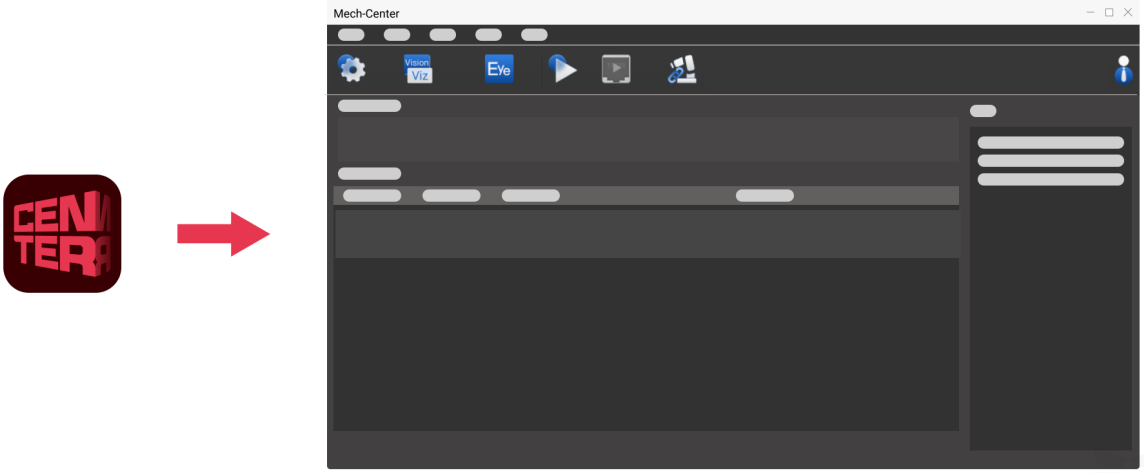
# Mech-Center Manual

**Mech-Mind**

**Jul 01, 2022**

# CONTENTS




Mech-Center is the **Control Center** of the Mech-Mind Software Suite independently developed by our company. With an intuitive interface, Mech-Center enables you to implement the global settings of the Mech-Mind Software Suite, backup and restore the whole project, and check the status of Mech-Viz, Mech-Vision, Mech-Eye Viewer, the robot, standard interface, and Adapter. It can also be used to activate and manage Mech-Interface.

---

Please refer to the section below to learn about the **User Interface and Functions** of Mech-Center.

*Quick Facts of Mech-Center*

---

Please refer to the section below to learn about the **Basic Instructions on Using Mech-Center**.

*Getting Started with Mech-Center*

---

Please refer to the section below to learn about **Mech-Interface**.

*Mech-Interface*

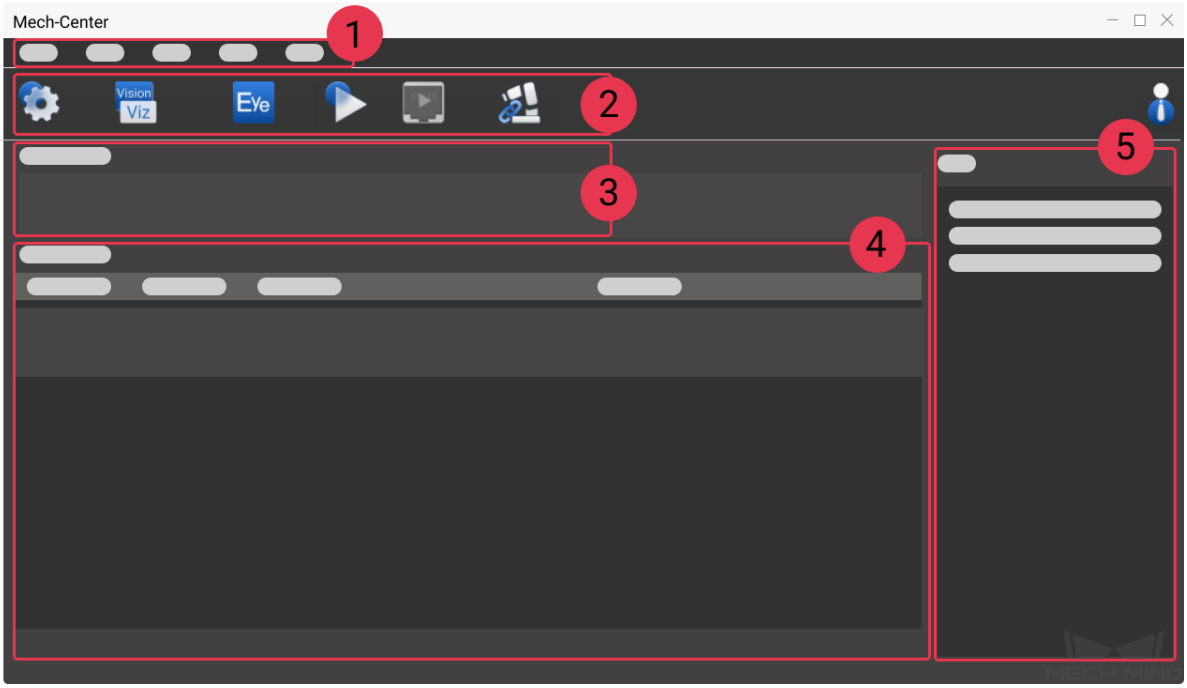
CHAPTER

ONE

# QUICK FACTS OF MECH-CENTER

Mech-Center is the **Control Center** of the Mech-Mind Software Suite independently developed by our company. With an intuitive interface, Mech-Center enables you to implement the global settings of the Mech-Mind Software Suite, backup and restore the whole project, and check the status of Mech-Viz, Mech-Vision, Mech-Eye Viewer, the robot, standard interface, and Adapter. It can also be used to activate and manage Mech-Interface.

The main interface of Mech-Center consists of 5 parts:

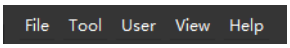



1. *Menu Bar* : Provides functions for managing projects, modifying user interface and view, generating Adapter, checking software version, etc.
2. *Toolbar* : Provides functions for deployment settings, starting Mech-Viz and Mech-Vision, connecting robot, and running the project.
3. *Service Status Bar* : Displays registered software, camera, robot, etc.
4. *Project Status Bar* : Displays the status, execution time, and details of Mech-Viz/Mech-Vision projects.

5. *Log Panel* : Displays the log of current projects and services in real time.

## 1.1 Menu Bar

Menu Bar consists of File, Tool, User, View, and Help, which provide basic functions.




### 1.1.1 File

| Options | Description | Shortcut |
|---|---|---|
| Backup Projects | Used to backup the project data of Mech-Mind Software Suite. It will save the Mech-Viz and Mech-Vision projects which are set to autoload, and Adapter projects and deployment settings of Mech-Center. | Ctrl+B |
| Restore Projects | Used to restore the selected backup project and settings. The autoload project will be replaced after restoring. Please make sure to backup the needed project. | Ctrl+R |
| Import Projects | Used to import projects from other paths. | Ctrl+I |
| Exit | Exit Mech-Mind Software Suite system. | Ctrl+Q |

**Attention:** In Operator mode, only Backup Projects and Exit are available.

### 1.1.2 Tool

| Options | Description |
|---|---|
| Pack Debug Data | Used to pack debug data of Mech-Mind Software Suite. You can customize the time period, package path, and save options. |
| Adapter Generator | Used to generate customized Adapter program. |
| Log Viewer | Used to view the log information of Mech-Viz / Mech-Vision. Please select the log file and then load it to view. You can view the log information you need by selecting a log level, entering a key word to filter, or searching. Click and drag the scrollbar at the bottom to view the log information after the specified point of time only. |
| Show Service Status | Used to display the statuses of the software, robot, interfaces, and other services. |

**Attention:** The Adapter Generator option is not available in Operator mode.

### 1.1.3 User

Modify Password: Only available in Administrator mode. Used to modify password. An operator cannot use this function.

### 1.1.4 View

Log: Check to display Log in the main interface.

### 1.1.5 Help

| Options | Description | Shortcut |
|---|---|---|
| User Manuals | Used to open the user manual of standard interface quickly. | F1 |
| Release Notes | Display the information about new features and bug fix of each versions. | N/A |
| About | Display the version and copyright information of the software. | N/A |

## 1.2 Toolbar

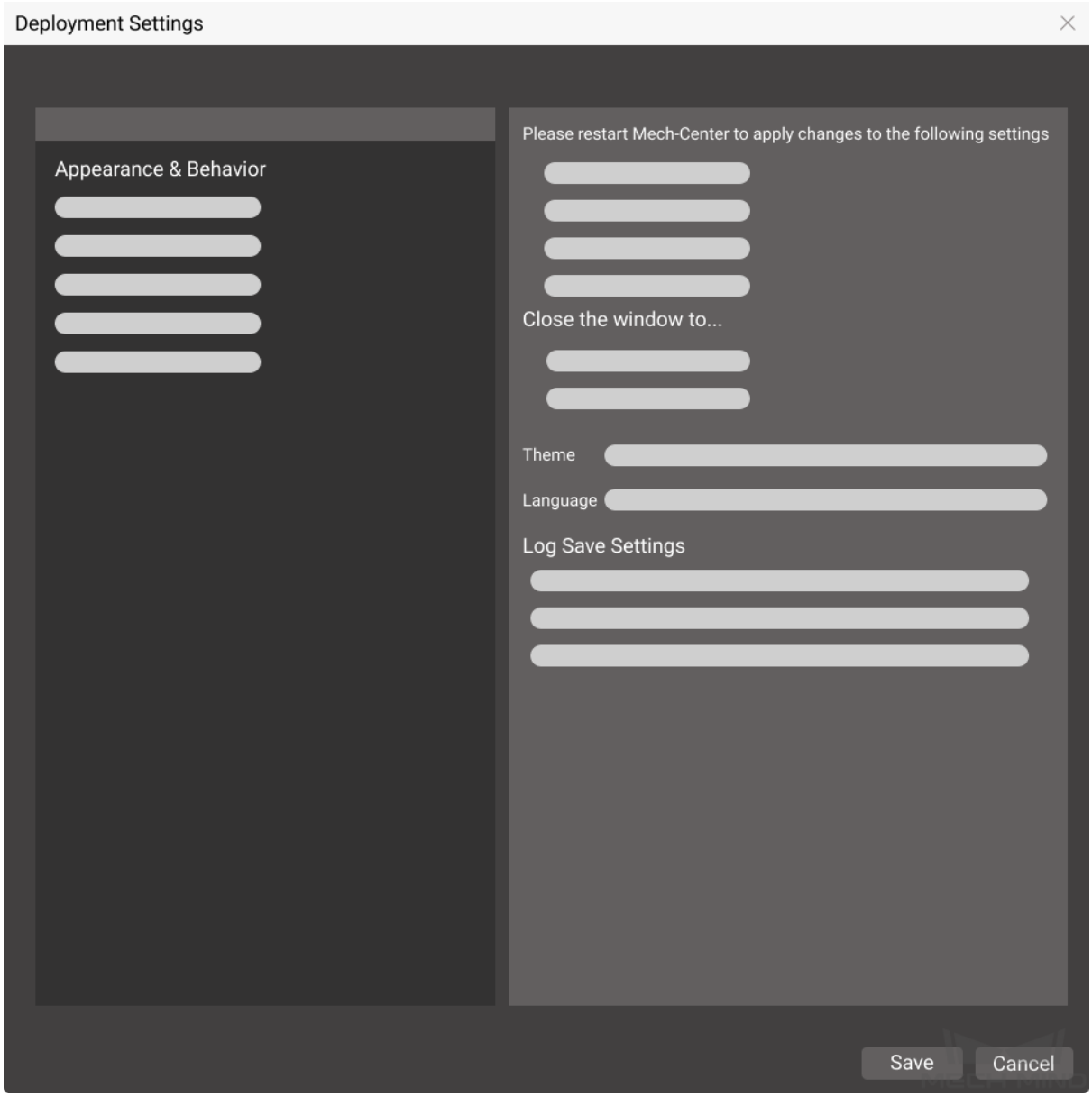
The Toolbar of Mech-Center includes seven buttons, which are Deployment Settings, Start Viz/Vision, Start Mech-Eye Viewer, Run, Start Interface, Connect Robot, and Administrator.

### 1.2.1 Deployement Settings

Deployment Settings is used to implement basic settings of Mech-Center, configure the path of each software, view the path of autoloaded projects, select external services, etc.

#### Appearance and Behavior

The Appearance and Behavior option is used to customize global settings, as shown below. You can configure relevant settings based on your own behavior, such as changing the theme color, switching interface language (Chinese, English, Janpanese, and Korean are now available), changing log save settings, etc.

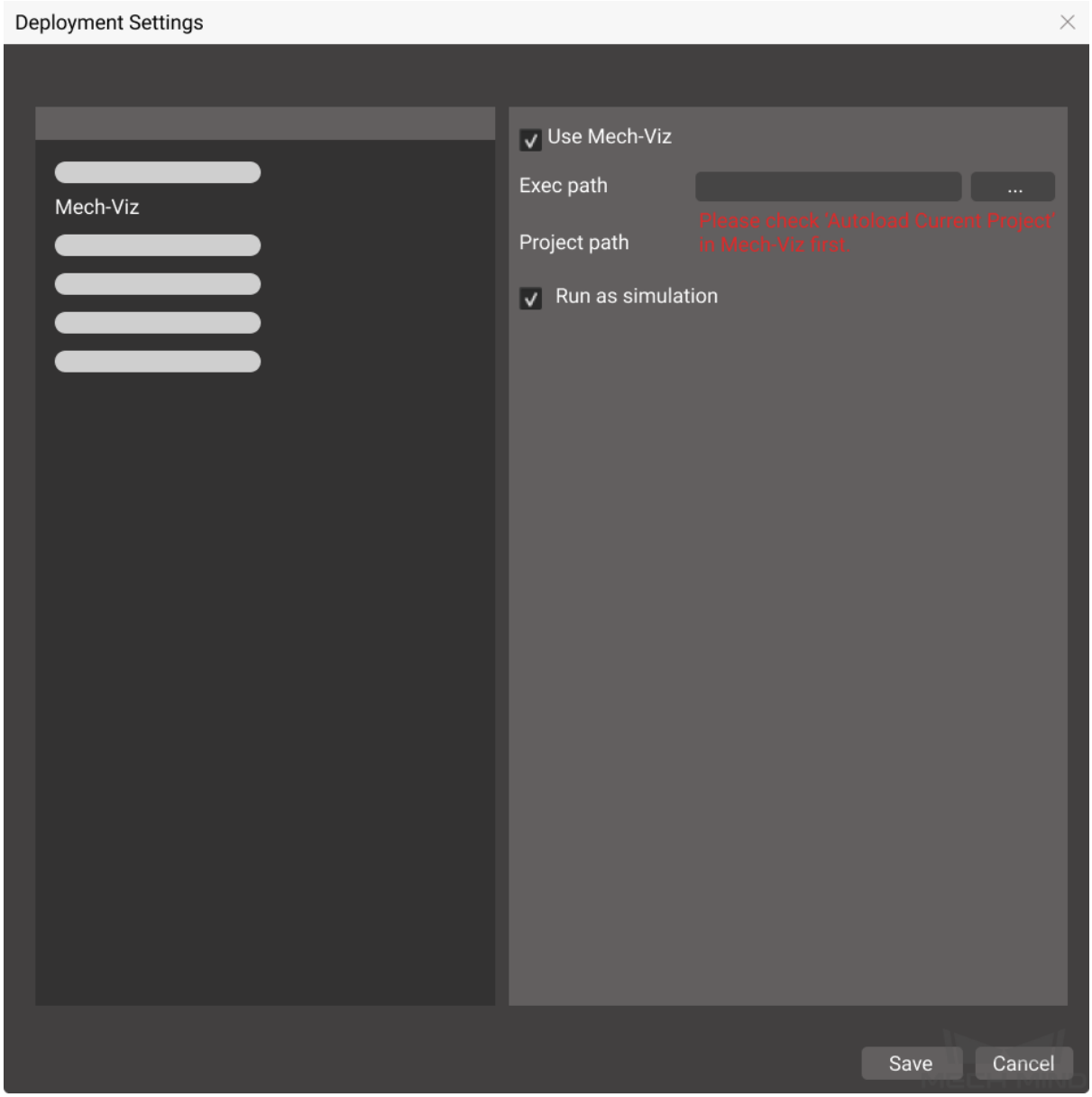



- If *Hide console when running* is checked, the console will be hid after opening Mech-Center.
- If *Run Mech-Center at PC startup* is checked, Mech-Center will be opened automatically after starting the PC.
- If *Open Mech-Viz/Mech-Vision/Mech-Interface automatically (if any)* is checked, Mech-Viz/Mech-Vision/Mech-Interface will be opened automatically after Mech-Center is opened.
- If *Minimize Mech-Viz/Mech-Vision to System Tray when opening* is checked, the opened Mech-Viz/Mech-Vision will be added in the system tray on the desktop.

**Attention:** Please save the changes and restart Mech-Center for the changes to take effect.

### Mech-Viz

The Mech-Viz option is used to set the open path of Mech-Viz software and display the path of autoloaded projects, as shown below.




Click on *Mech-Viz* on the left to start setting. *Use Mech-Viz* is checked by default.

Click on ··· next to the Exec path, and select the mmind_viz.exe file in the directory where Mech-Viz is installed to complete the path configuration.

Check *Autoload Current Project* in Mech-Viz, and the project path will be added automatically.

---

**Note:** If **Run as simulation** is checked, Mech-Viz will only simulate the project and will not guide

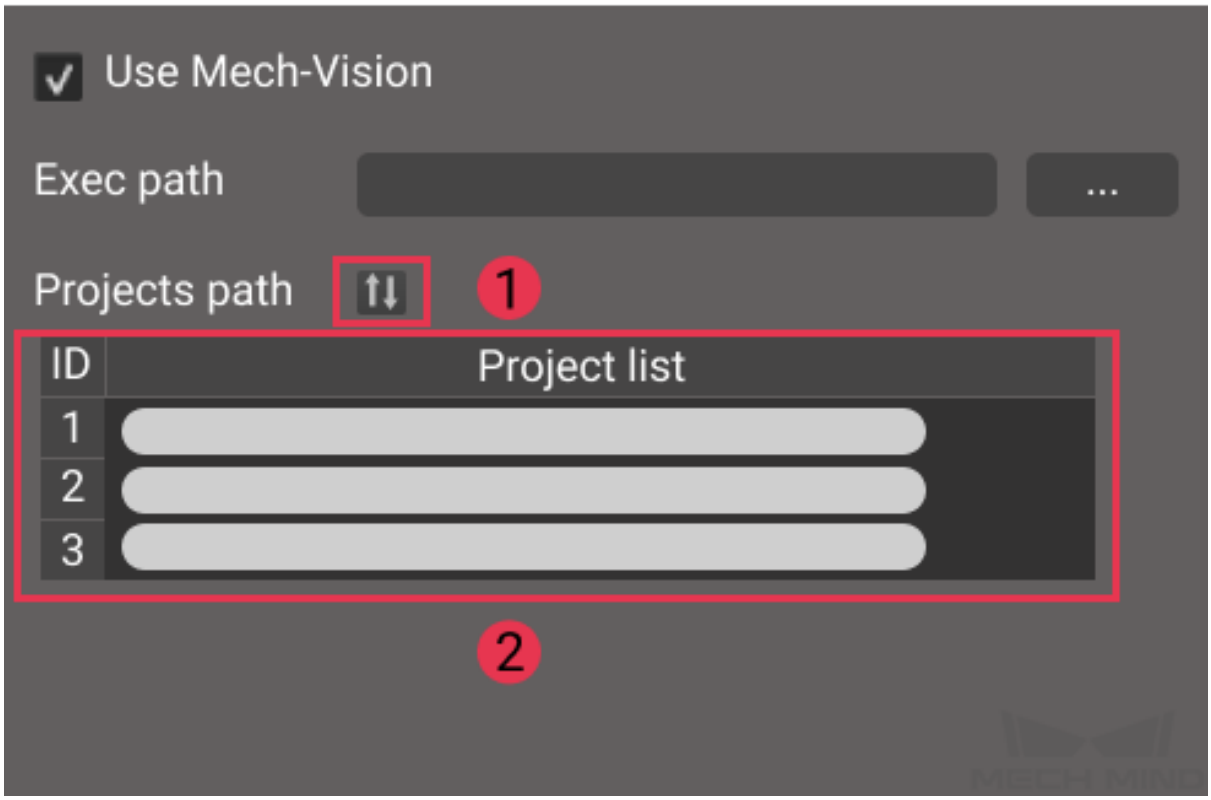
the real robot to move.

### Mech-Vision, Mech-Eye Viewer

The methods to configure Mech-Vision and Mech-Eye Viewer are similar to that of Mech-Viz.

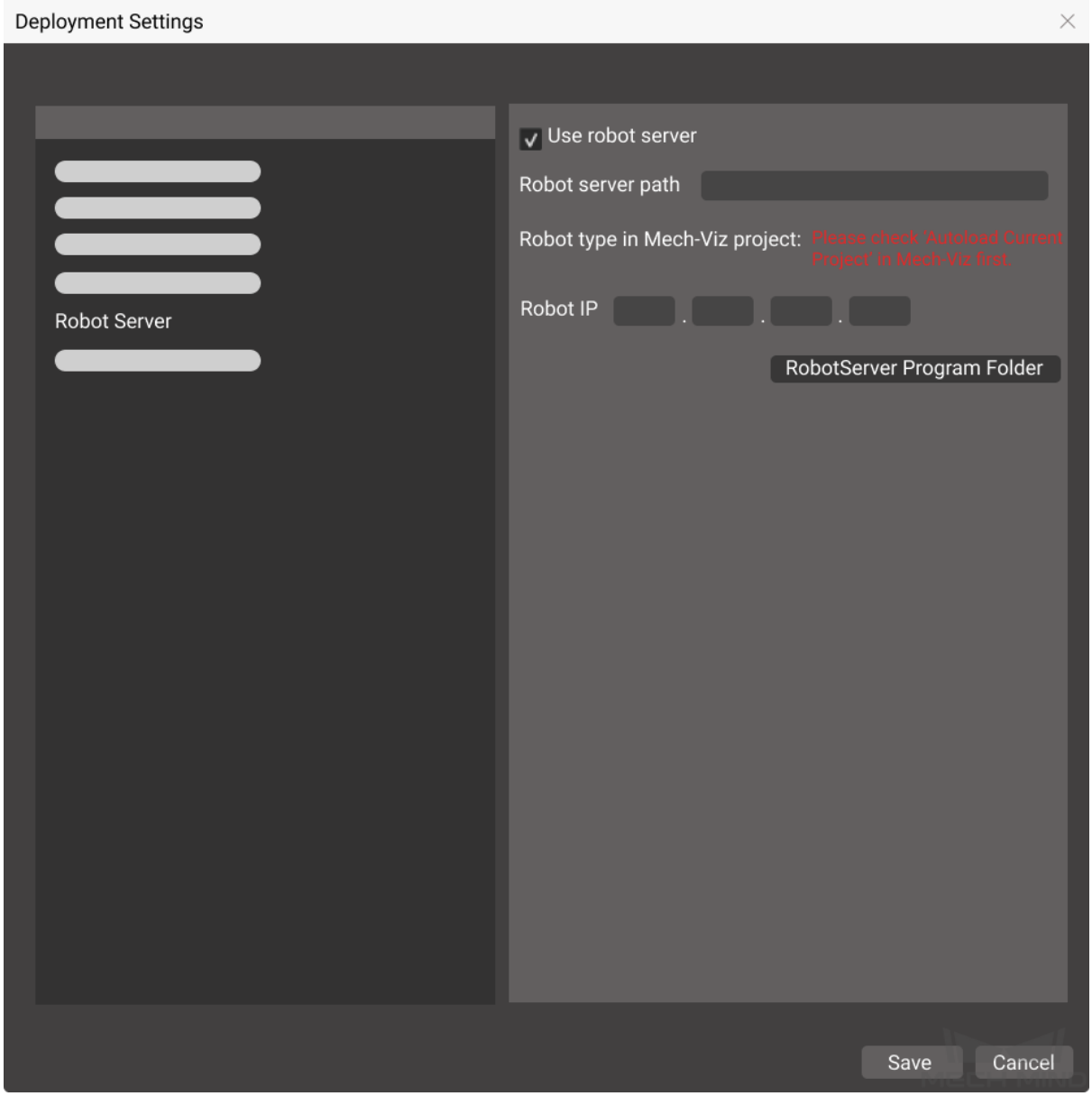
Please follow the instructions below to adjust the sequence of project path in the project list:

1. Open Mech-Vision first, select the project in the **Project List** and right-click with the mouse, and then check *Autoload Project* in the context menu.
2. Open Deployment Settings in Mech-Center, and click on  to synchronize the Mech-Vision project paths.
3. Press an hold the left mouse button on a project to drag up or down to adjust the sequence.



### Robot Server

The Robot Server option is used to adapt the robot to realize the full control of Mech-Viz software, as shown below.




Click on *Robot Server* to start setting. *Use robot server* is checked by default. The robot server files are in the directory where Mech-Center is installed, and the robot server path will be automatically added.

**Attention:** After loading the Mech-Viz project successfully, the robot type in Mech-Viz project will be filled automatically. Please make sure to enter the correct robot IP which is set in the actual project.

#### Mech-Interface

*Mech-Interface* is an unified external interface that realizes communication with third parties.

### 1.2.2 Start Viz/Vision

After opening Mech-Viz/Mech-Vision successfully, their icons will be displayed on the Service Status Bar.

**Attention:**

1. A version compatibility check will be performed when running a project. It is recommended to use Mech-Viz, Mech-Vision, and Mech-Center of versions 1.4.0 and above.
2. When Mech-Center detects that the version of either Mech-Viz or Mech-Vision is lower than 1.4.0, it will prompt that the version must be higher than 1.4.0, and an error message box will pop up after clicking *Run*.

### 1.2.3 Start Mech-Eye Viewer

After opening Mech-Eye Viewer successfully, its icon will be displayed on the Service Status Bar.

### 1.2.4 Run

Run the loaded projects in Mech-Viz and Mech-Vision.

### 1.2.5 Start Interface

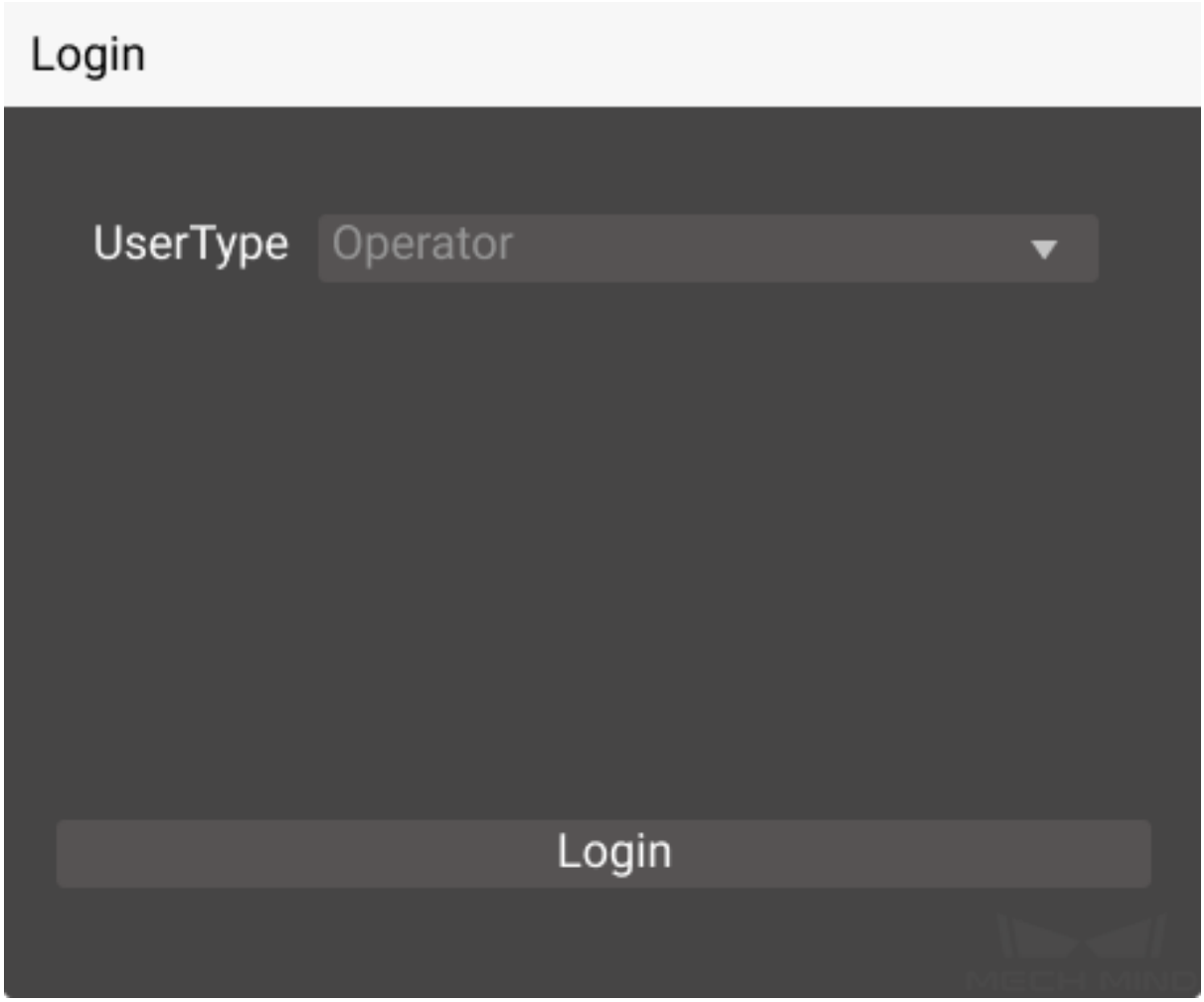
Used to start *Mech-Interface*.

### 1.2.6 Connect Robot

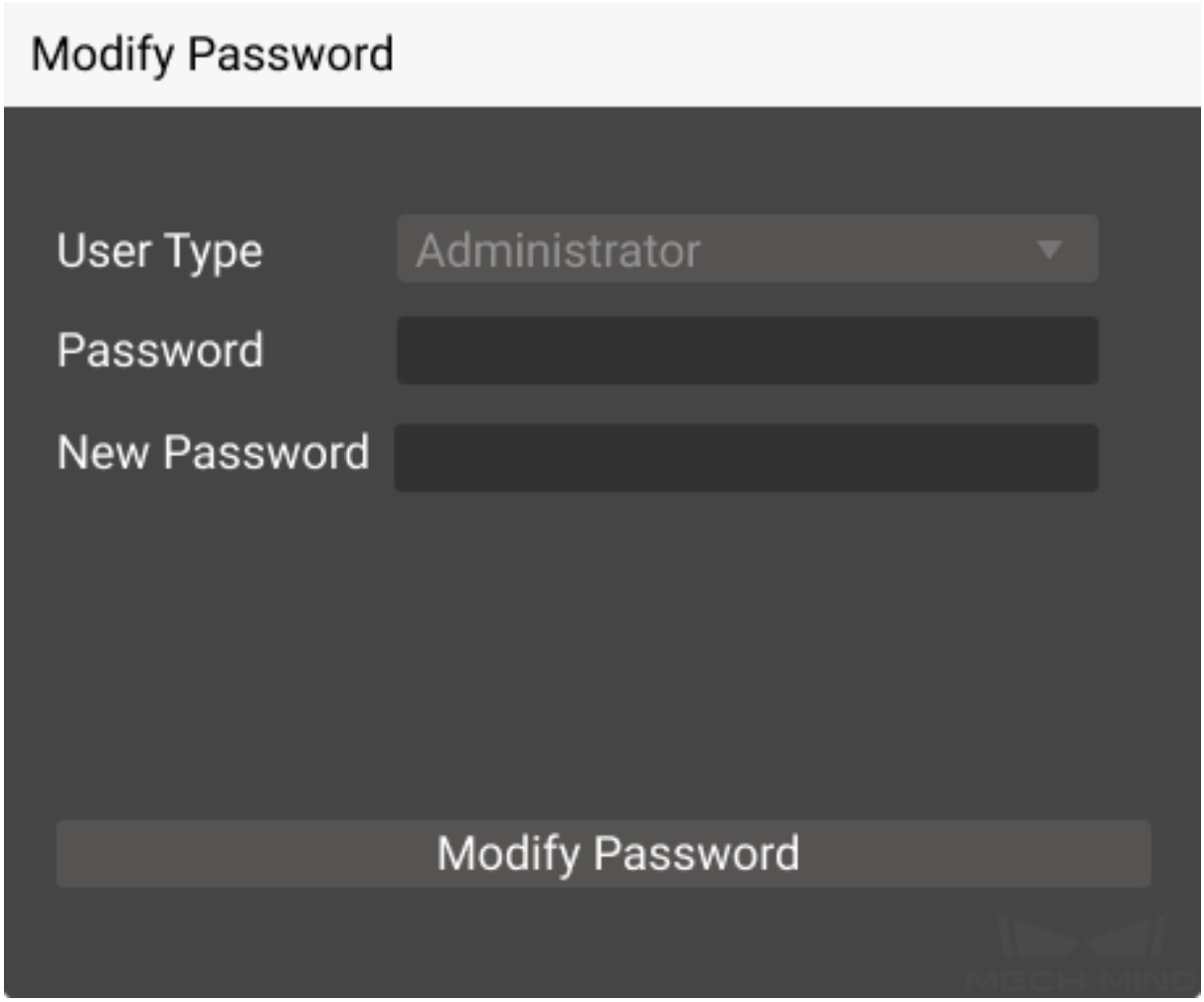
After connecting a real robot successfully, the robot icon will be displayed on the Service Status Bar.

### 1.2.7 Administrator

Mech-Center can be used in either Administrator mode or Operator mode, and the default one is Administrator mode. Operator mode does not enable to edit projects or adjust configurations, and the icon displayed on the toolbar will be the Operator. If you need to switch the mode, please click on the icon, select the user type and click on *Login* to confirm setting.

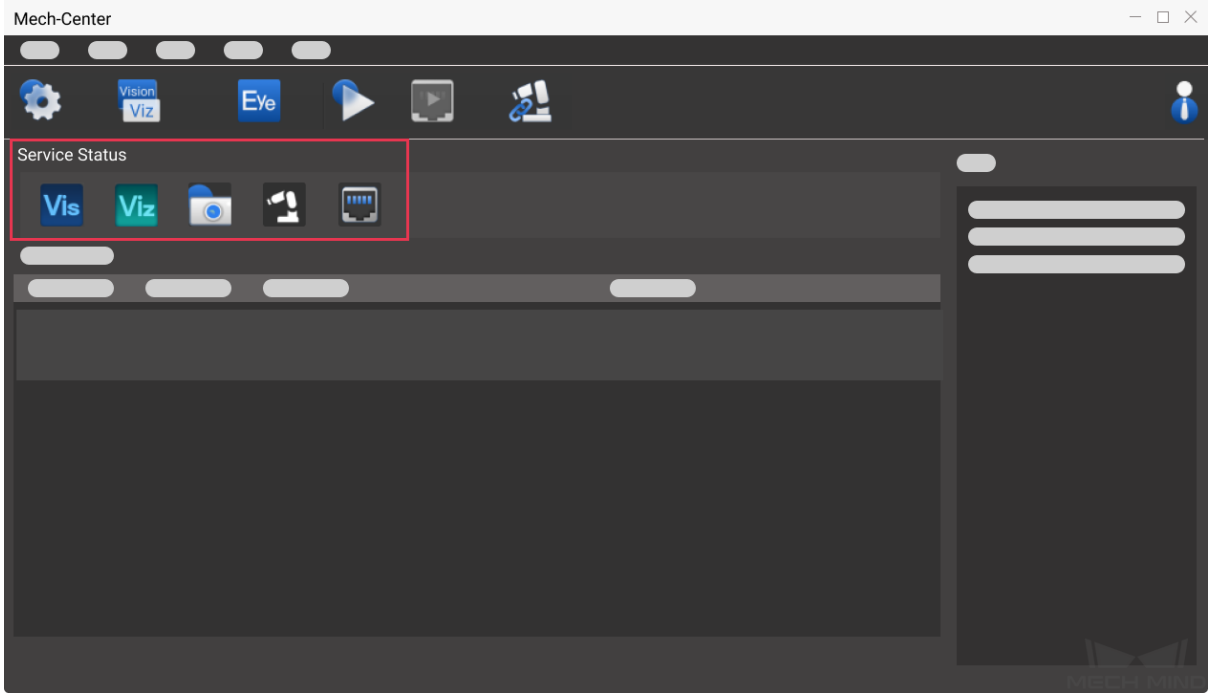



The Administrator mode requires a password to log in. If you need to modify the passward, please go to *User* → *Modify Password*.



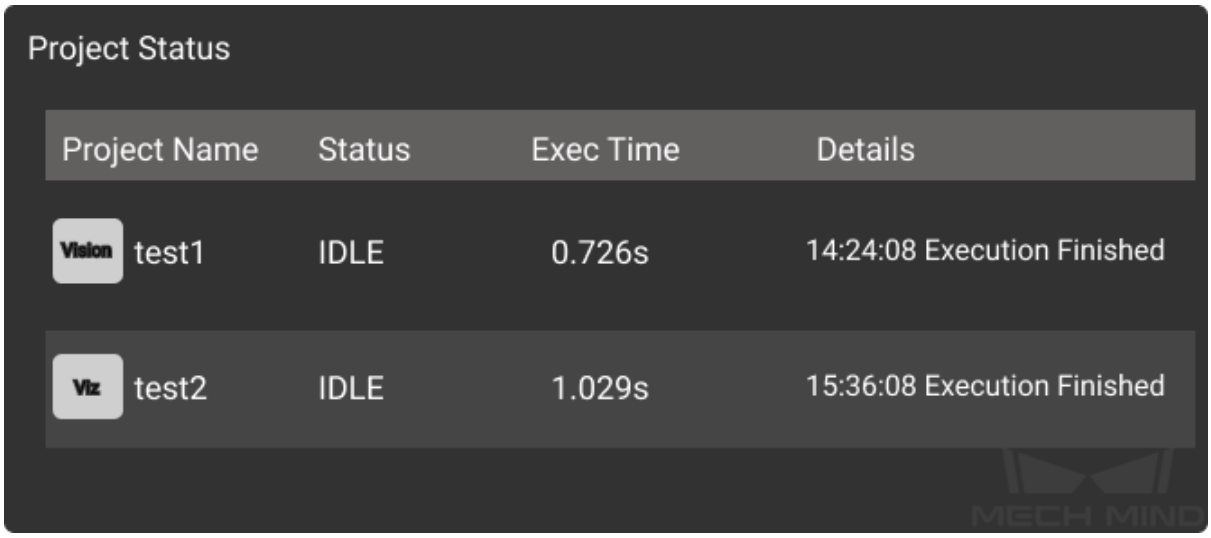

## 1.3 Service Status Bar

When Mech-Center is running, the Service Status Bar will display the icon of the software, camera, robot, and interface service which are activated. You can click on the icon to open corresponding window.




## 1.4 Project Status Bar

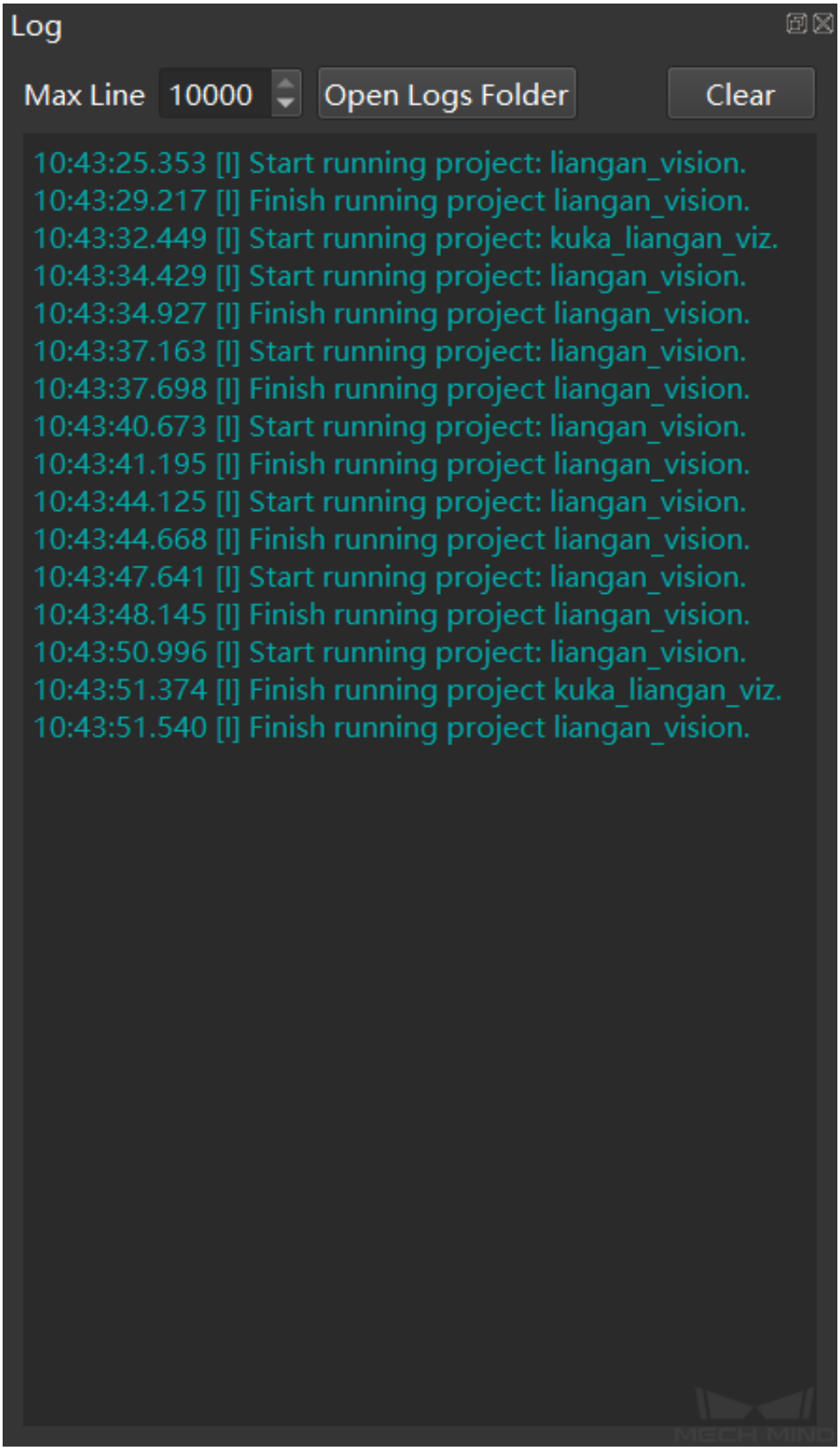
The Project Status Bar displays the project name, status, execution time, and details of Mech-Viz/Mech-Vision projects. You can learn the detailed project running status from the messages in it and the Log Panel on the right.



| Options | Description |
|---|---|
| Project Name | The name of Mech-Viz / Mech-Vision projects. |
| Status | Current statuses of the projects (IDLE or RUNNING). |
| Execution Time | The amount of time taken by a project to complete its execution. |
| Details | Record of the execution time and the corresponding status. An error message will be displayed if an error occurs. |

## 1.5 Log Panel

Display the log information of the current project and service in real time.


Log
Max Line
10000
Open Logs Folder
Clear
10:43:25.353 [I] Start running project: liangan_vision.
10:43:29.217 [I] Finish running project liangan_vision.
10:43:32.449 [I] Start running project: kuka_liangan_viz.
10:43:34.429 [I] Start running project: liangan_vision.
10:43:34.927 [I] Finish running project liangan_vision.
10:43:37.163 [I] Start running project: liangan_vision.
10:43:37.698 [I] Finish running project liangan_vision.
10:43:40.673 [I] Start running project: liangan_vision.
10:43:41.195 [I] Finish running project liangan_vision.
10:43:44.125 [I] Start running project: liangan_vision.
10:43:44.668 [I] Finish running project liangan_vision.
10:43:47.641 [I] Start running project: liangan_vision.
10:43:48.145 [I] Finish running project liangan_vision.
10:43:50.996 [I] Start running project: liangan_vision.
10:43:51.374 [I] Finish running project kuka_liangan_viz.
10:43:51.540 [I] Finish running project liangan_vision.

| Options | Description |
|---|---|
| Max Line | The maximum line displayed in the log panel (adjustable). |
| Open Logs Folder | Open the folder where the current log is saved. |
| Clear | Delete all logs. |

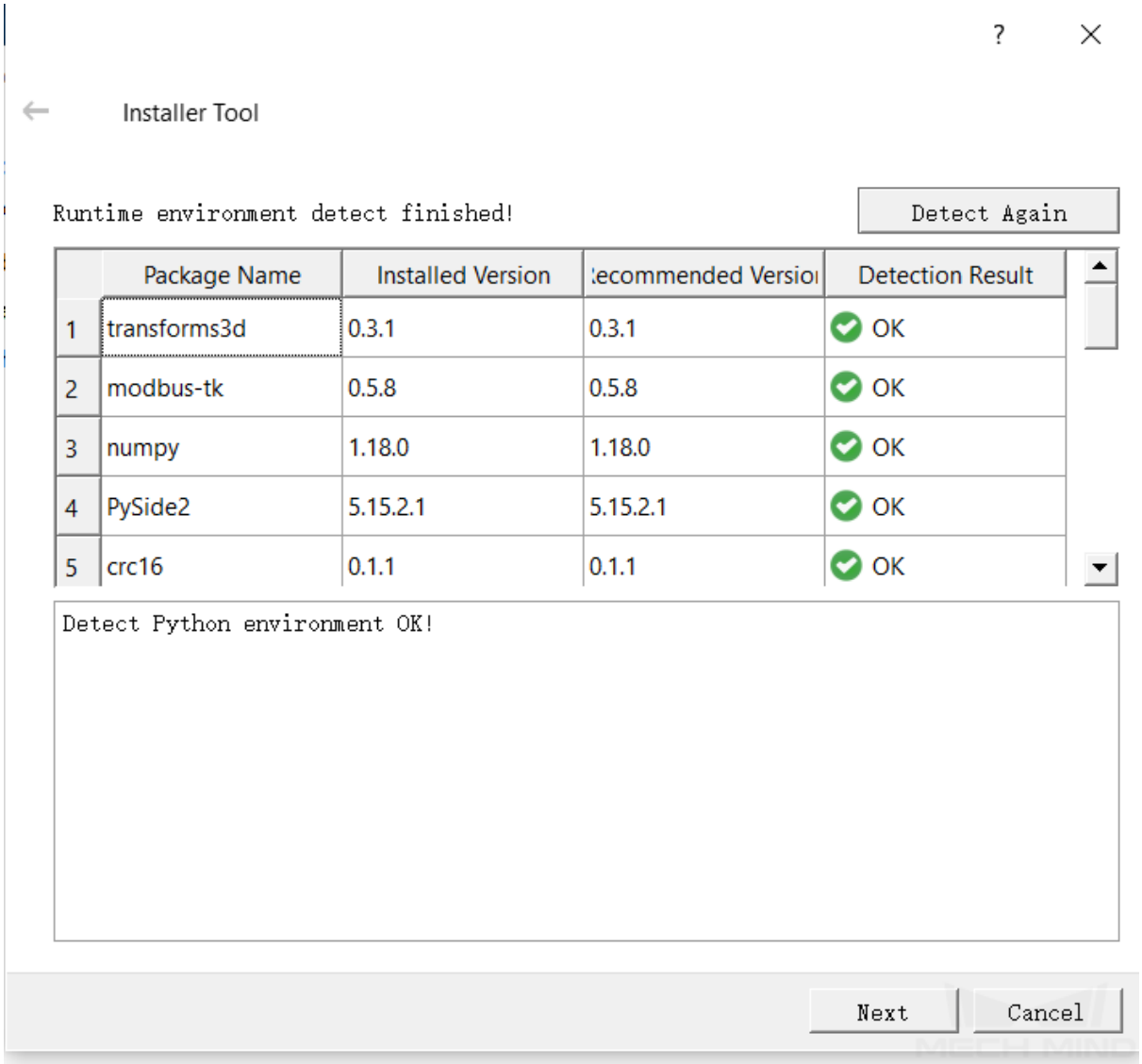
CHAPTER

TWO

# GETTING STARTED WITH MECH-CENTER

## 2.1 Install Mech-Center

1. Double-click on the Mech-Center .exe file to run the Setup Wizard, and then click on *Next*.
2. **Select an installation path** and then click on *Next*.
3. Choose whether to **keep custom settings** and **create desktop shortcuts** (usually checked by default) and then click on *Next*.
4. After installing Mech-Center, an environment check will be performed. Please download the missing components according to the notification. If the environment check is passed, click on *Next*. An message which reads "Integrated environment installed successfully" will be displayed, and then click on *Finish* to complete installation.



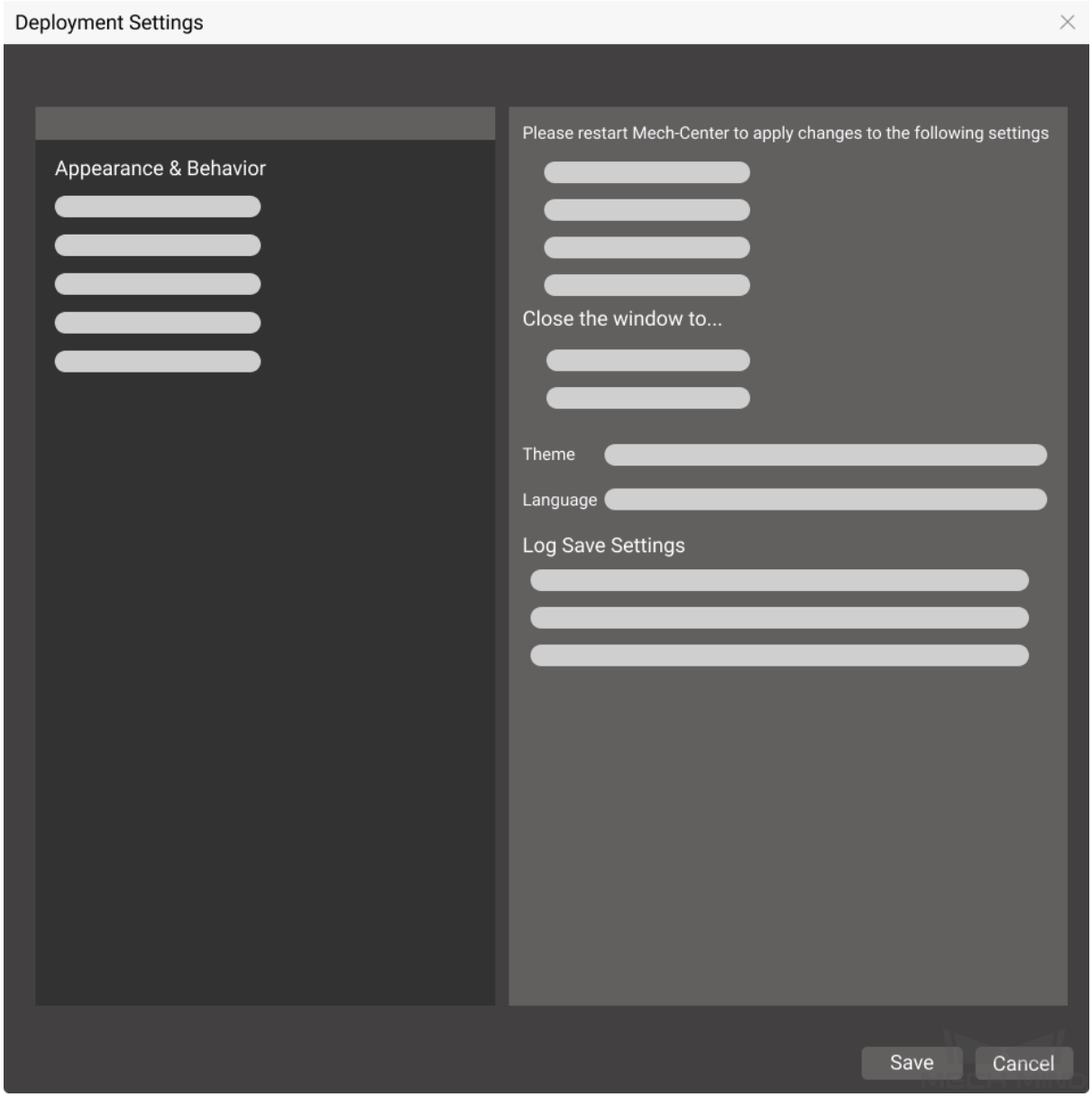

5. Check **Open Mech-Center** and click on *Done* will enable to run Mech-Center automatically after exit the Setup Wizard.

6. Now you have installed Mech-Center sucessfully. The icons  and  will be displayed on the desktop.

**Hint:** If you have any problems opening Mech-Center, please check the installation environment first. You need to double-click on  to enable **Runtime Environment Check Tool** to check if the environment meets all requirements.

## 2.2 Deployment Settings

Open Mech-Center and click on **Deployment Settings**  on the *Toolbar* to open the Deployment Settings window, as shown below. You can configure the path, parameter, and other settings in it. After configuration, please click on *Save*. For detailed instructions, please refer to *Deployment Settings*.



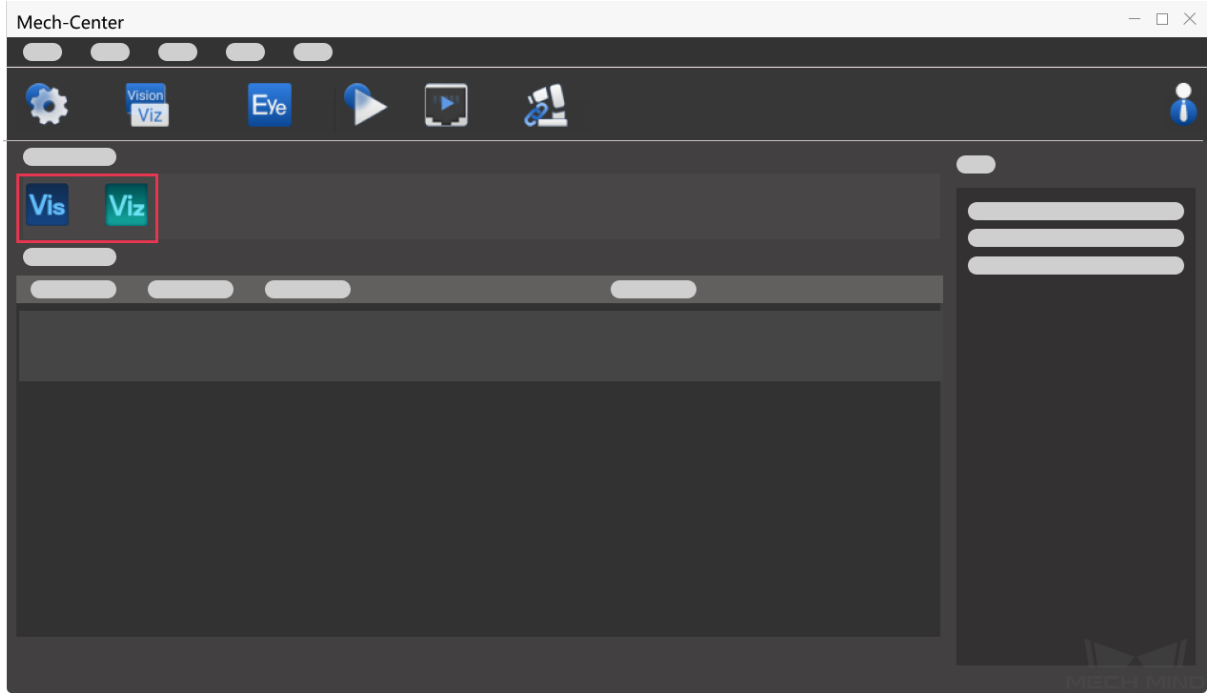

**Hint:**

- Please use Deployment Settings according to actual needs.

- If Mech-Vision, Mech-Viz, and Mech-Eye Viewer are installed successfully, their paths will be automatically added in Deployment Settings, and you do not need to add by yourself.

## 2.3 Open Projects

Click on **Start**  on the *Toolbar* to open Mech-Viz and Mech-Vision. Their icons will be displayed on the *Service Status Bar*.




### 2.3.1 Open Mech-Viz Project

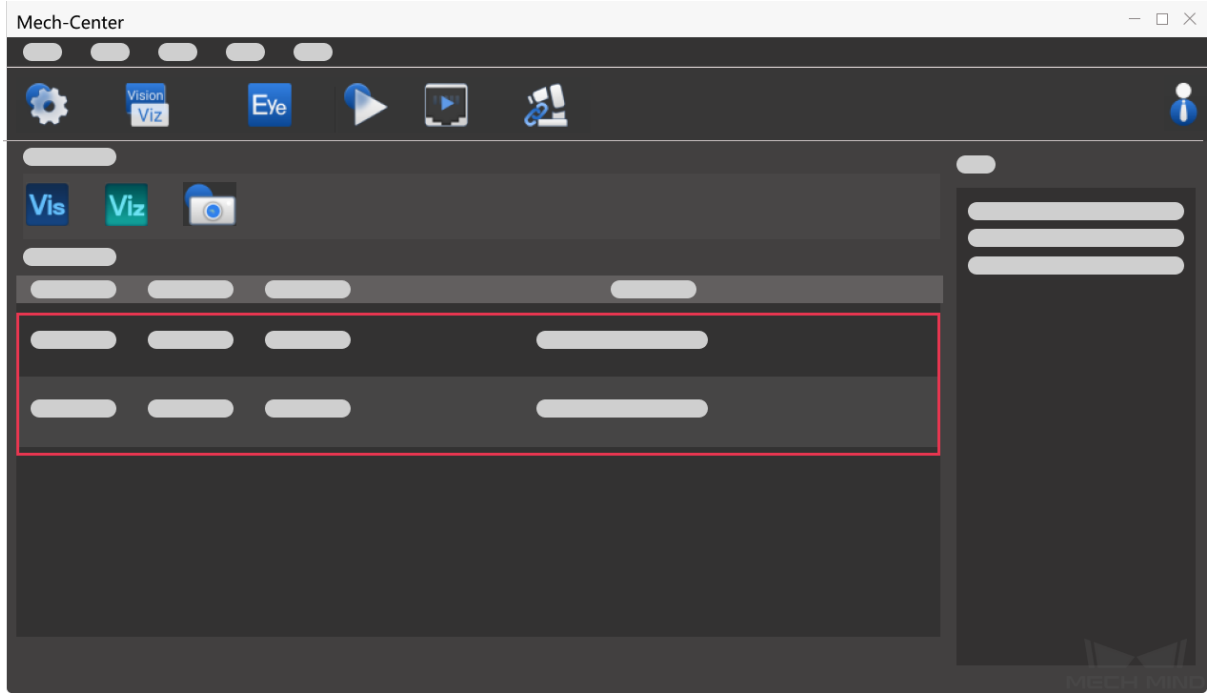
1. Click on  to enter the main interface of Mech-Viz.
2. Create a project: please refer to viz_first_project for detailed instructions. You can also open an existing project.
3. Check *Autoload Current Project* on the toolbar of Mech-Viz.
4. View project status: the project status will be displayed on the *Project Status Bar*.

**Hint:**

- After checking *Autoload Current Project* in Mech-Viz, the **Project path** will be automatically added.

### 2.3.2 Open Mech-Vision Project

1. Click on  to enter the main interface of Mech-Vision.
2. Create a project: please refer to typical_applications to create a project. You can also open an existing project.
3. Check *Autoload Project*: select the project in the **Project List** and right-click with the mouse, and then check *Autoload Project* in the context menu.
4. Add project path: go to *Deployment Settings* → *Mech-Vision*, click on  and then *Save* to add the project path.
5. View project status: the project status will be displayed on the *Project Status Bar*, as shown below.




### 2.3.3 Configure Camera Settings

If you need to check or configure settings of the camera, click on **Start Mech-Eye Viewer**  on the *Toolbar*. The icon will be displayed on the *Service Status Bar*.

## 2.4 Connect the Robot

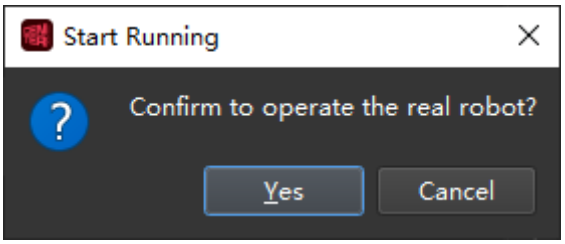
If the on-site robot you are using is not UR, you need to load the program files before connecting to Mech-Center. For detailed instructions, please see robot_integrations. Click on **Connect Robot** on the *Toolbar* to connect a real robot. If the connection is successful, a robot icon will be displayed on *Service Status Bar*.

For projects using Mech-Interface, please click on **Start Interface** on the *Toolbar*. After the interfce service is enabled, an icon will be displayed on *Service Status Bar*.

## 2.5 Run the Project

**Hint:** If you want to control a real robot, please go to *Deployment Settings* → *Mech-Viz* and uncheck *Run as simulation*.

Click on **Run**. If a window as shown below pops up, click on *Yes* to run the loaded project. Then you can view related information on the *Project Status Bar* while running the project.




**Attention:** When the real robot is working, please ensure the safety of personnel. When an emergency occurs, please press the emergency stop button on the robot teach pendant.

## 2.6 Example Mech-Viz Projects for Standard Interface

In the file location of Mech-Center, in the folder `tool\viz_project`, there are four example Mech-Viz projects for using Standard Interface.

**Check_collision**

For path planning and collision detection in vision-guided picking.

**Outer_move**

For the scenarios in which the robot needs to move to a pose passed in from an external client.

**Suction_zone**

For using multiple suction cup sections (or array grippers) through DO signals.

**Vision_result_reuse**

For the scenarios in which the vision result returned at a time need to be used multiple times for path planning and collision detection in vision-guided picking.

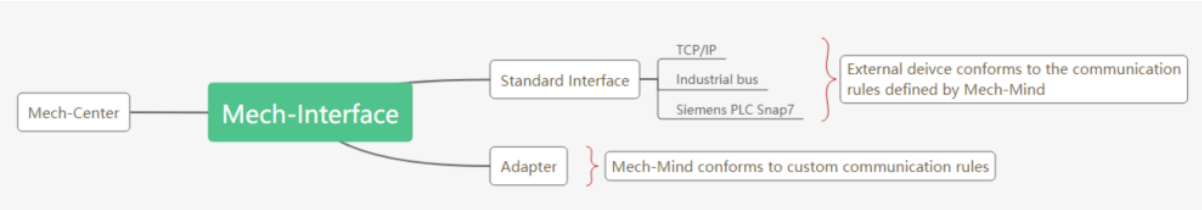
CHAPTER

THREE

# MECH-INTERFACE

Mech-Interface provides communication service with the outside for the Mech-Mind Software Suite. It transmits information between the outside (host computer, robot, etc.) and the Mech-Mind Software Suite.

Mech-Interface works in two ways: standardized **Standard Interface** and customized **Adapter**.




- When using Standard Interface, the external device needs to conform to the communication rules set by the Mech-Mind Software Suite.
- Adapter is the communication tool that conforms to customized rules.

If only the robot path planned by Mech-Viz or the vision points calculated by Mech-Vision need to be transmitted, please use **Standard Interface**, which is easy and fast but can only transmit limited data types.

## 3.1 Standard Interface

When only the pose points are needed and the Mech-Mind system is not needed to control the robot' s movement, the standard interface can be used to transmit the data. The standard interface only provides the most basic interface functions, such as sending vision points and task data. If more interface functions are required, please see *Adapter* for details. The standard interface is integrated in the Mech-Center software and no additional generation is required. The service can be started directly in the software.

### 3.1.1 Instructions

#### Start Mech-Interface:

Under *Deployment Settings* → *Mech-Interface*, check *Enable Mech-Interface*, and check the interface service type as *Standard Interface*, as shown in *Figure 1*.

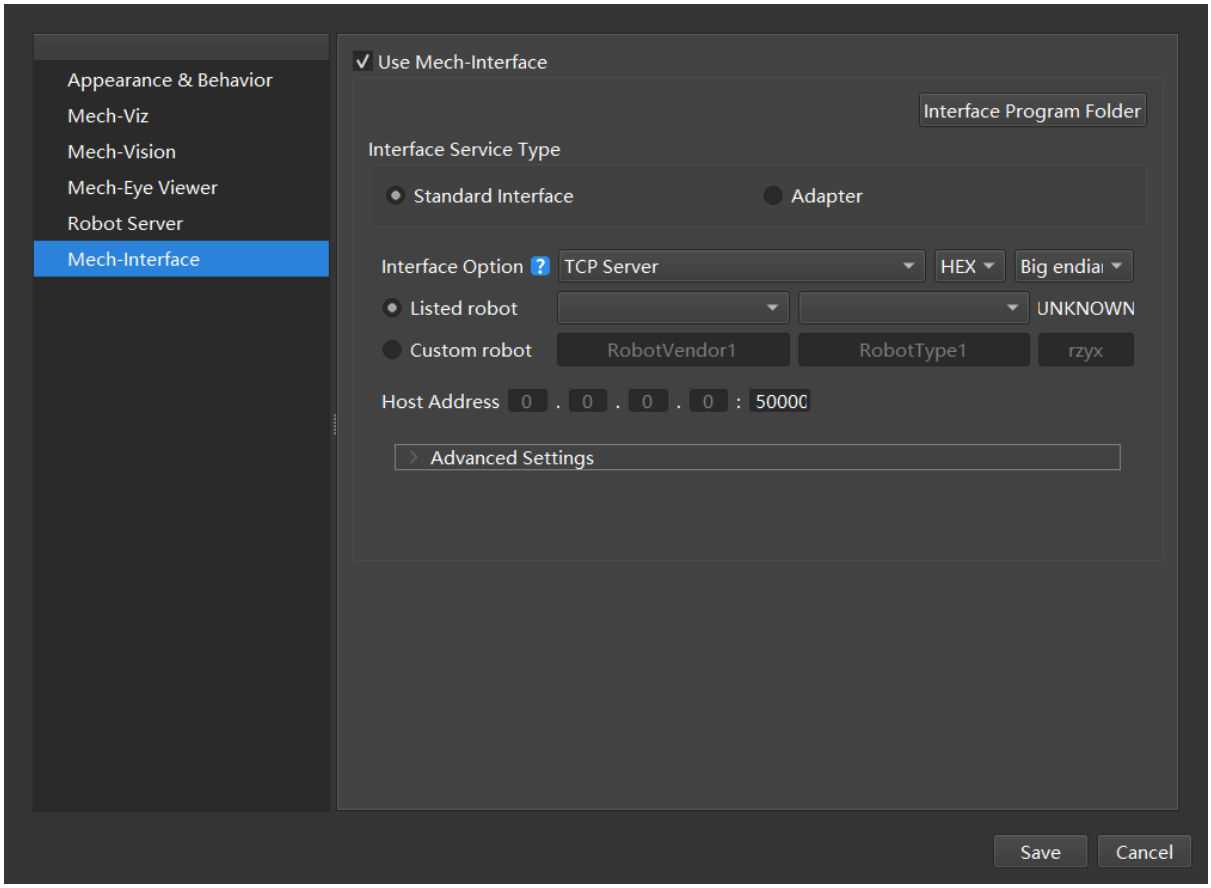



Figure 1. Start Mech-Interface

#### Interface Options, Host Address:

There are two types of external services: TCP Server and Siemens PLC Client. Please select as appropriate.

**Siemens PLC Client:** For Siemens PLC Client, the PLC IP and the DB Block Number need to be set. The default DB Block Number is 10.

TCP Server:

For TCP Server, the protocol format needs to be selected as ASCII or HEX. In HEX, plsese select Big Endian / Little Endian, i.e., “>” (big endian) or “<” (little endian). For TCP Server, the port number needs to be set as appropriate, and the default port number is 50000.

Please select the valid communication format to write the interface program based on the setup of the robot.

| Robot Brand | Sample Program Communication format |
|---|---|
| ABB | HEX |
| FANUC | HEX |
| KAWASAKI | ASCII |
| YASKAWA | ASCII |
| KUKA | HEX |
| Others | No sample program has been provided yet. Please write the interface program based on the robot's support for HEX and ASCII |

#### Select a robot:

Click  to select the corresponding robot brand and model.

After the setting is complete, save and restart Mech-Center. After restarting, click *Start Interface* on the interface to enable the standard interface.

## 3.2 Standard Interface Development Manual

### 3.2.1 Overview

- *Protocols*
- *Introduction to Commands*
- *Mech-Center Service Settings*

#### Protocols

#### TCP Server

Mech-Center provides a TCP Server (default port 50000) as an external service interface that supports the transfer of ASCII and HEX data.

#### Siemens PLC Client

To communicate with Siemens S7 series PLC, Mech-Center provides a PLC Client (default DB address: 10) based on Snap7 protocol as a communication interface.

### PROFINET

For communication using the PROFINET industrial bus, the conditions that need to be met include:

- The industrial computer or host supports the installation of standard PCI-e cards.
- HMS INpact 40 PIR card and Ixxat VCI driver software have been installed.
- Mech-Center 1.5 or above has been installed.
- Use the PROFINET General Station Description (GSD) file provided by Mech-Center.
- PROFINET communication is in the standard big-endian data format. The data contain 32-bit DINT pose data, and the PROFINET master station (especially the robot controller) needs to support 32-bit integer sending and receiving.

### EtherNet/IP

Mech-Center can be used as an EtherNet/IP slave station to connect to the EtherNet/IP industrial network.

To communicate using the EtherNet/IP industrial bus, the conditions include:

- The industrial computer or host supports the installation of standard PCI-e cards.
- HMS INpact 40 PIR card and Ixxat VCI driver software have been installed.
- Mech-Center 1.5 or above has been installed.
- Use the EtherNet/IP Station Description (GSD) file provided by Mech-Center.
- EtherNet/IP communication is in the standard big-endian data format. The data contain 32-bit DINT pose data, and the EtherNet/IP master station (especially the robot controller) needs to support 32-bit integer sending and receiving.

### Introduction to Commands

**Mech-Vision**

| Code & Command name | Function |
|---|---|
| 101: Start Mech-Vision Project | For scenarios using only Mech-Vision but not Mech-Viz. This command is for starting the running of the corresponding Mech-Vision project for image acquisition and vision data processing. |
| 102: Get Vision Results | For scenarios using only Mech-Vision but not Mech-Viz. This command is for reading the vision recognition results, i.e., target object pick points. |
| 103: Switch Mech-Vision Recipe | This command switches between the saved parameter recipes in Mech-Vision. Multiple parameter recipes are for recognizing different target objects. Parameters involved include recognition models, DL model files, etc. |

**Mech-Viz**

| Code & Command name | Function |
|---|---|
| 201: Start Mech-Viz Project | For scenarios using both Mech-vision and Mech-Viz. This command starts the Mech-Viz project, calls the corresponding Mech-Vision project, and plan the path for picking. |
| 202: Stop Mech-Viz Project | This command is for manually terminating the running of Mech-Viz. |
| 203: Select Mech-Viz Branch | This command is for controlling the branch_by_service_message Task (if there is one) in the Mech-Viz project to let the project run along the specified out port. |
| 204: Set Move Index | Set the index parameter of the **move** class Tasks in the Mech-Viz project. The "Move" class Skills that contain index parameters include move_list, move_grid. |
| 205: Get Planned Path | This command obtains the robot path planned by the Mech-Viz project. |
| 206: Get DO Signal List | This command gets the DO list, i.e., the list of array gripper (or suction cup section) control signals calculated by Mech-Viz, when using multiple suction cup sections for picking multiple objects at a time. |

**Others**

| Code & Command name | Function |
|---|---|
| 501: Input Object Dimensions to Mech-Vision | Input object dimensions, i.e., length, width, height of boxes, to set the 3D dimensions in the project when the Mech-Vision project has a Step that needs to read object dimensions from an external source. |
| 502: Input TCP to Mech-Viz | Set a dynamically changing target in the Mech-Viz project when there is an outer_move Task in the project. |
| 601: Notify | This command does not need to be initiated by the user. It will be executed when the Mech-Vision/Mech-Viz project raises a notification message by a **Notify** Step/Task. |
| 701: Calibration | For hand-eye calibration for cameras. This command obtains the calibration points and triggers the camera to take pictures, thus completing the calibration. The calibration points are from Mech-Vision. |
| 901: Get Software Status | Get the status of Mech-Mind Software Suite for checking the project status. |

**Note:** unified data unit is required for communication:

- The unit of joint angle and Euler angle is degree (°).
- The unit of XYZ coordinates in the flange pose (pose) is mm.

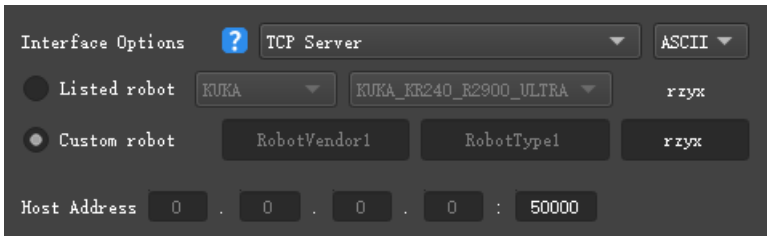
### Mech-Center Service Settings

#### TCP Server

The interface service is disabled by default. To enable, please click on *Deployment Settings* → *Mech-Interface* → *Use Mech-Interface* → *Standard Interface*.

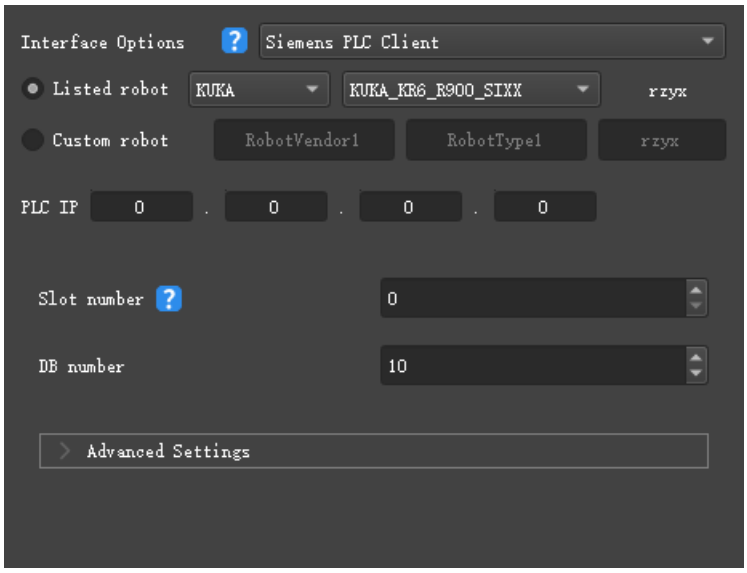
For using external services on the TCP server, please set the IP port according to your actual needs, the default port is 50000.

For TCP Server, please select the data type as ASCII or HEX. For HEX, please select between big endian and little endian.



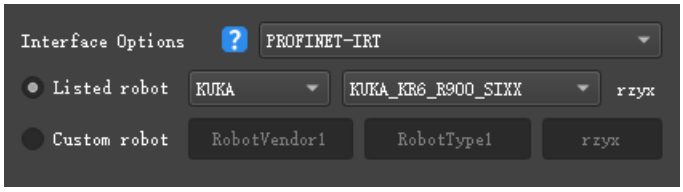

#### Siemens PLC Client

For Siemens PLC Client, please set the PLC IP, slot number, and DB number. The default slot number is 0, and the default DB number is 10.



### PROFINET

For PROFINET, the default board IP is 0.0.0.0, and the configuration can be made in the Siemens PLC programming software or **HMS IPconfig**.




### EthernNet/IP

For EtherNet/IP, the default board IP is 0.0.0.0, and the configuration can be made in **HMS IPconfig**.


### TCP/IP Socket Communication

#### Communication Settings

- Set the IP addresses of the robot and IPC to be in the same subnet. Command line `ping xxx.xxx.xxx.xxx` (the x' s are the fields for the IP address) can be used to test whether the network is connected.
- Open the **Deployment Settings** window in Mech-Center, select **Mech-Interface** in the left panel, and check **Enable Mech-Interface**.
- Select **Standard Interface** for **Interface Service Type**.
- For **Interface Options**, select **TCP Server**, and select **ASCII** or **HEX** according to what format the client program supports.
- Select the model of the robot that needs communication. The supported robot models are listed in the options. If the robot model used is not listed, please select **Custom robot** and set the Euler angle format of the robot.
- Set the port for TCP. Default: 50000.
- Advanced settings:
  - Set the max number of poses that can be transmitted each time. Default: 20. The reason is that the length of data that can be transmitted each time is limited.
  - Set the timeout period of receiving data from Mech-Viz. The default timeout period is 10 seconds. Mech-Viz needs some time for computing before outputting the data.
  - Set the timeout period of receiving data from Mech-Vision. The default timeout period is 10 seconds. Mech-Vision needs some time for computing before outputting the data.

- Click on *Save* to save the settings.

### 3.2.2 TCP/IP

Mech-Mind Software Suite can communicate with the following robots through TCP/IP:

- ABB
- YASKAWA
- FANUC
- KUKA
- Kawasaki

For setup instructions and other information specific to each robot, please refer to Robot Integrations - Standard Interface.

The commands are as follows:

- *Command 101: Start Mech-Vision Project*
- *Command 102: Get Vision Result*
- *Command 103: Switch Mech-Vision Recipe*
- *Command 201: Start Mech-Viz Project*
- *Command 202: Stop Mech-Viz Project*
- *Command 203: Select Mech-Viz Branch*
- *Command 204: Set Move Index*
- *Command 205: Get Planned Path*
- *Command 206: Get DO List*
- *Command 501: Input Object Dimensions to Mech-Vision*
- *Command 502: Input TCP to Mech-Viz*
- *Command 601: Notify*
- *Command 701: Calibration*
- *Command 901: Get Software Status*

#### Command 101: Start Mech-Vision Project

This command starts the running of the Mech-Vision project, which executes image capturing and performs vision recognition.

If the project works in the eye-in-hand mode, the robot pose for image capturing will be transmitted by this command into the project.

This command is for scenarios using only Mech-Vision.

### Command Sent

`101, project number, number of vision points, robot pose type, robot pose`

**Project number**

The integer ID number of the Mech-Vision project in Mech-Center, i.e., the number shown on the left of the project path in *Deployment Settings* → *Mech-Vision* in Mech-Center.

**Number of vision points**

The number of vision points (i.e., vision poses and their corresponding point clouds, labels, indices, etc.) to expect Mech-Vision to output.

`0`

Get all the vision points from the Mech-Vision project' s recognition results.

`integers > 0`

Get the specified number of vision points.

If the total number of vision points is smaller than the parameter value, all the available vision points will be returned.

If the total number of vision points is greater than or equal to the parameter value, vision points in the quantity of the parameter value will be returned.

---

**Note:** The command to obtain the vision points is command 102. In TCP/IP, due to the limit that a maximum of 20 vision points can be obtained by executing command 102 at a time, after executing command 102 for the first time, one of the parameters returned will indicate whether all the vision points requested have been returned; if not, please repeat executing command 102.

---

**Robot pose type**

This parameter indicates the type of the current pose of the real robot to input to Mech-Vision.

`0`

No robot pose needs to be transmitted by this command.

If the project works in the eye-to-hand mode, then image capturing has nothing to do with the robot' s pose, so no robot image capturing pose is needed by Mech-Vision.

`1`

The robot pose transmitted by this command is JPS.

`2`

The robot pose transmitted by this command is a flange pose.

**Robot pose**

This parameter is the robot pose needed when the project works in the eye-in-hand mode.

The robot pose is either JPS or flange pose, according to the setting of the parameter **robot pose type**.

### Data Returned

`101, status code`

**Status code**

If there is no error, status code 1102 will be returned. Otherwise, the corresponding error code will be returned.

### Test Samples

Command 101 is normally executed without error.

```
TCP send string = 101, 1, 10, 1, 0, -20.63239, -107.81205, 0, -92.81818, 0.00307
TCP received string = 101, 1102
```

Error: project ID number does not exist.

```
TCP send string = 101, 2, 10, 1, 0, -20.63239, -107.81205, 0, -92.81818, 0.00307
TCP received string = 101, 1011, 1
```

## Command 102: Get Vision Result

This command gets the vision result, i.e., vision points, after executing command 101.

**Note:** In TCP/IP, by default, command 102 can only fetch at most 20 vision points at a time. So, command 102 may need to be repeatedly executed until all the vision points required are obtained.

### Command Sent

`102, project number`

**Project number**

The integer ID number of the Mech-Vision project in Mech-Center, i.e., the number shown on the left of the project path in *Deployment Settings* → *Mech-Vision* in Mech-Center.

### Data Returned

`102, status code, sending completion status, number of vision points, reserved field, vision point, vision point, ...`

**Note:** The vision points (up to 20 vision points by default) are located at the tail of the data returned.

**Status code**

If there is no error, status code 1100 will be returned. Otherwise, the corresponding error code will be returned.

After executing this command, if the results from Mech-Vision have not been returned, Mech-Center will wait before sending the results to the robot. The default wait time is 10 seconds. If a timeout occurs, the timeout error status code will be returned.

**Sending completion status**

This parameter indicates whether all the vision points requested have been obtained.

`0`

Not all the vision points requested have been obtained. Please repeat executing command 102 until this parameter turns 1.

`1`

All the vision points requested have been obtained.

---

**Note:** If not all the vision points requested have been obtained and command 101 is executed at this time, the rest of the vision points that are not obtained will be cleared.

---

**Number of vision points**

The number of vision points returned from the Mech-Vision project by executing this command this time.

**Reserved field**

This field is not used.

The value defaults to 0.

**Vision point**

`pose, label, velocity`

**Pose**

A pose includes the Cartesian coordinates (XYZ) and Euler angles (ABC).

**Label**

The integer label assigned to the pose. If in the Mech-Vision project, the labels are strings, they need to be mapped to integers before outputting from the Mech-Vision project. If there are no labels in the Mech-Vision project, the label defaults to 0.

**Velocity**

The parameter defaults to 0 for command 102 because Mech-Vision does not provide the planning on velocity.

## Test Samples

### Test samples of normal execution and execution error

Command 102 is normally executed without errors.

```
TCP send string = 102, 1
TCP received string = 102, 1100, 1, 1, 0, 95.7806085592122, 644.5677779910724, 401.
↪1013614123109, 91.12068316085427, -171.13014981284968, 180.0, 0, 0
```

Error: no vision results.

```
TCP send string = 102, 1
TCP received string = 102, 1002, 1
```

### Test Sample of requesting vision points

The test sample below is obtaining 22 vision points by sending commands 101, 102, and 102 sequentially. Details are as follows:

- TCP/IP sends command 101, with content `101, 1, 0, 1, ⋯`, expecting to obtain all the available vision points.
- TCP/IP sends command 102 to obtain the vision results.
- TCP/IP receives the data returned by executing command 102. The content is `102, 1100, 0, 20, ⋯`, indicating 20 vision points have been obtained and not all vision points have been obtained.
- TCP/IP sends command 102 again to fetch the remaining vision points.
- TCP/IP receives the data returned by executing command 102 again. The content is `102, 1100, 1, 2, ...`, which includes 2 vision points and indicates all the vision points have been obtained.

```
TCP send string = 101, 1, 0, 1, -0, -20.63239, -107.81205, -0, -92.81818, 0.0016
TCP received string = 101, 1102
TCP send string = 102, 1

TCP received string = 102, 1100, 0, 20, 0, 95.7806085592122, 644.5677779910724, 401.
↪1013614123108, 31.12068316085427, ...
TCP received string = 78549940546, -179.99999999999991.0.0, 329.228345202334.712.7061697180302.
↪400.9702665047771, ...
TCP received string =39546, -83.62567351596952, -170.87955974536686, -179.99999999999937, 0, 0,
↪ 223.37118373658322, ...
TCP received string = 005627, 710.1004355953408, 400.82227273918835, -43.89328326393665, -171.
↪30845207792612, ...
TCP received string = 20.86318821742358, 838.7634193547805, 400.79807564314797, -102.
↪03947940869523, -171.149261231 ...
TCP received string = 390299920645, -179.99999999999994, 0, 0, 303.0722145720921, 785.
↪3254917220695, 400.75827437080, ...
TCP received string = 99668287.77.78291612041707, -171.53941633937786, 179.99999999899997, 0.0,
↪ 171.47819668864432, ...
TCP received string = 332193785, 400.6472716208158, -94.3418019038759, -171.10001228964776, -
↪179.39999999999994, ...
TCP received string = 92388542936, 807.5641001485708, 400.6021999602664, - 167.9834797197932.-
↪171.39671274951826, ...
TCP received string = 278.3198007132188, 780.5325992145735, 400.4924381003066, -174.
↪72728396633053, -171.422604771 ...
TCP recelved string = 3.99999999999994, 0, 0, 183.82195326381233, 862.5171519967056.400.
↪422966515846.-154. 17801945 ...
```

(continues on next page)

(continued from previous page)

```
TCP received string = 173.34301974982765, -180.0, 0, 0


TCP send string = 102, 1

TCP received string = 102, 1100, 1, 2, 0, 315.2017788478321, 592.1261793743445, 399.
↪60526335590957, 126.19602189220371, ...
TCP received string = 686127, -171.44430002882129, -1.3381805753922965e-15, 0, 0
```

### Command 103: Switch Mech-Vision Recipe

This command switches the parameter recipe used in Mech-Vision.

In Mech-Vision, what parameter settings a Step has can be modified by switching the parameter recipe.

Parameters involved in recipe switching usually include point cloud matching model, image matching template, ROI, confidence threshold, etc.

This command needs to be used before executing command 101 which starts the Mech-Vision project.

#### Command Sent

`103, project number, recipe number`

**Project number**

The integer ID number of the Mech-Vision project in Mech-Center, i.e., the number shown on the left of the project path in *Deployment Settings* → *Mech-Vision* in Mech-Center.

**Recipe number**

The identification number of the parameter recipe to switch to, i.e., the number on the left of the parameter recipe name in *Project Assistance* → *Parameter Recipe* → *Parameter Recipe Editor* in Mech-Vision.

#### Data Returned

`103, status code`

Status code

If there is no error, status code 1107 will be returned. Otherwise, the corresponding error code will be returned.

### Test Samples

Command 103 is executed normally without errors.

```
TCP send string = 103, 1, 2
TCP received string = 103, 1107
```

Error: invalid recipe number.

```
TCP send string = 103, 1, 2
TCP received string = 103, 1102
```

### Command 201: Start Mech-Viz Project

This command is for scenarios using both Mech-Vision and Mech-Viz.

This command starts the running of the Mech-Viz project, calls the corresponding Mech-Vision project, and lets the Mech-Viz project plan the robot path based on the vision points from Mech-Vision.

For the Mech-Viz project that needs starting, the option *Autoload* needs to be checked in Mech-Viz' s interface.

Please see *Example Mech-Viz Projects for Standard Interface* for the description of example Mech-Viz projects.

### Command Sent

`201, pose type, robot pose`

**Pose type**

`0`

The current pose of the robot is not needed by Mech-Viz and no pose will be sent.

If the project works in the eye-to-hand mode, no robot image capturing pose will be needed by the project.

In Mech-Viz, the simulated robot will move from the initial pose JPS = [0, 0, 0, 0, 0, 0] to the first target point in the planned path.

`1`

The robot pose will be sent to Mech-Viz and the pose sent is in JPS.

In Mech-Viz, the simulated robot will move from the input initial pose (i.e., the pose sent by this command) to the first target point in the planned path.

TCP is not supported at present.

**Note:** If in the scene, there are barriers that stand in the way from the initial pose JPS = [0, 0, 0, 0, 0, 0] to the first target point in the planned path, the pose type must be set to 1.

**Robot pose**

The current JPS of the real robot (if pose type is set to 1).

### Data Returned

`201, status code`

**Status code**

If there is no error, status code 2103 will be returned. Otherwise, the corresponding error code will be returned.

### Test Samples

Command 201 is normally executed without errors.

```
TCP send string = 201, 1, 0, -20.63239, -107.81205, 0, -92.81818, 0.00307
TCP received string = 201, 2103
```

Error: Mech-Viz does not support robot pose in TCP.

```
TCP send string = 201, 2, -0, 682.70355, 665.22266, 90, 179.99785, -89.99693
TCP received string = 201, 2015, 1
```

## Command 202: Stop Mech-Viz Project

Stop the running of the Mech-Viz project. This command is not needed when the Mech-Viz project does not fall into an infinite loop or can be stopped normally.

### Command Sent

202

### Data Returned

`202, status code`

**Status code**

If there is no error, status code 2104 will be returned. Otherwise, the corresponding error code will be returned.

### Test Sample

Command 202 is normally executed.

```
TCP send string = 202
TCP received string = 202, 2104
```

### Command 203: Select Mech-Viz Branch

This command specifies which branch the project should run along. For this command, the branching is implemented by a branch_by_service_message Task, and this command selects the branch by specifying an out port of the Task.

Before executing this command, the Mech-Viz project needs to be started by executing command 201.

When the Mech-Viz project runs to the branch_by_service_message Task, it will wait for command 203 to specify which out port of the Task, i.e., the branch, the project should run along.

#### Command Sent

`203, branching Task name, out port number`

**Branching Task name**

This parameter is for specifying which branch_by_service_message Task the branch selection should apply to.

The value should be an integer ([1, N])., and before running the project, the Task involved in this command should be named as an integer ([1, N]). The name should be unique in the project.

**Out port number**

This parameter is for specifying which out port of the specified Task, i.e., the branch, the project should run along. The value should be an integer ([1, N]).

---

**Note:** Out port number is the 1-based index of the specified out port on the Task. For example, if the specified out port is the second out port of the Task from left to right, the out port number is 2.

---

#### Data Returned

`203, status code`

**Status code**

If there is no error, status code 2105 will be returned. Otherwise, the corresponding error code will be returned.

#### Test Samples

Command 203 is executed normally without errors.

```
TOP send string = 203, 1, 1
TCP received string = 203, 2105
```

Error: invalid out port number.

```
TCP send string = 203, 1, 3
TCP received string = 203, 2018, 1
```

### Command 204: Set Move Index

This command is for setting the index parameter of a Task that involves sequential or separate motions or operations.

Tasks with index parameters include move_list, move_grid, custom_pallet_pattern, smart_pallet_pattern, etc.

Before executing this command, command 201 needs to be executed to start the Mech-Viz project.

#### Command Sent

`204, Task name, index value`

**Task name**

This parameter specifies which Task the index setting should apply to.

The value should be an integer ([1, N]), and the Task that needs index parameter setting by this command should be named as an integer ([1, N]). The name should be unique in the project.

**Index value**

The index parameter of the specified Task will be set to this value.

#### Data Returned

`204, status code`

**Status code**

If there is no error, status code 2106 will be returned. Otherwise, the corresponding error code will be returned.

#### Test Samples

Command 204 is executed normally without errors.

```
TCP send string = 204, 2, 6
TCP received string = 204, 2106
```

Error: Failed to set the index.

```
TCP send string = 204, 3, 6
TCP received string = 204, 2028, 1
```

### Command 205: Get Planned Path

This command gets the robot motion path planned by Mech-Viz after command 201 is executed to start the Mech-Viz project.

**Note:** InTCP/IP, by default, command 205 can only fetch at most 20 target points of the planned path at a time. So, command 205 may need to be executed repeatedly until all the target points required are obtained.

**Note:** If one of the target points in the path is not supposed to be sent to the robot, please rename the corresponding move Task by adding "_internal" to the end of the name (with an underscore; case insensitive).

#### Command Sent

```
205, target point type
```

**Target point type**

This parameter specifies the type of path target points to return from Mech-Viz.

`1`

The target points returned should be in JPS.

`2`

The target points returned should be in TCP.

#### Data Returned

```
205, status code, sending completion status, number of points, position of
"visual_move", target point, target point, ...
```

**Status code**

If there is no error, status code 2100 will be returned. Otherwise, the corresponding error code will be returned.

**Note:** When executing this command, if Mech-Viz has not yet had the planned robot motion path (the project is still running), Mech-Center will wait. The default wait time is 10 seconds. If a timeout occurs, a timeout error code will be returned.

**Sending completion status**

`0`

Not all the target points of the planned path have been obtained. Please repeat executing this command until this parameter' s value is 1.

`1`

All the target points of the planned path have been obtained.

**Note:** If the expected number of target points to transmit is greater than 20 (20 is the default setting), please execute command 205 multiple times until the returned value of this parameter is 1.

**Number of points**

This parameter indicates the number of path target points ([pose, label, velocity]) sent by executing this command this time.

Range: 0 to 20.

**Position of "visual_move"**

The position of the visual_move Task, i.e., the move to the vision pose (usually the pose for picking the object) in the entire robot motion path.

For example, if the path is composed of Tasks **move_1**, **move_2**, **visual_move**, **move_3** sequentially, the position of **visual_move** is 3.

If in the path there is no visual_move Task, the returned value will be 0.

**Target point**

[pose, label, velocity]

**Pose**

Cartesian coordinates (XYZ) and Euler angles (ABC), or JPS, according to the pose type set by command 205.

**Label**

Label is the integer label assigned to the pose. If in the Mech-Vision project, the labels are strings, they need to be mapped to integers before outputting from the Mech-Vision project. If there are no labels in the Mech-Vision project, the label defaults to 0.

**Velocity**

The non-zero velocity parameter percentage value for the move Task set in Mech-Viz.

### Test Samples

Command 205 is normally executed without error.

```
TCP send string = 205, 1

TCP received string =205, 2100, 1. 2, 2, 8.307755332057372, 15.163476541700463, -142.
↪1778810972881, -2.7756047848536745, -31.44046012182799, -96.94907235126934, 0, 64, 8.2␣
↪42574265592342, 12.130080796661591, -141.75872288706663-2.513533225987894, -34.8905853␣
↪039525, -97.19108378871277, 0, 32
```

Error: Mech-Vision runtime error.

```
TCP send string = 205, 1
TCP received string = 205, 2008, 1
```

### Command 206: Get DO List

This command gets the planned DO signal list when there are multiple grippers, such as suction cup sections, to control.

For using this command:

1. The Mech-Viz project’s name must be set to “suction_zone”.
2. The set_do_list Task must be named to “set_do_list_1”.
3. The set_do_list Task’s parameter “Get DO List from VisualMove” must be set to True.
4. The set_do_list Task must immediately follow a “visual_move” Task.
5. The name of the visual_move Task followed by the set_do_list Task must be selected in the lower part of the parameter panel of set_do_list.

Before calling this command, command 205 needs to be executed to obtain the planned motion path by Mech-Viz.

Please deploy the Mech-Viz project based on the template project at */Mech-Center/tool/viz_project/suction_zone*, and set the suction cup configuration file in the Mech-Viz project.

#### Command Sent

206

No parameters.

#### Data Returned

`206, status code, DO signal value, DO signal value, ..., DO signal value`

**Status code**

If there are no errors, status code 2102 will be returned. Otherwise, the corresponding error code will be returned.

**DO signal value**

There are 64 DO signal values, in integers, located at the tail of the data returned.

Range of valid DO values: [0, 999]. Placeholder value: -1.

### Test Samples

Command 206 is normally executed. The obtained DO signal values: 11, 12.

```
TCP send string: 206
TCP receive string: 206, 2102, 11, 12, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,␣
↪-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
↪1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
↪1, -1
```

Error: Mech-Viz did not provide a valid DO signal list.

```
TCP send string: 206
TCP receive string: 206, 2011, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,␣
↪-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
↪1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
↪1, -1
```

### Command 501: Input Object Dimensions to Mech-Vision

This command is for dynamically inputting object dimensions into the Mech-Vision project.

Please confirm the actual object dimensions before running the Mech-Vision project.

The Mech-Vision project should have the read_object_dimensions Step, and the Step' s parameter **Read Object Dimensions from Parameters** should be set to `True`.

### Command Sent

`501, project number, length, height, width`

**Project number**

The integer ID number of the Mech-Vision project in Mech-Center, i.e., the number shown on the left of the project path in *Deployment Settings* → *Mech-Vision* in Mech-Center.

**Length, height, width**

The object dimensions to input to the Mech-Vision project.

These values will be read by the read_object_dimensions Step.

Unit: mm

### Data Returned

`501, status code`

**Status code**

If there is no error, status code 1108 will be returned. Otherwise, the corresponding error code will be returned.

### Test Samples

Command 501 is normally executed without errors.

```
TCP send string: 501, 1, 100, 200, 300
TCP receive string: 501, 1108
```

Error: Error code 3002, missing height value.

```
TCP send string: 501, 1, 100, 200
TCP receive string: 501, 3002
```

## Command 502: Input TCP to Mech-Viz

This command is for dynamically inputting robot TCP into the Mech-Viz project.

The Task that receives the robot TCP is outer_move.

Please deploy the Mech-Viz project based on the template project at */Mech-Center/tool/viz_project/outer_move*, and put the outer_move Task to a proper position in the workflow.

This command needs to be executed before executing command 201.

### Command Sent

`502, TCP`

### Data Returned

`502, status code`

**Status code**

If there is no error, status code 2107 will be returned. Otherwise, the corresponding error code will be returned.

### Test Samples

Command 502 is normally executed without errors.

```
TCP send string: 502, 0, 10, 10, 20, 0, 0
TCP received string: 502, 2107
```

### Command 601: Notify

The user does not need to initiate this command.

When the Mech-Viz/Mech-Vision project runs to the **Notify** Task/Step, Mech-Center will send the custom notification message defined in the **Notify** Task/Step.

The **Notify** Task/Step must be named "Standard Interface Notify".

#### Command Sent

None

#### Data Returned

`601, custom notification message`

**Custom notification message**

The notification message defined in the **Notify** Task/Step. The message must be an integer.

#### Test Sample

When the notification message defined in the **Notify** Task/Step is set to integer 1000, the project will return 1000 when running through the **Notify** Task/Step.

```
TCP receive string = 601, 1000
```

### Command 701: Calibration

This command is for hand-eye calibration (camera extrinsic parameter calibration).

This command syncs the calibration status with Mech-Vision and fetches each calibration point that the robot needs to reach.

This command needs to be executed multiple times to complete the calibration.

#### Command Sent

`701, calibration status, flange pose, JPS`

**Calibration status**

`0`

Tell Mech-Vision to initiate the calibration.

`1`

The previous calibration point has been received by the robot.

`2`

The previous calibration point failed to be received by the robot.

**Flange pose**

The current flange pose of the robot.

**JPS**

The current JPS of the robot.

### Data Returned

`701, status code, calibration status, next calibration point's flange pose, next calibration point's JPS`

**Status code**

This status code is for indicating the status of receiving the calibration point. If the calibration point is transmitted normally, status code 7101 will be returned. Otherwise, the corresponding error code will be returned.

**Calibration status**

1

Calibration is in progress.

0

Calibration finished.

**Next calibration point' s flange pose**

The flange pose of the next calibration point the robot should move to.

**Next calibration point' s JPS**

The JPS of the next calibration point the robot should move to.

### Test Samples

Initiate the calibration.

```
TCP send string = 701, 0, 1371.62147, 25.6, 1334.3529, 148.58471, -179.24347, 88.75702, 88.
↪86102, -7.11107, -28.82309, -0.44014, -67.6509, 31.4764
TCP received string = 701, 7101, 0, 1271.6969, -743374, 1334.34094, -3128422, 1792412, -91.
↪11236, 93.28109, -12.0273, -32.8811, -0.37183, -68.41364, 27.02411
```

Obtain the calibration point (this process needs to be repeated to obtain multiple calibration points).

```
TCP send string = 701, 1, 1271.6969. -74.3374. 1334.34094. -3128422, 1792412 -91.11236, 93.
↪28109, -12.0273, -32.8811, -0.37183, -68.41364, 27.02411
TCP received string = 701, 7101, 0, 1471.62226, -74.40452, 1334.34235. 148.56924, -179.24432,␣
↪88.74148, 92.8367, -2.14999, -24.25433, -0.39222, -67.23261, 27.485225
```

Finish the calibration.

```
TCP send string = 701, 1, 1371.60876, 25.53615, 1384.45532. -20.82704. 179.22026, -72.77879,␣
↪88.88467, -7.42242.-26.68142, -0.2991, -69.95593, 39.26262
TCP received string = 701 7101, 1, 1371.62147.25.6, 1334.3529. 148, 58471. -179 24347, 88.
↪75702, 88.86102, -7.11107, -28.82309. -0.44014, -67.6509, 31.4764
```

(continues on next page)

(continued from previous page)

#### Command 901: Get Software Status

This command is designed for checking the software running status of Mech-Vision, Mech-Viz, and Mech-Center. At present, this command only supports checking whether Mech-Vision is ready for running the project.

#### Command Sent

`901`

No parameters.

#### Data Returned

`901, status code`

Status code

Software status.

#### Test Samples

Mech-Vision is ready for running the project.

```
TCP send string = 901
TCP received string = 901, 1101
```

Mech-Vision is not ready for running the project. Please open the project in Mech-Vision, right-click on the project in **Projects List**, and check **Autoload Project**.

```
TCP send string = 901
TCP received string = 901, 1001, 1
```

### 3.2.3 Siemens PLC

..Mech-Mind Software Suite can communicate with Siemens SIMATIC S7 PLCs through the Siemens S7 Standard Interface. For setup instructions, please refer to standard_interface_siemens_s7_tia_portal and standard_interface_siemens_s7_step_7.

The commands are as follows:

- *Command 101: Start Mech-Vision Project*
- *Command 102: Get Vision Result*
- *Command 103: Switch Mech-Vision Recipe*
- *Command 201: Start Mech-Viz Project*

- *Command 202: Stop Mech-Viz Project*
- *Command 203: Select Mech-Viz Branch*
- *Command 205: Get Planned Path*
- *Command 206: Get DO List*
- *Command 501: Input Object Dimensions to Mech-Vision*
- *Command 502: Input TCP to Mech-Viz*
- *Command 901: Get Software Status*

### Command 101: Start Mech-Vision Project

This command starts the running of the Mech-Vision project, which executes image capturing, and performs vision recognition.

If the project works in the eye-in-hand mode, the robot pose for image capturing will be transmitted by this command into the project.

This command is for scenarios using only Mech-Vision.

#### Command Sent

| Parameter | DB offset |
|---|---|
| Command code 101 | 2.0 |
| Project number | 8.0 |
| Number of vision points | 6.0 |
| Robot pose type | 4.0 |
| Robot pose | 12.0 (JPS) or 36.0 (flange pose) |

**Project number**

The integer ID number of the Mech-Vision project in Mech-Center, i.e., the number shown on the left of the project path in *Deployment Settings* → *Mech-Vision* in Mech-Center.

**Number of vision points**

The number of vision points (i.e., vision poses and their corresponding point clouds, labels, indices, etc.) to expect Mech-Vision to output.

`0`

Get all the vision points from the Mech-Vision project' s recognition results.

`integers > 0`

Get the specified number of vision points.

If the total number of vision points is smaller than the parameter value, all the available vision points will be returned.

If the total number of vision points is greater than or equal to the parameter value, vision points in the quantity of the parameter value will be returned.

**Note:** The command to obtain the vision points is command 102.

**Robot pose type**

This parameter indicates the type of the current pose of the real robot to input to Mech-Vision.

0

No robot pose needs to be transmitted by this command.

If the project works in the eye-to-hand mode, then image capturing has nothing to do with the robot' s pose, so no robot image capturing pose is needed by Mech-Vision.

1

The robot pose transmitted by this command is JPS.

2

The robot pose transmitted by this command is a flange pose.

**Robot pose**

This parameter is the robot pose needed when the project works in the eye-in-hand mode.

The robot pose is either JPS or flange pose, according to the setting of the parameter **robot pose type**.

### Data Returned

| Parameter | DB offset |
|---|---|
| Status code | 200.0 |

**Status code**

If there is no error, status code 1102 will be returned. Otherwise, the corresponding error code will be returned.

### Command 102: Get Vision Result

This command gets the vision result, i.e., vision points, after executing command 101.

### Command Sent

| Parameter | DB offset |
|---|---|
| Command code 102 | 2.0 |
| Project numner | 8.0 |

**Project number**

The integer ID number of the Mech-Vision project in Mech-Center, i.e., the number shown on the left of the project path in *Deployment Settings* → *Mech-Vision* in Mech-Center.

### Data Returned

| Parameter | DB offset |
|---|---|
| Status code | 200.0 |
| Sending status | 202.0 |
| Number of vision points | 204.0 |
| Reserved field | / |
| Poses | 208.0 |
| Labels | 1168.0 |

**Note:** The vision points (up to 40 vision points by default) are located at the tail of the data returned.

**Status code**

If there is no error, status code 1100 will be returned. Otherwise, the corresponding error code will be returned.

After executing this command, if the results from Mech-Vision have not been returned, Mech-Center will wait before sending the results to the robot. The default wait time is 10 seconds. If a timeout occurs, the timeout error status code will be returned.

**Sending status**

This parameter indicates whether the data returned includes newly arrived vision points.

1

The vision points in the data returned are new data.

**After PLC reads all the vision point data (poses and labels), please reset this field.**

**Note:** In Siemens PLC, by default, executing command 102 once can fetch at most 40 vision points (poses and labels) at a time. If the expected number of vision points is greater than 40, please execute command 102 multiple times.

**Number of vision points**

The number of vision points returned from the Mech-Vision project by executing this command this time.

Range: 0 to 40.

**Reserved field**

This field is not used.

The value defaults to 0.

**Poses**

A pose includes the Cartesian coordinates (XYZ) and Euler angles (ABC).

**Labels**

Each label is an integer label assigned to a pose. If in the Mech-Vision project, the labels are strings, they need to be mapped to integers before outputting from the Mech-Vision project. If there are no labels in the Mech-Vision project, the label defaults to 0.

### Command 103: Switch Mech-Vision Recipe

This command switches the parameter recipe used in Mech-Vision.

In Mech-Vision, what parameter settings a Step has can be modified by switching the parameter recipe.

Parameters involved in recipe switching usually include point cloud matching model, image matching template, ROI, confidence threshold, etc.

This command needs to be used before executing command 101 which starts the Mech-Vision project.

### Command Sent

| Parameter | DB offset |
|---|---|
| Command code 103 | 2.0 |
| Project number | 8.0 |
| Recipe numner | 10.0 |

**Project number**

The integer ID number of the Mech-Vision project in Mech-Center, i.e., the number shown on the left of the project path in *Deployment Settings* → *Mech-Vision* in Mech-Center.

**Recipe number**

The identification number of the parameter recipe to switch to, i.e., the number on the left of the parameter recipe name in *Project Assistance* → *Parameter Recipe* → *Parameter Recipe Editor* in Mech-Vision.

### Data Returned

| Parameter | DB offset |
|---|---|
| Status code | 200.0 |

**Status code**

If there is no error, status code 1107 will be returned. Otherwise, the corresponding error code will be returned.

### Command 201: Start Mech-Viz Project

This command is for scenarios using both Mech-Vision and Mech-Viz.

This command starts the running of the Mech-Viz project, calls the corresponding Mech-Vision project, and lets the Mech-Viz project plan the robot path based on the vision points from Mech-Vision.

For the Mech-Viz project that needs starting, the option *Autoload* needs to be checked in Mech-Viz' s interface.

Please see *Example Mech-Viz Projects for Standard Interface* for the description of example Mech-Viz projects.

### Command Sent

| Parameter | DB offset |
|---|---|
| Command code | 2.0 |
| Pose type | 4.0 |
| Robot pose | 12.0 |

**Pose type**

0

The current pose of the robot is not needed by Mech-Viz and no pose will be sent.

If the project works in the eye-to-hand mode, no robot image capturing pose will be needed by the project.

In Mech-Viz, the simulated robot will move from the initial pose JPS = [0, 0, 0, 0, 0, 0] to the first target point in the planned path.

1

The robot pose will be sent to Mech-Viz and the pose sent is in JPS.

In Mech-Viz, the simulated robot will move from the input initial pose (i.e., the pose sent by this command) to the first target point in the planned path.

TCP is not supported at present.

**Note:** If in the scene, there are barriers that stand in the way from the initial pose JPS = [0, 0, 0, 0, 0, 0] to the first target point in the planned path, the pose type must be set to 1.

**Robot pose**

The current JPS of the real robot (if pose type is set to 1).

### Data Returned

| Parameter | DB offset |
|---|---|
| Status code | 200.0 |

**Status code**

If there is no error, status code 2103 will be returned. Otherwise, the corresponding error code will be returned.

### Command 202: Stop Mech-Viz Project

Stop the running of the Mech-Viz project. This command is not needed when the Mech-Viz project does not fall into an infinite loop or can be stopped normally.

### Command Sent

| Parameter | DB offset |
|---|---|
| Command code 202 | 2.0 |

### Data Returned

| Parameter | DB offset |
|---|---|
| Status code | 200.0 |

**Status code**

If there is no error, status code 2104 will be returned. Otherwise, the corresponding error code will be returned.

### Command 203: Select Mech-Viz Branch

This command specifies which branch the project should run along. For this command, the branching is implemented by a branch_by_service_message Task, and this command selects the branch by specifying an out port of the Task.

Before executing this command, the Mech-Viz project needs to be started by executing command 201.

When the Mech-Viz project runs to the branch_by_service_message Task, it will wait for command 203 to specify which out port of the Task, i.e., the branch, the project should run along.

### Command Sent

| Parameter | DB offset |
|---|---|
| Command code 203 | 2.0 |
| Branching Task name | 60.0 |
| Out port number | 62.0 |

**Branching Task name**

This parameter is for specifying which branch_by_service_message Task the branch selection should apply to.

The value should be an integer ([1, N])., and before running the project, the Task involved in this command should be named as an integer ([1, N]). The name should be unique in the project.

**Out port number**

This parameter is for specifying which out port of the specified Task, i.e., the branch, the project should run along. The value should be an integer ([1, N]).

---

**Note:** Out port number is the 1-based index of the specified out port on the Task. For example, if the specified out port is the second out port of the Task from left to right, the out port number is 2.

---

### Data Returned

| Parameter | DB offset |
|---|---|
| Status code | 200.0 |

**Status code**

If there is no error, status code 2105 will be returned. Otherwise, the corresponding error code will be returned.

### Command 204: Set Move Index

This command is for setting the index parameter of a Task that involves sequential or separate motions or operations.

Tasks with index parameters include move_list, move_grid, custom_pallet_pattern, smart_pallet_pattern, etc.

Before executing this command, command 201 needs to be executed to start the Mech-Viz project.

### Command Sent

| Parameter | DB offset |
|---|---|
| Command code 204 | 2.0 |
| Task name | 64.0 |
| Index value | 66.0 |

**Task name**

This parameter specifies which Task the index setting should apply to.

The value should be an integer ([1, N]), and the Task that needs index parameter setting by this command should be named as an integer ([1, N]). The name should be unique in the project.

**Index value**

The index parameter of the specified Task will be set to this value.

### Data Returned

| Parameter | DB offset |
|---|---|
| Status code | 200.0 |

**Status code**

If there is no error, status code 2106 will be returned. Otherwise, the corresponding error code will be returned.

### Command 205: Get Planned Path

This command gets the robot motion path planned by Mech-Viz after command 201 is executed to start the Mech-Viz project.

---

**Note:** If one of the target points in the path is not supposed to be sent to the robot, please rename the corresponding move Task by adding "_internal" to the end of the name (with an underscore; case insensitive).

---

### Command Sent

| Parameter | DB offset |
|---|---|
| Command code 205 | 2.0 |
| Target point type | 4.0 |

**Target point type**

This parameter specifies the type of path target points to return from Mech-Viz.

1

The target points returned should be in JPS.

2

The target points returned should be in TCP.

### Data Returned

| Parameter | DB offset |
|---|---|
| Status code | 200.0 |
| Sending status | 202.0 |
| Number of points | 204.0 |
| Position of "visual_move" | 206.0 |
| Target points' poses | 208.0 |
| Target points' labels | 1168.0 |
| Target points' velocities | 1248.0 |

**Status code**

If there is no error, status code 2100 will be returned. Otherwise, the corresponding error code will be returned.

---

**Note:** When executing this command, if Mech-Viz has not yet had the planned robot motion path (the project is still running), Mech-Center will wait. The default wait time is 10 seconds. If a timeout occurs, a timeout error code will be returned.

---

**Sending status**

This parameter indicates whether the data returned includes newly arrived target points.

1

The target points in the data returned are new data.

**After PLC reads all the target point data (poses, labels, velocities), please reset this field.**

---

**Note:** In Siemens PLC, by default, executing command 205 once can fetch at most 40 target points (poses, labels, velocities) at a time. If the number of target points is greater than 40, please execute command 205 multiple times.

---

**Number of points**

This parameter indicates the number of path target points ([pose, label, velocity]) sent by executing this command this time.

Range: 0 to 40.

**Position of "visual_move"**

The position of the visual_move Task, i.e., the move to the vision pose (usually the pose for picking the object) in the entire robot motion path.

For example, if the path is composed of Tasks **move_1**, **move_2**, **visual_move**, **move_3** sequentially, the position of **visual_move** is 3.

If in the path there is no visual_move Task, the returned value will be 0.

**Poses**

Each pose includes Cartesian coordinates (XYZ) and Euler angles (ABC), or JPS, according to the target point type set by this command.

**Labels**

A label is the integer label assigned to a pose. If in the Mech-Vision project, the labels are strings, they need to be mapped to integers before outputting from the Mech-Vision project. If there are no labels in the Mech-Vision project, the label defaults to 0.

**Velocities**

A velocity is the non-zero velocity parameter percentage value for the correspoinding move Task in Mech-Viz.

### Command 206: Get DO List

This command gets the planned DO signal list when there are multiple grippers, such as suction cup sections, to control.

For using this command:

1. The Mech-Viz project’s name must be set to “suction_zone”.
2. The set_do_list Task must be named to “set_do_list_1”.
3. The set_do_list Task’s parameter “Get DO List from VisualMove” must be set to True.
4. The set_do_list Task must immediately follow a “visual_move” Task.
5. The name of the visual_move Task followed by the set_do_list Task must be selected in the lower part of the parameter panel of set_do_list.

Before calling this command, command 205 needs to be executed to obtain the planned motion path by Mech-Viz.

Please deploy the Mech-Viz project based on the template project at */Mech-Center/tool/viz_project/suction_zone*, and set the suction cup configuration file in the Mech-Viz project.

#### Command Sent

| Parameter | DB offset |
|---|---|
| Command code 206 | 2.0 |

No parameters.

### Data Returned

| Parameter | DB offset |
|---|---|
| Status code | 200.0 |
| DO signals | 1408.0 |

**Status code**

If there are no errors, status code 2102 will be returned. Otherwise, the corresponding error code will be returned.

**DO signal value**

There are 64 DO signal values, in integers, located at the tail of the data returned.

Range of valid DO values: [0, 999]. Placeholder value: -1.

### Command 501: Input Object Dimensions to Mech-Vision

This command is for dynamically inputting object dimensions into the Mech-Vision project.

Please confirm the actual object dimensions before running the Mech-Vision project.

The Mech-Vision project should have the read_object_dimensions Step, and the Step' s parameter **Read Object Dimensions from Parameters** should be set to `True`.

### Command Sent

| Parameter | DB offset |
|---|---|
| Command code 501 | 2.0 |
| Project number | 8.0 |
| [length, height, width] | 68.0 |

**Project number**

The integer ID number of the Mech-Vision project in Mech-Center, i.e., the number shown on the left of the project path in *Deployment Settings* → *Mech-Vision* in Mech-Center.

**[length, height, width]**

The object dimensions to input to the Mech-Vision project.

Those values will be read by the read_object_dimensions Step.

Unit: mm

### Data Returned

| Parameter | DB offset |
|---|---|
| Status code | 200.0 |

**Status code**

If there is no error, status code 1108 will be returned. Otherwise, the corresponding error code will be returned.

### Command 502: Input TCP to Mech-Viz

This command is for dynamically inputting robot TCP into the Mech-Viz project.

The Task that receives the robot TCP is outer_move.

Please deploy the Mech-Viz project based on the template project at */Mech-Center/tool/viz_project/outer_move*, and put the outer_move Task to a proper position in the workflow.

This command needs to be executed before executing command 201.

### Command Sent

| Parameter | DB offset |
|---|---|
| Command code 502 | 2.0 |
| TCP | 80.0 |

### Data Returned

| Parameter | DB offset |
|---|---|
| Status code | 200.0 |

**Status code**

If there is no error, status code 2107 will be returned. Otherwise, the corresponding error code will be returned.

### Command 901: Get Software Status

This command is designed for checking the software running status of Mech-Vision, Mech-Viz, and Mech-Center. At present, this command only supports checking whether Mech-Vision is ready for running the project.

#### Command Sent

| Parameter | DB offset |
|---|---|
| Command code 901 | 2.0 |

No parameters.

#### Data Returned

| Parameter | DB offset |
|---|---|
| Status code | 200.0 |

**Status code**

Software status.

### 3.2.4 PROFINET

Mech-Mind Software Suite can communicate with Siemens SIMATIC S7 PLCs through the PROFINET Standard Interface. For setup instructions, please refer to standard_interface_profinet.

#### Protocol

#### From Mech-Center to PLC

#### Module Data Structures

#### Control_Output

| Bit | Data |
|---|---|
| 7 | / |
| 6 | / |
| 5 | / |
| 4 | Command execution complete (Bool) |
| 3 | Data ready (Bool) |
| 2 | Camera exposure complete (Bool) |
| 1 | Trigger Acknowldge (Bool) |
| 0 | Heartbeat (Bool) |

### Status Code

Status code. INT32

### Calib_Cam_Status

Calibration status. INT8

### Send_Pose_Num

Number of poses sent. INT8

### Visual_Point_Index

Position of visual_move in the planned path. INT8

### DO List

| Byte | Data |
|---|---|
| 0 | DO list 1: signal 0 to 7 |
| 1 | DO list 2: signal 8 to 15 |
| 2 | DO list 3: signal 16 to 23 |
| 3 | DO list 4: signal 24 to 31 |
| 4 | DO list 5: signal 32 to 39 |
| 5 | DO list 6: signal 40 to 47 |
| 6 | DO list 7: signal 48 to 55 |
| 7 | DO list 8: signal 56 to 63 |

### Notify Message

Integer message. INT32

### Send_Pose_Type

Type of pose sent. INT8

### Target_Pose

A pose in Cartesian coordinates and Euler angles can be represented by:

[X, Y, Z, A, B, C]

JPS consists of up to 6 joint angles:

[J1, J2, J3, J4, J5, J6]

| Byte | Data |
|---|---|
| 0 to 3 | X or J1 |
| 4 to 7 | Y or J2 |
| 8 to 11 | Z or J3 |
| 12 to 15 | A or J4 |
| 16 to 19 | B or J5 |
| 20 to 23 | C or J6 |

### Target_Label

Label of target point. INT32

### Target_Speed

Velocity of target point. sINT32

### Ext_Output_Data

Reserved module for other data for transmission.

This moduel takes up 40 bytes (INT32[1: 10], 10 INT32 integers in total).

### Module Functions

### Heartbeat

System heartbeat that flips every 1 second.

### Trigger Acknowledge

Trigger Acknowledge = 1 means Mech-Mind Software Suite has been triggered sucessfully by the Trigger signal.

Tigger Acknowledge will stay at 1 until the Trigger signal is reset to 0.

### Exposure Complete

When the camera completes the exposure, Exposure Complete will be set to 1, indicating that the object can be moved or the robot working eye-in-hand can move.

..Please see :ref:`` for instructions on using this module.

### Data Ready

This module is for indicating that new data has been sent from Mech-Center to the PLC, and the PLC can read the data.

This module' s signal is for command 102 or command 205.

### Command Complete

This module is for indicating that the execution of a command has been completed, and the data returned by the command can be read.

For command 102 and 205, only when the last byte of data has been returned will this module' s signal be set to 1.

### Status Code

The command execution status code returned from Mech-Center.

It may be a normal status code or an error code.

### Calib_Cam_Status

For command 701: calibration.

1 means the calibration is in progress.

0 means the calibration has been completed.

### Send_Pose_Num

The number of poses sent by executing the command at the time.

### Visual_Point_Index

The position of of the visual_move Task in the planned path.

For example, if the planned path consists of the following move Tasks: **move_1**, **move_2**, **move_3**, **visual_move**, **move_4**, then Visual_Point_Index = 4.

### DO List

The 64 INT8 DO signals for controlling multiple suction cup sections or array gripper.

### Notify Message

The customized integer message sent by a **Notify** Task/Step named "Standard Interface Notify" from Mech-Viz/Mech-Vision.

### Send_Pose_Type

1 means JPS. 2 means TCP.

### Target_Pose

Cartesian coodinates and Euler angles, or JPS.

**Note:** The data from this module should be divided by 10000 before using.

### Target_Label

Correspoinding non-negative integer labels corresponding to the poses.

### Target_Speed

Velocity parameter percentage value of the move Task correpsonding to the target point (pose).

Range: 0 to 100

### From PLC to Mech-Center

#### Module Data Structures

#### Control_input

| Bit | Data |
|---|---|
| 7 | / |
| 6 | / |
| 5 | / |
| 4 | Reset Notify (Bool) |
| 3 | Data Acknowledge (Bool) |
| 2 | Reset Exposure_Complete (Bool) |
| 1 | Trigger (Bool) |
| 0 | Comm Enable (Bool) |

#### Command

Command code. INT32

#### Calib_Rob_Status

0: The calibration starts.

1: The robot has normally moved to the last calibration point sent.

2: The robot failed to move to the last calibration point sent.

#### Robot_Pose_Type

Type of robot pose.

#### Req_Pose_Num

Number of target points requested. INT8

#### Vision_Proj_Num

Mech-Vision project ID number.

### Vision_Recipe_Num

Mech-Vision parameter recipe number.

The identification number of the parameter recipe to switch to, i.e., the number on the left of the parameter recipe name in *Project Assistance* → *Parameter Recipe* → *Parameter Recipe Editor* in Mech-Vision.

### Viz_Task_Name

Mech-Viz branching Task name.

### Viz_Task_Value

Mech-Viz branching Task out port number.

### Req_Pose_Type

Type of target points requested. INT8

### Robot_Pose_JPS

JPS includes up to 6 joint position data (6 INT32 integers):

[J1, J2, J3, J4, J5, J6]

| Byte | Data |
|---|---|
| 0 to 3 | J1 |
| 4 to 7 | J2 |
| 8 to 11 | J3 |
| 12 to 15 | J4 |
| 16 to 19 | J5 |
| 20 to 23 | J6 |

### Robot_Pose_TCP

A TCP includes Cartesian coordinates (X, Y, Z) and Euler angles (A, B, C), 6 INT32 integers in total.

[X, Y, Z, A, B, C]

| Byte | Data |
|---|---|
| 0 to 3 | X |
| 4 to 7 | Y |
| 8 to 11 | Z |
| 12 to 15 | A |
| 16 to 19 | B |
| 20 to 23 | C |

### Ext_Input_Data

Reserved module for other data for transmission.

This moduel takes up 40 bytes (INT32[1: 10], 10 INT32 integers in total).

#### Module Functions

#### Comm Enable

0: Communication disabled. Mech-Center will ignore the Trigger signal. 1: Communication enabled. The Trigger signal will work and Mech-Center will receive commands.

#### Trigger

If Trigger = 1, Mech-Center will read the command sent and the command will be executed.

Trigger Acknowlege can be reset once Mech-Center receives the Trigger signal.

The upward segment of the signal is considered as 1.

#### Reset_Exposure

If Reset_Exposure = 1, Exposure Complete will be set to 0.

#### Data Acknowledge

If Data Acknowledge = 0, the PLC has not read the data from Mech-Center and the data are kept at the port.

If Data Acknowledge = 1, the PLC has read the data from Mech-Center and Mech-Center can write the data of the next round.

Data Acknowledge is for acknowledging having read the data returned by executing command 102 or command 205.

Data Acknowledge can be reset at heartbeat flip or when Data Ready = 0.

#### Reset Notify

If Reset Notify = 1, the content of Notify Message will be cleared.

### Command

The command code.

### Calib_Rob_Status

0: Calibration starts, and command 701 is sent.

1: Calibration in progress and the robot moves normally.

### Robot_Pose_Type

0: No image capturing robot pose is needed (eye-to-hand mode).

1: The image capturing robot pose sent is in JPS.

2: The image capturing robot pose sent is a flange pose.

### Req_Pose_Num

Number of vision points to request from Mech-Vision.

0: Request all the available vision points from the vision results in Mech-Vision.

### Vision_Proj_Num

Mech-Vision project ID number.

### Vision_Recipe_Num

Mech-Vision parameter recipe numner

### Viz_Task_Name

Mech-Viz Task name.

### Viz_Task_Value

The value to set in the Mech-Viz Task’s index parameter.

### Req_Pose_Type

The pose type to expect Mech-Viz to return,

1: JPS.

2: TCP.

### Robot_Pose_JPS

Robot JPS for image capturing.

Please multiply the JPS data by 10000 before setting to the module.

### Robot_Pose_TCP (Robot_Pose_Flange)

Robot flang pose for image capturing.

Please multiply the pose data by 10000 before setting to the module.

### Ext_Input_Data

Reserved module for other data for transmission.

This moduel takes up 40 bytes (INT32[1: 10], 10 INT32 integers in total).

### Profinet Commands

- *Command 101: Start Mech-Vision Project*
- *Command 102: Get Vision Result*
- *Command 103: Switch Mech-Vision Recipe*
- *Command 201: Start Mech-Viz Project*
- *Command 202: Stop Mech-Viz Project*
- *Command 203: Select Mech-Viz Branch*
- *Command 205: Get Planned Path*
- *Command 206: Get DO List*
- *Command 501: Input Object Dimensions to Mech-Vision*
- *Command 502: Input TCP to Mech-Viz*
- *Command 901: Get Software Status*

### Command 101: Start Mech-Vision Project

This command starts the running of the Mech-Vision project, which executes image capturing, and performs vision recognition.

If the project works in the eye-in-hand mode, the robot pose for image capturing will be transmitted by this command into the project.

This command is for scenarios using only Mech-Vision.

#### Command Sent

| Module | Description |
|---|---|
| Command | 101 |
| Vision_Proj_Num | Project number |
| Req_Pose_Num | Number of vision points |
| Robot_Pose_Type | Robot pose type |
| Robot_Pose_JPS / Robot_Pose_TCP | Robot pose |

**Project number**

The integer ID number of the Mech-Vision project in Mech-Center, i.e., the number shown on the left of the project path in *Deployment Settings* → *Mech-Vision* in Mech-Center.

**Number of vision points**

The number of vision points (i.e., vision poses and their corresponding point clouds, labels, indices, etc.) to expect Mech-Vision to output.

`0`

Get all the vision points from the Mech-Vision project' s recognition results.

`integers > 0`

Get the specified number of vision points.

If the total number of vision points is smaller than the parameter value, all the available vision points will be returned.

If the total number of vision points is greater than or equal to the parameter value, vision points in the quantity of the parameter value will be returned.

**Note:** The command to obtain the vision points is command 102.

**Robot pose type**

This parameter indicates the type of the current pose of the real robot to input to Mech-Vision.

`0`

No robot pose needs to be transmitted by this command.

If the project works in the eye-to-hand mode, then image capturing has nothing to do with the robot' s pose, so no robot image capturing pose is needed by Mech-Vision.

1

The robot pose transmitted by this command is in JPs.

2

The robot pose transmitted by this command is a flange pose.

**Robot pose**

This parameter is the robot pose needed when the project works in the eye-in-hand mode.

The robot pose is either in JPs or flange pose, according to the setting of the parameter "robot pose type" .

### Data Returned

Status code

If there is no error, status code 1102 will be returned. Otherwise, the corresponding error code will be returned.

### Command 102: Get Vision Result

This command gets the vision result, i.e., vision points, after executing command 101.

---

**Note:** In PROFINET, by default, command 102 can only fetch at most 20 vision points at a time. So, command 102 may need to be repeatedly executed until all the vision points required are obtained.

---

### Command Sent

| Module | Description |
|---|---|
| Command | 102 |
| Vision_Proj_Num | Project number |

**Project number**

The integer ID number of the Mech-Vision project in Mech-Center, i.e., the number shown on the left of the project path in *Deployment Settings* → *Mech-Vision* in Mech-Center.

### Data Returned

| Module | Description |
|---|---|
| Status code | / |
| Send_Pose_Num | Number of vision points |
| Send_Pose_Type | Pose type |
| Target_Pose | Poses in vision points |
| Target_Label | Labels in vision points |

**Note:** The vision points (up to 20 vision points) are located at the tail of the data returned.

**Status code**

If there is no error, status code 1100 will be returned. Otherwise, the corresponding error code will be returned.

After executing this command, if the results from Mech-Vision have not been returned, Mech-Center will wait before sending the results to the robot. The default wait time is 10 seconds. If a timeout occurs, the timeout error status code will be returned.

**Number of vision points**

The number of vision points returned from the Mech-Vision project by executing this command this time.

**Pose type**

The pose type is vision points.

This module' s value defaults to 2, meaning the pose type is TCP pose.

**Poses in vision points**

A pose includes the Cartesian coordinates (XYZ) and Euler angles (ABC).

**Labels in vision points**

The integer label assigned to the pose. If in the Mech-Vision project, the labels are strings, they need to be mapped to integers before outputting from the Mech-Vision project. If there are no labels in the Mech-Vision project, the label defaults to 0.

### Command 103: Switch Mech-Vision Recipe

This command switches the parameter recipe used in Mech-Vision.

In Mech-Vision, what parameter settings a Step has can be modified by switching the parameter recipe.

Parameters involved in recipe switching usually include point cloud matching model, image matching template, ROI, confidence threshold, etc.

This command needs to be used before executing command 101 which starts the Mech-Vision project.

### Command Sent

| Module | Description |
|---|---|
| Command | 103 |
| Vision_Proj_Num | Project number |
| Vision_Recipe_Num | Recipe number |

**Project number**

The integer ID number of the Mech-Vision project in Mech-Center, i.e., the number shown on the left of the project path in *Deployment Settings* → *Mech-Vision* in Mech-Center.

**Recipe number**

The identification number of the parameter recipe to switch to, i.e., the number on the left of the parameter recipe name in *Project Assistance* → *Parameter Recipe* → *Parameter Recipe Editor* in Mech-Vision.

### Data Returned

**Status code**

If there is no error, status code 1107 will be returned. Otherwise, the corresponding error code will be returned.

### Command 201: Start Mech-Viz Project

This command is for scenarios using both Mech-Vision and Mech-Viz.

This command starts the running of the Mech-Viz project, calls the corresponding Mech-Vision project, and lets the Mech-Viz project plan the robot path based on the vision points from Mech-Vision.

For the Mech-Viz project that needs starting, the option *Autoload* needs to be checked in Mech-Viz' s interface.

Please see *Example Mech-Viz Projects for Standard Interface* for the description of example Mech-Viz projects.

### Command Sent

| Module | Description |
|---|---|
| Command | 201 |
| Robot_Pose_Type | Pose type |
| Robot_Pose_JPS | Pose |

**Pose type**

0

The current pose of the robot is not needed by Mech-Viz and no pose will be sent.

If the project works in the eye-to-hand mode, no robot image capturing pose will be needed by the project.

In Mech-Viz, the simulated robot will move from the initial pose JPs = [0, 0, 0, 0, 0, 0] to the first target point in the planned path.

1

The robot pose will be sent to Mech-Viz and the pose sent is in JPs.

In Mech-Viz, the simulated robot will move from the input initial pose (i.e., the pose sent by this command) to the first target point in the planned path.

TCP is not supported at present.

---

**Note:** If in the scene, there are barriers that stand in the way from the initial pose JPs = [0, 0, 0, 0, 0, 0] to the first target point in the planned path, the pose type must be set to 1.

---

**Pose**

The current JPs of the real robot (if pose type is set to 1).

---

**Note:** Before setting as the value of the pose module, the numerical values of the pose need to be multiplied by 10000, to transform floating point numbers into integers.

---

### Data Returned

**Status code**

If there is no error, status code 2103 will be returned. Otherwise, the corresponding error code will be returned.

### Command 202: Stop Mech-Viz Project

Stop the running of the Mech-Viz project. This command is not needed when the Mech-Viz project does not fall into an infinite loop or can be stopped normally.

### Command Sent

| Module | Description |
|---|---|
| Command | 202 |

### Data Returned

**Status code**

If there is no error, status code 2104 will be returned. Otherwise, the corresponding error code will be returned.

### Command 203: Select Mech-Viz Branch

This command specifies which branch the project should run along. For this command, the branching is implemented by a branch_by_service_message Task, and this command selects the branch by specifying an out port of the Task.

Before executing this command, the Mech-Viz project needs to be started by executing command 201.

When the Mech-Viz project runs to the branch_by_service_message Task, it will wait for command 203 to specify which out port of the Task, i.e., the branch, the project should run along.

### Command Sent

| Module | Description |
|---|---|
| Command | 203 |
| Viz_Task_Name | branching Task name |
| Viz_Task_Value | out port number |

**Branching Task name**

This parameter is for specifying which branch_by_service_message Task the branch selection should apply to.

The value should be an integer ([1, N])., and before running the project, the Task involved in this command should be named as an integer ([1, N]). The name should be unique in the project.

**Out port number**

This parameter is for specifying which out port of the specified Task, i.e., the branch, the project should run along. The value should be an integer ([1, N]).

---

**Note:** Out port number is the 1-based index of the specified out port on the Task. For example, if the specified out port is the second out port of the Task from left to right, the out port number is 2.

---

### Data Returned

**Status code**

If there is no error, status code 2105 will be returned. Otherwise, the corresponding error code will be returned.

### Command 204: Set Move Index

This command is for setting the index parameter of a Task that involves sequential or separate motions or operations.

Tasks with index parameters include move_list, move_grid, custom_pallet_pattern, smart_pallet_pattern, etc.

Before executing this command, command 201 needs to be executed to start the Mech-Viz project.

### Command Sent

| Module | Description |
|---|---|
| Command | 204 |
| Viz_Task_Name | Task name |
| Viz_Task_Value | Index value |

**Task name**

This parameter specifies which Task the index setting should apply to.

The value should be an integer ([1, N]), and the Task that needs index parameter setting by this command should be named as an integer ([1, N]). The name should be unique in the project.

**Index value**

The index parameter of the specified Task will be set to this value.

### Data Returned

**Status code**

If there is no error, status code 2106 will be returned. Otherwise, the corresponding error code will be returned.

### Command 205: Get Planned Path

This command gets the robot motion path planned by Mech-Viz after command 201 is executed to start the Mech-Viz project.

**Note:** If one of the target points in the path is not supposed to be sent to the robot, please rename the corresponding move Task by adding "_internal" to the end of the name (with an underscore; case insensitive).

**Note:** In PROFINET, by default, command 205 can only fetch at most 20 target points of the planned path at a time. So, command 205 may need to be executed repeatedly until all the target points required are obtained.

#### Command Sent

| Module | Description |
|---|---|
| Command | 205 |
| Req_Pose_Type | Target point type |

**Target point type**

This parameter specifies the type of poses in the path target points to return from Mech-Viz.

1

The target points returned should be in JPs.

2

The target points returned should be in TCP.

#### Data Returned

| Module | Description |
|---|---|
| Status code | / |
| Send_Pose_Num | Number of points |
| Send_Pose_Type | Pose type in targe points |
| Visual_Point_Index | Position of "visual_move" |
| Target_Pose | Poses in target points |
| Target_Label | Labels in target points |
| Target_Speed | Velocities in target points |

**Status code**

If there is no error, status code 2100 will be returned. Otherwise, the corresponding error code will be returned.

**Note:** When executing this command, if Mech-Viz has not yet had the planned robot motion path (the project is still running), Mech-Center will wait. The default wait time is 10 seconds. If a timeout occurs, a timeout error code will be returned.

**Number of points**

This parameter indicates the number of path target points ([pose, label, velocity]) sent by executing this command this time.

**Note:** In PROFINET, by default, command 205 can only send at most 20 target points at a time. So, command 205 may need to be repeatedly executed until all the target points required are sent.

**Pose type in targe points**

Same as the sent value in module "Req_Pose_Type".

1

JPs

2

TCP

**Position of "visual_move"**

The position of the visual_move Task, i.e., the move to the vision pose (usually the pose for picking the object) in the entire robot motion path.

For example, if the path is composed of Tasks **move_1**, **move_2**, **visual_move**, **move_3** sequentially, the position of **visual_move** is 3.

If in the path there is no visual_move Task, the returned value will be 0.

**Poses in target points**

A pose includes Cartesian coordinates (XYZ) and Euler angles (ABC), or JPs, according to the pose type set by command 205.

**Labels in target points**

Label is the integer label assigned to the pose. If in the Mech-Vision project, the labels are strings, they need to be mapped to integers before outputting from the Mech-Vision project. If there are no labels in the Mech-Vision project, the label defaults to 0.

**Velocities in target points**

A velocity value is the non-zero velocity parameter percentage value for the move Task set in Mech-Viz.

### Command 206: Get DO List

This command gets the planned DO signal list when there are multiple grippers, such as suction cup sections, to control.

For using this command:

1. The Mech-Viz project’s name must be set to “suction_zone”.
2. The set_do_list Task must be named to “set_do_list_1”.
3. The set_do_list Task’s parameter “Get DO List from VisualMove” must be set to True.
4. The set_do_list Task must immediately follow a “visual_move” Task.
5. The name of the visual_move Task followed by the set_do_list Task must be selected in the lower part of the parameter panel of set_do_list.

Before calling this command, command 205 needs to be executed to obtain the planned motion path by Mech-Viz.

Please deploy the Mech-Viz project based on the template project at */Mech-Center/tool/viz_project/suction_zone*, and set the suction cup configuration file in the Mech-Viz project.

#### Command Sent

| Module | Description |
|---|---|
| Command | 206 |

#### Data Returned

| Module | Description |
|---|---|
| Status code | / |
| DO List | DO signal values |

**Status code**

If there are no errors, status code 2102 will be returned. Otherwise, the corresponding error code will be returned.

**DO signal values**

There are 64 DO signal values, in integers, located at the tail of the data returned.

Range of valid DO values: [0, 999]. Placeholder value: -1.

### Command 501: Input Object Dimensions to Mech-Vision

This command is for dynamically inputting object dimensions into the Mech-Vision project.

Please confirm the actual object dimensions before running the Mech-Vision project.

The Mech-Vision project should have the read_object_dimensions Step, and the Step' s parameter **Read Object Dimensions from Parameters** should be set to `True`.

#### Command Sent

| Module | Description |
|---|---|
| Command | 501 |
| Vision_Proj_Num | Project number |
| Ext_Input_Data | [length, height, width] |

**Project number**

The integer ID number of the Mech-Vision project in Mech-Center, i.e., the number shown on the left of the project path in *Deployment Settings* → *Mech-Vision* in Mech-Center.

**Length, height, width**

The object dimensions to input to the Mech-Vision project.

Those values will be read by the read_object_dimensions Step.

Unit: mm

#### Data Returned

**Status code**

If there is no error, status code 1108 will be returned. Otherwise, the corresponding error code will be returned.

### Command 502: Input TCP to Mech-Viz

This command is for dynamically inputting robot TCP into the Mech-Viz project.

The Task that receives the robot TCP is outer_move.

Please deploy the Mech-Viz project based on the template project at */Mech-Center/tool/viz_project/outer_move*, and put the outer_move Task to a proper position in the workflow.

This command needs to be executed before executing command 201.

### Command Sent

| Module | Description |
|---|---|
| Command | 502 |
| Ext_Input_Data | TCP |

### Data Returned

**Status code**

If there is no error, status code 2107 will be returned. Otherwise, the corresponding error code will be returned.

### Command 901: Get Software Status

This command is designed for checking the software running status of Mech-Vision, Mech-Viz, and Mech-Center. At present, this command only supports checking whether Mech-Vision is ready for running the project.

### Command Sent

| Module | Description |
|---|---|
| Command | 901 |

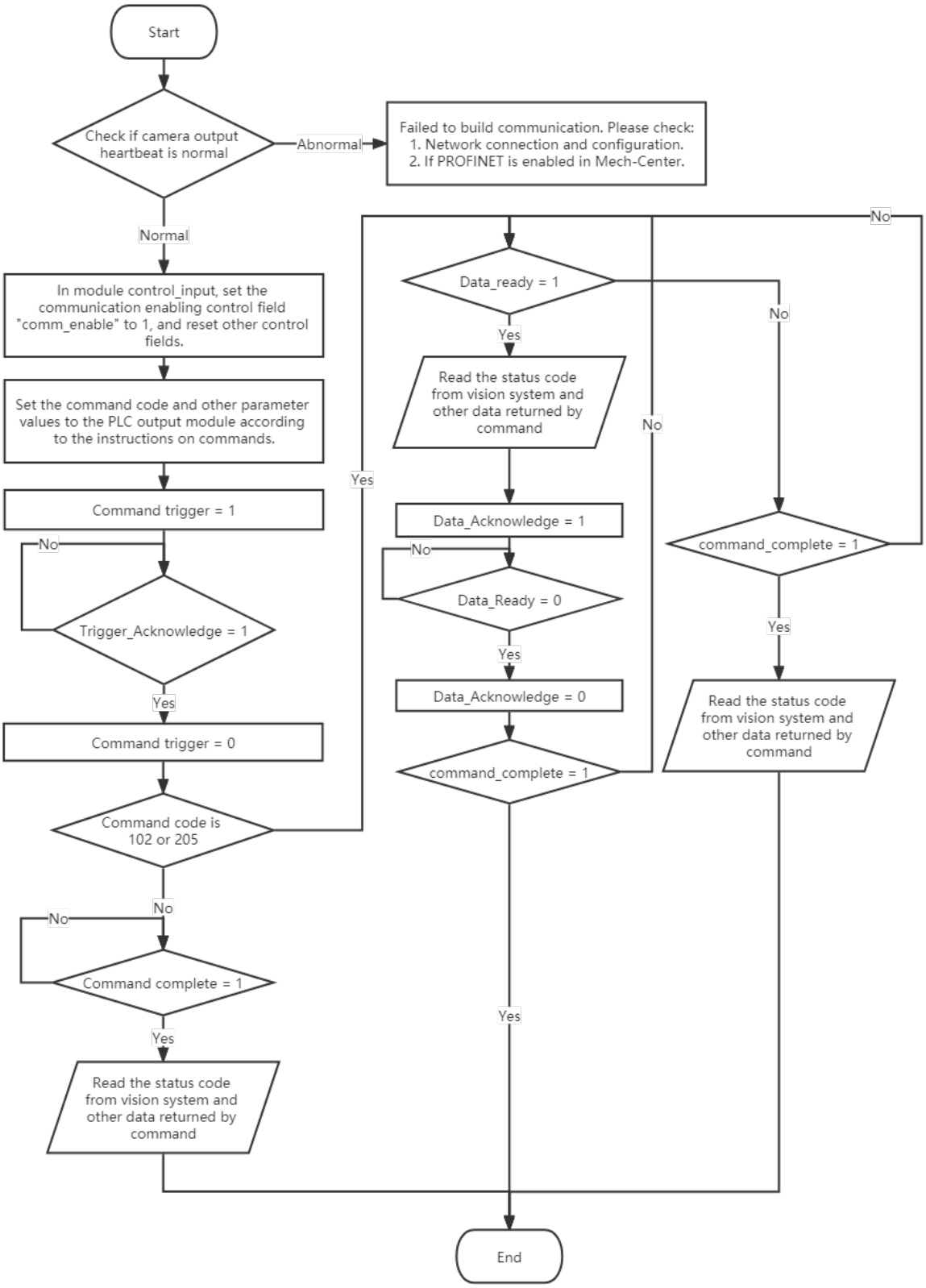
### Data Returned

**Status code**

Software status.

If there is no error, 1101 will be returned. Otherwise, the corresponding error code will be returned.

## Communication Control Flowchart



### 3.2.5 EtherNet/IP

For development instructions for EtherNet/IP, please see *PROFINET*.

Mech-Mind Software Suite can communicate with some Keyence and Omron PLCs through the EtherNet/IP Standard Interface. For setup instructions, please refer to standard_interface_ethernetip_keyence and standard_interface_ethernetip_omron.

### 3.2.6 Status Codes

The following are lists of status codes used in Standard Interface communication. Troubleshooting instructions will be added soon!

#### Overview

| Range | Category |
|---|---|
| 1001–1099 | Mech-Vision error codes |
| 1100–1199 | Mech-Vision normal status codes |
| 2001–2099 | Mech-Viz error codes |
| 2100–2199 | Mech-Viz normal status codes |
| 3001–3099 | Mech-Center error codes |
| 3100–3199 | Mech-Center normal status codes |
| 7001–7099 | Hand-eye calibration error codes |
| 7100–7199 | Hand-eye calibration normal status codes |

#### Mech-Vision

### Mech-Vision Error Codes

| Code | Meaning |
|---|---|
| 1001 | Mech-Vision: project not registered |
| 1002 | Mech-Vision: no vision results |
| 1003 | Mech-Vision: no point cloud in ROI |
| 1004 | Mech-Vision: parameter setting failed |
| 1005 | Mech-Vision: invalid pose type |
| 1006 | Mech-Vision: invalid pose data |
| 1007 | Mech-Vision: computing |
| 1008 | Status code not in use |
| 1009 | Mech-Vision: number of poses and number of motion params do not match |
| 1010 | Mech-Vision: number of poses and number of labels do not match |
| 1011 | Mech-Vision: project ID number does not exist |
| 1012 | Mech-Vision: parameter recipe number out of range |
| 1013 | Mech-Vision: parameter recipe not set |
| 1014 | Mech-Vision: parameter recipe name does not exist |
| 1015 | Mech-Vision: project runtime error |
| 1016 | Mech-Vision: failed to start deep learning server |
| 1017 | Mech-Vision: invalid label mapping |
| 1018 | Mech-Vision: wrong number of vision points |
| 1019 | Mech-Vision: execution timed out |
| 1020 | Mech-Vision: not executed |
| 1021 | Mech-Vision: failed to set box dimensions; please confirm if Step read_object_dimensions is in the project |
| 1022 | Mech-Vision: invalid setting values of object dimensions |
| 1023 | Mech-Vision: failed to connect to camera |

### Mech-Vision Normal Status Codes

| Code | Meaning |
|---|---|
| 1100 | Mech-Vision: successfully obtained vision points |
| 1101 | Mech-Vision: ready |
| 1102 | Mech-Vision: successfully triggered project |
| 1107 | Mech-Vision: successfully switched parameter recipe |
| 1108 | Mech-Vision: successfully set box dimensions |

### Mech-Viz

### Mech-Viz Error Codes

| Code | Meaning |
|---|---|
| 2001 | Mech-Viz: project not registered |
| 2002 | Mech-Viz: project is running |
| 2003 | Mech-Viz: vision results from Mech-Vision not received |

continues on next page

Table 1 – continued from previous page

| Code | Meaning |
|---|---|
| 2004 | Mech-Viz: failed to reach vision point from Mech-Vision |
| 2005 | Mech-Viz: failed to calculate robot JPS |
| 2006 | Error code not in use |
| 2007 | Mech-Viz: path planning failed |
| 2008 | Mech-Viz: project runtime error |
| 2009 | Mech-Viz: TCP not provided |
| 2010 | Mech-Viz: path not reachable |
| 2011 | Mech-Viz: DO list not provided |
| 2012 | Mech-Viz: invalid pose type |
| 2013 | Mech-Viz: invalid pose data |
| 2014 | Mech-Viz: project not set |
| 2015 | Mech-Viz: pose of TCP type not supported |
| 2016 | Mech-Viz: parameter setting failed |
| 2017 | Mech-Viz: failed to stop execution |
| 2018 | Mech-Viz: invalid branch_by_service_message Task out port number |
| 2019 | Mech-Viz: failed to set branch_by_service_message Task; please confirm whether the Task name exists |
| 2020 | Mech-Viz: motion error—singularity |
| 2021 | Mech-Viz: MoveL calculation mismatch |
| 2022 | Mech-Viz: not executed |
| 2023 | Mech-Viz: project file error |
| 2024 | Mech-Viz: invalid branch_by_service_message Task name |
| 2025 | Mech-Viz: execution timed out |
| 2026 | Mech-Viz: invalid name of Task with index parameter |
| 2027 | Mech-Viz: invalid index value |
| 2028 | Mech-Viz: index setting failed; please confirm whether the Task name exists |
| 2029 | Mech-Viz: failed to set target point of outer_move Task |
| 2030 | Mech-Viz: invalid vision point |
| 2031 | Mech-Viz: robot self-collision detected |
| 2032 | Mech-Viz: collision between robot and scene object detected |
| 2033 | Mech-Viz: collision detected as point cloud collision point count exceeds threshold |
| 2034 | Mech-Viz: collision detected as point cloud collision area exceeds threshold |
| 2035 | Mech-Viz: collision detected as point cloud collision volume exceeds threshold |
| 2036 | Mech-Viz: vision service did not capture image |
| 2037 | Mech-Viz: no vision results from vision service |
| 2038 | Mech-Viz: no point cloud in ROI in vision results |
| 2039 | Mech-Viz: no vision point for planning |
| 2040 | Mech-Viz: failed to plan paths for some of vision points from vision result reuse |
| 2041 | Mech-Viz: vision service not registered |

### Mech-Viz Normal Status Codes

| Code | Meaning |
|---|---|
| 2100 | Mech-Viz: successfully executed |
| 2101 | Mech-Viz: successfully stopped execution |
| 2102 | Mech-Viz: successfully sent DO list |
| 2103 | Mech-Viz: successfully started |
| 2104 | Mech-Viz: successfully stopped |
| 2105 | Mech-Viz: branch successfully set |
| 2106 | Mech-Viz: index successfully set |
| 2107 | Mech-Viz: target point of outer_move Task successfully set |

### Mech-Center

### Mech-Center Error Codes

| Code | Meaning |
|---|---|
| 3001 | Mech-Center: illegal command |
| 3002 | Mech-Center: interface command length or format error |
| 3003 | Mech-Center: client disconnected |
| 3004 | Mech-Center: server disconnected |
| 3005 | Mech-Center: calling Mech-Vision timed out |
| 3006 | Mech-Center: unknown error |
| 3007 | Mech-Center: data acknowledge signal timed out |

### Mech-Center Normal Status Codes

| Code | Meaning |
|---|---|
| 3100 | Mech-Center: client connection normal |
| 3101 | Mech-Center: server connection normal |
| 3102 | Mech-Center: waiting for client to connect |

### Hand-Eye Calibration

### Hand-Eye Calibration Error Codes

| Code | Meaning |
|---|---|
| 7001 | Calibration: returned argument error |
| 7002 | Calibration: Mech-Vision did not output calibration point |
| 7003 | Calibration: robot failed to reach calibration point |

#### Hand-Eye Calibration Normal Status Codes

| Code | Meaning |
|---|---|
| 7100 | Calibration: robot successfully reached calibration point |
| 7101 | Calibration: Mech-Vision normally output calibration point |

### 3.2.7 Appendix

#### Use Mech-Viz for Collision Detection

If collision detection is required, please build the Mech-Viz project by referring to the sample project at *Mech-Center/tool/viz_project/check_collision*.

Please note:

1. **check_collision** is only a sample project. In the project, except for move Tasks, the Tasks in the workflow are not supposed to be deleted or modified in their positions in the workflow.
2. Please select the actual robot model in use for the project.
3. The move Tasks can be deleted, added, or modified according to the actual needs.

#### Use Mech-Viz for Controlling Suction Cup Sections or Array Gripper

If controlling multiple suction cup sections or an array gripper, please build the Mech-Viz project by referring to the sample project at *Mech-Center/tool/viz_project/suction_zone*.

Please note:

1. **suction_zone”** ** is only a sample project. In the project, except for move Tasks, the Tasks in the workflow are not supposed to be deleted or modified in their positions in the workflow.
2. Please select the actual robot model in use for the project.
3. The move Tasks can be deleted, added, or modified according to the actual needs.
4. Please configure the suction cup file in the project.
5. DO list can only be obtained after image capturing.

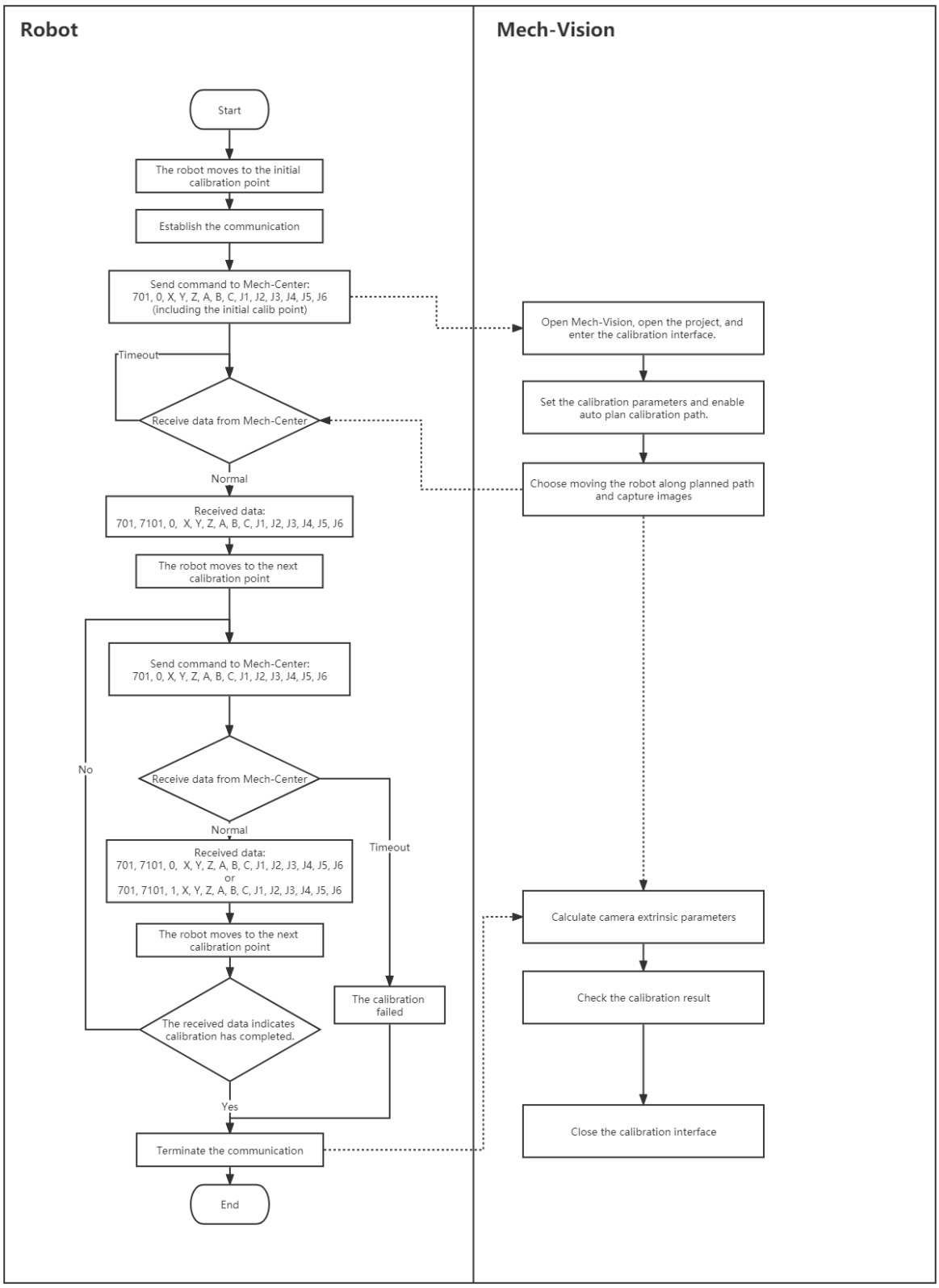
#### Map String Labels to Integer Labels in Mech-Vision

The Step to map string labels to integer labels in Mech-Vision is label_mapping.

Label mapping can be done by configuring a mapping file in the Step’ s parameters. A sample label mapping file is as follows:

```
{
    "Medium", "3",
    "Large", "2",
    "Small", "1"
}
```

## Flowchart of Calibration



**Note:**

1. The robot pose sent and received can be either a flange pose ([X, Y, Z, A, B, C]) or JPs ([J1, J2, J3, J4, J5, J6]).
2. The last point sent from Mech-Vision is the initial calibration point input to Mech-Vision at first, telling the robot to move back to the initial position, and no image capturing will be performed for the last point sent.
3. In the data sent from Mech-Vision, the third argument (value: 0 or 1) indicates whether the calibration has finished. 0 means not finished and the next calibration point is on the way; 1 means the calibration has finished.

#### Add Signal for Exposure Completion in Mech-Vision

For Profinet and Ethernet/IP, a camera Exposure Complete signal can be used to shorten the system cycle time.

When the Mech-Vision project takes a long time to run, the system cycle time can be shortened by moving the robot immediately after camera exposure.

In the Mech-Vision project, please make the following modification to implement the Camera Exposure Complete signal:

1. Add a notify_vision Step, and connect it to the control flow port of the capture_images_from_camera Step.
2. For the capture_images_from_camera Step, set the parameter **Trigger Control Flow When Output** to True.
3. Name the notify_vision Step to "Standard Interface Notify", and set the message content to "1001". Please do not change the content later.

After the settings above, when the camera finishes exposure, an Exposure Complete signal will be sent, and please reset the signal using Exposure Complete Reset after receiving it.

If Mech-Center does not receive the reset signal for over 10 seconds, it will raise an error message: **Mech-Center data confirmation signal timeout**.

If more types of data need to be transmitted between the outside and the Mech-Mind Software Suite, please use the **Adapter**. The Adapter can be used for communication customization, but the customization involves higher time and labor costs.

## 3.3 Adapter

When the Standard Interface cannot meet customer needs, a customized Adapter is needed to transmit data.

Adapter can be customized according to the different needs of customers. The communication program is divided into two parts, for the robot and the software suite respectively. The part of robot needs to be manually written, and the part of software suite is generally generated by the Adapter generator. For specific usage, please see *Adapter Generator*.

### 3.3.1 Adapter Enable Method

Under the *Deployment Settings —Mech-Interface* tab, check *Enable Mech-Interface*. The interface is shown in the figure below.

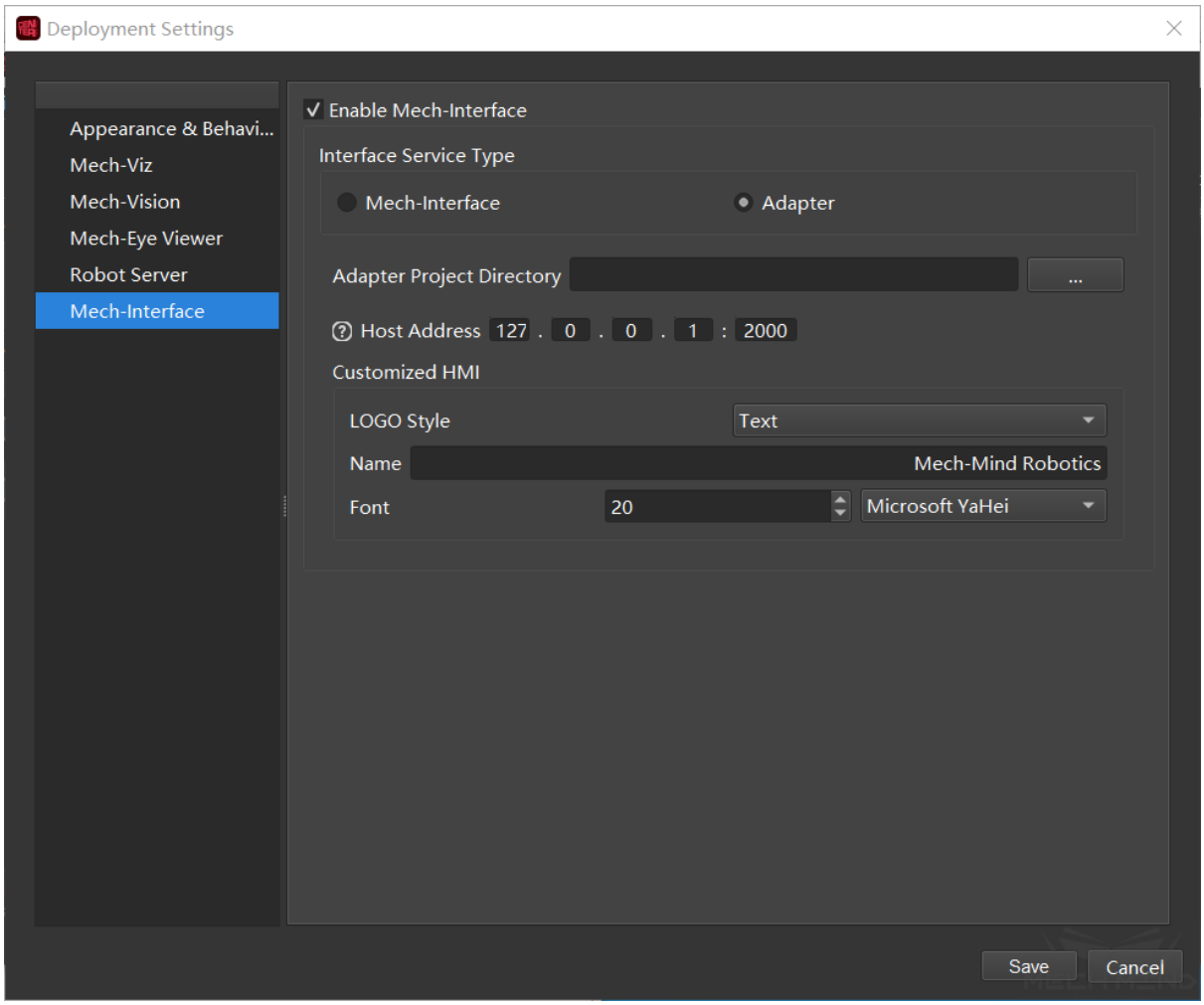



Figure 1 Enable Adapter

- Interface service type: Select Adapter.
- Adapter project folder: Select the folder path where Adapter is located.
- Host address (and port): If the other party is a Client, fill in 0.0.0.0; if the other party is a Server, fill in the other party' s IP. The port must be consistent with the other party.
- Custom LOGO style: text is selected by default. If there are no special requirements, you can ignore it.

After the setting is complete, save and restart Mech-Center. After restarting, click *Start interface service* on the interface to enable the Adapter.

**Tip:** For the detailed interface protocol in the Adapter, please refer to the Adapter Programming Guide

under the Mech-Center installation path. The specific address is *center\docs\en\Adapter Programming Guide*.

## 3.4 Adapter Generator

The adapter generator is a component integrated in Mech-Center.

Use the Adapter generator to get an Adapter that only sends visual points through a series of settings. If the actual situation is complicated, you can simply modify the generated code to meet the actual needs according to the actual situation.

The following will introduce how to use the Adapter generator.

**Tip:** For the convenience of configuration, the corresponding component has its detailed description. You can hover the mouse cursor over the component to view it.

### 3.4.1 How to Use the Adapter Generator

#### Adapter Generator Location

The location of the Adapter generator is as follows *Figure 1*as shown.

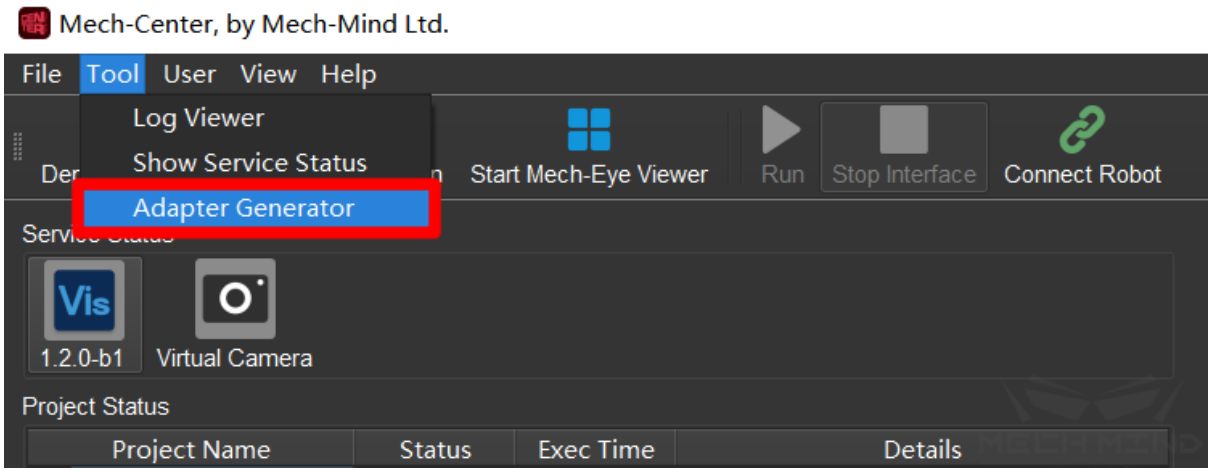



Figure 1 Adapter generator location

### Network Configuration-Server or Client

In this step, you need to set the name of the Adapter, the Client or Server, and the communication format. The interface is shown in *Figure 2*. Click *Next* in the lower right corner when finished.

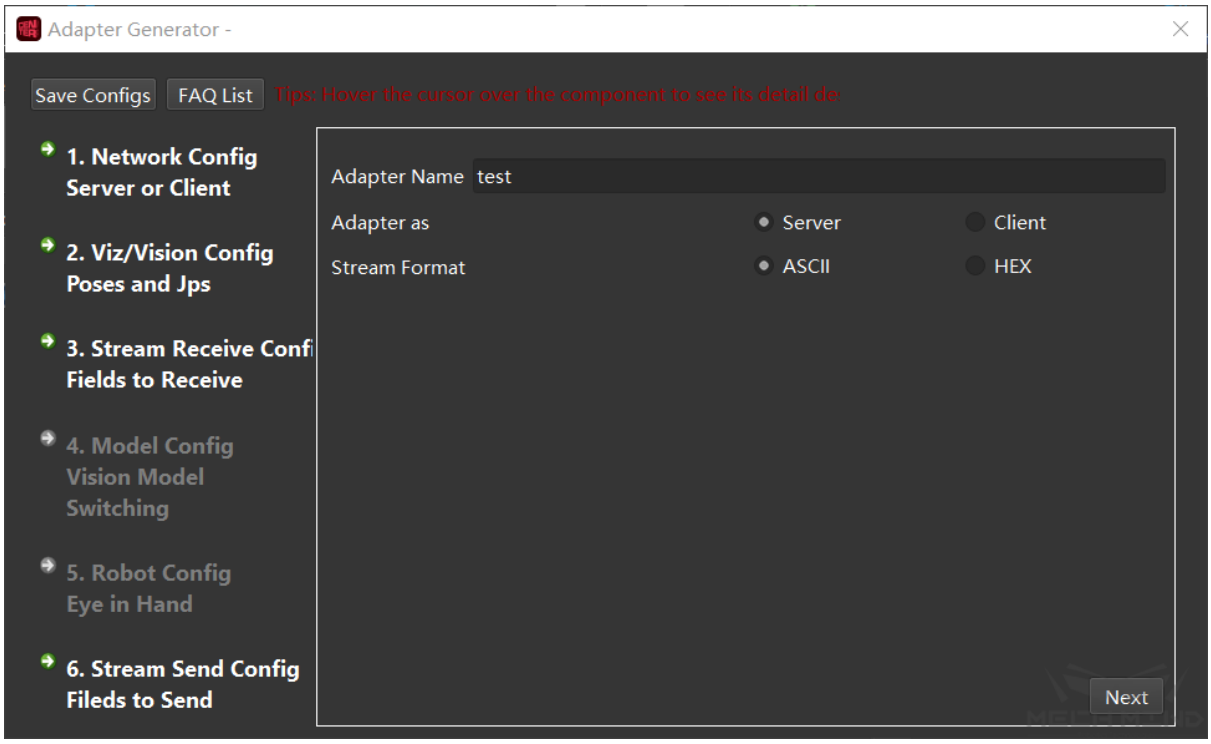



Figure 2 Network configuration-Server or Client

- Choose Server or Client: You need to configure the corresponding adapter host IP and port in *Settings*.
- Bind port: It only takes effect in Client mode. If the Server has port restrictions on the Client, check this option.
- Select the communication format: The options are ASCII string and hexadecimal. Hexadecimal (HEX) needs to specify the endianness.

### Mech-Viz/Mech-Vision Configuration-Pose and Joint Angle

This step is to set the number and form of poses. The interface is shown in *Figure 3*. Click *Next* in the lower right corner when finished.

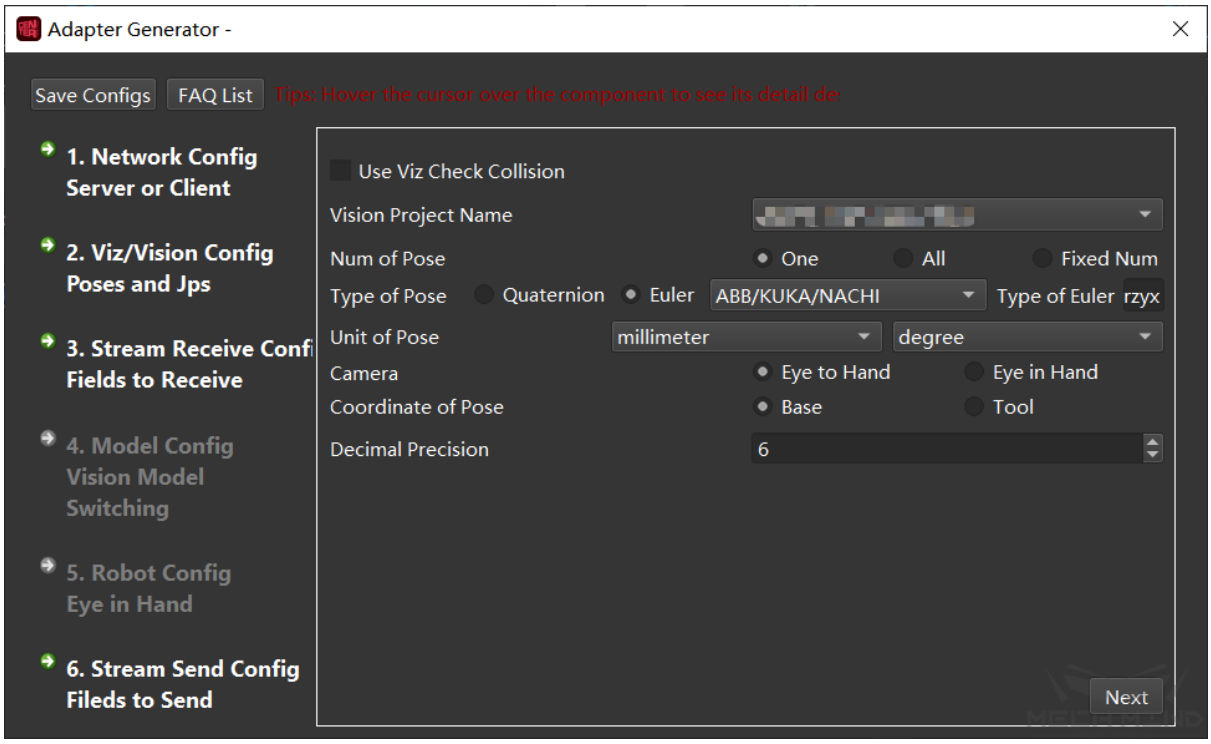



Figure 3 Mech-Viz/Mech-Vision configuration-pose and joint angle

- Vision project Name: It is necessary to configure the Vision program path and Vision project path in *Settings* in advance.
- Num of Pose: Select the number of poses to be sent to the other party.
- Type of Pose: You can choose quaternion or Euler angle.
- Unit of Pose: Use millimeters and degrees in general situation.
- Camera: Choose the camera installation method. There are three methods which are ETE, ETH and EIH.
- Coordinate of Pose: Determine which coordinate system the sent Pose point is based on. In general, Pose is based on the Base Coordinate System. Pose can only be based on the Tool Coordinate System if the robot unable to give the end pose of the robot when in EIH.
- Decimal Precision: Determine the decimal places of the sent Pose point. The maximum number of digits is 10.
- Use Mech-Viz Check Collisions: After checking it, the visual points are calculated by Mech-Viz detection, filtering the points that failed in the planning, and filtering out the collision-free grasping poses. This option requires Mech-Viz to create a project. The Mech-Viz project needs to be at least from the home point to visual_move. You can refer to the example project check_collision in /tool/viz_project under the Mech-Center installation path.

**Attention:** As shown in the sample project, the photo is triggered by visual_look, and non-moving tasks must exist, and the name cannot be changed, including: notify_1, notify_2, visual_look_1.

- Mech-Viz Config for Generator: The interface appears after clicking "Use Mech-Viz to Check Collisions", as shown in *Figure 4*. Click *Save* in the lower right corner when finished.

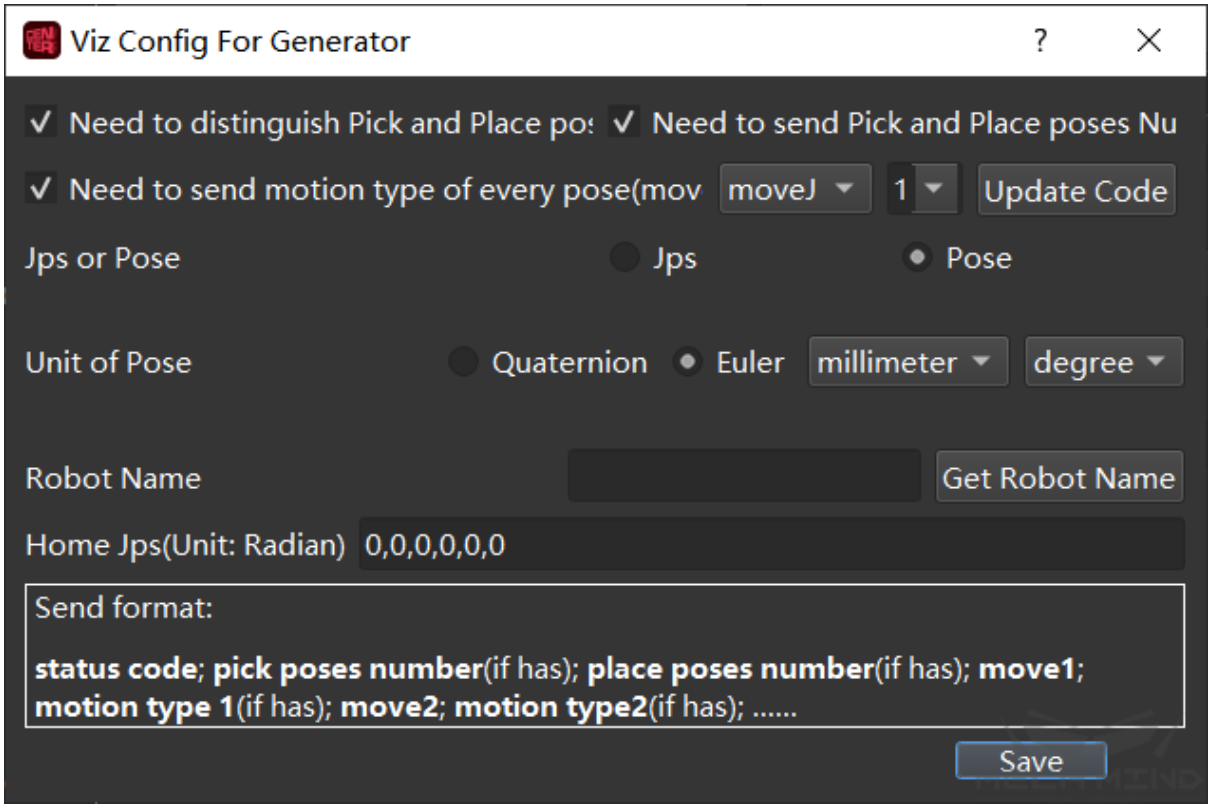



Figure 4 Mech-Viz config settings

1. Need to distinguish Pick and Place poses: pick points are all points before *visual movement* (including visual movement), and placement points are all points after *visual movement*. In some scenarios, robots may need to distinguish between grabbing actions and placing actions according to their tasks.
2. Need to send Pick and Place poses Number: If the number of points is large, you can bring the quantity field of the pick points and placement points. After checking, this field will be brought only if the default number of points is greater than one.
3. Need to send motion type of every pose: The motion mode of the mobile task in Mech-Viz is divided into joint motion or linear motion.
4. Update Code: The default joint motion corresponding code is 1, and the linear motion corresponding code is 2. The code can be customized, and the updated value will take effect after the change.
5. Jps or Pose: The way to send pose, Jps is used by default. If you choose Pose, you need to check *Send tool pose* in the "Other" tab in Mech-Viz. As shown in the figure:

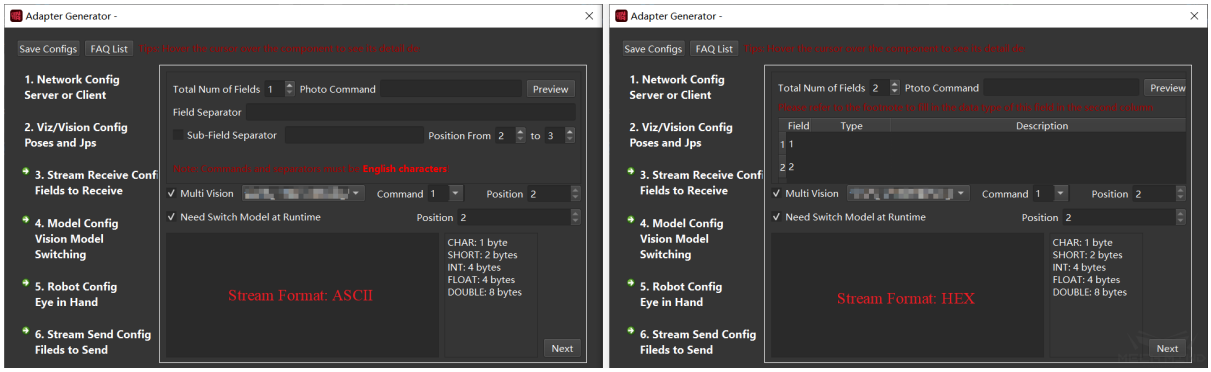



Figure 5 Setting the pose of the sending tool

6. Unit of Pose: Set the corresponding unit according to the selected sending format. The unit is generally degrees and millimeters.
7. Robot Name: Using Mech-Viz to simulate robot motion requires a real robot service. The generated Adapter will simulate this service. The robot name is the name of the service and needs to be consistent with the robot name in Mech-Viz. You can load the project in Mech-Viz and check *Automatically load current project*, then return to the settings and click *Get robot name* to successfully add the robot name.
8. Home Jps(Unit: Radians): It refers to the reference origin of the movement in Mech-Viz, the unit is radians, separated by commas. You can edit a *movement* from Mech-Viz as the origin and copy the joint position of the origin.

### Receiving Data Format Configuration-receiving Field

This step is to set the format of the receiving field. The interface is shown in *Figure 6*. What needs to be set are: photo instruction, multiple projects (instruction code), dynamically switch templates (template instruction code), as well as the total number of fields, field types, field separators and sub-field separators. Click *Next* in the lower right corner when finished.

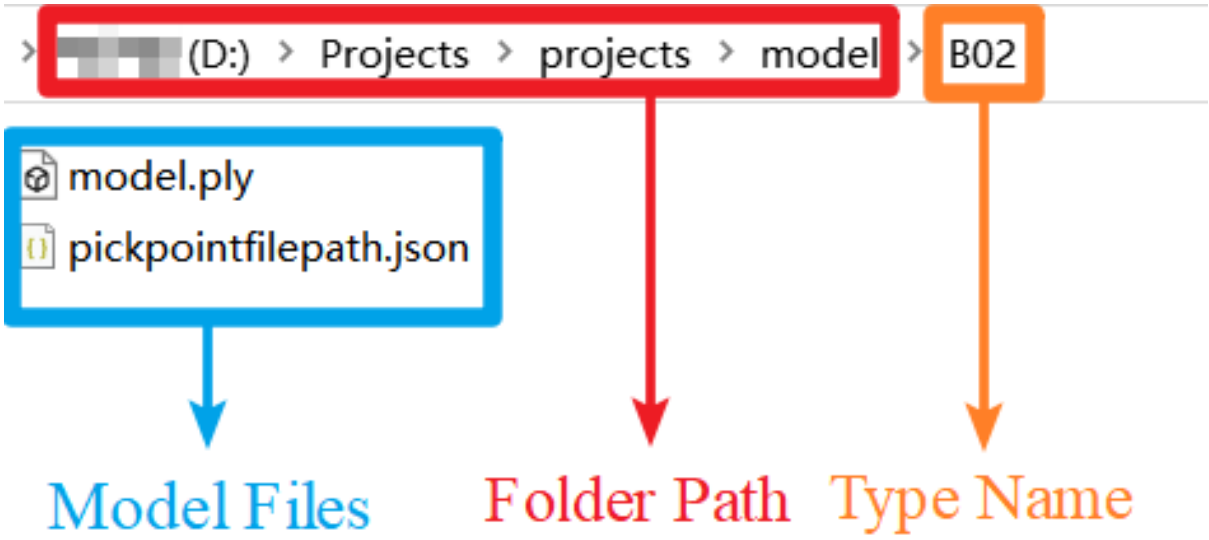



Figure 6 Received data format configuration-receive field

- Photographing instruction: External send a photographing instruction to the Mech-Mind Software Suite, so that the camera can take photos. When the communication format is ASCII code, it is recommended to use letters, such as letters *p* and the field position is 1 by default. When the communication format is hexadecimal (HEX), an integer in hexadecimal form is required, such as *0xff* or *ff*.
- Multiple projects (instruction code): This setting is optional. When there are multiple Vision projects in a project, different Vision projects need to be called according to external instructions, and the instruction code is configurable.

**Note:** Each project corresponds to a unique command code, and the field position is unique, and cannot overlap with other fields.

- Need to dynamically switch the template (template instruction code): this setting is optional. The dynamic switch template means that the object to be recognized has multiple models, and the template file of the current model needs to be dynamically switched during operation.

**Note:** The field position is unique and cannot overlap with other fields.

- Total number of fields: related to the number of parameters that need to be set, and the value range is 1~10. There must be a camera instruction in the field.
- Field type: It needs to be set when the communication format is hexadecimal (HEX). The available types are CHAR, SHORT, INT, FLOAT, DOUBLE.
- Field separator and sub-field separator: They need to be set when the communication format is ASCII. If there are more than two fields, you need to fill in the field separator; if there are

additional separators in the additional information, the sub-field separator is also necessary, and you can specify the start and end range of the sub-field.

### Template Configuration-Vision Template Switching

When the *Need to dynamically switch templates* option is checked in the previous step, this configuration needs to be set. Click *Next* in the lower right corner when finished.

- Template folder: The folder where templates are stored according to the type. The folder hierarchy is shown in *Figure 7*.

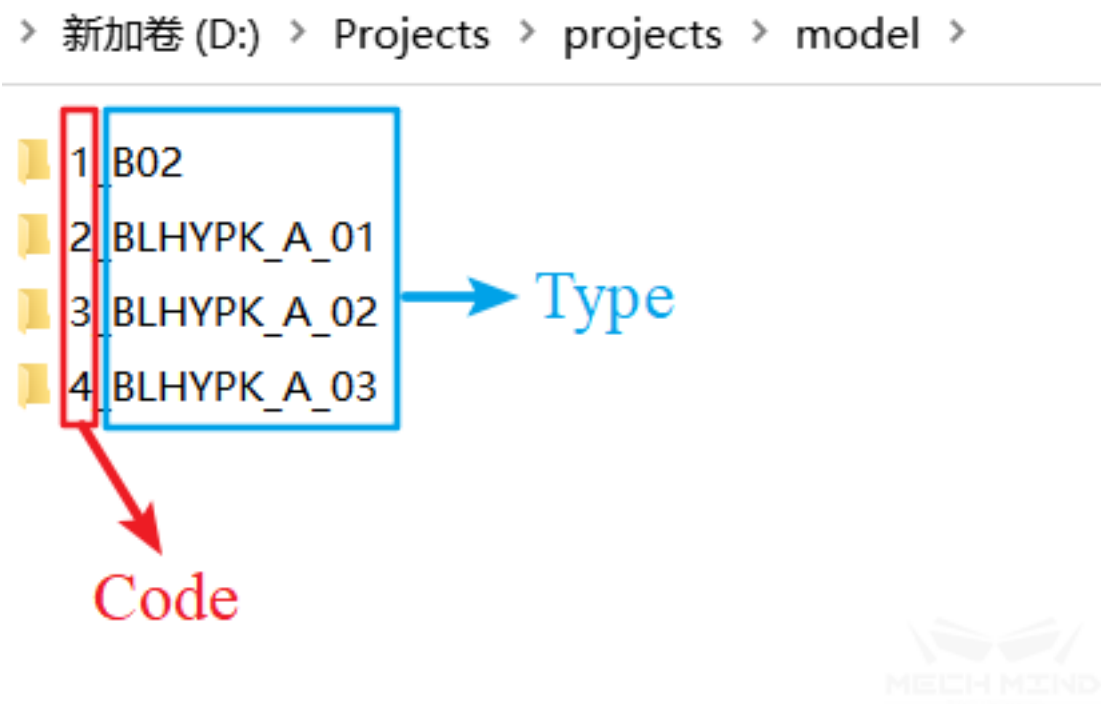



Figure 7 Template folder

- Type and type code: After loading the template folder, all types under the folder will be parsed into the type drop-down box. If you need the corresponding type code, please name the template subfolder according to the rule: code_type, as shown in the figure below:

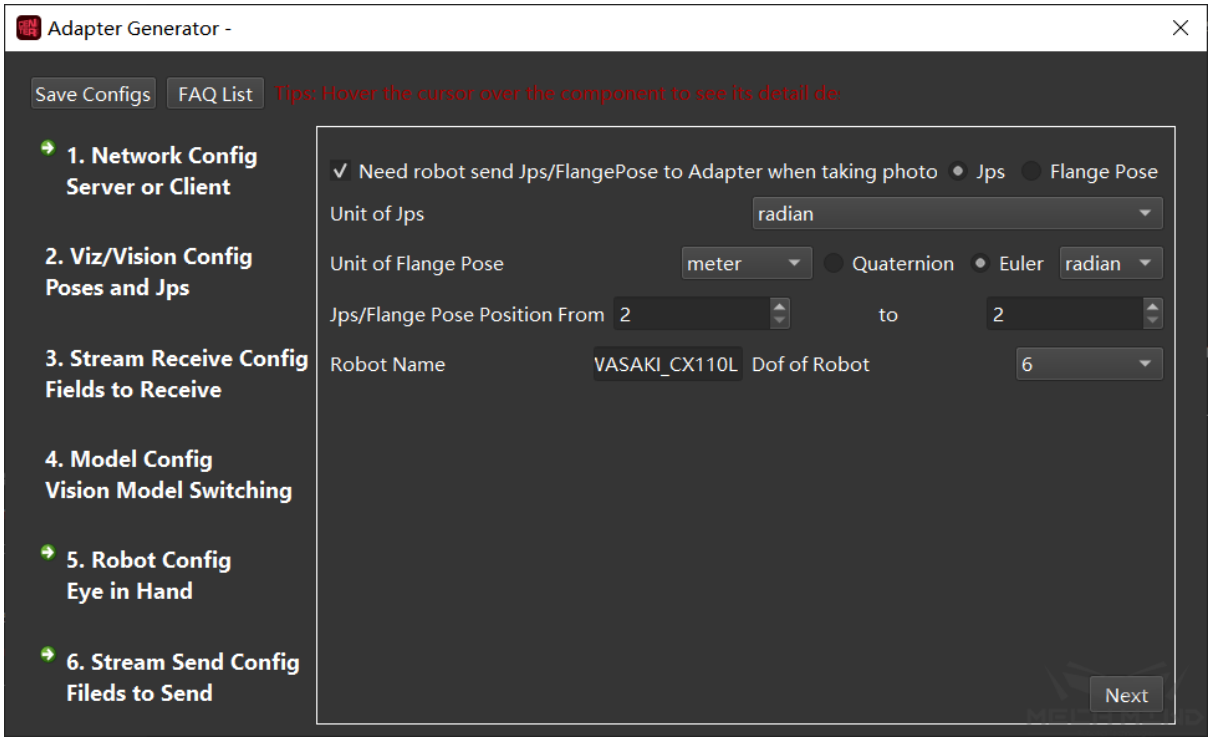



Figure 8 Type and type code

- Switching templates on the interface: After selecting and starting the Adapter, a sub-interface will be generated, and the templates can be switched manually.
- Step name: Need to set the name of the Step of the template, which means the step name of the corresponding step in Mech-Vision.
- Step type: It indicates the type of corresponding step, currently only supports 3d_coarse_matching, 3d_fine_matching, 3d_coarse_matching_multiple_models, 3d_fine_matching_multiple_models, map_to_multi_pick_points, read_object_dimensions, read_poses_from_file, add_labels_to_poses, each step type has corresponding attributes, and you can select attributes according to your needs.
- After selecting attributes, click + or - to add or delete steps.

**Attention:** In order to name the template files uniformly, please name the template files with these attribute names. For example, the point cloud template file, please name it modelfile.ply, and the pick point please name it pickpointfilepath.json, and it is not case-sensitive.

### Robot Configuration-Eye in Hand

In the case of Eye in Hand, if the customer needs us to provide a Pose based on the Base coordinate system, the robot needs to provide the joint angle or flange posture when taking pictures. This step will set the pose format of the robot when taking pictures. The interface is shown in *Figure 9*. Click *Next* in the lower right corner when finished.

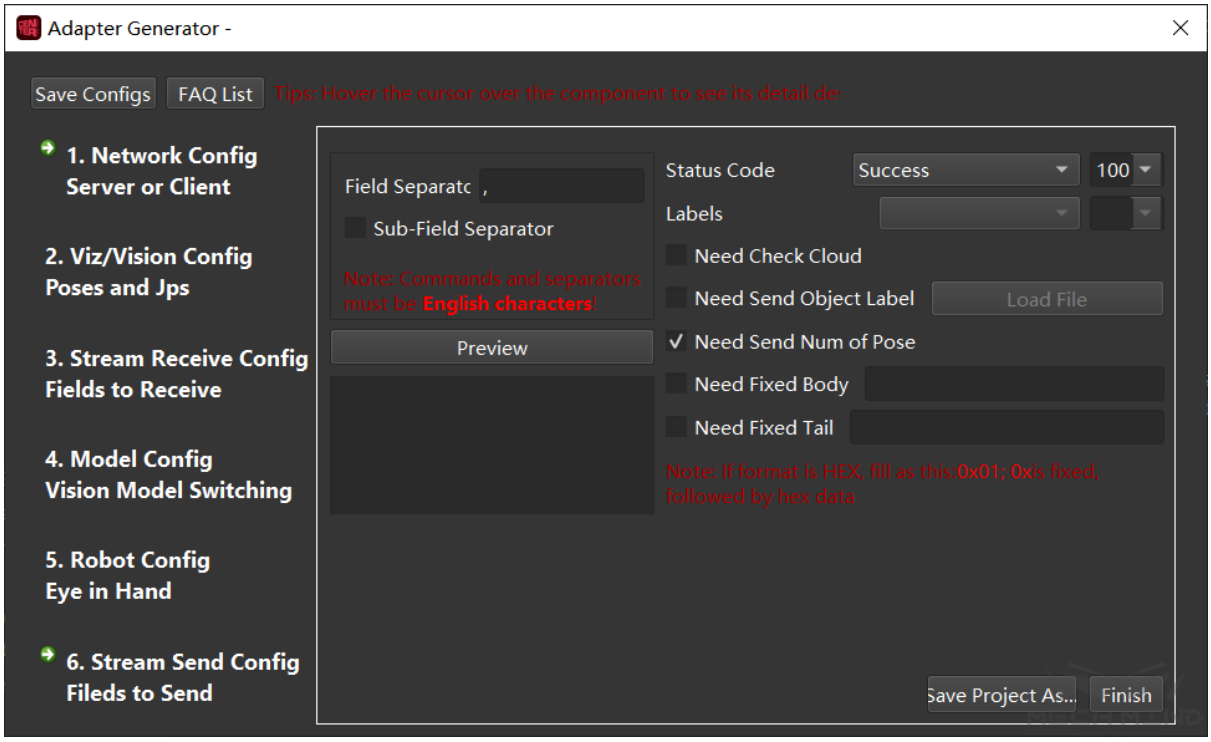



Figure 9 Robot configuration-Eye in Hand

- Pose form: You can choose joint angle or flange pose.
- Joint angle unit: Degree or radian can be selected.
- Flange pose unit: Position can choose meters or millimeters, attitude (rotation) can choose quaternion or Euler angle, Euler angle unit can choose degrees or radians.
- Joint angle/flange attitude field position: It is the position of the start and end fields of the pose in the total field.

**Attention:** The index position starts counting from 1, and the index position 1 is the camera instruction!

- Robot name: The name used to identify the robot service, which needs to be consistent with the robot name in Mech-Viz.
- Robot degree of freedom: currently supports 4-axis and 6-axis robots, select the corresponding robot degree of freedom according to the actual project.

### Send Data Format Configuration-send Field

This step is to set the format of sending Pose. The interface is shown in *Figure 10*. Click *Next* in the lower right corner when finished.

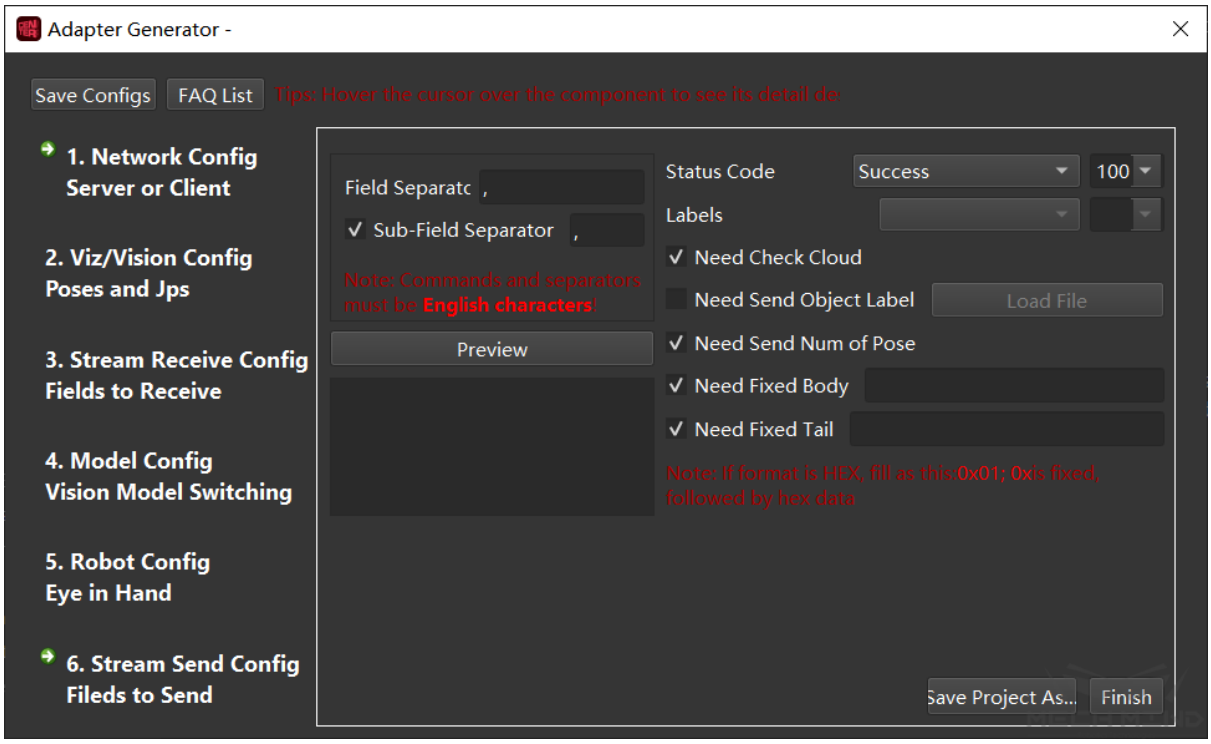



Figure 10 Sending data format configuration-sending field

- Field Separator and sub-field separator: Set the separator form.
- Status Code: Set the sending status, each status corresponds to a unique status code.
- Need Check Cloud: After checking, the point clouds will be checked, if the point clouds does not exist, the corresponding status code will be output.
- Need Send Object Labels: Sending object labels means to send to the other party according to the label recognized by Vision, each label is connected to the Pose; when the other party is inconvenient to parse the label string, you can also specify the code of the corresponding label, which needs to load the label file of all label strings. It should be noted that the label file format must be a json array format.
- Need Send Num of Pose: Send the number of Pose of this time.
- Need Fixed Body: When the vision is not recognized, send a message to the other party (the message after the error code).
- Need Fixed Tail: After checking it, a fixed tail mark will be added after the data.

**Attention:** When the communication format is hexadecimal (HEX), it is necessary to set the status code, the number of poses, and the numeric type of poses.

After configuring all the above settings, click *Finish* or *Save project as* to save the Adapter.