Softwarehandbuch
2D Grasping-Kit
Smart Grasping – Natives Protokoll V4

Original-Softwarehandbuch

Hand in hand for tomorrow

Impressum

Technische Änderungen:
Änderungen im Sinne technischer Verbesserungen sind uns vorbehalten.

Dokumentennummer: 1614334

Auflage: 02.00 | 02.04.2025 | de

Sehr geehrte Kundin,

sehr geehrter Kunde,

vielen Dank, dass Sie unseren Produkten und unserem Familienunternehmen als führendem Technologieausrüster für Roboter und Produktionsmaschinen vertrauen.

Unser Team steht Ihnen bei Fragen rund um dieses Produkt und weiteren Lösungen jederzeit zur Verfügung. Fragen Sie uns und fordern Sie uns heraus. Wir lösen Ihre Aufgabe!

Mit freundlichen Grüßen

Ihr SCHUNK-Team

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

Betriebsanleitung bitte vollständig lesen und produktnah aufbewahren.

Inhaltsverzeichnis

1 Protokoll – Allgemein

Low-Level-Pipeline-Schnittstelle für Robotersteuerungen.

Das Protokoll ist ein Kommunikationsprotokoll, das auf dem TCP/IP-Stack aufbaut. Das Design des Protokolls ist einfach und direkt, sodass es selbst auf älteren Robotern und Industrie-Steuerungen implementiert werden kann.

1.1 Konventionen

- **Port**: Der Server hört auf TCP-Port 42001
- **Byte-Reihenfolge**: Die Bytereihenfolge ist big-endian
- **Gleitkommazahlen**: Single IEEE-Norm (IEC-60559)
- **Ganzzahlen mit Vorzeichen**: Ganzzahlen werden im Zweierkomplement dargestellt
- **Maßeinheiten**: Sofern nicht anders angegeben, werden Längen in Meter [m] und Winkel in Bogenmaß [rad] gemessen.

1.2 Datentypen

Folgende Datentypen sind definiert:

- **Ganzzahlen mit Vorzeichen**: int8, int16, int32
- **Ganzzahlen ohne Vorzeichen**: uint8, uint16, uint32
- **Gleitkommazahlen**: float32

Das Zahlensuffix eines Datentyps gibt dessen Größe in Bits an (z. B. ist int16 zwei Byte lang)

1.3 Versionierung

Der Server ist abwärtskompatibel.

Clients, die frühere Protokollversionen verwenden, werden wie erwartet bedient und bemerken keinen Versionskonflikt. Der Server antwortet im Präfix immer mit derselben Versionsnummer, die auch in der Anfrage verwendet wurde.

Ist der Server jedoch veraltet und seine Protokollversion niedriger als die des Clients, antwortet der Server stets mit einer leeren Nachricht. Das Längenfeld im Nachrichtenpräfix wird auf null gesetzt, und das Versionsfeld enthält die höchste vom Server unterstützte Protokollversion. In diesem Fall sollten Clients den Benutzer darauf hinweisen, dass der Server veraltet und mit der aktuellen Client-Version nicht kompatibel ist.

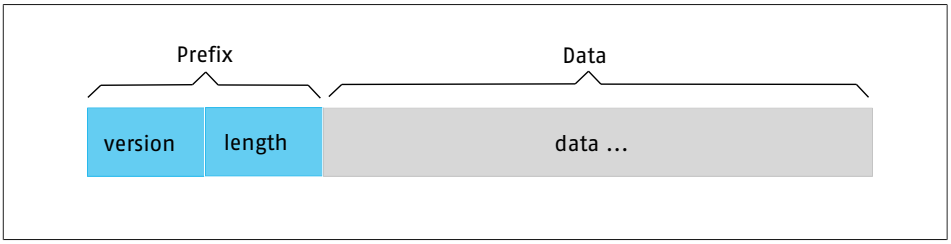
1.4 Aufbau

Jede Nachricht besteht aus zwei Teilen: einem unveränderlichen Präfix, das über verschiedene Protokollversionen hinweg gleichbleibt, und einem versionsspezifischen Teil. Die Details dieser Teile werden in den jeweiligen Kapiteln erläutert.

2 Protokoll – Präfix

Unveränderlicher Teil der Low-Level-Pipeline-Schnittstelle für Robotersteuerungen.

2.1 Aufbau



2.2 Präfix

Alle Nachrichten beginnen mit einem *Präfix*, das die Protokollversion und die Größe der verbleibenden Nachricht definiert.

Das Präfix ist unveränderlich und bleibt über verschiedene Protokollversionen hinweg gleich.

	Byte index	Name	Typ
Präfix	0	version	uint16
	2	length	uint32
Daten	6	...	...

- **version**: Protokollversion.
- **length**: Die Größe der restlichen Nachricht in Byte (max. 4 GB), ohne das *Präfix* selbst (6 B).

2.3 Daten

Eigentlicher Inhalt der Nachricht. Die Struktur ist versionsspezifisch und wird im Versionskapitel beschrieben.

3 Protokoll – V4

Version 4 der Low-Level-Pipeline-Schnittstelle für Robotersteuerungen.

3.1 Allgemeines

- Nachrichtengröße: Nachrichten sind immer 80 Byte lang. Daten
- Datentypen: Nur Ganzzahlen (mit und ohne Vorzeichen). Keine Gleitkommazahlen.
- Aufbau: Festes Layout, bei dem jeder Byte-Index für ein bestimmtes Feld reserviert ist und nicht wiederverwendet wird.

3.2 Aufbau

Byte index	Daten
0	Präfix
6	Header
10 – 79	Body

Anfragen und Antworten folgen einem festen Layout mit einer globalen Zuordnung von Feldern zu bestimmten Byte-Indizes. Jeder Byte-Index ist einem bestimmten Feld zugewiesen und wird nicht wiederverwendet. Je nach Nachrichtentyp enthalten nur bestimmte Abschnitte des Nachrichteninhalts sinnvolle Werte, während andere ignoriert werden. Diese ignorierten Abschnitte sind als <offset> oder <padding> gekennzeichnet.

	Anfrage			*Antwort*		
	Byte index	**Name**	**Typ**	**Byte index**	**Name**	**Typ**
Präfix	0	version	unit16	0	version	uint16
	2	length	unit32	2	length	uint32
Header	6	comm type	unit8	6	comm type	uint8
	7	reply code	unit8	7	reply code	uint8
	8	reply counter	unit8	8	reply counter	uint8
	9	msg type	unit8	9	msg type	uint8
Body	10	client	unit8	10	version	uint16
	11	grasp mode	unit8	12	state	uint8
	12	object id	uint16	13	grasp mode	uint8
	14	selection criterion	unit8	14	object id	uint16
	15	pose format	unit8	16	instance id	uint16
	16	grasp feedback	unit8	18	pose format	uint8

Anfrage			*Antwort*		
Byte index	**Name**	**Typ**	**Byte index**	**Name**	**Typ**
17	*<offset>*	–	19	reference frame	uint8
18	project id	uint16	20	gripper stroke	int32
20	tool id	uint16	24	object offset	int32[3]
22	workspace id	uint16	36	center offset	int32[2]
24	tool pose	int32[7]	44	grasp pose	int32[7]
52 – 79	*<padding>*	–	72	candidate count	uint16
			74	object count	uint16
			76 – 79	*<padding>*	–

3.2.1 Präfix

Jede Nachricht beginnt mit einem festen Präfix, das die Protokollversion und die Länge der restlichen Nachricht enthält.

	Byte index	Wert	Name	Typ
Präfix	0	4	version	uint16
	2	74	length	uint32
Data	6	…	…	…

Bemerkung: Das Längenfeld gibt die Größe der verbleibenden Nachricht an. In dieser Protokollversion beträgt die verbleibende Länge immer 74 Byte.

Layout siehe ▸ 2.1 [🗋 6].

3.2.2 Header

Der *Header* ist der Teil, der den Nachrichtentyp und dessen Zweck kodiert.

	Byte index	Name	Typ
Präfix	0	…	…
Header	6	▸ Kommunikationstyp [🗋 9]	uint8
	7	▸ Antwortcode [🗋 9]	uint8
	8	▸ Antwortzähler [🗋 10]	uint8
	9	▸ Meldungstyp [🗋 10]	uint8
Body	10 – 79	…	…

3.2.2.1 comm type

Gibt an, ob die Nachricht eine Anfrage oder eine Antwort ist.

Wert	Beschreibung
01	Anfrage
02	Antwort

3.2.2.2 reply code

Der *reply code* gibt den Ergebnisstatus der verarbeiteten Anfrage an.

Clients sollten den *reply code* in jeder Antwortnachricht überprüfen und nur fortfahren, wenn der *reply code* auf SUCCESS gesetzt ist. Wenn der *reply code* nicht auf SUCCESS gesetzt ist, kann der Rest der Nachricht ignoriert werden. Der *reply counter* wird jedoch weiterhin hochgezählt.

Wert	Name	Beschreibung
Allgemein		
1	SUCCESS	Die Anfrage wurde erfolgreich bearbeitet.
2	ERROR	Fehler bei der Bearbeitung der Anfrage.
Nachrichtenspezifisch		
3	NO_OBJECT	Kein Objekt gefunden.
4	NO_GRASP	Keinen Griff gefunden.
5	INVALID_OBJECT_CLASS	Die Objekt ID/Klasse ist entweder ungültig oder existiert nicht.
6	CAMERA_NOT_CONNECTED	Die Kamera ist nicht verbunden.
7	CAMERA_NOT_CALIBRATED	Die Kamera ist nicht kalibriert.
8	ROBOT_NOT_CALIBRATED	Der Roboter ist nicht kalibriert.
9	WORKSPACE_NOT_CALIBRATED	Der Arbeitsraum ist nicht kalibriert.
10	NO_ACTIVE_PROJECT	Kein aktives Projekt vorhanden.
11	NO_ACTIVE_TOOL	Kein aktives Werkzeug vorhanden.

HINWEIS

Der *reply code* wird bei Anfragen ignoriert.

3.2.2.3 reply counter

Ein Zähler, der sich mit jeder Antwort erhöht und auf null zurückgesetzt wird, wenn er 256 erreicht.

HINWEIS

Der *reply counter* wird bei Anfragen ignoriert.

3.2.2.4 msg type

siehe Kapitel ▸ 3.3 [🗋 10]

3.3 Nachrichten

3.3.1 Nachrichtentypen

Gibt den Zweck der Nachricht an und wie der Nachrichteninhalt interpretiert werden soll.

Wert	Name	Beschreibung
Allgemeines		
1	▸ GET_PROTOCOL_VERSION [🗋 11]	Protokollversion des Servers abfragen.
2	▸ GET_STATE [🗋 12]	Aktuellen Status des Servers abfragen.
3	▸ REGISTER_CLIENT [🗋 13]	Client-System auf dem Server registrieren.
4	▸ SET_PROJECT [🗋 14]	Aktives Projekt auf dem Server festlegen.
5	▸ SET_TOOL [🗋 15]	Aktives Werkzeug auf dem Server festlegen.
6	▸ SET_WORKSPACE [🗋 16]	Aktiven Arbeitsraum auf dem Server festlegen.
Greifen		
16	▸ GET_GRASP [🗋 17]	Griff für die aktuelle Szene erhalten.
17	▸ GRASP_FEEDBACK [🗋 21]	Server Feedback zum letzten Griff geben.
Erkennung		
32	▸ GET_OBJECT_COUNT [🗋 22]	Anzahl der Objekte in der aktuellen Szene abrufen.
Roboterstatus		
48	▸ TOOL_POSE [🗋 23]	Server über die aktuelle Werkzeugpose benachrichtigen.

3.3.2 Nachrichten

3.3.2.1 GET_PROTOCOL_VERSION

Gibt die höchste unterstützte Protokollversion des Servers zurück. Dies ist nützlich, um festzustellen, ob die Version des Clients veraltet ist oder nicht. Sollte dies der Fall sein, wird dem Client empfohlen, den Anwender entsprechend zu informieren.

Anfrage:

	Byte index	Name	Wert	Typ
Präfix	0	version	4	uint16
Header	2	length	74	uint32
	6	comm type	1	uint8
	7	reply code	–	uint8
	8	reply counter	–	uint8
	9	msg type	1	uint8
Body	10–79	*<padding>*	–	–

Antwort:

	Byte index	Name	Wert	Typ
Präfix	0	version	4	uint16
Header	2	length	74	uint32
	6	comm type	2	uint8
	7	reply code	x	uint8
	8	reply counter	x	uint8
	9	msg type	1	uint8
Body	10	version	x	uint16
	10–79	*<padding>*	–	–

Version: Höchste unterstützte Protokollversion des Servers.

3.3.2.2 GET_STATE

Ruft den Status des Servers ab.

Anfrage:

	Byte index	Name	Wert	Typ
Präfix	0	version	4	uint16
Header	2	length	74	uint32
	6	comm type	1	uint8
	7	reply code	–	uint8
	8	reply counter	–	uint8
	9	msg type	2	uint8
Body	10–79	*<padding>*	–	–

Antwort:

	Byte index	Name	Wert	Typ
Präfix	0	version	4	uint16
Header	2	length	74	uint32
	6	comm type	2	uint8
	7	reply code	x	uint8
	8	reply counter	x	uint8
	9	msg type	1	uint8
Body	10	*<offset>*	–	–
	12	state	x	uint8
	13–79	*<padding>*	–	–

state:

Wert	Name	Beschreibung
1	INIT	Der Server wird initialisiert.
2	OPERATIONAL	Der Server ist betriebsbereit und kann Anfragen bearbeiten.
3	STOPPED	Der Server hat den Betrieb gestoppt.
4	ERROR	Auf dem Server ist ein kritischer Fehler aufgetreten.

3.3.2.3 REGISTER_CLIENT

Registriert das Client-System (zu Informationszwecken).

Diese Nachricht sollte direkt nach dem Aufbau der Verbindung gesendet werden. Es gibt jedoch keine Einschränkungen, wann und wie oft sie gesendet werden kann. Wenn das Client-System nicht offiziell in der Liste aufgeführt ist (siehe Client-Tabelle in den Anfragedetails), kann diese Nachricht einfach ignoriert werden.

Anfrage:

Enthält Informationen über das Robotersystem des Clients.

	Byte index	Name	Wert	Type
Präfix	0	version	4	uint16
	2	length	74	uint32
Header	6	comm type	1	uint8
	7	reply code	–	uint8
	8	reply counter	–	uint8
	9	msg type	3	uint8
Body	10	client	x	uint8
	11 – 79	*<padding>*	–	–

client:

Wert	Name	Anbieter
1	UR	Universal Robots
2	KUKA	KUKA
3	Yaskawa	Yaskawa
4	FANUC	FANUC
5	ABB	ABB
6	HORST	fruitcore robotics
128	Siemens PLC	Siemens
129	Allen-Bradley PLC	Allen-Bradley

Antwort:

Der Antwort-Body ist leer.

	Byte index	Name	Wert	Typ
Präfix	0	version	4	uint16
	2	length	74	uint32
Header	6	comm type	2	uint8
	7	reply code	x	uint8
	8	reply counter	x	uint8
	9	msg type	3	uint8
Body	10–79	*<padding>*	–	–

3.3.2.4 SET_PROJECT

Legt das aktive Projekt auf dem Server fest.

Anfrage:

	Byte index	Name	Wert	Type
Präfix	0	version	4	uint16
	2	length	74	uint32
Header	6	comm type	1	uint8
	7	reply code	–	uint8
	8	reply counter	–	uint8
	9	msg type	4	uint8
Body	10	*<offset>*	–	–
	18	project id	x	uint16
	20 – 79	*<padding>*	–	–

project id: Die ID des zu aktivierenden Projekts.

Antwort:

Der Antwort-Body ist leer. Der reply code gibt an, ob die Operation erfolgreich war oder nicht.

	Byte index	Name	Wert	Typ
Präfix	0	version	4	uint16
	2	length	74	uint32
Header	6	comm type	2	uint8
	7	reply code	x	uint8
	8	reply counter	x	uint8
	9	msg type	4	uint8
Body	10–79	*<padding>*	–	–

3.3.2.5 SET_TOOL

Legt das aktive Werkzeug auf dem Server fest.

Anfrage:

	Byte index	Name	Wert	Typ
Präfix	0	version	4	uint16
	2	length	74	uint32
Header	6	comm type	1	uint8
	7	reply code	–	uint8
	8	reply counter	–	uint8
	9	msg type	5	uint8
Body	10	*<offset>*	–	–
	20	tool id	x	uint16
	22 – 79	*<padding>*	–	–

tool id: ID des zu aktivierenden Werkzeugs

Antwort:

Der Antwort-Body ist leer. Der *reply code* gibt an, ob die Operation erfolgreich war.

	Byte index	Name	Wert	Typ
Präfix	0	version	4	uint16
	2	length	74	uint32
Header	6	comm type	2	uint8
	7	reply code	x	uint8
	8	reply counter	x	uint8
	9	msg type	5	uint8
Body	10–79	*<padding>*	–	–

3.3.2.6 SET_WORKSPACE

Legt den aktiven Arbeitsraum auf dem Server fest.

Anfrage:

	Byte index	Name	Wert	Typ
Präfix	0	version	4	uint16
	2	length	74	uint32
Header	6	comm type	1	uint8
	7	reply code	–	uint8
	8	reply counter	–	uint8
	9	msg type	6	uint8
Body	10	*<offset>*	–	–
	22	workspace id	x	uint8
	24 – 79	*<padding>*	–	–

workspace id: ID des zu aktivierenden Arbeitsraums.

Antwort:

Der Antwort-Body ist leer. Der *reply code* gibt an, ob die Operation erfolgreich war.

	Byte index	Name	Wert	Typ
Präfix	0	version	4	uint16
	2	length	74	uint32
Header	6	comm type	2	uint8
	7	reply code	x	uint8
	8	reply counter	x	uint8
	9	msg type	6	uint8
Body	10–79	*<padding>*	–	–

3.3.2.7 GET_GRASP

Fordert einen Griff für die aktuelle Szene an.

Anfrage:

	Byte index	Name	Wert	Typ
Präfix	0	version	4	uint16
	2	length	74	uint32
Header	6	comm type	1	uint8
	7	reply code	–	uint8
	8	reply counter	–	uint8
	9	msg type	16	uint8
Body	10	*<offset>*	–	–
	11	grasp mode	x	uint8
	12	object id	x	uint16
	14	selection criterion	x	uint8
	15	pose format	x	uint8
	16 – 79	*<padding>*	–	–

grasp mode:

Wert	Name	Beschreibung
1	ACTIVE_GRASP	Der gefundene Griff muss der **aktiven**, benutzerdefinierten Griffdefinition des Zielobjekts entsprechen.
2	ANY_GRASP	Der gefundene Griff muss mit **einer** der benutzerdefinierten Griffdefinitionen des Zielobjekts übereinstimmen.
3	AUTO_GRASP	Der Server bestimmt den optimalen Griff, wobei benutzerdefinierte Griffdefinitionen nach Möglichkeit bevorzugt werden.

- *object id*: Gibt die Objekt-ID des Zielobjekts an. Alle Objekte mit der angegebenen Objekt-ID werden als Kandidaten betrachtet. Ist die Objekt-ID null, werden alle Objekte, die kein Hindernis darstellen, als Kandidaten betrachtet. Das Zielobjekt wird dann anhand des angegebenen Auswahlkriteriums aus diesen Kandidaten ausgewählt.
- *selection criterion*: Definiert die Methode zur Auswahl des Zielobjekts aus den verfügbaren Kandidaten.

Wert	Name	Beschreibung
1	RANDOM	Wählt einen zufälligen Kandidaten aus.

Wert	Name	Beschreibung
2	CENTRAL	Wählt den Kandidaten aus, der sich am zentralsten im Bild befindet. Ist dieser Kandidat nicht greifbar, schlägt der Griff fehl.
3	ISOLATED	Wählt den am isoliertesten gelegenen Kandidaten unter allen Objekten aus. Ist dieser Kandidat nicht greifbar, wird der nächstisolierte Kandidat ausgewählt usw.
4	ROWS	Wählt die Kandidaten Zeile für Zeile von links nach rechts aus. Wenn ein Kandidat nicht greifbar ist, schlägt der Griff fehl.

pose format: Gibt das in der Antwortnachricht zu verwendende Posenformat an. Siehe ▸ 5 [🗋 25]

Antwort:

	Byte index	Name	Wert	Typ
Präfix	0	version	4	uint16
	2	length	74	uint32
Header	6	comm type	2	uint8
	7	reply code	x	uint8
	8	reply counter	x	uint8
	9	msg type	6	uint8
Body	10	*<offset>*	–	–
	13	grasp mode	x	uint8
	14	object id	x	uint16
	16	instance id	x	uint8
	18	pose format	x	uint8
	19	reference frame	x	uint8
	20	gripper stroke	x	int32
	24	object offset	x	int32[3]
	36	center offset	x	int32[2]
	44	grasp pose	x	int32[7]
	72	candidate count	x	uint16
	74	object count	x	uint16
	76–79	*<padding>*	–	–

- *grasp mode*: Gibt den verwendeten Greifmodus an (siehe Anfragedetails).

Hinweis: Dieser Greifmodus ist nicht unbedingt derselbe wie der angeforderte Greifmodus. Wird z. B. ein automatischer Griff angefordert, bevorzugt der Server dennoch benutzerdefinierte Griffe und greift nur dann auf automatische Griffe zurück, wenn erstere entweder nicht existieren oder nicht anwendbar sind.

- *object id*: ID des Zielobjekts
- *instance id*: Instanz-ID des Zielobjekts.
- *pose format*: Das Posenformat, das für das Feld *grasp pose* verwendet wird. Siehe ▸ 5 [🗋 25]
- *reference frame*: Gibt das Bezugssystem der Greifpose an. Wenn die Kamera am Roboter montiert ist, ist dies häufig das Werkzeug oder der Roboterflansch. Bei Kameras, die auf einem externen Stativ montiert sind, ist dies oft die Roboterbasis oder das Weltbezugssystem. Das spezifische Bezugssystem hängt davon ab, wie der Roboter mit der Kamera kalibriert wurde..

Wert	Name	Beschreibung
1	BASE	Basis/Welt (Kamera auf externem Stativ montiert)
2	TOOL	Werkzeug/Flansch (Kamera am Roboter montiert)

- *gripper stroke*: Der Abstand in Mikrometern zwischen den Greiferfingern, der vor dem Annähern an das Objekt eingestellt werden muss. Dieses Feld ist nur für Greifer relevant und kann bei anderen Werkzeugtypen ignoriert werden.

  Hinweis: Es wird angenommen, dass sich der Greifer in der Position Null befindet, wenn sich die Finger berühren.
- *object offset*: Gibt die (x, y, rz)-Pose des Objekts im Bezugssystem des Werkzeugs zum Zeitpunkt des Griffs an.
  - Anordnung: [x, y, rz]
  - x, y: Position in Mikrometern
  - rz: Drehung in Mikrograd um die z-Achse des Werkzeugs

  Der *object offset* kann verwendet werden, um Objekte nach dem Greifen immer in der gleichen Pose abzulegen. Ein Beispiel für die Reihenfolge der Operationen ist wie folgt:

  1. Nach dem Greifen das Werkzeug an eine fest definierte Pose anfahren.
  2. Das Werkzeug um -rz um die z-Achse drehen.

3. Das Werkzeug um (-x, -y) relativ zu seinem eigenen Bezugssystem verschieben.

4. Das Objekt ist nun unabhängig vom angewandten Griff einheitlich positioniert.

Wichtig: Das Werkzeugkoordinatensystem des Roboters muss mit dem vom Server definierten Werkzeugkoordinatensystem übereinstimmen. Die Achsen beider Koordinatensysteme müssen gleich ausgerichtet sein.

- *center offset*: Gibt den (x, y) -Versatz relativ zur Greifpose an, um den Mittelpunkt des Zielobjekts mit dem Zentrum des Kamerabilds auszurichten. Die Greifpose entspricht dem Werkzeugkoordinatensystem zum Zeitpunkt des Greifens. Dieses Feld ist nur relevant, wenn die Kamera auf dem Roboter montiert ist.

  Diese Anpassung hilft, perspektivische Verzerrungen zu minimieren, insbesondere bei höheren Objekten. Das Werkzeug kann (ohne das Objekt zu greifen) an die angepasste Position bewegt werden. Anschließend kann ein neuer Greifvorgang mit dem Auswahlkriterium CENTRAL angefordert werden, um das nun zentral im Kamerabild positionierte Objekt zu greifen.

  Um das Werkzeug an die angepasste Position zu bewegen, kann es um (-x, -y) relativ zur Greifpose verschoben werden.

  Wichtig: Das Werkzeugkoordinatensystem des Roboters muss mit dem vom Server definierten Werkzeugkoordinatensystem übereinstimmen. Die Achsen beider Koordinatensysteme müssen gleich ausgerichtet sein.
- *grasp pose*: Die Greifpose. Das Bezugssystem wird im Feld reference frame angegeben. Das Posenformat wird im Feld *pose format* angegeben.
- *tool id*: ID des verwendeten Werkzeugs
- *candidate count*:Die Anzahl der Kandidatenobjekte, einschließlich des ausgewählten. Wenn der Wert 1 beträgt, bedeutet dies, dass nach dem Greifvorgang keine Objekte mit der angegebenen object id mehr vorhanden sind.
- *object count*: Die Gesamtzahl aller erkannten Objekte.

3.3.2.8 GRASP_FEEDBACK

Stellt dem Server optionales Feedback zum letzten Greifvorgang bereit.

Anfrage:

	Byte index	Name	Wert	Typ
Präfix	0	version	4	uint16
	2	length	74	uint32
Header	6	comm type	1	uint8
	7	reply code	–	uint8
	8	reply counter	–	uint8
	9	msg type	17	uint8
Body	10	*<offset>*	–	–
	16	grasp feedback	x	uint8
	11 – 79	*<padding>*	–	–

grasp feedback: Rückmeldung über das letzte Greifergebnis.

Wert	Name	Beschreibung
1	OK	Der Griff war okay.
2	BAD	Der Griff war nicht okay.

Antwort:

Der Antwort-Body ist leer.

	Byte index	Name	Wert	Typ
Präfix	0	version	4	uint16
	2	length	74	uint32
Header	6	comm type	2	uint8
	7	reply code	x	uint8
	8	reply counter	x	uint8
	9	msg type	6	uint8
Body	10 – 79	*<padding>*	–	–

3.3.2.9 GET_OBJECT_COUNT

Fordert die Anzahl der Objekte in der aktuellen Szene an.

Anfrage:

	Byte index	Name	Wert	Typ
Präfix	0	version	4	uint16
	2	length	74	uint32
Header	6	comm type	1	uint8
	7	reply code	–	uint8
	8	reply counter	–	uint8
	9	msg type	32	uint8
Body	10	*<offset>*	–	–
	12	object id	x	uint16
	14 – 79	*<padding>*	–	–

object id: ID der zu zählenden Objekte. Ist der Wert Null, werden alle Objekte gezählt.

Antwort:

	Byte index	Name	Wert	Typ
Präfix	0	version	4	uint16
	2	length	74	uint32
Header	6	comm type	2	uint8
	7	reply code	x	uint8
	8	reply counter	x	uint8
	9	msg type	32	uint8
Body	10	*<offset>*	–	–
	72	candidate count	x	uint16
	74	object count	x	uint16
	76–79	*<padding>*	–	–

- *candidate count*: Anzahl der gefundenen Objekte mit der angeforderten ID.
- *object count*: Gesamtzahl aller erkannten Objekte.

3.3.2.10 TOOL_POSE

Teilt dem Server die aktuelle Werkzeugpose im Basis-/Weltkoordinatensystem des Roboters mit.

Anfrage:

	Byte index	Name	Wert	Typ
Präfix	0	version	4	uint16
	2	length	74	uint32
Header	6	comm type	1	uint8
	7	reply code	–	uint8
	8	reply counter	–	uint8
	9	msg type	48	uint8
Body	10	*<offset>*	–	–
	15	pose format	x	uint8
	16	*<offset>*	–	–
	24	tool pose	x	int32[7]
	52–79	*<padding>*	–	–

- *pose format*: Legt das Posenformat fest. Siehe dazu Abschnitt ▸ 5 [🗋 25], um alle unterstützten Formate zu sehen.
- *tool pose*: Werkzeugpose im Basis-/Weltkoordinatensystem des Roboters.

Antwort:

Der Antwort-Body ist leer.

	Byte index	Name	Wert	Typ
Präfix	0	version	4	uint16
	2	length	74	uint32
Header	6	comm type	2	uint8
	7	reply code	x	uint8
	8	reply counter	x	uint8
	9	msg type	48	uint8
Body	10–79	*<padding>*	–	–

4 Visualisierung

Eine Visualisierung des letzten Erkennungsergebnisses wird als Bildressource bereitgestellt, auf die über HTTP zugegriffen werden kann:

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

Für ältere Clients, die nicht mit vielen Pixeln umgehen können, ist auch eine niedrig aufgelöste Version der Visualisierung verfügbar:

`http://<server-ip>/monitor/latest_result_compressed`

5 Posenformate

Die folgenden Posenformate werden unterstützt:

Wert	Name	Typ	Aufbau	Beschreibung
Allgemeine Formate				
1	Quaternion	int32[7]	Position: x, y, z; Orientierung: qx, qy, qz, qw	Position und unnormalisiertes Quaternion. *Bemerkung: Um das Quaternion zu normalisieren, das Quaternion durch 10^6 teilen.* *Einheiten*: Mikrometer und Radiant.
2	Rotations-vektor	int32[7]	Position: x, y, z; Orientierung: x, y, z, –	Position und unnormalisierter Rotationsvektor. *Einheiten*: Mikrometer und Mikroradiant. *Anbieter*: Universal Robots
Anbieterspezifische Formate				
16	WPR	int32[7]	Position: x, y, z; Orientierung: W, P, R, –	Position und Ausrichtung WPR mit W, das sich um die feste x-Achse dreht, dann P, das sich um die feste y-Achse dreht, dann R, das sich um die feste z-Achse dreht. *Rotationskonvention*: x-y-z (extrinsisch). *Einheiten*: Mikrometer und Mikrograd. *Anbieter*: Fanuc, Yaskawa
17	ABC	int32[7]	Position: x, y, z; Orientierung: A, B, C, –	Position und Ausrichtung ABC mit A, das sich um die z-Achse dreht, dann B, das sich um die y'-Achse dreht, dann C, das sich um die x''-Achse dreht. *Rotationskonvention*: x-y-z (intrinsisch). *Einheiten*: Mikrometer und Mikrograd. *Anbieter*: Kuka

6 Beispiele

Angenommen, eine TCP-Verbindung zum Server wurde erfolgreich eingerichtet.

Für einfache Greifaufgaben besteht die grundlegendste Nachrichtenfolge aus:

1. ▸ GET_PROTOCOL_VERSION [🗋 26]
2. ▸ GET_STATE [🗋 27]
3. ▸ REGISTER_CLIENT [🗋 28]
4. ▸ SET_PROJECT [🗋 29]
5. ▸ GET_GRASP [🗋 30]

6.1 GET_PROTOCOL_VERSION

Nach dem Verbindungsaufbau ist als Erstes die höchste unterstützte Protokollversion des Servers zu prüfen.

	Anfrage				*Antwort*			
	Index	**Name**	**Wert**	**Typ**	**Index**	**Name**	**Wert**	**Typ**
Präfix	0	version	4	unit16	0	version	4	uint16
	2	length	74	unit32	2	length	74	uint32
Header	6	comm type	1	unit8	6	comm type	2	uint8
	7	reply code	0	unit8	7	reply code	1	uint8
	8	reply counter	0	unit8	8	reply counter	0	uint8
Body	9	msg type	1	unit8	9	msg type	1	uint8
	10 – 79	*<padding>*	0	–	10	version	4	uint16
					12 – 79	*<padding>*	0	–

Daten

```
000 004 000 000 000 074 001 000
000 001 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
```

Daten

```
000 004 000 000 000 074 002 001
000 001 000 004 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
```

In diesem Beispiel antwortet der Server mit version= 4. Dies bedeutet, dass der Client auf dem neuesten Stand ist.

6.2 GET_STATE

Bevor weitere Anfragen an den Server gesendet werden, ist der Status des Servers zu überprüfen.

	Anfrage				*Antwort*			
	Index	**Name**	**Wert**	**Typ**	**Index**	**Name**	**Wert**	**Typ**
Präfix	0	version	4	unit16	0	version	4	uint16
	2	length	74	unit32	2	length	74	uint32
Header	6	comm type	1	unit8	6	comm type	2	uint8
	7	reply code	0	unit8	7	reply code	1	uint8
	8	reply counter	0	unit8	8	reply counter	1	uint8
Body	9	msg type	2	unit8	9	msg type	2	uint8
	10 – 79	*<padding>*	0	–	10	*<offset>*	0	–
					12	state	2	
					13 – 79	*<padding>*	0	–

Daten

```
000 004 000 000 000 074 001 000
000 002 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
```

Daten

```
000 004 000 000 000 074 002 001
001 002 000 002 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
```

In diesem Beispiel antwortet der Server mit state = 2. Das bedeutet, dass der Server funktionsfähig ist und der Vorgang fortgesetzt werden kann.

6.3 REGISTER_CLIENT

Client als Siemens SPS registrieren.

	Anfrage				*Antwort*			
	Index	**Name**	**Wert**	**Typ**	**Index**	**Name**	**Wert**	**Typ**
Präfix	0	version	4	unit16	0	version	3	uint16
	2	length	74	unit32	2	length	74	uint32
Header	6	comm type	1	unit8	6	comm type	2	uint8
	7	reply code	0	unit8	7	reply code	1	uint8
	8	reply counter	0	unit8	8	reply counter	2	uint8
Body	9	msg type	3	unit8	9	msg type	3	uint8
	10	client	128	unit8	10 – 79	*<padding>*	0	–
	11 – 79	*<padding>*	0	–				

Daten

```
000 004 000 000 000 074 001 000
000 003 128 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
```

Daten

```
000 004 000 000 000 074 002 001
002 003 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
```

Der Antwort-Body ist leer.

6.4 SET_PROJECT

Projekt mit dem Index 5 aktivieren, das die Objekte enthält, die erkannt und gegriffen werden sollen.

	Anfrage				*Antwort*			
	Index	**Name**	**Wert**	**Typ**	**Index**	**Name**	**Wert**	**Typ**
Präfix	0	version	4	unit16	0	version	4	uint16
	2	length	74	unit32	2	length	74	uint32
Header	6	comm type	1	unit8	6	comm type	2	uint8
	7	reply code	0	unit8	7	reply code	1	uint8
	8	reply counter	0	unit8	8	reply counter	3	uint8
Body	9	msg type	4	unit8	9	msg type	4	uint8
	10	*<offset>*	0	-	10 - 79	*<padding>*	0	-
	18	project id	5	unit16				
	20 - 79	*<padding>*	0	-				

Daten

```
000 004 000 000 000 074 001 000
000 004 000 000 000 000 000 000
000 000 000 005 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 005 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
```

Daten

```
000 004 000 000 000 074 002 001
003 004 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
```

In diesem Beispiel war die Aktivierung des Projekts mit Index 5 erfolgreich.

Der Antwort-Body ist leer.

Bemerkung: Den reply code im Antwort-Header überprüfen und nur fortfahren, wenn der reply code 1 (Erfolg) ist.

6.5 GET_GRASP

Anforderung eines automatischen Griffs für das am zentralsten gelegene Objekt. Die Greifpose soll in Quaternion-Darstellung zurückgegeben werden.

	Anfrage				*Antwort*			
	Index	**Name**	**Wert**	**Typ**	**Index**	**Name**	**Wert**	**Typ**
Präfix	0	version	4	unit16	0	version	4	uint16
	2	length	74	unit32	2	length	74	uint32
Header	6	comm type	1	unit8	6	comm type	2	uint8
	7	reply code	0	unit8	7	reply code	1	uint8
	8	reply counter	0	unit8	8	reply counter	4	uint8
Body	9	msg type	16	unit8	9	msg type	16	uint8
	10	*<offset>*	0	-	10	*<offset>*	0	-
	11	grasp mode	3	unit8	13	grasp mode	2	uint8
	12	object id	0	unit16	14	object id	3	uint16
	14	selection criterion	2	unit8	16	instance id	0	uint8
	15	pose format	1	unit8	18	pose format	1	uint8
	16 - 79	*<padding>*	0	-	19	reference frame	2	uint8
					20	gripper stroke	86000	int32
					24	object offset	[16112, 8102, -25210142]	int32[3]
					36	center offset	[-123911, -225418, 68]	int32[2]
					44	grasp pose	[853798, 1111998, 502790, 775990, 630744, 0, 0]	int32[7]
					72	candidate count	2	uint16
					74	object count	4	uint16
					76 - 79	*<padding>*	0	-

Anfrage				*Antwort*			
Index	**Name**	**Wert**	**Typ**	**Index**	**Name**	**Wert**	**Typ**
Daten				**Daten**			

```
000 004 000 000 000 074 001 000
000 016 000 003 000 000 002 001
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
000 000 000 000 000 000 000 000
```

```
000 004 000 000 000 074 002 001
004 016 000 000 000 002 000 003
000 000 001 002 000 001 079 240
000 000 062 240 000 000 031 166
254 127 082 226 255 254 027 249
255 252 143 118 000 013 007 038
000 016 247 190 000 007 172 006
000 011 215 054 000 009 159 216
000 000 000 000 000 000 000 000
000 002 000 004 000 000 000 000
```

In diesem Beispiel ist der zurückgegebene Griff ein benutzerdefinierter Griff (grasp mode = 2), und die Objekt-ID des ausgewählten zentralen Objekts ist 3. Obwohl ein automatischer Griff angefordert wurde, bevorzugt das System immer benutzerdefinierte Griffe und greift nur auf automatische Griffe zurück, wenn letzterer entweder nicht existiert oder nicht anwendbar ist.

In der Antwort ist ersichtlich, dass die Kandidatenanzahl 2 beträgt, was bedeutet, dass nach der Anwendung des Griffs noch ein Kandidatenobjekt in der Szene verbleibt.

Bemerkung: Das Anfordern eines Griffs kann aus verschiedenen Gründen fehlschlagen, wie z. B. einer nicht kalibrierten Kamera, der Unfähigkeit, das Objekt zu lokalisieren, oder einem unpraktischen Griff. Es ist wichtig, den reply code im Antwort-Header zu überprüfen und nur fortzufahren, wenn der Antwortcode 1 (Erfolg) ist.

Schritte 1 bis 4 werden nur einmal ausgeführt. Schritt 5 wird typischerweise in einer Schleife wiederholt.

SCHUNK SE & Co. KG
Spanntechnik | Greiftechnik | Automatisierungstechnik

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

Folgen Sie uns | *Follow us*

Wir drucken nachhaltig | *We print sustainable*

04-2025

02.00 | 2D Grasping-Kit | Softwarehandbuch | de | 1614334