ksapi

Handhabung und Beispiel-FB

Dokumentation
Version 1.1.0

Andreas Schüller

Lehrstuhl für Prozessleittechnik
Prof. Dr.-Ing. Ulrich Epple
RWTH Aachen
D-52064 Aachen, Deutschland
Telefon +49 (0) 241 80 94339
Fax +49 (0) 241 80 92238
www.plt.rwth-aachen.de

Inhaltsverzeichnis

1 Die Bibliothek *ksapi* **1**

1.1 setString . . . 1
1.1.1 Welche Einstellungen kann man vornehmen? . . . 1
1.1.2 Wie starte ich den Sendevorgang? . . . 1
1.2 getString . . . 2
1.2.1 Welche Einstellungen kann man vornehmen? . . . 2
1.2.2 Wie starte ich den Lesevorgang? . . . 2
1.3 ~~getEPidentifiers~~ –> funktioniert noch nicht . . . 2
1.3.1 ~~Welche Einstellungen kann man vornehmen?~~ . . . 3
1.3.2 ~~Wie starte ich den Lesevorgang?~~ . . . 3
1.4 Übersicht der implementierten Variablentypen . . . 4
1.5 createObject . . . 4
1.5.1 Welche Einstellungen kann man vornehmen? . . . 4
1.5.2 Wie starte ich den Sendevorgang? . . . 5
1.6 renameObject . . . 5
1.6.1 Welche Einstellungen kann man vornehmen? . . . 5
1.6.2 Wie starte ich den Sendevorgang? . . . 5
1.7 deleteObject . . . 5
1.7.1 Welche Einstellungen kann man vornehmen? . . . 5
1.7.2 Wie starte ich den Sendevorgang? . . . 6
1.8 linkObject . . . 6
1.8.1 Welche Einstellungen kann man vornehmen? . . . 6
1.8.2 Wie starte ich den Sendevorgang? . . . 6
1.9 unlinkObject . . . 6
1.9.1 Welche Einstellungen kann man vornehmen? . . . 6
1.9.2 Wie starte ich den Sendevorgang? . . . 7

2 Die Beispielbibliothek *ksapiexample* **7**

2.1 setStringexample . . . 7
2.2 Der FB *client* . . . 8
2.3 Die Links **assoc* . . . 8

3 Anhang **9**

3.1 Übersicht der Variablennamen und -bedeutungen . . . 9
3.2 ~~Fehlercodes~~ . . . 10

1 Die Bibliothek *ksapi*

Die Bibliothek *ksapi* ist eine Sammlung verschiedener Klassen. Jede Klasse liest oder setzt jeweils einen Variablentyp von/auf einem anderen Server. ~~Dazu wird eine KS-Connection zu einem Server auf einem Host aufgebaut und ein KS-Service auf dieser Connection übernimmt dann den Datentransfer.~~ Die Variablennamen in der *ksapi* sind konsistent mit denen in der *ksapiexample*. Im Folgenden werden die Klassen *setString*, *getString* und *getEPidentifier* exemplarisch beschrieben. Alle anderen Variablentypen funktionieren analog. Eine Übersicht, welche Variablen bereits implementiert sind, findet man in Kapitel 1.4.

1.1 setString

1.1.1 Welche Einstellungen kann man vornehmen?

Zum einen können natürlich der Host, der Server und der Pfad des Empfängers eingestellt werden sowie der zu sendende String. Dies kann entweder über die Set-Funktionen der einzelnen Variablen geschehen oder durch Parameterübergabe an die entsprechende Funktion *setandsubmit*. ~~Zum anderen kann auch die Anzahl der Fehler, die auftreten dürfen, eingegeben werden und man kann entscheiden, ob die Verbindung nach dem Ende des Sendevorgangs offen bleiben soll oder nicht.~~
Alle Änderungen der Variablen dürfen nur über die entsprechenden Set-Methoden vorgenommen werden; bei der Parameterübergabe wird dies intern gemacht. Wenn die Variablen direkt gesetzt werden, werden eventuell wichtige Aufräumaktionen nicht durchgeführt.

1.1.2 Wie starte ich den Sendevorgang?

Zuerst muss eine Rückgabemethode definiert werden. Was das ist, ist im nächsten Abschnitt erklärt. Danach gibt es zwei Möglichkeiten, den Sendevorgang zu starten:

1. Man setzt die benötigten Variablen über ihre Set-Funktionen und ruft die Methode *submit* auf

2. Man übergibt der Methode *setandsubmit* die benötigten Variablen als Parameter

Das weitere Vorgehen ist wieder einheitlich und läuft im Hintergrund ab. ~~Falls noch keine Verbindung existiert, wird eine erzeugt, ansonsten wird die bestehende weiter benutzt. Auf der Verbindung wird dann ein KS-Service erzeugt, der den String dann an den Empfänger sendet. Danach wird der Service wieder geschlossen und eine Meldung an die Rückgabemethode übergeben. Falls die entsprechende Variable gesetzt ist, wird die Verbindung auch geschlossen, sonst ist sie danach wieder offen.~~

Was ist die Rückgabemethode? Da das Objekt *setString* nicht weiß, wer ihn aufgerufen hat, kann es auch keine Rückmeldung an den übergeordneten Prozess geben. Deswegen muss in der aufrufenden Klasse eine Methode mit der entsprechenden Syntax (sie muss vom Typ *OV_FNCPTR_KSAPI_FNC_RET* sein) erzeugt werden, die dann entsprechend mit dem aufrufenden Objekt interagieren kann. Dieses Objekt weiß, wo die *setString* in der Hierarchie

liegt, kann also den Weg rückwärts entlanggehen. In dieser Methode können dann die eventuell auftretenden Fehler behandelt werden.

Welche Fehler können auftreten? Beim Senden können verschiedene Fehler auftreten, die an die Rückgabemethode übergeben werden. Eine Übersicht steht unter Kapitel 3.2.

1.2 getString

1.2.1 Welche Einstellungen kann man vornehmen?

Zum einen können natürlich der Host, der Server und der Pfad der Variable, die gelesen werden soll, eingestellt werden. Dies kann entweder über die Set-Funktionen der einzelnen Variablen geschehen oder durch Parameterübergabe an die entsprechende Funktion *setandsubmit*. Zum anderen kann auch die Anzahl der Fehler, die auftreten dürfen, eingegeben werden und man kann entscheiden, ob die Verbindung nach dem Ende des Sendevorgangs offen bleiben soll oder nicht.
Alle Änderungen der Variablen dürfen nur über die entsprechenden Set-Methoden vorgenommen werden; bei der Parameterübergabe wird dies intern gemacht. Wenn die Variablen direkt gesetzt werden, werden eventuell wichtige Aufräumaktionen nicht durchgeführt.

1.2.2 Wie starte ich den Lesevorgang?

Auch hier muss zuerst eine Rückgabemethode definiert werden. Zusätzlich zu den Aufgaben, die in Kapitel 1.1.2 erklärt sind, muss in dieser Methode auch die Weiterverarbeitung des gelesenen Strings erfolgen. Danach gibt es zwei Möglichkeiten, den Lesevorgang zu starten:

1. Man setzt die benötigten Variablen über ihre Set-Funktionen und ruft die Methode *submit* auf
2. Man übergibt der Methode *setandsubmit* die benötigten Variablen als Parameter

Das weitere Vorgehen ist wieder einheitlich und läuft im Hintergrund ab. Falls noch keine Verbindung existiert, wird eine erzeugt, ansonsten wird die bestehende weiter benutzt. Auf der Verbindung wird dann ein KS-Service erzeugt, der den String dann vom Empfänger liest. Danach wird der Service wieder geschlossen und eine Meldung an die Rückgabemethode übergeben. Falls die entsprechende Variable gesetzt ist, wird die Verbindung auch geschlossen, sonst ist sie im Anschluss wieder offen.

Welche Fehler können auftreten? Auch beim Lesen können verschiedene Fehler auftreten, die an die Rückgabemethode übergeben werden. Eine Übersicht steht im Anhang unter Kapitel 3.2.

1.3 getEPidentifiers –> funktioniert noch nicht

Die Klasse *getEPidentifiers* gibt eine Liste der Identifier aller unter dem angegebenen Pfad sich befindenden Kinder an. Diese müssen von der Klasse *ov_domain* sein. Die Klasse liest die

EP-Objekte als ganzes aus, gibt aber nur die Identifier zurück. Sollten andere Attribute benötigt werden, muss man die reqcb-Methode entsprechend ändern.

1.3.1 Welche Einstellungen kann man vornehmen?

Zum einen können natürlich der Host, der Server und der Pfad der Domain, deren Kinder man lesen möchte, sowie ein regulärer Ausdruck, der die Identifier filtert, eingestellt werden. Dies kann entweder über die Set-Funktionen der einzelnen Variablen geschehen oder durch Parameterübergabe an die entsprechende Funktion *setandsubmit*. Zum anderen kann auch die Anzahl der Fehler, die auftreten dürfen, eingegeben werden und man kann entscheiden, ob die Verbindung nach dem Ende des Sendevorgangs offen bleiben soll oder nicht.
Alle Änderungen der Variablen dürfen nur über die entsprechenden Set-Methoden vorgenommen werden; bei der Parameterübergabe wird dies intern gemacht. Wenn die Variablen direkt gesetzt werden, werden eventuell wichtige Aufräumaktionen nicht durchgeführt.

1.3.2 Wie starte ich den Lesevorgang?

Auch hier muss zuerst eine Rückgabemethode definiert werden. Zusätzlich zu den Aufgaben, die in Kapitel 1.1.2 erklärt sind, muss in dieser Methode auch die Weiterverarbeitung der gelesenen Strings, die in einem Vektor gespeichert werden, erfolgen. Danach gibt es zwei Möglichkeiten, den Lesevorgang zu starten:

1. Man setzt die benötigten Variablen über ihre Set-Funktionen und ruft die Methode *submit* auf
2. Man übergibt der Methode *setandsubmit* die benötigten Variablen als Parameter

Das weitere Vorgehen ist wieder einheitlich und läuft im Hintergrund ab. Falls noch keine Verbindung existiert, wird eine erzeugt, ansonsten wird die bestehende weiter benutzt. Auf der Verbindung wird dann ein KS-Service erzeugt, der die EP-Objekte vom Empfänger liest und die Identifier speichert. Danach wird der Service wieder geschlossen und eine Meldung an die Rückgabemethode übergeben. Falls die entsprechende Variable gesetzt ist, wird die Verbindung auch geschlossen, sonst ist sie im Anschluss wieder offen.

Welche Fehler können auftreten? Auch beim Lesen können verschiedene Fehler auftreten, die an die Rückgabemethode übergeben werden. Eine Übersicht steht im Anhang unter Kapitel 3.2.

1.4 Übersicht der implementierten Variablentypen

Variablentyp	Klasse	Aufruf
~~EP~~	~~getEPidentifiers~~	~~setandsubmit(pobj, host, server, path, namemask)~~
INT	getInt setInt	setandsubmit(pobj, host, server, path) setandsubmit(pobj, host, server, path, sendint)
INTVEC	getIntVec setIntVec	setandsubmit(pobj, host, server, path) setandsubmit(pobj, host, server, path, sendintvec, sendintveclength)
STRING	getString setString	setandsubmit(pobj, host, server, path) setandsubmit(pobj, host, server, path, sendstring)
STRINGVEC	getStringVec setStringVec	setandsubmit(pobj, host, server, path) setandsubmit(pobj, host, server, path, sendstringvec, sendstringlength)
BOOL	getBool setBool	setandsubmit(pobj, host, server, path) setandsubmit(pobj, host, server, path, sendbool)
BOOLVEC	getBoolVec setBoolVec	setandsubmit(pobj, host, server, path) setandsubmit(pobj, host, server, path, sendboolvec, sendboolveclength)
DOUBLE	getDouble setDouble	setandsubmit(pobj, host, server, path) setandsubmit(pobj, host, server, path, senddouble)
DOUBLEVEC	getDoubleVec setDoubleVec	setandsubmit(pobj, host, server, path) setandsubmit(pobj, host, server, path, senddoublevec, senddoubleveclength)
SINGLE	getSingle setSingle	setandsubmit(pobj, host, server, path) setandsubmit(pobj, host, server, path, sendsingle)
SINGLEVEC	getSingleVec setSingleVec	setandsubmit(pobj, host, server, path) setandsubmit(pobj, host, server, path, sendsinglevec, sendsingleveclength)
UINT	getUInt setUInt	setandsubmit(pobj, host, server, path) setandsubmit(pobj, host, server, path, senduint)
UINTVEC	getUIntVec setUIntVec	setandsubmit(pobj, host, server, path) setandsubmit(pobj, host, server, path, senduintvec, senduintveclength)

Alle weiteren Variablentypen sind noch nicht implementiert, können aber einfach hinzugefügt werden.

1.5 createObject

1.5.1 Welche Einstellungen kann man vornehmen?

Zum einen können natürlich der Host, der Server und der Pfad des zu erzeugenden Objektes eingestellt werden sowie der Pfad der zu erzeugenden Klasse (librarypath) und die Position in der Hierarchie (position, element). Dies kann entweder über die Set-Funktionen der einzelnen Variablen geschehen oder durch Parameterübergabe an die entsprechende Funktion *setandsubmit*.
Alle Änderungen der Variablen dürfen nur über die entsprechenden Set-Methoden vorgenommen werden; bei der Parameterübergabe wird dies intern gemacht. Wenn die Variablen direkt gesetzt werden, werden eventuell wichtige Aufräumaktionen nicht durchgeführt.

1.5.2 Wie starte ich den Sendevorgang?

Auch hier muss zuerst eine Rückgabemethode definiert werden. Danach gibt es zwei Möglichkeiten, den Sendevorgang zu starten:

1. Man setzt die benötigten Variablen über ihre Set-Funktionen und ruft die Methode *submit* auf

2. Man übergibt der Methode *setandsubmit* die benötigten Variablen als Parameter

Das weitere Vorgehen ist wieder einheitlich und läuft im Hintergrund ab.

1.6 renameObject

1.6.1 Welche Einstellungen kann man vornehmen?

Zum einen können natürlich der Host, der Server und der Pfad des umzubenennenden Objektes eingestellt werden sowie der neue Pfad unter dem das Objekt angelegt werden soll. Dies kann entweder über die Set-Funktionen der einzelnen Variablen geschehen oder durch Parameterübergabe an die entsprechende Funktion *setandsubmit*.
Alle Änderungen der Variablen dürfen nur über die entsprechenden Set-Methoden vorgenommen werden; bei der Parameterübergabe wird dies intern gemacht. Wenn die Variablen direkt gesetzt werden, werden eventuell wichtige Aufräumaktionen nicht durchgeführt.

1.6.2 Wie starte ich den Sendevorgang?

Auch hier muss zuerst eine Rückgabemethode definiert werden. Danach gibt es zwei Möglichkeiten, den Sendevorgang zu starten:

1. Man setzt die benötigten Variablen über ihre Set-Funktionen und ruft die Methode *submit* auf

2. Man übergibt der Methode *setandsubmit* die benötigten Variablen als Parameter

Das weitere Vorgehen ist wieder einheitlich und läuft im Hintergrund ab.

1.7 deleteObject

1.7.1 Welche Einstellungen kann man vornehmen?

Es können der Host, der Server und der Pfad des zu löschenden Objektes eingestellt werden. Dies kann entweder über die Set-Funktionen der einzelnen Variablen geschehen oder durch Parameterübergabe an die entsprechende Funktion *setandsubmit*.
Alle Änderungen der Variablen dürfen nur über die entsprechenden Set-Methoden vorgenommen werden; bei der Parameterübergabe wird dies intern gemacht. Wenn die Variablen direkt gesetzt werden, werden eventuell wichtige Aufräumaktionen nicht durchgeführt.

1.7.2 Wie starte ich den Sendevorgang?

Auch hier muss zuerst eine Rückgabemethode definiert werden. Danach gibt es zwei Möglichkeiten, den Sendevorgang zu starten:

1. Man setzt die benötigten Variablen über ihre Set-Funktionen und ruft die Methode *submit* auf

2. Man übergibt der Methode *setandsubmit* die benötigten Variablen als Parameter

Das weitere Vorgehen ist wieder einheitlich und läuft im Hintergrund ab.

1.8 linkObject

1.8.1 Welche Einstellungen kann man vornehmen?

Zum einen können natürlich der Host, der Server und der Pfad des zu verlinkenden Objektes eingestellt werden sowie der Pfad des Objektes mit dem es verlinkt werden soll (linkedpath) und die Position in der Hierarchie (position, element). Dies kann entweder über die Set-Funktionen der einzelnen Variablen geschehen oder durch Parameterübergabe an die entsprechende Funktion *setandsubmit*.
Alle Änderungen der Variablen dürfen nur über die entsprechenden Set-Methoden vorgenommen werden; bei der Parameterübergabe wird dies intern gemacht. Wenn die Variablen direkt gesetzt werden, werden eventuell wichtige Aufräumaktionen nicht durchgeführt.

1.8.2 Wie starte ich den Sendevorgang?

Auch hier muss zuerst eine Rückgabemethode definiert werden. Danach gibt es zwei Möglichkeiten, den Sendevorgang zu starten:

1. Man setzt die benötigten Variablen über ihre Set-Funktionen und ruft die Methode *submit* auf

2. Man übergibt der Methode *setandsubmit* die benötigten Variablen als Parameter

Das weitere Vorgehen ist wieder einheitlich und läuft im Hintergrund ab.

1.9 unlinkObject

1.9.1 Welche Einstellungen kann man vornehmen?

Zum einen können natürlich der Host, der Server und der Pfad des zu unlinkenden Objektes eingestellt werden sowie der Pfad des Objektes mit dem es verlinkt ist (linkedpath). Dies kann entweder über die Set-Funktionen der einzelnen Variablen geschehen oder durch Parameterübergabe an die entsprechende Funktion *setandsubmit*.
Alle Änderungen der Variablen dürfen nur über die entsprechenden Set-Methoden vorgenommen werden; bei der Parameterübergabe wird dies intern gemacht. Wenn die Variablen direkt gesetzt werden, werden eventuell wichtige Aufräumaktionen nicht durchgeführt.

1.9.2 Wie starte ich den Sendevorgang?

Auch hier muss zuerst eine Rückgabemethode definiert werden. Danach gibt es zwei Möglichkeiten, den Sendevorgang zu starten:

1. Man setzt die benötigten Variablen über ihre Set-Funktionen und ruft die Methode *submit* auf

2. Man übergibt der Methode *setandsubmit* die benötigten Variablen als Parameter

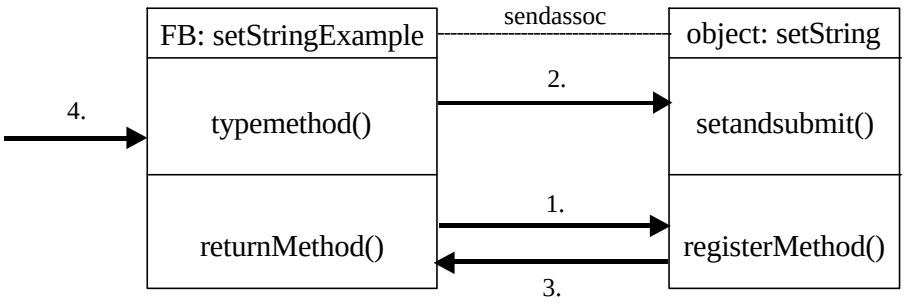
Das weitere Vorgehen ist wieder einheitlich und läuft im Hintergrund ab.

2 Die Beispielbibliothek *ksapiexample*

Die Beispielbibliothek *ksapiexample* ist ein einfacher Anwendungsfall der *ksapi*. In ihr werden alle dort implementierten Variablentypen verwendet. Sie besteht aus je einem Funktionsbaustein pro Typ, einem Client-FB und einem Link, die im Folgenden genauer erläutert werden. Exemplarisch wird hier die Klasse *setStringexample* detailliert erläutert, alle anderen Variablentypen funktionieren analog.

2.1 setStringexample

Der Zweck dieser Klasse ist es, einen String, der in einem Eingabebaustein eingegeben werden kann, an einen Receiver zu senden. Dazu wird die Bibliothek *ksapi* verwendet, es ist also darauf zu achten, dass die Bibliothek *ksapi* vor *ksapiexample* geladen wird.
Die folgende Grafik stellt die Abläufe, die für den Nutzer entscheidend sind, dar. Eine detailliertere Beschreibung befindet sich im an die Abbildung folgenden Text.



Der FB *setStringExample* ist ein Userinterface, in das alle benötigten Daten zum Versenden eines Strings eingegeben werden können. Im Anhang (3.1) ist eine tabellarische Auflistung der einzelnen Variablen und ihrer Bedeutung.

1. Als erstes muss die Rückgabemethode gesetzt werden. Dazu wird die Methode *registerMethod* mit dem Methodentabelleneintrag der selbstgeschriebenen Rückgabemethode aufgerufen.

2. Anschließend ruft man die Methode *setandsubmit* mit den benötigten Parametern auf
3. Nachdem der ~~KS~~-Vorgang beendet ist, wird die Rückgabemethode ausgeführt
4. Dann kann man sich um die Auswertung der Rückgabewerte kümmern.

2.2 Der FB *client*

Der FB *client* ist ein Beispiel für einen ganz einfachen Client-Baustein. Er kann Variablen annehmen, was jedesmal durch Setzen von *reset* auf TRUE quittiert werden muss. Dabei ist zu beachten, dass jedes Setzen quittiert werden muss, es geht also nicht, dass z.B. zuerst ein String gesetzt wird und direkt im Anschluss noch ein Integer. Gelesen werden kann jedoch immer.

2.3 Die Links **assoc*

Die Links dienen nur dazu, die Klassen der *ksapi* als Kinder der Klassen der *ksapiexample* leicht wiederzufinden. Da diese Links nur für diesen Zweck verwendet werden, braucht man nicht bei evtl. weiteren Kindern einen Test durchzuführen, ob das richtige Kind erreicht wurde.

3 Anhang

3.1 Übersicht der Variablennamen und -bedeutungen

Variable	Bedeutung	STANDARDWERT
*send**	Hier kann der Wert der Variablen eingetragen werden, die versendet werden soll.(nur bei set)	verschiedene Werte
*received**	Enthält den Wert der gelesenen Variable (nur bei get).	NULL
host	Der Host, auf dem der Receiver ausgeführt wird.	127.0.0.1
server	Der Server, auf dem der Receiver ausgeführt wird.	FB_TESTLIB
path	Der Pfad, wo der Receiver auf dem Server liegt.	/TECHUNITS/CLIENT.*
librarypath	Der Pfad der zu erzeugenden Klasse (nur bei createObject).	/LIBRARIES/*
newpath	Der Pfad, auf den das Objekt umbenannt werden soll (nur bei renameObject).	/TECHUNITS/*
linkedpath	Der Pfad des Objektes mit dem verlinkt werden soll (nur bei (un)linkObject).	/TECHUNITS/CLIENT.*
position	Position in der Hierarchie (nur bei createObject und linkObject).	0=DEFAULT, 1=ANFANG, 2=ENDE, 3=BEVOR, 4=NACH
element	Der Pfad, auf den sich *position* bezieht (nur bei createObject und linkObject).	/TECHUNITS/*
~~*maxtries*~~	~~Die Anzahl der Versuche, wie oft bei einem Fehler wiederholt werden soll, kann hier eingestellt werden.~~	~~3~~
~~*close*~~	~~Hier kann eingestellt werden, ob die Verbindung nach dem Senden geschlossen werden soll.~~	~~FALSE~~
~~*namemask*~~	~~Ein regulärer Ausdruck zum Filtern der gelesenen Identifier (nur getEPidentifier).~~	~~FALSE~~
send	Wird auf true gesetzt, wenn der Kommunikationsvorgang gestartet werden soll.	FALSE
reset	Stellt den Anfangszustand wieder her.	FALSE

Bis auf die letzten beiden Variablen treten diese sowohl in der *sendapi* als auch im *sendexample* auf.

3.2 Fehlercodes

#	Fehlerbeschreibung
1	Der Sendevorgang wurde erfolgreich abgeschlossen
2	Fehler bei der Eingabe, nicht alle Variablen wurden gesetzt
3	Es wurde keine Rückgabemethode gesetzt
4	Es ist ein Fehler beim Erzeugen eines OV-Objektes aufgetreten
5	Die Verbindung hat einen Fehler erzeugt
6	Der Service hat einen Fehler erzeugt
7	Die Variable, die gelesen werden soll, hat den falschen Typ.
8	Der Receiver hat die Nachricht nicht angenommen
9	Speicherfehler beim dynamischen Vektor in getEPidentifier