SCHUNK

Software manual
2D Grasping-Kit
Smart Grasping Native Protocol

Original software manual

Hand in hand for tomorrow

Imprint

Technical changes:
We reserve the right to make alterations for the purpose of technical improvement.

Document number: 1559896

Version: 01.00 | 30/08/2023 | en

Dear Customer,

Thank you for trusting our products and our family-owned company, the leading technology supplier of robots and production machines.

Our team is always available to answer any questions on this product and other solutions. Ask us questions and challenge us. We will find a solution!

Best regards,

Your SCHUNK team

Customer Management
Tel. +49-7133-103-2503
Fax +49-7133-103-2189
cmg@de.schunk.com

Please read the operating manual in full and keep it close to the product.

Table of Contents

1 Simple Protocol General

Low-level pipeline interface for robot controllers.

The simple protocol is a communication protocol sitting on top of the TCP/IP stack. The protocol design is simple and straightforward, and thus feasible to implement even on the most archaic robots and industrial controllers.

1.1 Conventions

- **Port**: The server listens on tcp-port 42001
- **Endianness**: The order of multi-byte fields is big-endian
- **Floating-point numbers**: Single IEEE-Standard (IEC-60559)
- **Signed integers**: Signed integers are represented in two's complement notation
- **Units of measure**: Unless otherwise specified, lengths are measured in meters [m], and angles are measured in radians [rad]

1.2 Data Types

Following data types are define:

- **signed integers**: int8, int16, int32
- **unsigned integers**: uint8, uint16, uint32
- **floating point numbers**: float32

The number suffix of a data type denotes its size in bits (e. g. int16 is two bytes long).

1.3 Versioning

The server is backwards compatible.

Clients using earlier protocol versions will be served as expected and won't even notice the version mismatch. The server always replies with the same version number in the prefix as the one used in the requests.

However, if a server is outdated such that its protocol version is lower than the client's protocol version, the server will always reply with an empty message. The message prefix will have its length-field set to zero and its version-field set to the server's highest supported protocol version. In this case, clients are advised to alert the user that their server is outdated and incompatible with their current client version.

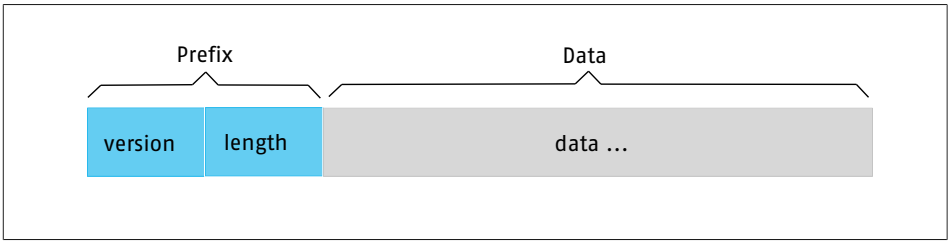
1.4 Layout

Each message frame comprises two parts: an invariant prefix that remains consistent across different protocol versions and a version-specific section. The details of these parts are elaborated in their respective chapters.

2 Simple Protocol Prefix

Invariant part of the low-level pipeline interface for robot controllers.

2.1 Layout



2.2 Prefix

All messages start with a *prefix* defining the protocol version and the size of the remaining message.

The prefix is invariant and does not change between different protocol versions.

	version	length
type	uint16	uint32

- **version**: The protocol version.
- **length**: The size of the *remaining* message in bytes (max. 4 GB), excluding the prefix itself (6 B). The server will not verify the correctness of this field. An incorrect length value will corrupt the session and, if the corruption remains unnoticed, lead to undefined behavior.

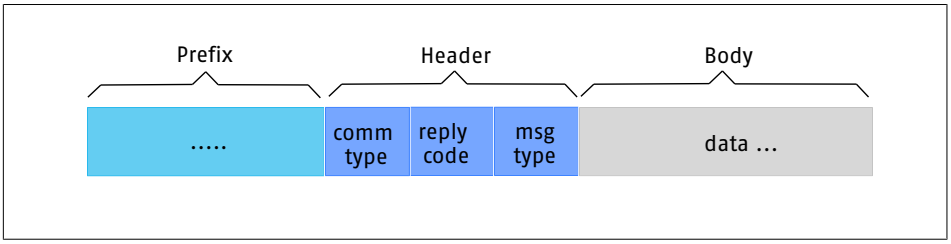
2.3 Data

The actual message content. The structure is version-specific and is described in the version chapters.

3 Simple Protocol v2

Version 2 of the low-level pipeline interface for robot controllers.

3.1 Layout



3.2 Prefix

see ▸ 2.2 [🗋 6].

3.3 Header

The *header* is the part that encodes the message type and its intent.

	comm type	reply code	msg type
type	uint8	uint8	uint16

3.3.1 comm type

value	description
0x00	request
0x01	response

3.3.2 reply code

value	name	description
0x00	success	Receiver processed the message succesfully
0x01	failure	Receiver encountered a failure processing the message (msg type is zeroed)

NOTE

The reply code is ignored in requests.

3.3.3 msg type

The message types and their body layouts are explained in detail in the next paragraph:

value	name	description
general		
0x00 00	reserved/unused	
0x00 01	GET_PROTOCOL_VERSION	Get the server's protocol version
0x00 02	GET_STATE	Get the server's current state
0x00 03	REGISTER_CLIENT	Register the client's robot system to the server
grasping		
0x01 00	GET_GRASP	Get a grasp for the current scene
0x01 01	ACKNOWLEDGE_GRASP	Acknowledge that a grasp has been successful

3.4 Message Types

3.4.1 General

3.4.1.1 GET_PROTOCOL_VERSION

Returns the server's highest supported protocol version.

This is useful to see if the client's version is outdated or not. If this is the case, the client is advised to inform the user accordingly.

Request Body

The request body is empty.

Response Body

	version
type	uint16

- **version**: The server's protocol version.

3.4.1.2 GET_STATE

Retrieves the server's state.

Request Body

The request body is empty.

Response Body

	state
type	uint8

- **state**:

value	name	description
0x01	INIT	The server is initializing
0x02	OPERATIONAL	The server is operational and ready to serve requests
0x03	STOPPED	The server has stopped operating
0x04	ERROR	The server has encountered a critical error

3.4.1.3 REGISTER_CLIENT

Registers the client's robot system (for informational purposes). This message should be sent right after the connection has been established. However, there are no restrictions on when to send and how often.

Request Body

Contains information about the client's robot system. If the client's robot system is not officially supported (see client table), then simply do not send anything.

	client
type	uint8

- **client**:

value	name	vendor
0x01	UR	Universal Robots
0x02	Kuka	Kuka
0x03	Yaskawa	Yaskawa
0x04	Fanuc	Fanuc
0x05	ABB	ABB
0x06	HORST	fruitcore robotics
0x80	Siemens PLC	Siemens

Response Body

The response body is empty.

3.4.2 Grasping

3.4.2.1 GET_GRASP

Requests an exterior two-finger grasp for the current scene.

Request Body

	object class	mode	pose format
type	uint8	uint8	uint8

- **object class:**The class id of the target object. The value 0 denotes a random object.
- **mode**:

value	name	description
0x01	ACTIVE_GRASP	The returned grasp must match the target object's active grasp definition
0x02	ANY_GRASP	The returned grasp must match any of the target object's grasp definitions
0x03	AUTO_GRASP	ANY_GRASP mode with model-free grasp-planning as fallback

- **pose format**: Specifies the pose format to use in the response message.

value	name	description
0x01	Matrix	Homogeneous 4x4 transformation matrix in row-order
0x02	Angle-axis	[x,y,z,rx,ry,rz], with [x, y, z] the position and [rx, ry, rz] the unnormalized angle-axis vector

Response Body

	result code	object class	object instance	pose format	pose	stroke	center offset
type	uint8	uint8	uint16	uint8	see details	float32	float32 [3]

- **result code**: Indicates the outcome of the grasp request.

value	name	description
0x01	OK	The server has found a valid grasp
0x02	NO_OBJECT	The server could not find the requested object (all fields after "result code" are zeroed)
0x03	NO_GRASP	The server could not find a valid grasp (all fields after "object instance" are zeroed)
0xFF	ERROR	The server has encountered an internal error (all fields after "result code" are zeroed).

- **object class**: The class id of the target object.
- **object instance**: The instance id of the target object.
- **pose format**: The pose format used for the pose-field.

value	name	pose type	description
0x01	Matrix	float32[16]	Homogeneous 4x4 transformation matrix in row-order
0x02	Angle-axis	float32[6]	[x,y,z,rx,ry,rz], with [x, y, z] the position and [rx, ry, rz] the unnormalized angle-axis vector

- **pose**: The grasp pose in the robot's base coordinate frame. The type of this field depends on the value in the *pose format*-field.
- **stroke**: The distance between both fingers to set before approaching the object (in [m]).
  Note: SCHUNK makes the assumption that the gripper is at position zero when both fingers touch each other.
- **center offset**: The translational [x,y,z]-offset from the object center to the grasp point in [m], expressed in the robot's base coordinate frame.

3.4.2.2 ACKNOWLEDGE_GRASP

Acknowledges that a specified object has been successfully grasped.

Request Body

	object instance
type	uint16

- **object instance**: The instance id of the target object.

Response Body

	result code	object instance
type	uint8	uint16

- **result code**:

value	name	description
0x01	OK	The successful grasp has been acknowledged
0x02	NO_OBJECT	The target object does not exist

- **object instance**: The instance id of the target object.

4 Visualization

A visualization of the latest detection result is provided as an image resource that can be accessed via HTTP at:

`http://<Server-IP>/monitor/latest_result`

5 Examples

Get the server's highest supported protocol version (which is version 2 in this example):

```
request:  0x00 0x01 0x00 0x00 0x00 0x04 | 0x00 0x00 0x00 0x01 |

response: 0x00 0x01 0x00 0x00 0x00 0x06 | 0x01 0x00 0x00 0x01 | 0x0002
```

Get an auto grasp for a random object in the current scene in angle-axis representation:

```
request:  0x00 0x01 0x00 0x00 0x00 0x07 | 0x00 0x00 0x01 0x00 | 0x00 0x03 0x02

response: 0x00 0x01 0x00 0x00 0x00 0x25 | 0x01 0x00 0x01 0x00 | 0x01 0x01 0x00
0x01 0x02 0x3F 0x80 0x00 0x00 0x40 0x40 0x00 0x00 0x40 0x40 0x00 0x00 0x40
0xE0 0x00 0x00 0x11 0x22 0x33 0x44 0x11 0x22 0x33 0x44 0x42 0x8A 0x00 0x00
```

Send a request with an unknown msg type:

```
request:  0x00 0x01 0x00 0x00 0x00 0x04 | 0x00 0x00 0x00 0x0F |

response: 0x00 0x01 0x00 0x00 0x00 0x04 | 0x01 0x01 0x00 0x00 |
```

Get the state of an outdated server:

```
request:  0x00 0x01 0x00 0x00 0x00 0x04 | 0x00 0x00 0x00 0x02 |

response:  0x00 0x00 0x00 0x00 0x00 0x00 |
```

SCHUNK SE & Co. KG
Toolholding and Workholding | Gripping Technology | Automation Technology

Bahnhofstr. 106 – 134
D-74348 Lauffen/Neckar
Tel. +49-7133-103-0
Fax +49-7133-103-2399
info@de.schunk.com
schunk.com

Folgen Sie uns | *Follow us*

Wir drucken nachhaltig | *We print sustainable*

08–2023

01.00 | 2D Grasping-Kit | Software manual | en | 1559896