



Programming manual
Vision Sensor Toolbox

GB

O2x

KUKA LBR iisy

Inhaltsverzeichnis

1	Preliminary note	3
1.1	Symbols used	3
1.2	Legal and copyright information	3
1.3	Disclaimer of warranties	3
1.4	Open source information	4
2	Safety instructions	5
3	Intended use	6
4	System requirements	7
4.1	Compatible devices	7
5	Toolbox setup	8
5.1	Installing and configuring the toolbox	8
5.1.1	Installing the toolbox	8
5.1.2	Adding the capability to the world scene	8
5.1.3	Configuring the capability	9
5.2	Setting the device configuration	11
5.2.1	Configuring network settings	11
5.2.2	Configuring interface settings	12
5.3	Application setup	12
5.3.1	Adding a new application to the device	13
5.3.2	Configuring the trigger mode for the application	13
5.3.3	Robot sensor calibration	14
5.3.4	Creating a new model	14
5.3.5	Selecting the interface presets	14
5.3.6	Logic layer configuration	15
5.4	Object quality inspection	16
5.5	Robot pick & place	16
5.5.1	Offsetting object coordinates	17
5.6	Classification	18
5.7	ifm 2D Vision Sensor Toolbox Methods	20
5.7.1	IFM_TriggerSensor	20
5.7.2	IFM_ActivateApplication	21
5.7.3	IFM_WritePCICInputStringData	22
5.7.4	IFM_ReadStringOutputData	23
5.7.5	IFM_RequestIOLogicState	24
5.7.6	IFM_SetIOLogicState	24
5.7.7	IFM_ExecuteButtonFunctionality	25
5.7.8	IFM_GetCurrentError	26
5.7.9	IFM_GetObjectPose	27
5.7.10	IFM_GetReadingResult	27
5.8	Example programs	28
5.8.1	Robot Pick & Place example	29
5.8.2	Object inspection example	29
5.8.3	Classification example	30
5.8.4	Sharing string data between devices example	30
5.8.5	Setting and reading IO states example	31
5.8.6	Handling error codes example	31
5.9	Error codes	32

1 Preliminary note

You will find instructions, technical data, approvals and further information using the QR code on the unit / packaging or at documentation.ifm.com.

1.1 Symbols used

- ✓ Requirement
- Instructions
- ▷ Reaction, result
- [...] Designation of keys, buttons or indications
- Cross-reference
-  Important note
Non-compliance may result in malfunction or interference.
-  Information
Supplementary note

1.2 Legal and copyright information

© All rights reserved by ifm electronic gmbh. No part of these instructions may be reproduced and used without the consent of ifm electronic gmbh.

All product names, pictures, companies or other brands used on our pages are the property of the respective rights owners.

1.3 Disclaimer of warranties

ifm electronic gmbh disclaims to the fullest extent authorized by law any and all warranties, whether express or implied, including, without limitation, any implied warranties of title, non-infringement, quiet enjoyment, integration, merchantability or fitness for a particular purpose.

Without limitation of the foregoing, ifm expressly does not warrant that:

- the software will meet your requirements or expectations,
- the software or the software content will be free of bugs, errors, viruses or other defects,
- any results, output, or data provided through or generated by the software will be accurate, up-to-date, complete or reliable,
- the software will be compatible with third party software,
- any errors in the software will be corrected.

Demo software and templates

Demo software and templates are provided “as is” (that is: excluding warranty) and “as available”, without any warranty of any kind, either express or implied. The user acknowledges and agrees to use the software at user’s own risk. In no event shall ifm be held liable for any direct, indirect, incidental or consequential damages arising out of the use of or incorrect use the software. The user may use the software solely for demonstration purposes and to assess the software functionalities and capabilities.

Customer-specific software

1. The software created and used has been put together by ifm especially for the customer using modular software components made by ifm for numerous applications (standard software modules) and adapted to the contractual service required (customer-specific application program).

2. Upon complete payment of the purchase price for the customer-specific application program, ifm transfers the non-exclusive, locally and temporarily unrestricted usage right thereof to the customer, without the customer acquiring any rights of any kind to the standard software module on which the individual or customer-specific adaptation is based. Notwithstanding these provisions, ifm reserves the right to produce and offer customer-specific software solutions of the same kind for other customers based on other terms of reference. In any case ifm retains a simple right of usage of the customer-specific solution for internal purposes.
3. By accepting the program, the user acknowledges and agrees to use the software at user's own risk. By accepting the program, the user also acknowledges that the software meets the requirements of the specifications agreed upon. ifm disclaims any and all warranties, in particular regarding fitness of the software for a particular purpose.

1.4 Open source information



For more open source information see: documentation.ifm.com.

2 Safety instructions

- The document is aimed at people with IT expertise.
- The document does not claim to be exhaustive and to apply to all cases, but merely illustrate the technical aspects.
- Whether the presented exemplary configuration can be transferred to the corresponding application must be evaluated and verified by the system administrator.

ATTENTION

Adjustment of the system environment

- ▷ Other applications running on the same system may be affected.
 - ▶ Ensure that other applications are not impaired in their function.
-

3 Intended use

The Programmer's guide describes

- the robot-sensor calibration via the ifm Vision Assistant software,
- the parameter setting of an application for an ifm vision sensor for the following use cases
 - pick & place
 - classification
 - quality Inspection
- The setup and usage of the ifm 2D Vision Sensor Toolbox.

4 System requirements

The following software, hardware and open ports are required to setup and calibrate the ifm vision sensor.

Ports

The following ports must be enabled in the firewall:

- UDP: 3321
- TCP: http: 80 and 8080
- TCP: 50010

Software

The following software is required as a minimum:

- Ifm Vision Assistant software: $\geq 2.8.5$
- O2D5xx sensor firmware: $\geq 1.30.10629$
- O2U5xx sensor firmware: $\geq 1.33.8$
- iiQKA.OS ≥ 1.2

Hardware

The following hardware is required as a minimum:

- marker pattern (can be generated as a printout in the calibration wizard)
- tool tip for approaching the marker points
- KUKA iisy with KR C5 controller

4.1 Compatible devices

In automation technology, vision sensors are an integral part of assembly and production tasks as well as quality control. They combine lighting, lens, camera chip, evaluation and process connections for integration into the higher-level control system in a robust, industrial-grade housing.

The O2D5xx object detection sensor with orientation-independent contour inspection solves tasks such as presence and position control, quality control as well as sorting and counting tasks. It is ideal for applications where the shape of the object is repeatable and defined.

Applications for the O2I5xx multi-code reader range from product tracking and control to product identification. The sensor can read 1D and 2D codes regardless of their position, same as the O2I4xx sensor.

The O2U5xx sensor combines the two worlds of O2D and O2I and greatly expands the range of possibilities.

The term O2X5xx has been created as an umbrella term for all these sensors.

5 Toolbox setup

During setup, the O2X5xx sensor is connected, calibrated and the process interface is configured. All subsequent steps are described in detail in the ifm Vision Assistant software manual. The respective chapter is indicated for each step.

The software manual is available in the [download area of the respective sensor](#) or inside the ifm Vision Assistant via the [Help] tab.

The O2X5xx sensor is connected to the robot via a physical network connection with the controller on port XF5. An Ethernet cable with M12 connector is recommended for data transmission, for example [EVC927](#) (5 m length).



It is advisable to use a switch to ensure communication between the robot and the sensor as well as to the device with the parameter setting software.

5.1 Installing and configuring the toolbox

5.1.1 Installing the toolbox

To install the toolbox on the KUKA iisy, it is necessary to download the toolbox file from the KUKA marketplace. Save the unzipped toolbox on a USB stick. Insert the USB stick into the USB port of the controller.

To install the toolbox to your World scene, take the following steps.

- ▶ Login with the smartPAD.
- ▶ Select the software in the drop-down menu.
- ▶ Install the ifm O2D/O3D Vision Toolbox.

Check if the toolbox is installed in the System Information:

SYSTEM INFORMATION		
INFO	<u>TOOLBOXES</u>	SMARTPAD EULA
	NAME	VERSION
▼	IFM O2X Vision Toolbox ifm efector gmbh	1.0.0
▼	iiQKA OS robot data for lbr3r760-iisy 230V	1.2.6
▼	iiQKA.OS Translations	1.2.6
▼	KUKA iiQKA.Gripper CoAct Toolbox	1.3.3
▼	KUKA iiQKA.Gripper Toolbox	1.3.3

Fig. 1: System information

5.1.2 Adding the capability to the world scene

To add the toolbox to your World scene, take the following steps.

- ▶ Select [Scene] in the drop-down menu.
- ▶ Check that you are in the World Scene. Otherwise, the sensor cannot be added.
- ▶ Select [Devices] in the below drop-down menu.
- ▶ Add the ifm 2D Vision Sensor capability to the World scene.
- ▶ Rename the capability and choose a suitable name like e.g., the sensor model itself (strongly recommended if using more than one sensor).



Feel free to add more devices if required, as shown in the images below (O2D510 and O2D500).

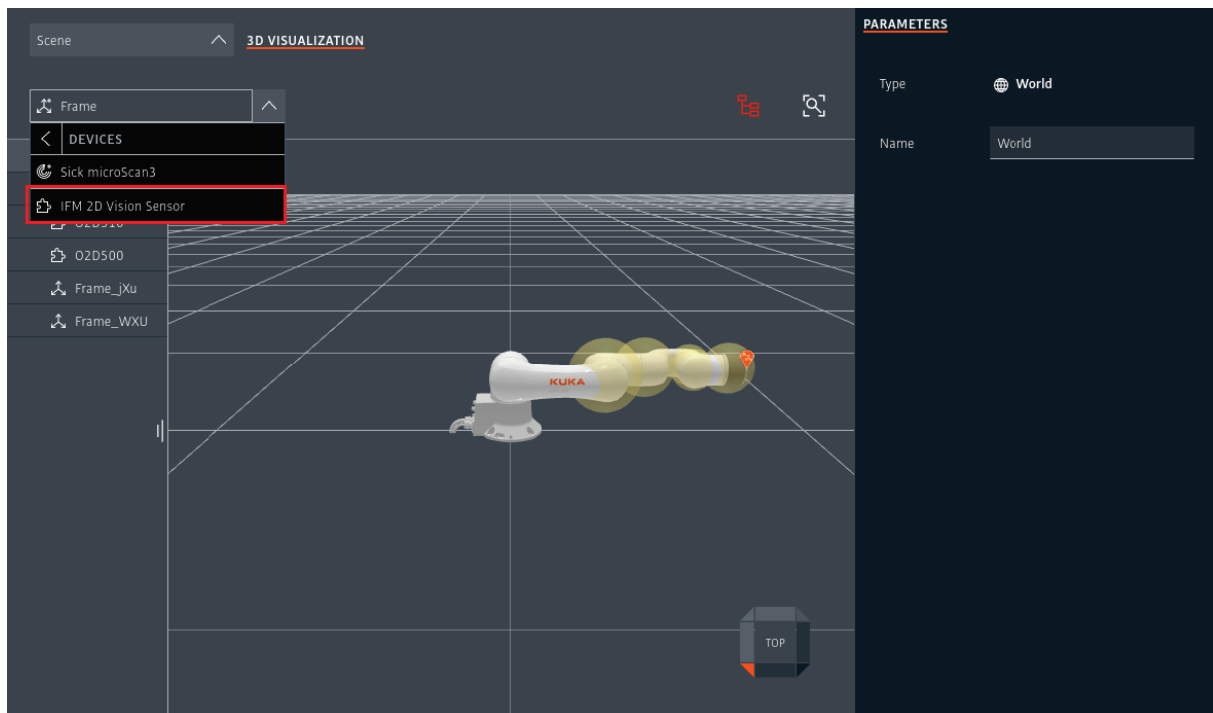


Fig. 2: Adding more devices

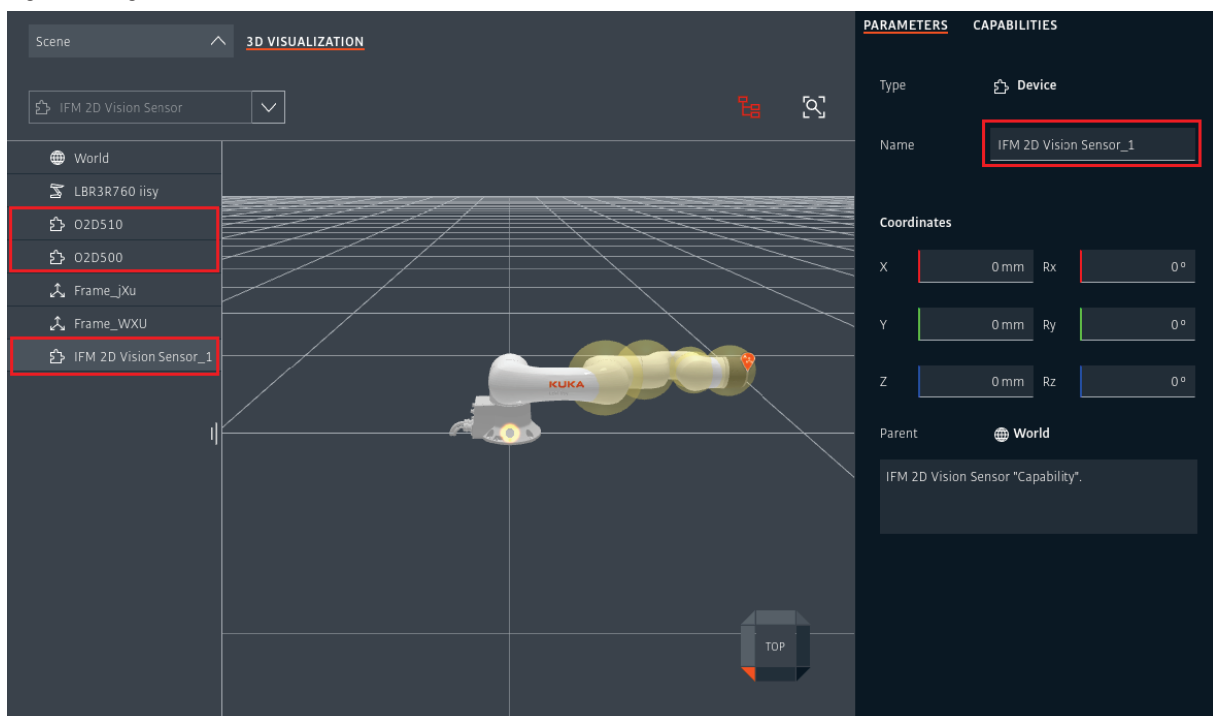


Fig. 3: Adding more devices

5.1.3 Configuring the capability

To configure the toolbox, take the following steps.

- ▶ Select [Setup] in the drop-down menu.
- ▶ Select [Capabilities] in the below drop-down menu.

- Rename the capability and choose a suitable name like e.g., the sensor model itself as prefix (strongly recommended if using more than one sensor).

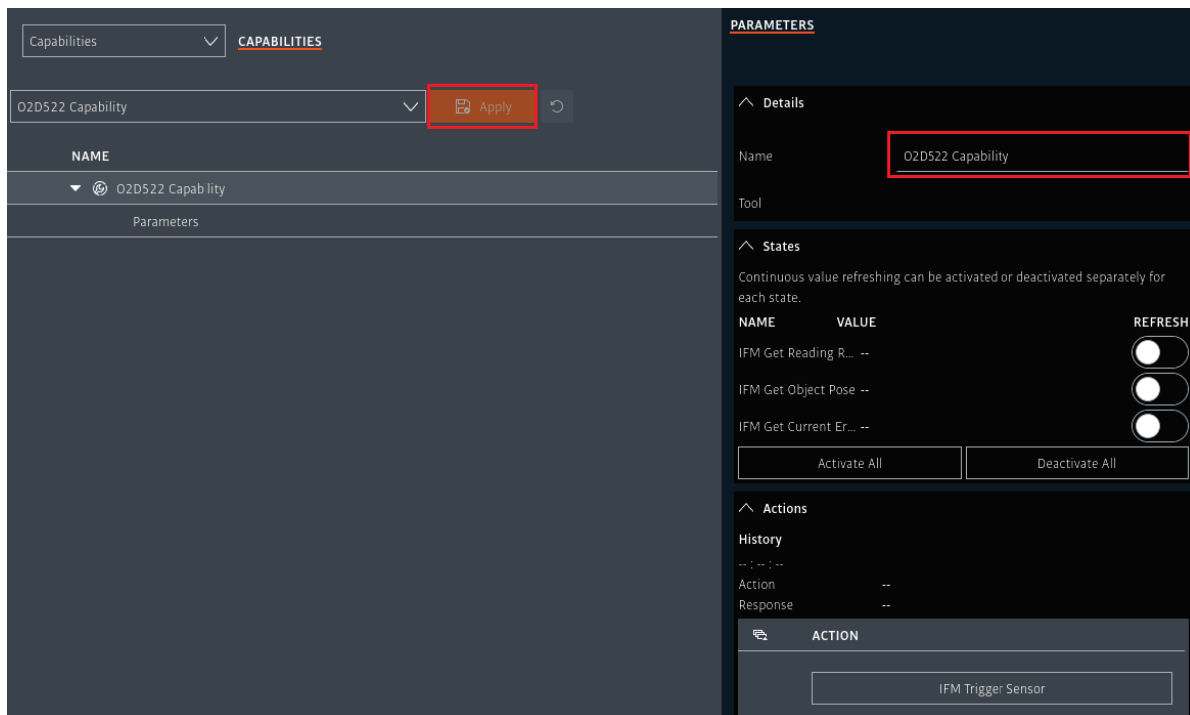


Fig. 4: Choosing a name

- Save the setting by clicking on the [Apply] button.
- Switch to the [Parameters] view.
- Change the Device Ip Address, the Device Tcp Port and create a new frame for the Object Pose or choose an existing one.
- Save the setting by clicking on the [Apply] button.

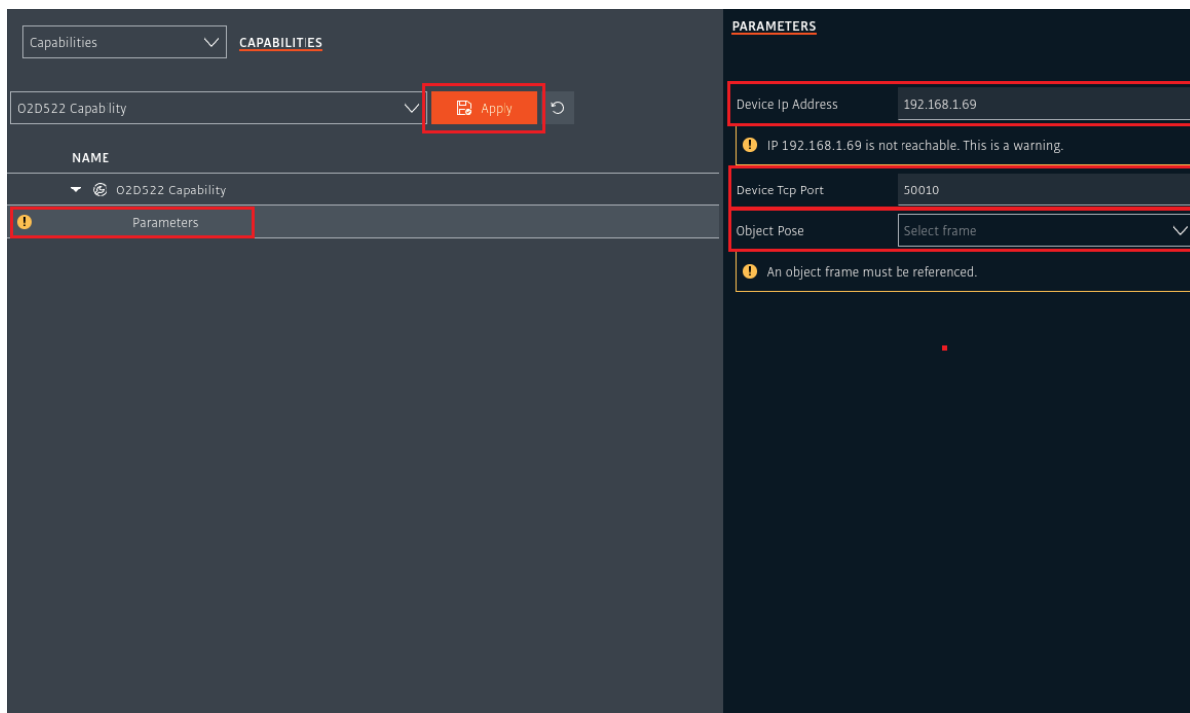


Fig. 5: Settings

5.2 Setting the device configuration

Open the ifm Vision Assistant and connect to the device. You can use the [Find sensor] feature to search for ifm Vision Devices via UDP.

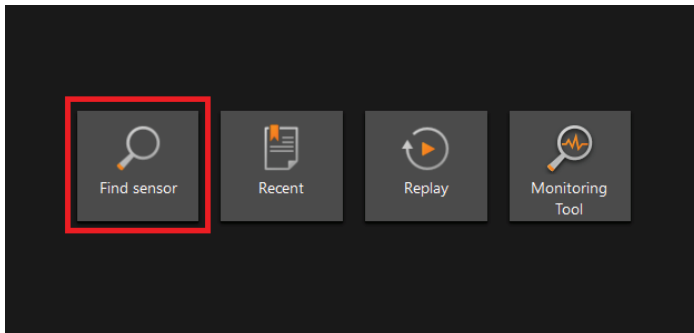


Fig. 6: Find sensor

After connecting to the device, the device settings for this chapter can be made in the [Device Setup] on the left navigation panel in the parameter setting software.

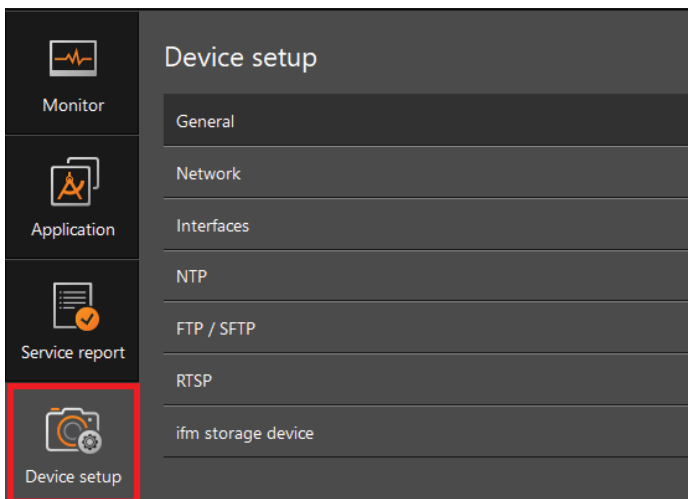


Fig. 7: Device setup

5.2.1 Configuring network settings

Once the sensor and the robot are physically connected, the network settings can be configured on the sensor.

► Set the following [IP address] and [Subnet mask] for the sample program:

▷ Example for IP address: 192.168.1.69

▷ Example for subnet mask: 255.255.255.0

► Select the [Save] button.



The sensor and the robot must be in the same network and the set IP addresses may only be used for the corresponding devices.

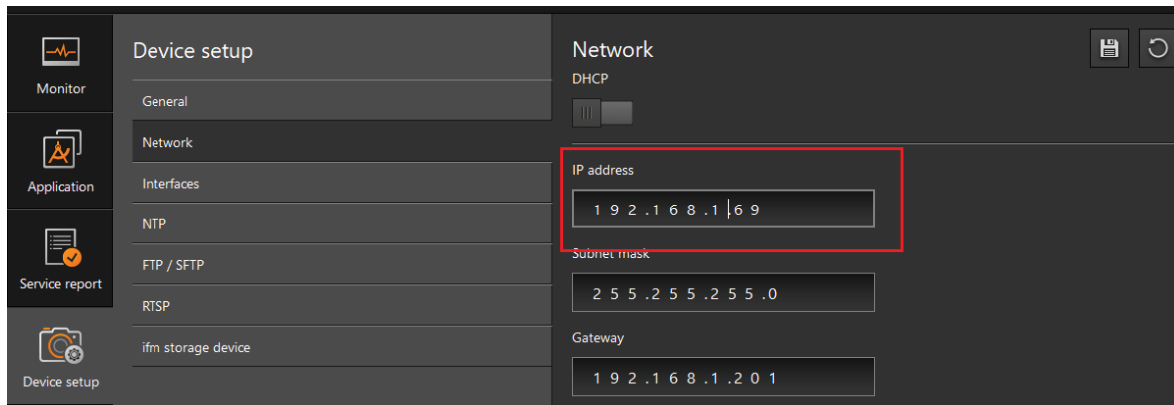


Fig. 8: Network settings

5.2.2 Configuring interface settings

- ▶ Set the following [Process interface version] and the [TCP/IP port for PCIC].
 - ▷ Process interface version: Version 3 (with length and ticket)
 - ▷ Example for port: 50010
- ▶ Activate [PCIC TCP/IP schema auto update] for application switching. In that case, the interface scheme from the application you switched into will not be overwritten by the previous application interface scheme.
- ▶ Select the [Save] button.

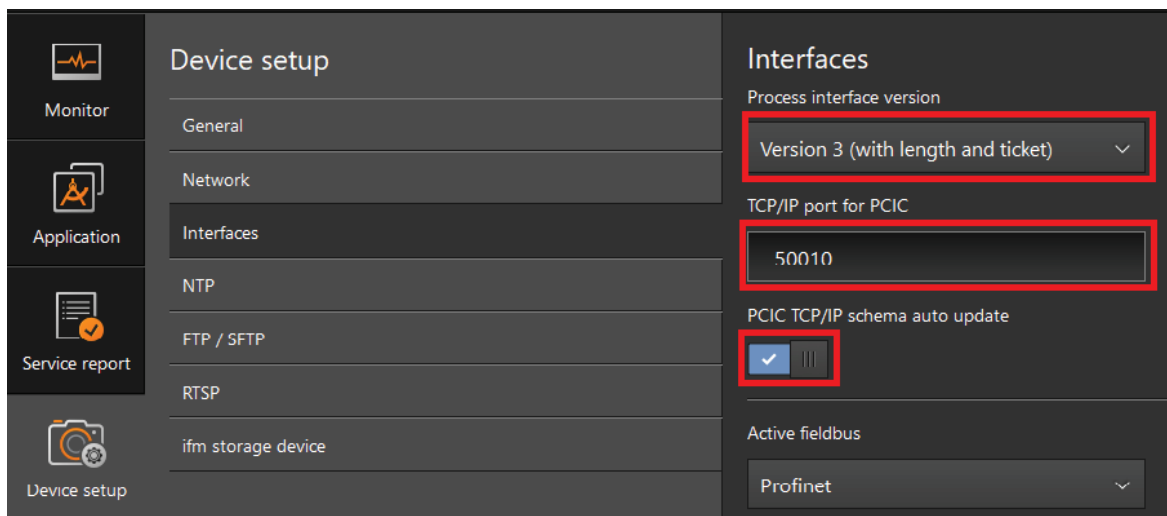


Fig. 9: Interface settings

5.3 Application setup

Depending on your required application, the following applications can be implemented together or individually with the ifm Vision Toolbox and the KUKA iisy cobot in the logic layer. The PCIC interface preset can handle all those applications without doing any changes in the interface output configuration.

- Object Quality Inspection
- Robo Pick & Place (requires Robot sensor calibration)
- Classification

5.3.1 Adding a new application to the device

Add a new application to the device by using one of the following wizards:

- User-defined mode (→ [Configuring the trigger mode for the application](#) 13)
- Measurement and robot calibration (→ [Robot sensor calibration](#) 14)

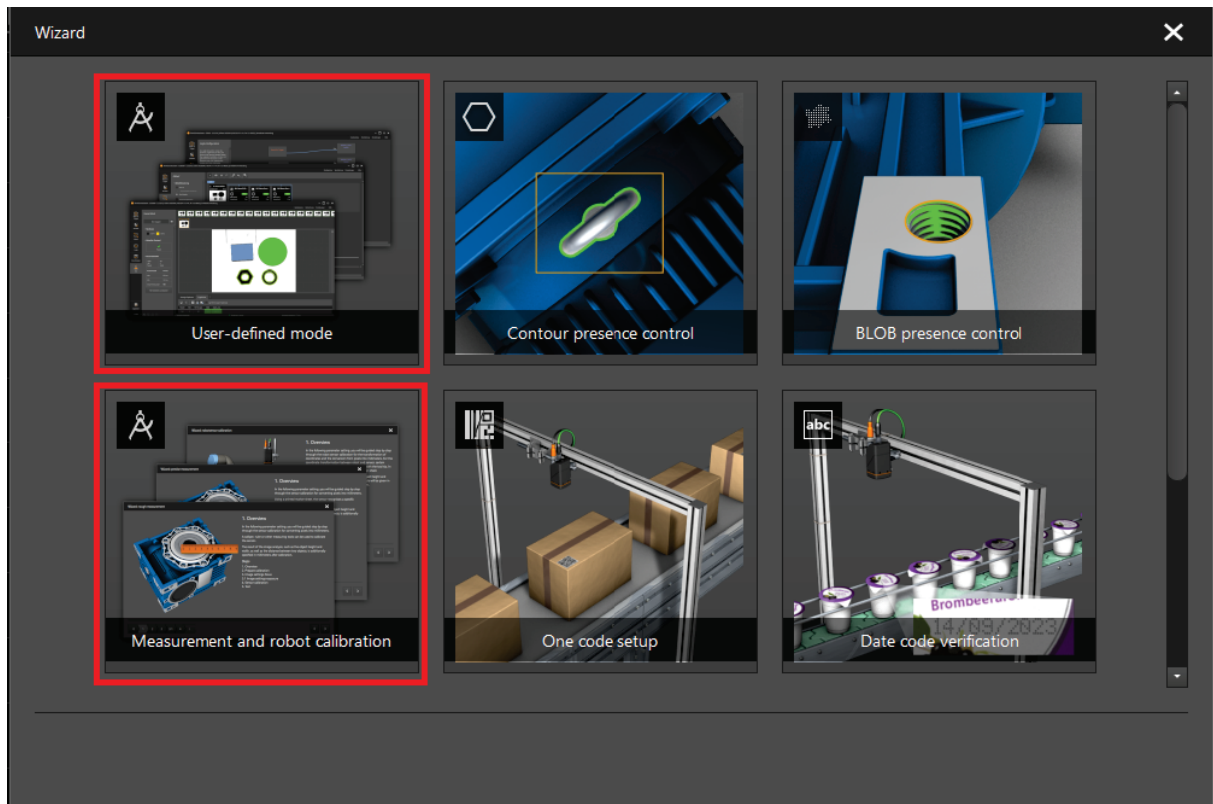


Fig. 10: Adding a application

5.3.2 Configuring the trigger mode for the application

► Select following [Trigger mode] for your application:

▷ Trigger mode: [Process Interface]

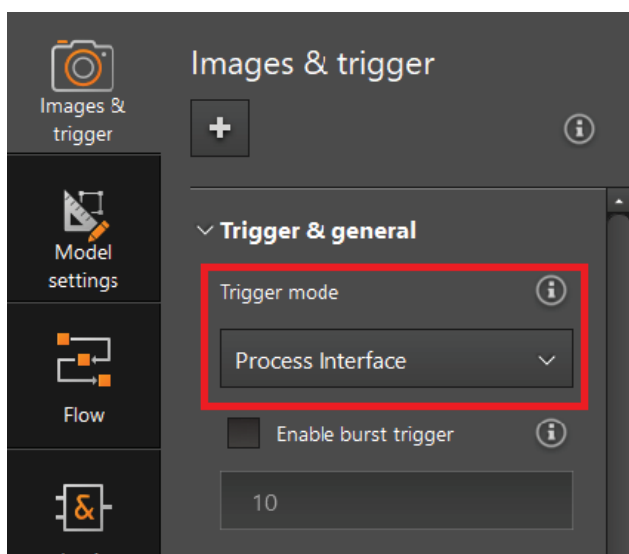


Fig. 11: Trigger mode

5.3.3 Robot sensor calibration

- Follow all calibration steps from chapter **Robot sensor calibration** in the software manual and observe all notes and tips.



The calibration is essential for robot pick & place applications due to the transformation of the image coordinate system of the sensor to the world coordinate system of the robot.

5.3.4 Creating a new model

Create a new or multiple models depending on your application. Follow all modelling steps from chapter **Models** in the software manual and observe all notes and tips.



For a simple pick & place application, it is recommended to use a [Contour detection] model described in chapter Contour detection model in the software manual.

Create a new model in the ifm Vision Assistant:

- Select the [Model settings] area.
- Select the [Create new model] button.
- Select a model which fits your application requirements.

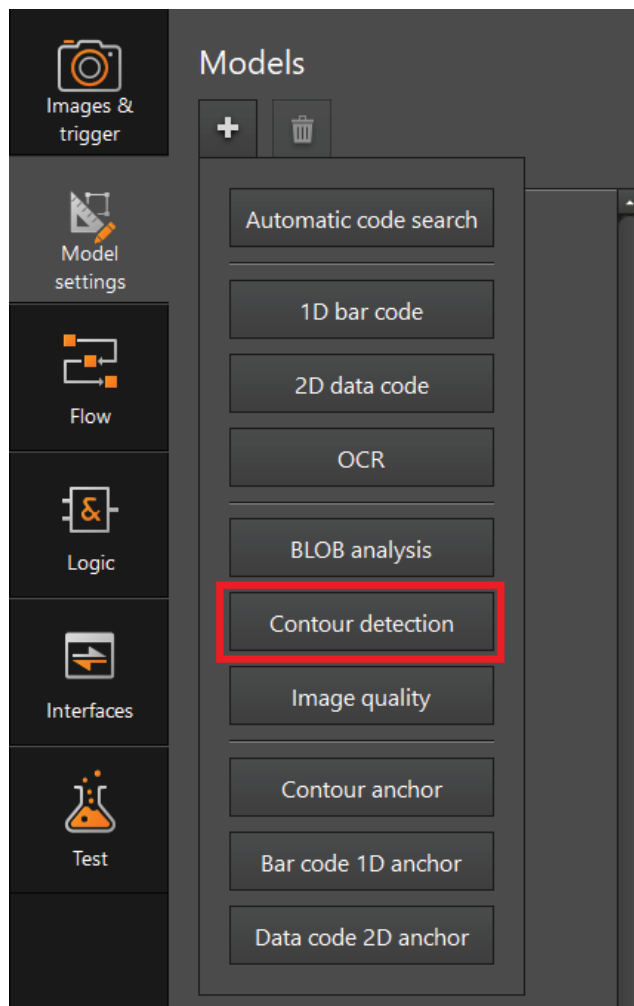


Fig. 12: Selecting a model

5.3.5 Selecting the interface presets

The configuration of the process interface is used to set the structure of the output via TCP/IP. Default settings can be used for common robot manufacturers:

Select an interface preset in the ifm Vision Assistant:

- ▶ Select the [Interfaces] area.
- ▶ Click on the drop-down menu in [Presets].
- ▶ Select one of the following presets, depending on the model used for object detection:
 - ▷ [KUKA iisy robots contour]
 - ▷ [KUKA iisy robots BLOB]

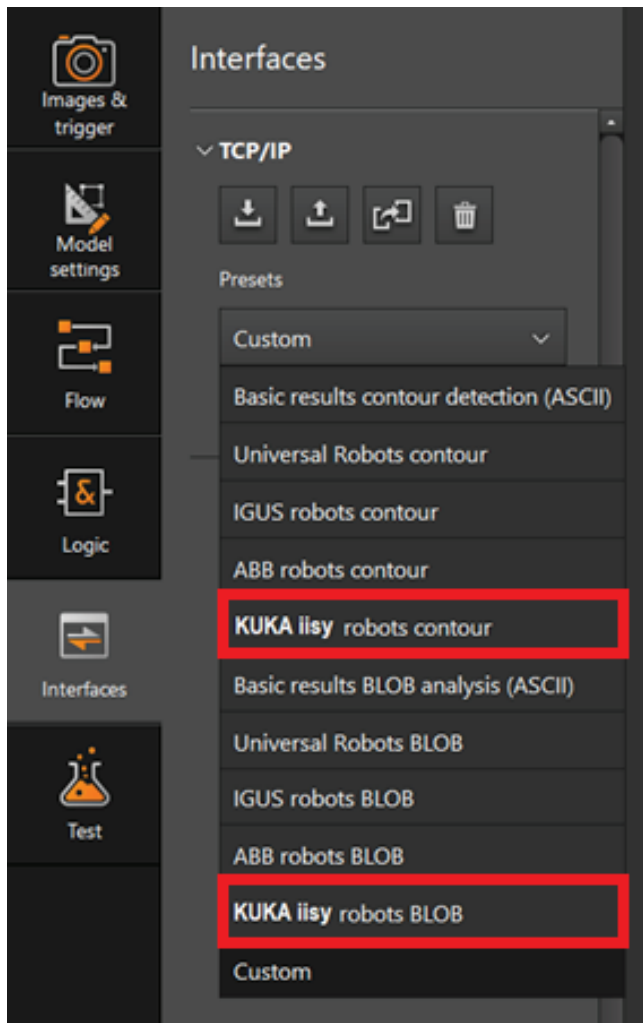


Fig. 13: Interface presets

The [KUKA iisy robots contour] present provides the data output structure (syntax) for your KUKA iisy cobot. By default, the data for following uses cases is provided:

- Object Quality Inspection
- Robo Pick & Place
- Classification



The Classification task requires some configuration in the Logic layer. ([→ Classification 18](#))

5.3.6 Logic layer configuration

Depending on your use case, the following applications can be implemented together or individually in the logic layer:

- Object Quality Inspection
- Robo Pick & Place

- Classification

5.4 Object quality inspection

For a simple object quality inspection application, there is no need to modify the logic layer when using the KUKA iisy interface preset. The element [Reading result (Pass/Fail)] is provided for all models in the [Interface] preset by default.

The two images below show the relationship between element [State definition pass/fail] in the logic and the element [Reading result (Pass/Fail)] in the process interface.

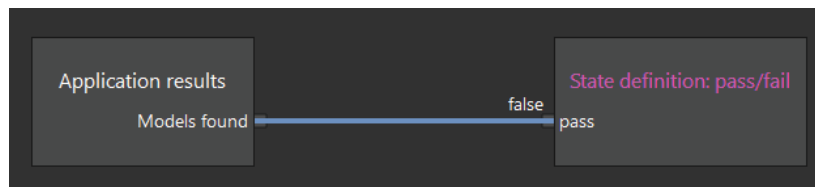


Fig. 14: Element State definition pass/fail



Fig. 15: Connected elements

Feel free to create your own pass/fail result output configuration by editing the [Logic] of your application like in following example:

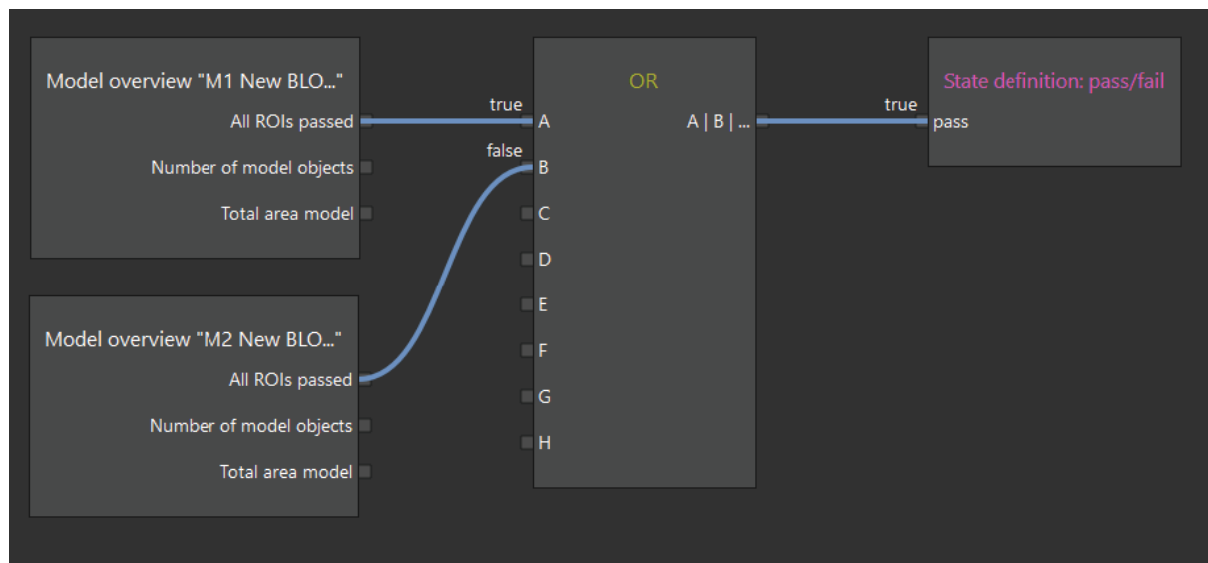


Fig. 16: pass/fail result output configuration

5.5 Robot pick & place

For a simple robot pick & place task there is no need to modify the logic layer in case of using the KUKA iisy interface preset. The elements [X (calibrated)], [Y (calibrated)], [Z (calibrated)], [180.0], [0.0] and [angle (calibrated)] that represent the 6 degrees of freedom X, Y, Z, Rx, Ry, Rz are provided for all contour models in the [Interface] preset by default.



Be aware that in case of using more than one contour detection model or a combination of an anchor tracking model with a contour detection model, you must modify the output configuration in the [Logic] layer and add an element [String Output] in the interface.

Feel free to create your own coordinates output configuration by editing the [Logic] of your application like in following example:

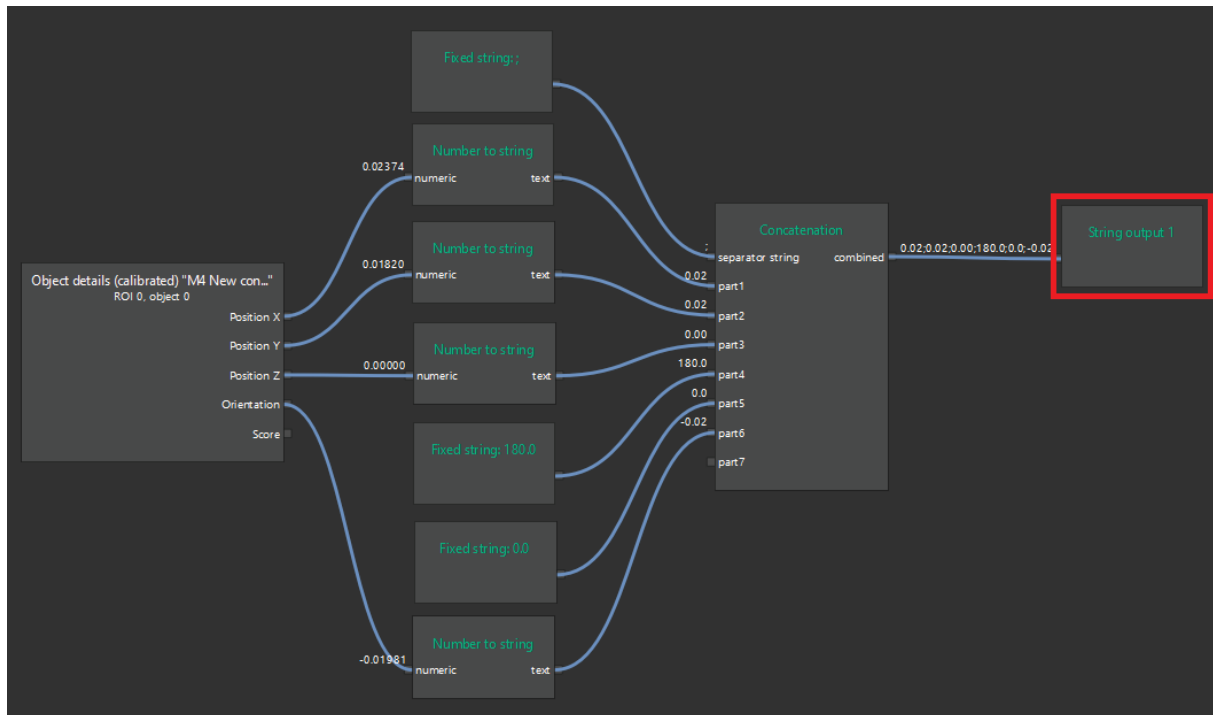


Fig. 17: logic example

In case of using your own [Logic] configuration, you can delete the six default elements responsible for the object coordinates and simply add the [String output] element to your interface configuration to provide the required data to your KUKA iisy cobot like in the image below.

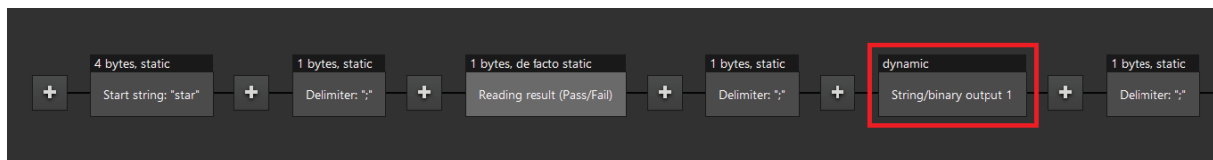


Fig. 18: Adding string output element

5.5.1 Offsetting object coordinates

In most cases, the Z value of an object coordinate must be shifted so that the robot's gripper will not hit the object.

There, it is possible to offset the elements [X (calibrated)], [Y (calibrated)], [Z (calibrated)], [180.0], [0.0] and [angle (calibrated)] that represent the 6 degrees of freedom X, Y, Z, Rx, Ry, Rz, if needed.

Simply add the required offset value in the property window of the element itself in the [Interface] scope.

Select the required element and value.

- Select the [Interfaces] area.
- Click on the drop-down menu in Presets.

The most frequently used offset value is probably the Z value. In this case, this causes the TCP to approach 100 mm above the detected object, thus avoiding a collision with the object.

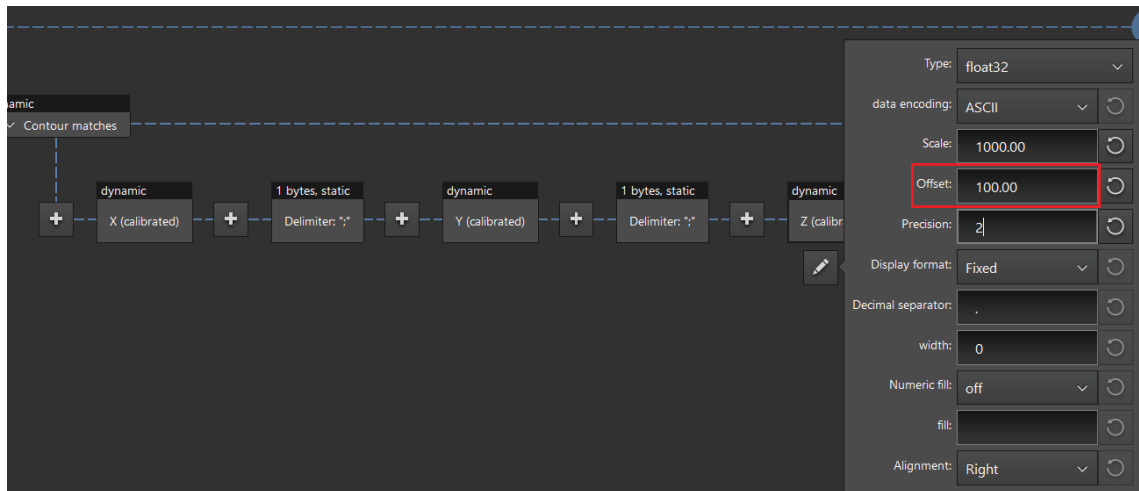


Fig. 19: Setting the offset value

To lower the gripper by 100 mm to grip the object, a relative movement of -100 mm in direction Z can be added to the program with a Move node, after moving to the coordinates received from the sensor.

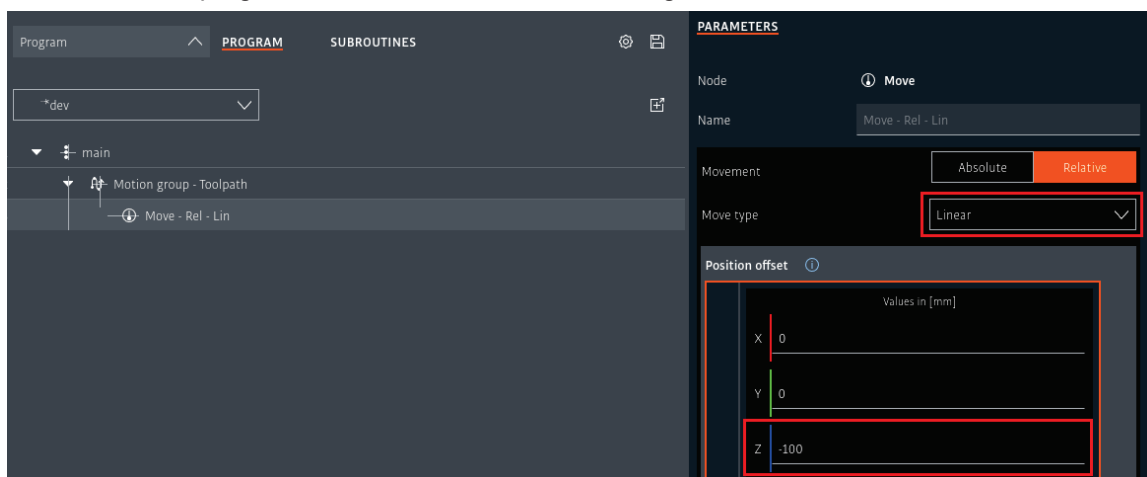


Fig. 20: Offsetting object coordinates

5.6 Classification

For a simple classification task, you need to modify the [Logic] configuration. There are several options for modelling the classification via the logic layer. The first option is to use the [Find first wildcard match] element in the logic layer, as shown in the below image.

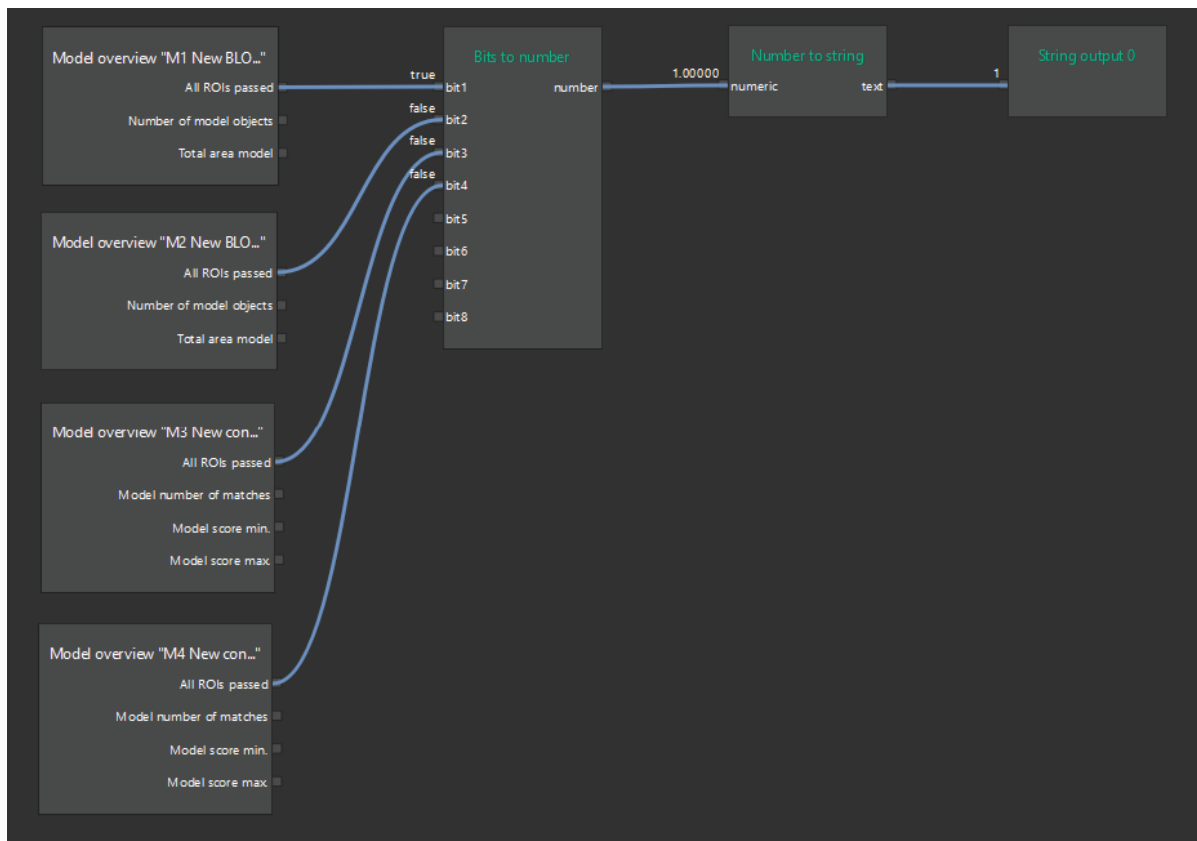


Fig. 21: [Find first wildcard match] element

Another option is to use the [Bits to number] element:

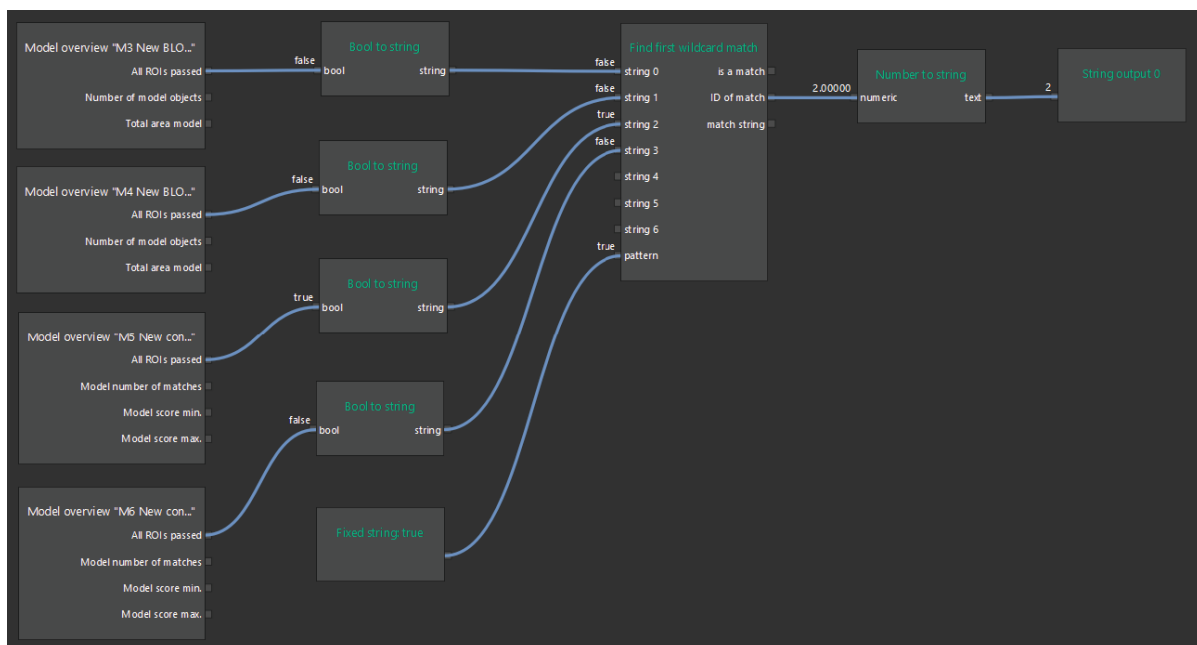


Fig. 22: [Bits to number] element



Make sure to also set the [First Fit] parameter in the [Flow] space. Otherwise, two passed models will lead to wrong classification.

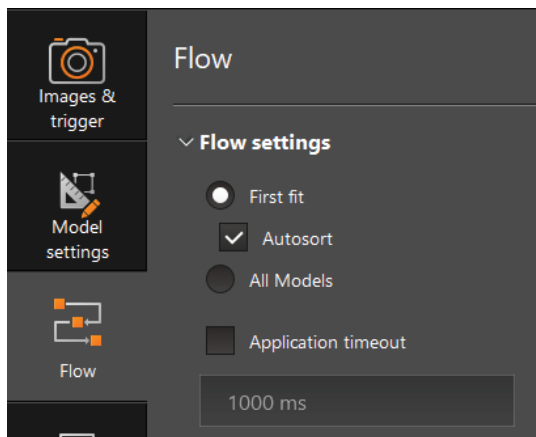


Fig. 23: Flow settings

5.7 ifm 2D Vision Sensor Toolbox Methods

The toolbox provides the following methods that trigger events on the sensor side.

- IFM_TriggerSensor
- IFM_ActivateApplication
- IFM_WritePCICInputStringData
- IFM_ReadStringOutputData
- IFM_RequestIOLogicState
- IFM_SetIOLogicState
- IFM_ExecuteButtonFunctionality

There are also better methods to show the current status of the toolbox or application:

- IFM_GetCurrentError
- IFM_GetObjectPose
- IFM_GetReadingResult

All the above-mentioned methods can be used in the Capabilities view:

- ▶ Select [Setup] in the drop-down menu.
- ▶ Select [Capabilities] in the below drop-down menu.

5.7.1 IFM_TriggerSensor

This method sends a synchronous trigger command to the sensor and is mandatory for receiving new data from the sensor. As a response, the PCIC answer is provided by the sensor to the capability. Other methods will decode their data from the received answer.

The Response value of the method itself can be one of the following:

- SUCCESSFUL
- FAILED

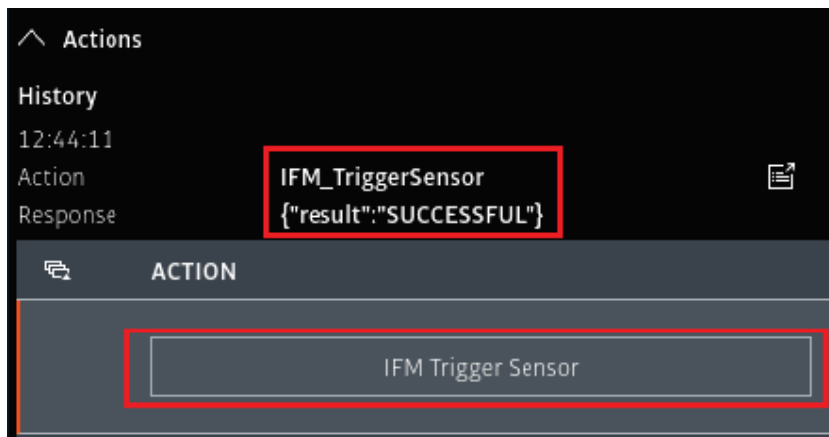


Fig. 24: IFM_TriggerSensor

5.7.2 IFM_ActivateApplication

This method will activate an application on the sensor. Enter a number between 1 and 32 into the textbox.

The Response value of the method itself can be one of the following:

- SUCCESSFUL
- FAILED



Fig. 25: IFM_ActivateApplication

It is important that the application exists in the application list of the sensor and that IO application switching is deactivated.

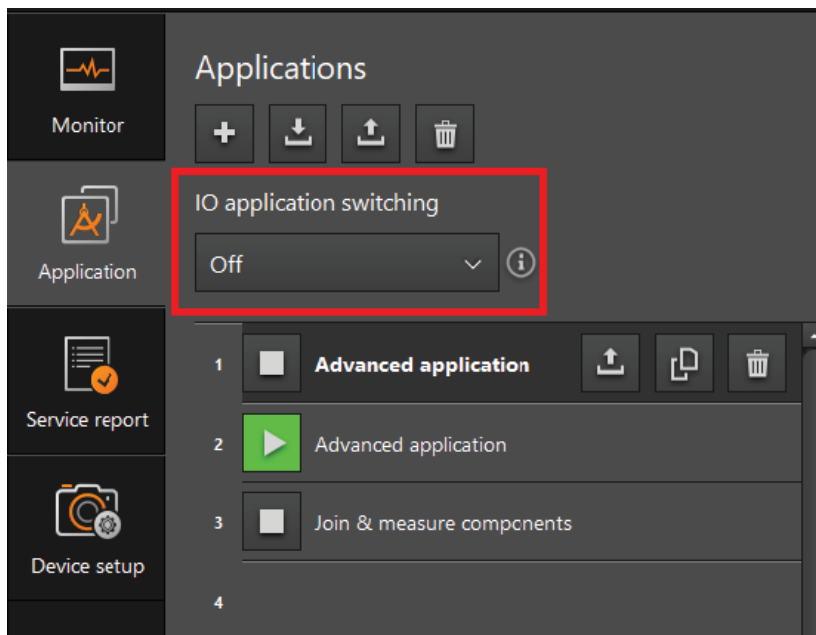


Fig. 26: IO application switching is deactivated

5.7.3 IFM_WritePCICInputStringData

Overwrites the string data of a specific (ID) string container used in the logic layer.

The Response value of the method itself can be one of the following:

- SUCCESSFUL
- FAILED

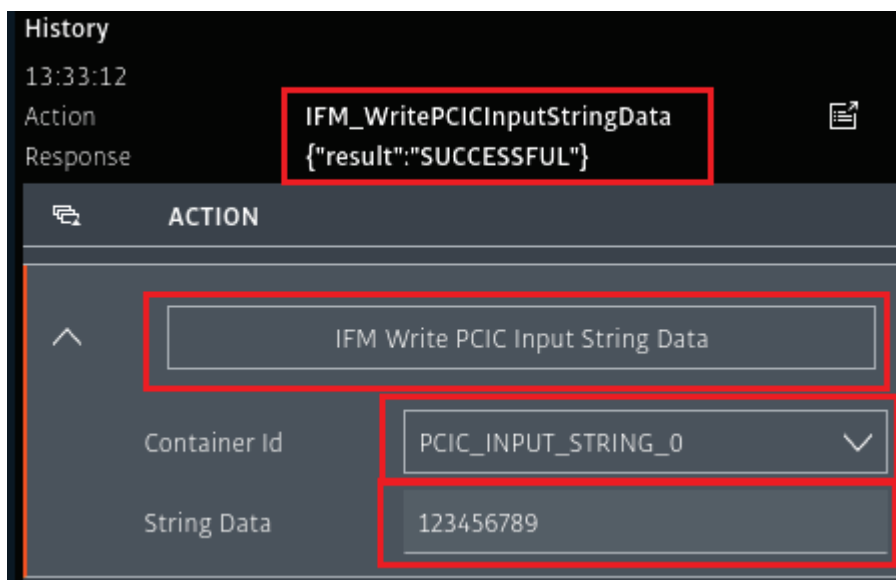


Fig. 27: IFM_WritePCICInputStringData

The element/container [PCIC input string] must be available in the [Logic] like you can see in the example blow.

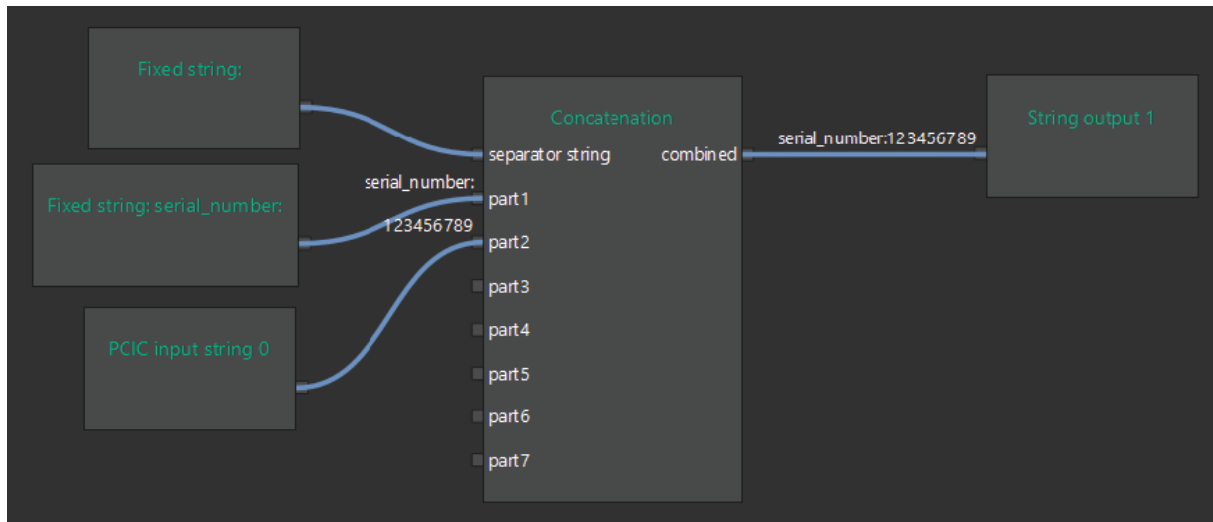


Fig. 28: element/container [PCIC input string]

5.7.4 IFM_ReadStringOutputData

Read the string output data of a specific (ID) string output container used in the logic layer and routed to the PCIC output. The string output if a specific (ID) container is generated during the evaluation of an application. A trigger may be needed to receive the required answer.

The Response value of the method itself can be one of the following:

- SUCCESSFUL
- FAILED

The string output data will also be forwarded as a string in the output variable stringOutputData.

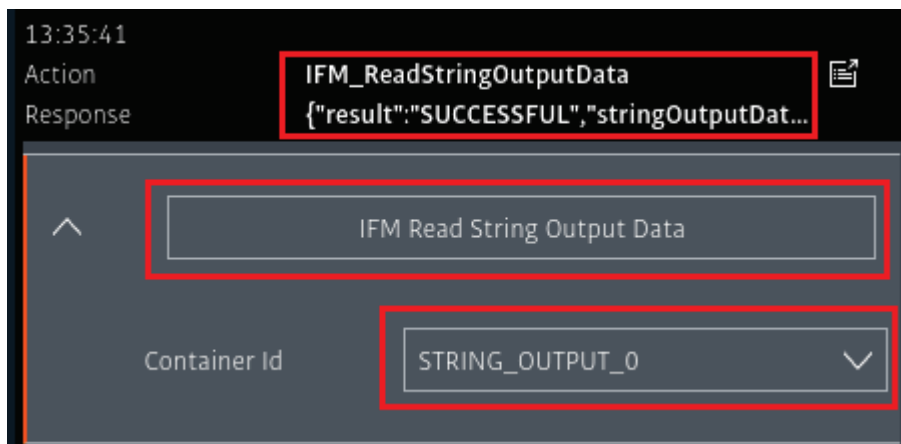


Fig. 29: IFM_ReadStringOutputData



You are reading the string output data provided by the [String/binary output] elements in the [Interface] and not of the [Logic] elements itself. A trigger is mandatory for receiving the data.



Fig. 30: [String/binary output] elements

5.7.5 IFM_RequestIOLogicState

Requests the state of a specific IO.

The Response value of the method itself can be one of following:

- SUCCESSFUL
- FAILED

The state will also be forwarded as a “LOW” or “HIGH” in the output variable stringOutputData.

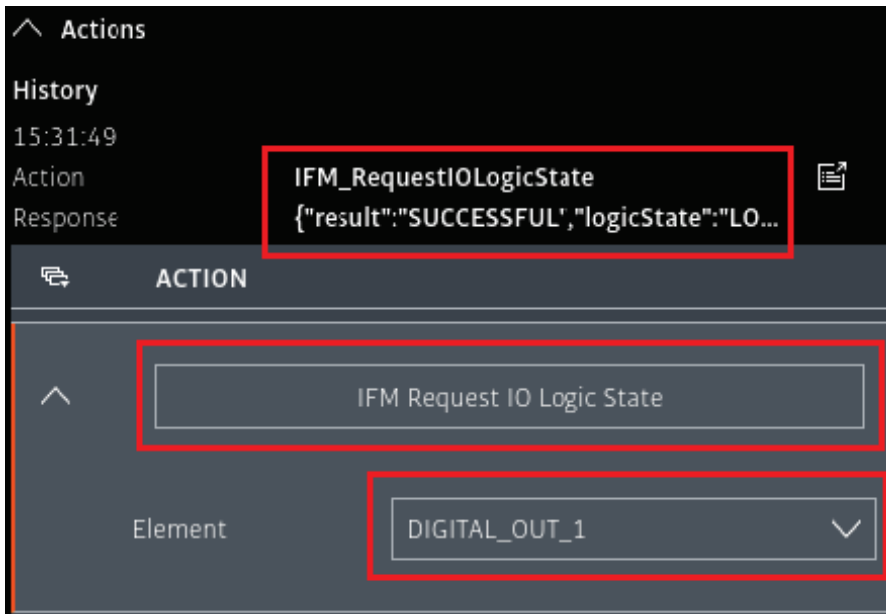


Fig. 31: IFM_RequestIOLogicState

It is important that the element is available in the [Logic] configuration at least.

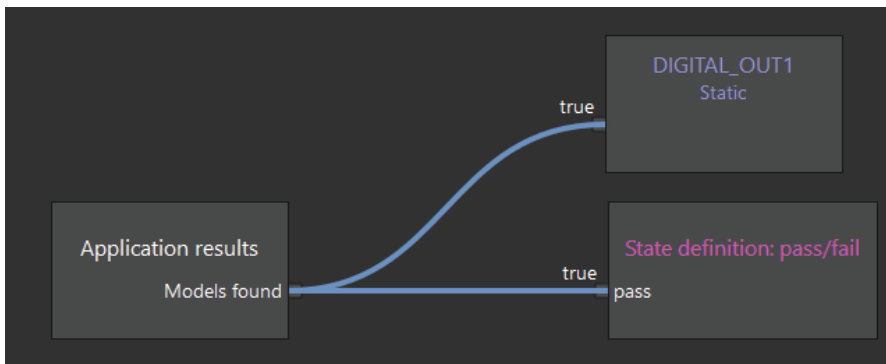


Fig. 32: IFM_RequestIOLogicState

5.7.6 IFM_SetIOLogicState

Sets the state of a specific IO.

The Response value of the method itself can be one of following:

- SUCCESSFUL
- FAILED

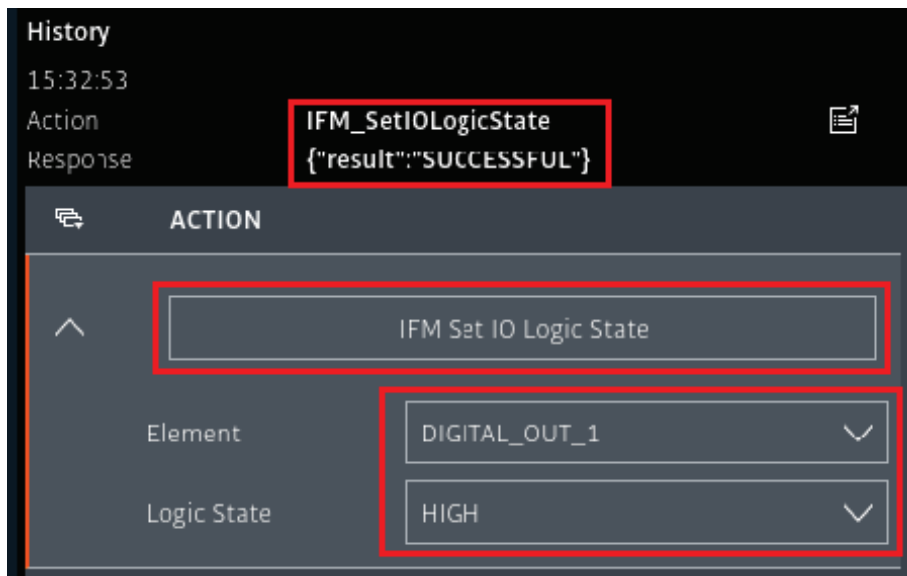


Fig. 33: IFM_SetIOLogicState

It is important that the element [PCIC O-command] is connected to the [DIGITAL_OUT] element in the [Logic] configuration.

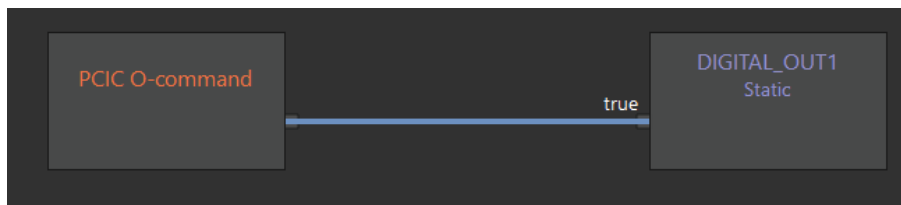


Fig. 34: element PCIC O-command

5.7.7 IFM_ExecuteButtonFunctionality

Overwrites the string data of a specific (ID) string container used in the logic layer.

The Response value of the method itself can be one of following:

- SUCCESSFUL
- FAILED

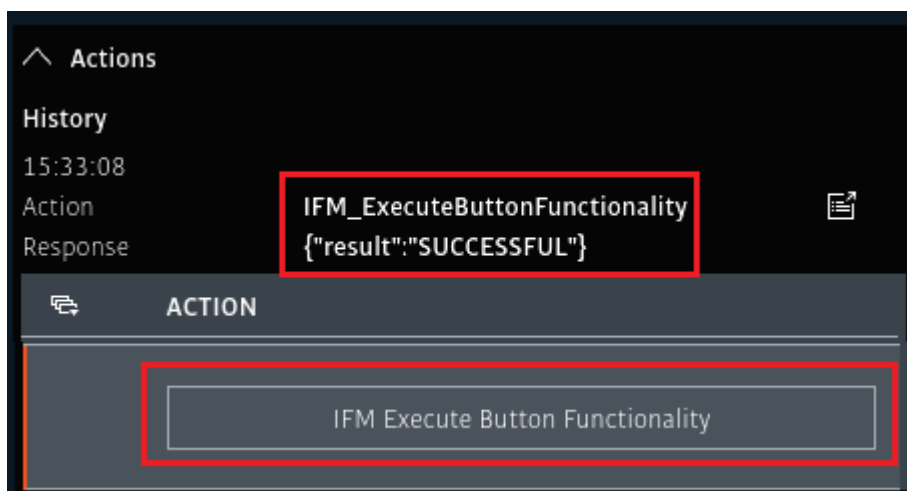


Fig. 35: IFM_ExecuteButtonFunctionality

The following functions can be selected on the sensor with the ifm Vision Assistant:

- Autofocus
- Auto-exposure of first image

- Teach reference-run and refocus

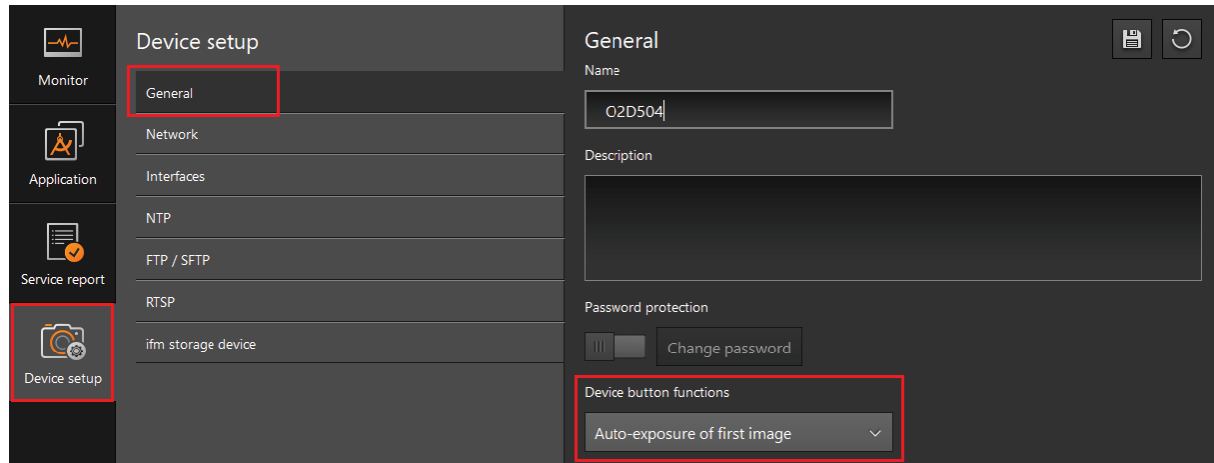


Fig. 36: Device button function

5.7.8 IFM_GetCurrentError

You can check for current errors by activating the refresh checkbox and by clicking the button marked with the red arrow in the image below.

This will open a window where you can find the following return variables that will provide the error codes and the corresponding description as a string value:

- ErrorState (NO_ERROR, ERROR)
- deviceErrorCode (string)
- deviceErrorCodeDescription (string)
- applicationErrorCode (string)
- applicationErrorCodeDescription (string)

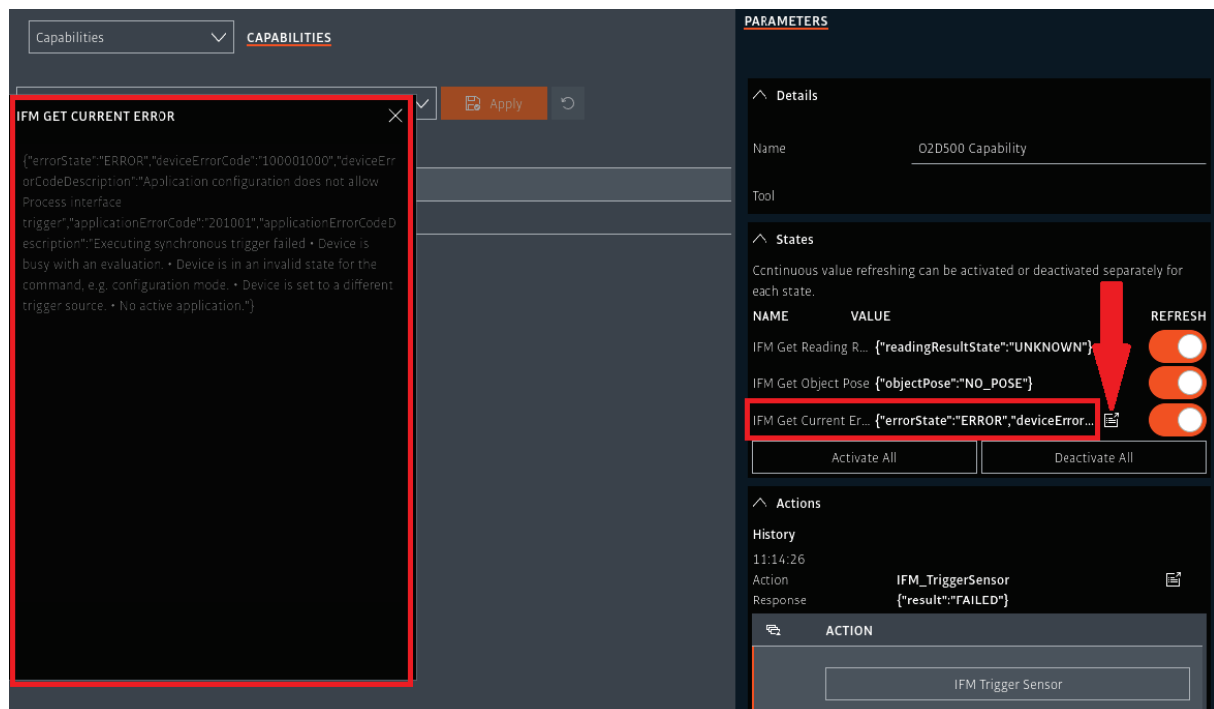


Fig. 37: IFM_GetCurrentError

This may be helpful in case of any issues with the sensor or the application. All error codes are described in this chapter: [Error codes](#) (→ [32](#))

5.7.9 IFM_GetObjectPose

This state method is checking for a found object in the answer from the device. If an object was found, the object coordinates are transformed and stored in the referenced object frame. ([→ Configuring the capability 9](#))

The Response value of the method itself can be one of following:

- NO_POSE
- INVALID_COORDINATES
- OBJECT_FOUND
- NO_CONTOUR_FOUND
- NO_FRAME

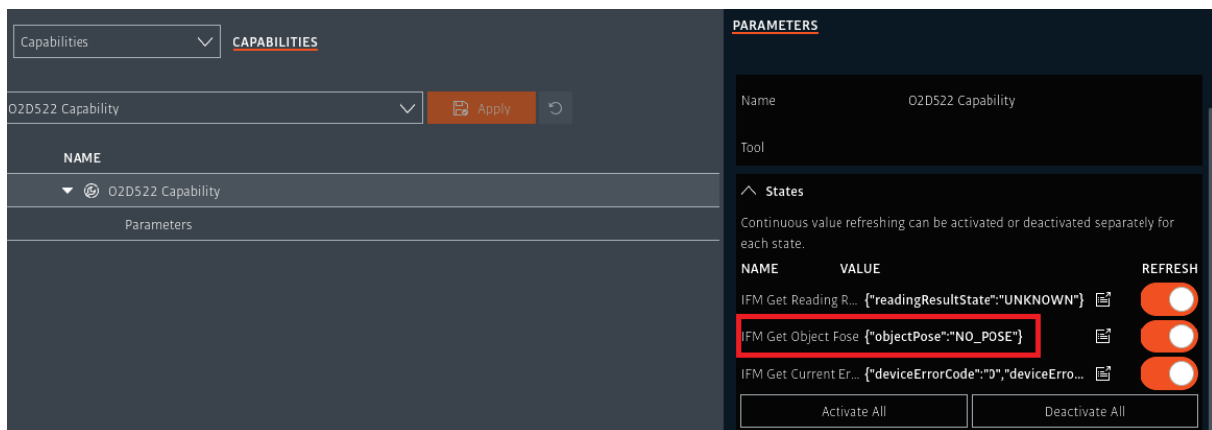


Fig. 38: IFM_GetObjectPose

5.7.10 IFM_GetReadingResult

Overwrites the string data of a specific (ID) string container used in the logic layer.

The Response value of the method itself can be one of the following:

- UNKNOWN
- PASS
- FAIL

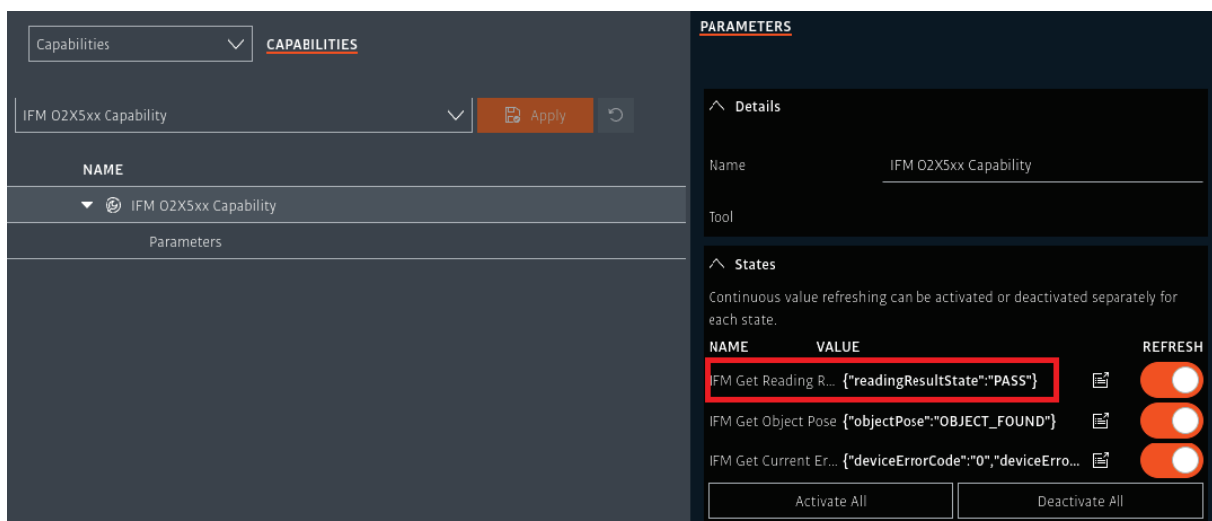


Fig. 39: IFM_GetReadingResult_1

You can see the relation between the [Interface] element on the sensor and the value from this method.



Fig. 40: Relation between [Interface] element on the sensor and this method

Also see the relation between the [Interface] element on the sensor above and the [Logic] element below.

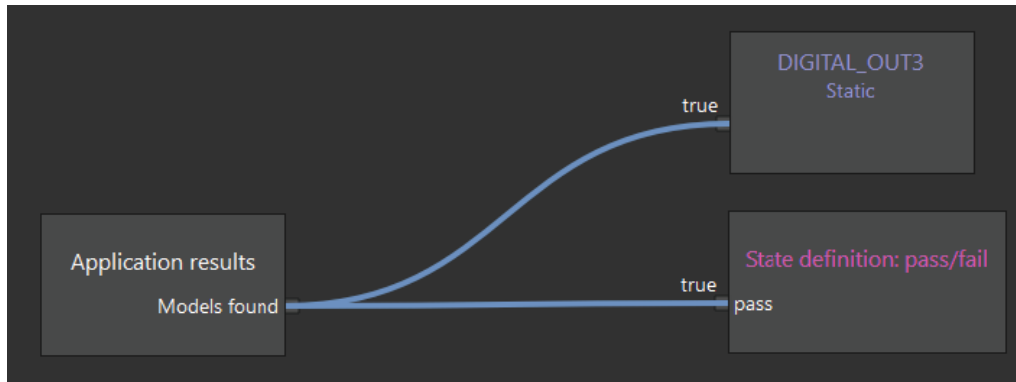


Fig. 41: Relation between [Interface] element on the sensor and [Logic] element

5.8 Example programs

In the following chapters you can see simple example programs for Robot Pick & Place, Object Inspection and Classification tasks.

Every method node described in the chapters above can be added to your own program. For selecting the method open the [Node Palette] by clicking on the symbol shown below.

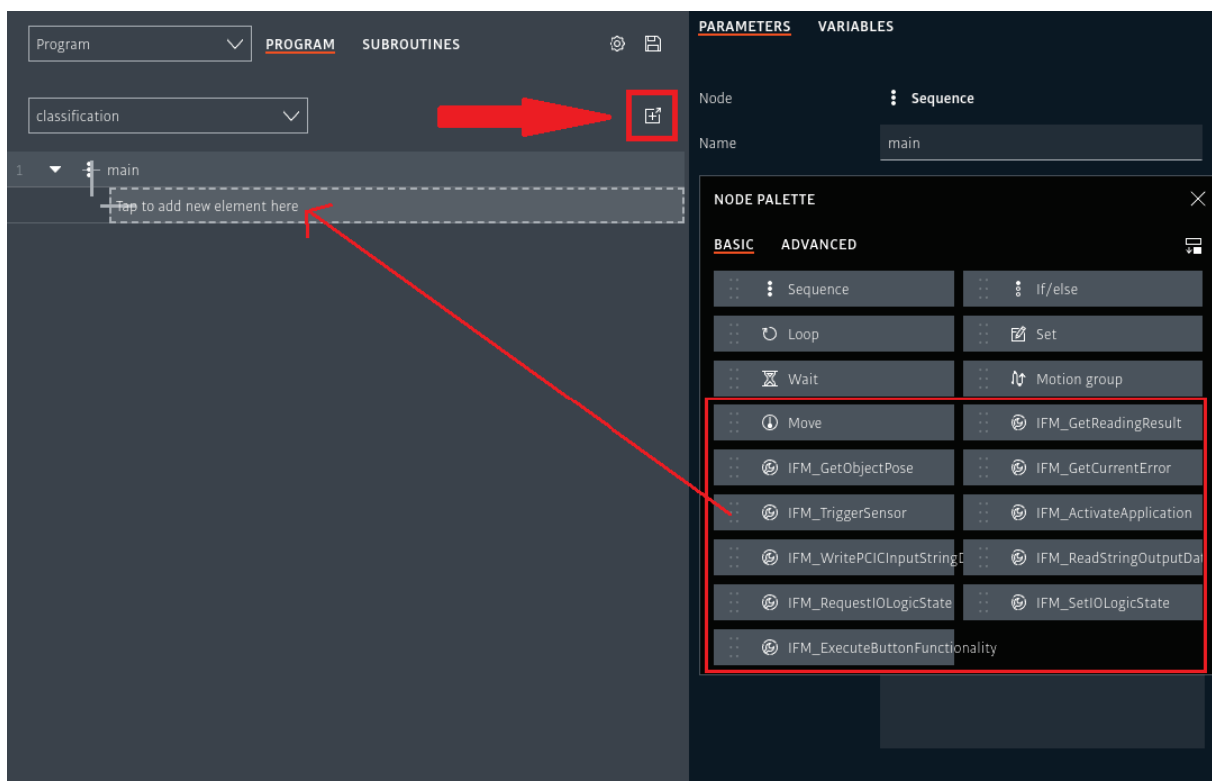


Fig. 42: Node palette

► Click on the node needed for your own application program.

▷ The node will be added to the empty slot automatically.

5.8.1 Robot Pick & Place example

The screenshot displays the Vision Sensor Toolbox interface for a program named `*RobotPP`. The main workspace shows a sequence of steps:

- 1. `main`
- 2. `Loop - Infinite`
- 3. `IFM_TriggerSensor - IFM O2X5xx Capability` (Trigger device to receive the answer)
- 4. `IFM_GetObjectPose - IFM O2X5xx Capability` (Get the Object Pose state)
- 5. `IF - get-object_pose(...) = OBJECT_FOUND` (Check if state == OBJECT_FOUND)
- 6. `Motion group - Toolpath`
 - 7. `Move - SPL - Frame_Object` (Move above object)
 - 8. `Move Z - 100mm` (Lower gripper by 100 mm for gripping the object)

The right sidebar shows the **PARAMETERS** and **VARIABLES** sections. Under **PARAMETERS**, the **Node** is set to `Sequence` and the **Name** is `main`. The **Description** field is empty, with a placeholder text "Enter description...".

Fig. 43: Example for Robot Pick & Place

5.8.2 Object inspection example

The screenshot displays the Vision Sensor Toolbox interface for a program named `*objectInsepection`. The main workspace shows a sequence of steps:

- 1. `main`
- 2. `Loop - Infinite`
- 3. `IFM_TriggerSensor - IFM O2X5xx Capability` (Trigger device to receive the answer)
- 4. `IFM_GetReadingResult - IFM O2X5xx Capability` (Read the Reading Result state)
- 5. `IF - get-reading_result_state(...) = PASS` (Check if state == PASS)
- 6. `Wait - Millis - 1000` (do something ...)

The right sidebar shows the **PARAMETERS** and **VARIABLES** sections. Under **PARAMETERS**, the **Node** is set to `Sequence` and the **Name** is `main`. The **Description** field is empty, with a placeholder text "Enter description...".

Fig. 44: Example for Object inspection

5.8.3 Classification example

The screenshot displays the Vision Sensor Toolbox interface for a program named `*classification`. The main editor shows a sequence of steps:

- 1. `main`
- 2. `Loop - Infinite`
- 3. `IFM_TriggerSensor - IFM O2X5xx Capability` (Trigger device to receive the answer)
- 4. `IFM_ReadStringOutputData - IFM O2X5xx Capability` (Read the string output data from `STRING_OUTPUT_0`)
- 5. `IF - get-string_output_data(...) = "M12x45 screw"` (Check if data = "M12x45 screw")
- 6. `IFM_ActivateApplication - IFM O2X5xx Capability` (Activate application with number 3)
- 7. `Wait - Millis - 1000` (do something ...)

The right-hand panel shows the `PARAMETERS` and `VARIABLES` sections. Under `PARAMETERS`, the `Node` is set to `Sequence` and the `Name` is `main`. The `Description` field is empty, with a placeholder text "Enter description...".

Fig. 45: Example for Classification

5.8.4 Sharing string data between devices example

The screenshot displays the Vision Sensor Toolbox interface for a program named `*shareData`. The main editor shows a sequence of steps:

- 1. `main`
- 2. `Loop - Infinite`
- 3. `IFM_TriggerSensor - O2D500 capability` (Trigger device to receive the answer from O2D500)
- 4. `IFM_ReadStringOutputData - O2D500 capability` (Read string output data from `STRING_CONTAINER_0` from O2D500)
- 5. `IFM_WritePCIInputStringData - O2D510 Capability` (Write the O2D500 data to the O2D510 device)
- 6. `IFM_TriggerSensor - O2D510 Capability` (Trigger device to receive the answer from O2D510)
- 7. `IFM_ReadStringOutputData - O2D510 Capability` (Read string output data from O2D510)
- 8. `IF - get-string_output_data(...) = "data_from_O2D500"` (Check if data == "data_from_O2D500")
- 9. `Wait - Millis - 1000` (do something ...)

The right-hand panel shows the `PARAMETERS` and `VARIABLES` sections. Under `PARAMETERS`, the `Node` is set to `Sequence` and the `Name` is `main`. The `Description` field is empty, with a placeholder text "Enter description...".

Fig. 46: Example for sharing data between devices

5.8.5 Setting and reading IO states example

The screenshot displays the Vision Sensor Toolbox interface. The left pane shows a program tree with a 'main' node containing an infinite loop. Inside the loop, three steps are defined:

- Step 3: IFM_RequestIOLogicState - O2D500 capability. Red annotation: **Read logic state of DIGITAL_OUT_1 of O2D500 device**
- Step 4: IF - get-logic_state(...) = HIGH. Red annotation: **Check if state == HIGH**
- Step 5: IFM_SetIOLogicState - O2D510 Capability. Red annotation: **Set logical state of DIGITAL_OUT_3 to HIGH of O2D510 device**

The right pane shows the 'PARAMETERS' tab for the 'main' node, with a 'Sequence' type and a description field.

Fig. 47: Example for setting and reading IO states

5.8.6 Handling error codes example

The screenshot displays the Vision Sensor Toolbox interface. The left pane shows a program tree with a 'main' node containing four steps:

- Step 2: IFM_TriggerSensor - O2D500 Capability. Red annotation: **Trigger device**
- Step 3: IFM_GetCurrentError - O2D500 Capability. Red annotation: **Read error states**
- Step 4: IF - get-error_state(...) = ERROR. Red annotation: **Check if state == ERROR**
- Step 5: Wait - Millis - 1000. Red annotation: **do something ...**

The right pane shows the 'PARAMETERS' tab for the 'main' node, with a 'Sequence' type and a description field.

Fig. 48: Example for handling error codes

5.9 Error codes

Device error codes are provided by the device by default. Application error codes are provided by exceptions or misbehaviour of the toolbox / capability. Both error codes can be read out in the capability space or in the program. ([→ Handling error codes example](#) 31)

Overview of all provided error codes:

Codes	Description	Reason	Scope
0	No error detected		device
100000001	Maximum number of connections exceeded		device
100000002	Internal failure during a D-Bus call		device
100000003	Unspecified internal error		device
100000004	Generic invalid parameter		device
100000005	Invalid command		device
100001000	Application configuration does not allow Process interface trigger		device
100001001	Video mode does not allow Process interface trigger		device
100001002	There is no application configured		device
100001003	Invalid image id in I? command		device
100001004	Invalid pin ID in o/O? command		device
100001005	Invalid pin configuration in o/O? command		device
100001006	Invalid conversion type selected		device
100001007	No trigger was run yet		device
100001008	Missed decoded frame		device
100001009	No more segments left		device
100001010	Command bit 4 not reset to 0		device
100001012	No such block in logic layer		device
100001013	Device is not in run or simulation mode		device
100001014	Internal temperature too high for view indicator usage		device
100001015	No button function configured		device
100001016	Button function already running		device
100001017	Button function caused an error		device
100001018	Device in invalid state for button function		device
100001019	Invalid temporal parameter ID		device
100001020	Temporal parameter out of range		device
100001021	Session request failed		device

Codes	Description	Reason	Scope
10001022	No view indicators available		device
110001001	Boot timeout		device
110001002	Fatal software error		device
110001003	Unknown hardware		device
110001006	Trigger overrun		device
110001007	Ethernet configuration was changed. The socket is going to be closed.		device
110002000	Short circuit on Ready for Trigger		device
110002001	Short circuit on OUT1		device
110002002	Short circuit on OUT2		device
110002003	Reverse feeding		device
110003000	Vled overvoltage		device
110003001	Vled undervoltage		device
110003002	Vmod overvoltage		device
110003003	Vmod undervoltage		device
110003004	Mainboard overvoltage		device
110003005	Mainboard undervoltage		device
110003006	Supply overvoltage		device
110003007	Supply undervoltage		device
110003008	VFEMon alarm		device
110003009	PMIC supply alarm		device
110004000	Illumination overtemperature		device
120000001	NTP server not available		device
120000002	other NTP error		device
101013	Application number is not available in the device	No application found in index <Application Number> in application list	application
101022	Application number is invalid	Application number must be between 1 and 32	application
101048	Device is not in run mode	Device is in wrong operation mode e.g., parametrization mode with ifm Vision Assistant	application
200001	Unable to connect to device	• Issue with socket connection to device. • Too many socket connect requests (reboot may help).	application
200002	Socket timeout error	No response after default timeout value of 10 seconds.	application
200003	TCP client is not connected	Unable to establish a socket connection.	application
200004	TCP/IP protocol version not implemented		application
200005	Unknown TCP/IP protocol version		application

Codes	Description	Reason	Scope
201001	Executing synchronous trigger failed	• Device is busy with an evaluation. • Device is in an invalid state for the command, e.g., configuration mode. • Device is set to a different trigger source. • No active application.	application
201002	Reading next answer failed	Connection issues	application
201003	Activating application failed	• Application not available. • <Application Number> contains wrong value. • External application switching activated. • Device is in an invalid state for this command e.g., configuration mode. • For devices with hardware application switching: hardware application switching is enabled.	application
201004	Requesting occupancy of application list failed	• Invalid state (e.g., no application active). • No application stored on the device (E? command will return 100001002).	application
201005	Requesting current error state failed		application
201006	Overwriting data of a string failed	• Invalid argument or invalid state (other than run mode). • Not existing string with input container-ID.	application
201007	Executing configured button functionality failed	• No button function configured. • Button function already running. • Button function caused an error. • Device is in an invalid state for this command, e.g., configuration mode.	application
201008	Internal failure sending command	Issues with network stream.	application
201009	Setting logic state of an IO failed	• Invalid state (e.g., configuration mode). • Wrong ID or IO state. The id and the state must be number 0...9. • Output is not configured as manual override.	application
201010	Requesting logic state of an IO failed	• Invalid state (e.g., configuration mode). • Wrong ID or IO state. The id and the state must be number 0...9. • Output is not configured as manual override.	application

Codes	Description	Reason	Scope
202001	Device answer is empty	• Trigger not triggered yet. • Trigger command was not successful. • Connection issues.	application
201011	Reading string output data failed	• Invalid state (e.g., configuration mode) • Output string data is not provided by device answer.	application
201012	No object reference frame	No object frame specified in Capability parameters.	application
201013	Invalid coordinates in device answer	Coordinates invalid in the received device answer. Check PCIC interface of sensor with ifm Vision Assistant.	application
201014	Contour not found	Contour model was not able to find the contour in the image. Check image settings and contour model.	application