- ✓ Ensure TMcraft Packer is running in the environment with .NET 6.0 installed.

- What if the TMcraft Program is closed unexpectedly

When developing a TMcraft Shell, it is important for developers, particularly system Integrators, to consider what actions should be taken if their TMcraft Shell Program may shut down unexpectedly. Here are some advices:

- ✓ Ensure thorough testing of the program on the TMflow Simulator before deploying it on the Robot Controller.
- ✓ Verify that the **Back to TMflow** checkbox is checked, enabling users to disable the TMcraft Shell in case of an unexpected shutdown.
- ✓ However, if unchecking the 'Back to TMflow' checkbox is needed, for instance, to restrict end-user access, developers should add a back door, like a password-protected button, for maintenance, which calls `SystemProvider.ShowTMflow`.
- ✓ If the program crashes at startup and the **Back to TMflow** checkbox is unchecked, disabling the TMcraft Shell becomes impossible, even for developers. In this case, reach out to TMflow technical support.

