LibOVKS API Referenz

Dirk Meyer <dirk@plt.rwth-aachen.de>
Christian Poensgen <christian@plt.rwth-aachen.de>

LibOVKS API Referenz

1 ACPLT/KS-Klientenverbindungen

Die Definitionen zu den ACPLT/KS-Klientenverbindungen unter OV befinden sich in der Headerdatei `ov_ksclient.h`.

1.1 Datentypen

1.1.1 OV_KSCLIENT_CONNECTION

Datentyp
OV_KSCLIENT_CONNECTION

Syntax

```
#include "ov_ksclient.h"

typedef void*   OV_KSCLIENT_CONNECTION;
```

Beschreibung
Zeiger auf ein Objekt, das eine ACPLT/KS-Klientenverbindung unter OV repräsentiert.

Anmerkungen
Als Anwender des Objekts braucht man den Inhalt (die Daten) des Objekts nicht zu kennen. Der Zeiger hat nicht wirklich den Typ `void*`, tatsächlich sehen die Daten auch je nach Implementierung unterschiedlich aus.

Siehe auch
Section 1.3.1 [ov_ksclient_connection_create()], page 4.

1.1.2 OV_KSCLIENT_CONNECTION_STATE

Datentyp
OV_KSCLIENT_CONNECTION_STATE

Syntax

```
#include "ov_ksclient.h"

#define OV_CCS_CLOSED   0x00000001
#define OV_CCS_OPEN     0x00000002
#define OV_CCS_BUSY     0x00000004

typedef enum_t   OV_KSCLIENT_CONNECTION_STATE;
```

Beschreibung
Eine ACPLT/KS-Klientenverbindung unter OV kann drei Zustände annehmen: geschlossen (`OV_CCS_CLOSED`), offen (`OV_CCS_OPEN`) oder aktiv, d.h. in Bearbeitung (`OV_CCS_BUSY`).

Siehe auch
Section 1.3.6 [ov_ksclient_connection_getstate()], page 7.

1.2 Funktionstypen

1.2.1 OV_FNC_KSCLIENT_OPENCALLBACK

Funktionstyp
OV_FNC_KSCLIENT_OPENCALLBACK

Syntax

```
#include "ov_ksclient.h"

typedef void OV_DLLFNCEXPORT OV_FNC_KSCLIENT_OPENCALLBACK(
    OV_KSCLIENT_CONNECTION   *pconn,
    OV_RESULT                result,
    OV_POINTER               userdata
);
```

Parameter

`pconn` Zeiger auf das C-Objekt, das die Verbindung repräsentiert, die zu öffnen versucht wurde

`result` Fehlercode, der angibt, ob und ggf. welcher Fehler beim Verbindungsaufbau aufgetreten ist

`userdata` Die Anwenderdaten, die beim Aufruf der Funktion `ov_ksclient_connection_open()` mit übergeben wurden

Beschreibung
Diese Callback-Funktion wird aufgerufen, nachdem durch den erfolgreichen Aufruf der Funktion `ov_ksclient_connection_open()` im Hintergrund versucht wurde, eine Verbindung zu einem ACPLT/KS-Server aufzubauen, die durch das Verbindungsobjekt mit dem Zeiger `pconn` repräsentiert wird. Der Fehlercode `result` gibt an, ob der Verbindungsaufbau erfolgreich war (`KS_ERR_OK`) bzw. welcher Fehler aufgetreten ist. Im Falle eines erfolgreichen Verbindungsaufbaus ist die Verbindung nun in dem Zustand `OV_CCS_OPEN`, andernfalls im Zustand `OV_CCS_CLOSED`.

Diese Funktion muß vom Anwender implementiert werden und ermöglicht ihm, auf das Gelingen oder Mißlingen eines Verbindungsaufbaus zu reagieren.

Rückgabewert
keiner

Siehe auch
Section 1.3.3 [ov_ksclient_connection_open()], page 5.

1.2.2 OV_FNC_KSCLIENT_REQUESTCALLBACK

Funktionstyp
OV_FNC_KSCLIENT_REQUESTCALLBACK

Syntax

```
#include "ov_ksclient.h"

typedef void OV_DLLFNCEXPORT OV_FNC_KSCLIENT_REQUESTCALLBACK(
    OV_KSCLIENT_CONNECTION   *pconn,
    OV_RESULT                result,
    OV_KSCLIENT_SERVICE      *psvc,
    OV_POINTER               userdata
);
```

Parameter

`pconn` Zeiger auf das C-Objekt, das die Verbindung repräsentiert, auf der versucht wurde, einen Service-Request (Dienst) zu versenden

`result` Fehlercode, der angibt, ob und ggf. welcher Fehler beim Ausführen des Dienstes aufgetreten ist

`psvc` Zeiger auf das Dienstobjekt

`userdata` Die Anwenderdaten, die beim Aufruf der Funktion `ov_ksclient_connection_sendrequest()` mit übergeben wurden

Beschreibung
Diese Callback-Funktion wird aufgerufen, nachdem durch den erfolgreichen Aufruf der Funktion `ov_ksclient_connection_sendrequest()` im Hintergrund versucht wurde, einen Dienst auf der Verbindung auszuführen, die durch das Verbindungsobjekt mit dem Zeiger `pconn` repräsentiert wird. Der Fehlercode `result` gibt an, ob der Dienst erfolgreich ausgeführt werden konnte (`KS_ERR_OK`) bzw. welcher Fehler aufgetreten ist. Im Falle einer erfolgreichen Dienstausführung enthält das durch `psvc` referenzierte Dienstobjekt nun neben den Serviceparametern auch die zugehörige Antwort. Im Falle einer erfolgreichen Diensteausführung ist die Verbindung nun in dem Zustand `OV_CCS_OPEN`, andernfalls wird sie geschlossen und befindet sich im Zustand `OV_CCS_CLOSED`.

Diese Funktion muß vom Anwender implementiert werden und ermöglicht ihm, auf das Gelingen oder Mißlingen einer Dienstausführung zu reagieren und dabei insbesondere die Ergebnisse der Dienstausführung in Empfang zu nehmen.

Rückgabewert
keiner

Siehe auch
Section 1.3.5 [ov_ksclient_connection_sendrequest()], page 7.

1.3 Funktionen

1.3.1 ov_ksclient_connection_create()

Funktionsname
ov_ksclient_connection_create()

Aufgabe
Erzeugen eines neuen Klientenverbindungsobjekts

Syntax

```
#include "ov_ksclient.h"

OV_KSCLIENT_CONNECTION* OV_DLLFNCEXPORT ov_ksclient_connection_create(
    OV_STRING   hostname,
    OV_STRING   servername
);
```

Parameter

`hostname` Name oder IP-Adresse des Hostrechners, mit dem die Verbindung hergestellt werden soll, z.B. `"acplt.plt.rwth-aachen.de"`. Kann eine IP-Adresse enthalten.

`servername`
Name des Servers auf dem Hostrechner, mit dem die Verbindung hergestellt werden soll, z.B. `"ovserver"`

Beschreibung
Beim Aufruf der Funktion wird ein neues Verbindungsobjekt erzeugt, das eine Klientenverbindung zu einem ACPLT/KS-Server `servername` auf dem Hostrechner `hostname` repräsentiert.

Rückgabewert
Zurückgegeben wird ein Zeiger auf das die Verbindung repräsentierende Verbindungsobjekt oder `NULL`, falls das Erzeugen des Objekts fehlschlägt.

Anmerkungen
Die Verbindung wird nach dem Erzeugen des Verbindungsobjekts *nicht* automatisch geöffnet.

1.3.2 ov_ksclient_connection_delete()

Funktionsname
ov_ksclient_connection_delete()

Aufgabe
Löschen eines bestehenden Klientenverbindungsobjekts

Syntax

```
#include "ov_ksclient.h"

void OV_DLLFNCEXPORT ov_ksclient_connection_delete(
    OV_KSCLIENT_CONNECTION   *pconn
);
```

Parameter

`pconn` Zeiger auf das Verbindungsobjekt, das die zu löschende Verbindung repräsentiert

Beschreibung
Beim Aufruf der Funktion wird das Verbindungsobjekt mit dem Zeiger `pconn` gelöscht, das die zu entfernende Klientenverbindung zu einem ACPLT/KS-Server repräsentiert.

Rückgabewert
keiner

Anmerkungen
Vor dem Löschen der Verbindung wird diese automatisch mit `ov_ksclient_connection_close()` geschlossen.

1.3.3 ov_ksclient_connection_open()

Funktionsname
ov_ksclient_connection_open()

Aufgabe
Öffnen einer bestehenden Klientenverbindung und asynchrones Warten auf den Verbindungsaufbau

Syntax

```
#include "ov_ksclient.h"

OV_RESULT OV_DLLFNCEXPORT ov_ksclient_connection_open(
    OV_KSCLIENT_CONNECTION         *pconn,
    OV_FNC_KSCLIENT_OPENCALLBACK   *callbackfnc,
    void                           *userdata
);
```

Parameter

`pconn` Zeiger auf das Verbindungsobjekt, das die zu öffnende Verbindung repräsentiert

`callbackfnc`
Zeiger auf die Callback-Funktion, die aufgerufen werden soll, wenn die Verbindung hergestellt worden oder ein Fehler aufgetreten ist

`userdata` Die Anwenderdaten, die beim Aufruf der Callback-Funktion übergeben werden sollen

Beschreibung
Beim Aufruf der Funktion wird versucht, die Verbindung zu einem ACPLT/KS-Server, die durch das Verbindungsobjekt mit dem Zeiger `pconn` repräsentiert wird, zu öffnen.
Wenn der Funktionsaufruf nicht unmittelbar fehlschlägt (siehe Rückgabewert), so wird sowohl bei erfolgreichem Aufbau der Verbindung als auch im Fehlerfall (z.B. Timeout des Netzwerks) die durch `callbackfnc` spezifizierte Callback-Funktion *asynchron* aufgerufen.

Rückgabewert
Zurückgegeben wird ein Fehlercode, der anzeigt, ob ein Fehler aufgetreten ist. Im Falle von `KS_ERR_OK` wird der Verbindungsaufbau im Hintergrund gestartet und asynchron auf die Beendigung des Verbindungsaufbaus gewartet.

Anmerkungen
Eine Verbindung kann nur dann geöffnet werden, wenn sie sich im Zustand `OV_CCS_CLOSED` befindet.

1.3.4 ov_ksclient_connection_close()

Funktionsname
ov_ksclient_connection_close()

Aufgabe
Schließen einer bestehenden Klientenverbindung

Syntax

```
#include "ov_ksclient.h"

void OV_DLLFNCEXPORT ov_ksclient_connection_close(
    OV_KSCLIENT_CONNECTION   *pconn
);
```

Parameter

`pconn` Zeiger auf das Verbindungsobjekt, das die zu schließende Verbindung repräsentiert

Beschreibung
Beim Aufruf dieser Funktion wird die Verbindung zu einem ACPLT/KS-Server, die durch das Verbindungsobjekt mit dem Zeiger `pconn` repräsentiert wird, geschlossen.
Bearbeitet die Verbindung noch einen Dienst, so wird zunächst die aktuelle Dienstbearbeitung abgebrochen und die beim Aufruf der Funktion `ov_ksclient_connection_sendrequest()` definierte Callback-Funktion mit dem Parameter `result = KS_ERR_GENERIC` aufgerufen, so daß die Bearbeitung des aktiven Serviceobjekts noch beendet werden kann.

Rückgabewert
keiner

Anmerkungen
Wird gerade versucht, die Verbindung im Hintergrund zu öffnen, so wird diese Operation abgebrochen, *ohne* die Callback-Funktion, die beim Aufruf der Funktion `ov_ksclient_connection_open()` definiert wurde, aufzurufen.

1.3.5 ov_ksclient_connection_sendrequest()

Funktionsname
ov_ksclient_connection_sendrequest()

Aufgabe
Versenden eines ACPLT/KS-Dienste-Request auf einer bestehenden Klientenverbindung und asynchrones Warten auf die Antwort

Syntax

```
#include "ov_ksclient.h"

OV_RESULT OV_DLLFNCEXPORT ov_ksclient_connection_sendrequest(
    OV_KSCLIENT_CONNECTION            *pconn,
    OV_KSCLIENT_SERVICE               *psvc,
    OV_FNC_KSCLIENT_REQUESTCALLBACK   *callbackfnc,
    void                              *userdata
);
```

Parameter

`pconn` Zeiger auf das Verbindungsobjekt, das die Verbindung repräsentiert, auf der der Request versendet werden soll

`psvc` Zeiger auf das Serviceobjekt, das u.a. den Request enthält und in dem bei erfolgreicher Dienstausführung die Antwort abgelegt wird

`callbackfnc`
Zeiger auf die Callback-Funktion, die aufgerufen werden soll, wenn der Dienst erfolgreich ausgeführt worden und die Antwort eingetroffen oder ein Fehler aufgetreten ist

`userdata` Die Anwenderdaten, die beim Aufruf der Callback-Funktion übergeben werden sollen

Beschreibung
Beim Aufruf der Funktion wird versucht, den durch das Serviceobjekt mit dem Zeiger `psvc` repräsentierten Dienst auf der Verbindung zu einem ACPLT/KS-Server, die durch das Verbindungsobjekt mit dem Zeiger `pconn` repräsentiert wird, zu versenden.
Wenn der Funktionsaufruf nicht unmittelbar fehlschlägt (siehe Rückgabewert), so wird sowohl bei erfolgreichem Ausführen des Dienstes (d.h. beim Eintreffen der Antwort) als auch im Fehlerfall (z.B. Timeout des Netzwerks) die durch `callbackfnc` spezifizierte Callback-Funktion *asynchron* aufgerufen.

Rückgabewert
Zurückgegeben wird ein Fehlercode, der anzeigt, ob ein Fehler aufgetreten ist. Im Falle von `KS_ERR_OK` wird das Versenden des Diensterequests im Hintergrund gestartet und asynchron auf die Antwort gewartet.

Anmerkungen
Ein Dienst-Request kann nur dann versendet werden, wenn die Verbindung sich im Zustand `OV_CCS_OPEN` befindet. Der Antwortteil des übergebenen Serviceobjekts mit dem Zeiger `psvc` muß mit Nullen initialisiert sein, d.h., das Serviceobjekt muß neu angelegt oder der Antwortobjektbaum des Dienstes mit `ov_ksclient_service_freeresult()` gelöscht worden sein.

1.3.6 ov_ksclient_connection_getstate()

Funktionsname
ov_ksclient_connection_getstate()

Aufgabe
Abfragen des Zustands einer bestehenden Klientenverbindung
Syntax

```
#include "ov_ksclient.h"

OV_KSCLIENT_CONNECTION_STATE OV_DLLFNCEXPORT ov_ksclient_connection_getstate(
    OV_KSCLIENT_CONNECTION   *pconn
);
```

Parameter

`pconn` Zeiger auf das Verbindungsobjekt, das die Verbindung repräsentiert, deren Zustand abgefragt werden soll

Beschreibung
Beim Aufruf der Funktion wird der aktuelle Zustand der Verbindung zu einem ACPLT/KS-Server abgefragt, die durch das Verbindungsobjekt mit dem Zeiger `pconn` repräsentiert wird.
Rückgabewert
Zurückgegeben wird der Zustand der Verbindung.

1.3.7 ov_ksclient_connection_settimeouts()

Funktionsname
ov_ksclient_connection_settimeouts()
Aufgabe
Setzen der Timeouts für Verbindungsaufbau und Serviceausführung bei einer bestehenden Klientenverbindung
Syntax

```
#include "ov_ksclient.h"

void OV_DLLFNCEXPORT ov_ksclient_connection_settimeouts(
    OV_KSCLIENT_CONNECTION   *pconn,
    const OV_UINT            timeout_open,
    const OV_UINT            timeout_request
);
```

Parameter

`pconn` Zeiger auf das Verbindungsobjekt, das die Verbindung repräsentiert, deren Timeouts gesetzt werden sollen

`timeout_open`
Timeout in Sekunden beim Öffnen einer Verbindung durch die Funktion `ov_ksclient_connection_open()`

`timeout_request`
Timeout in Sekunden beim Versenden eines Diensterequests auf einer Verbindung durch die Funktion `ov_ksclient_connection_sendrequest()`

Beschreibung
Beim Aufruf der Funktion werden die Timeouts der Verbindung zu einem ACPLT/KS-Server gesetzt, die durch das Verbindungsobjekt mit dem Zeiger `pconn` repräsentiert wird.
Rückgabewert
keiner
Anmerkungen
Die gesetzten Timeouts treten erst beim nächsten Aufruf der Funktion `ov_ksclient_connection_open()` bzw. `ov_ksclient_connection_sendrequest()` in Kraft.

1.3.8 ov_ksclient_connection_gettimeouts()

Funktionsname
ov_ksclient_connection_gettimeouts()

Aufgabe
Abfragen der Timeouts für Verbindungsaufbau und Serviceausführung bei einer bestehenden Klientenverbindung

Syntax

```
#include "ov_ksclient.h"

void OV_DLLFNCEXPORT ov_ksclient_connection_gettimeouts(
    OV_KSCLIENT_CONNECTION   *pconn,
    OV_UINT                  *ptimeout_open,
    OV_UINT                  *ptimeout_request
);
```

Parameter

`pconn` Zeiger auf das Verbindungsobjekt, das die Verbindung repräsentiert, deren Timeouts abgefragt werden sollen

`ptimeout_open`
Zeiger auf die Variable, die das gesetzte Timeout in Sekunden beim Öffnen einer Verbindung durch die Funktion `ov_ksclient_connection_open()` als Ergebnis aufnehmen soll

`ptimeout_request`
Zeiger auf die Variable, die das gesetzte Timeout in Sekunden beim Versenden eines Diensterequests auf einer Verbindung durch die Funktion `ov_ksclient_connection_sendrequest()` als Ergebnis aufnehmen soll

Beschreibung
Beim Aufruf der Funktion werden die Timeouts der Verbindung zu einem ACPLT/KS-Server abgefragt, die durch das Verbindungsobjekt mit dem Zeiger `pconn` repräsentiert wird.

Rückgabewert
siehe Parameter `ptimeout_open` und `ptimeout_request`

2 ACPLT/KS-Dienste (Klientenfunktionalität)

Die Definitionen zu den ACPLT/KS-Diensten aus Sicht von ACPLT/KS-Klienten unter OV befinden sich in der Headerdatei `ov_ksclient.h`.

2.1 Datentypen

2.1.1 OV_KSCLIENT_SERVICE

Datentyp
OV_KSCLIENT_SERVICE

Syntax

```
#include "ov_ksclient.h"

struct OV_KSCLIENT_SERVICE {

    /* the service ID (e.g. KS_GETPP) */
    KS_SVC                          serviceid;

    /* the service request parameters */
    union {
        KS_GETPP_PAR                getpp;
        KS_GETEP_PAR                getep;
        KS_GETVAR_PAR               getvar;
        KS_SETVAR_PAR               setvar;
        KS_EXGDATA_PAR              exgdata;
        KS_CREATEOBJECT_PAR         createobject;
        KS_DELETEOBJECT_PAR         deleteobject;
        KS_RENAMEOBJECT_PAR         renameobject;
        KS_LINK_PAR                 link;
        KS_UNLINK_PAR               unlink;
        KS_GETCANONICALPATH_PAR     getcanonicalpath;
    } params;

    /* the result of the service request */
    union {
        KS_GETPP_RES                getpp;
        KS_GETEP_RES                getep;
        KS_GETVAR_RES               getvar;
        KS_SETVAR_RES               setvar;
        KS_EXGDATA_RES              exgdata;
        KS_CREATEOBJECT_RES         createobject;
        KS_DELETEOBJECT_RES         deleteobject;
        KS_RENAMEOBJECT_RES         renameobject;
        KS_LINK_RES                 link;
        KS_UNLINK_RES               unlink;
        KS_GETCANONICALPATH_RES     getcanonicalpath;
    } result;

    /* the A/V module used */
```

```
    KS_AVMODULE                     *pavmodule;

    /* the associated connection, if the service is pending */
    OV_KSCLIENT_CONNECTION          *pconn;
};

typedef struct OV_KSCLIENT_SERVICE   OV_KSCLIENT_SERVICE;
```

Beschreibung
Objekt, das einen ACPLT/KS-Dienst aus Klientensicht unter OV repräsentiert. Es enthält die Service-ID `serviceid` des Dienstes. Diese ist der Diskriminator für die Unions `params` und `result`, die jeweils das oberste Objekt des Serviceparameter- bzw. Antwortobjektbaums eines Dienstes enthalten.
Der Zeiger `pavmodule` referenziert ein ACPLT/KS-A/V-Modul unter OV und ist für künftige Ergänzungen gedacht (siehe Headerdatei `ov_ksclient_avmodule.h`).
Der Zeiger `pconn` zeigt auf das Verbindungsobjekt, das den Dienst bearbeitet bzw. ist `NULL`, falls der Dienst zur Zeit nicht bearbeitet wird.

Siehe auch
Section 2.2.1 [ov_ksclient_service_create()], page 11.

2.2 Funktionen

2.2.1 ov_ksclient_service_create()

Funktionsname
ov_ksclient_service_create()

Aufgabe
Erzeugen eines neuen Serviceobjekts

Syntax

```
#include "ov_ksclient.h"

OV_KSCLIENT_SERVICE* OV_DLLFNCEXPORT ov_ksclient_service_create(
    KS_SVC          serviceid,
    KS_AVMODULE     *pavmodule
);
```

Parameter

`serviceid`
: Service-ID des ACPLT/KS-Dienstes, der durch das Serviceobjekt repräsentiert wird

`pavmodule`
: Zeiger auf ein A/V-Modulobjekt, das den beim Ausführen des Dienstes zu verwendenden A/V-Mechanismus repräsentiert. Soll keine Authentifizierung durchgeführt werden, so ist dieser Parameter `NULL`.

Beschreibung
Beim Aufruf der Funktion wird ein neues Serviceobjekt erzeugt, das einen auf einer Klientenverbindung zu einem ACPLT/KS-Server auszuführenden Dienst repräsentiert und sowohl die Parameter als auch die Antwort des Dienstes aufnehmen kann.
Die in dem Serviceobjekt enthaltenen obersten Objekte der Serviceparameter- bzw. Antwortobjektbäume werden mit Nullen initialisiert.

Rückgabewert
Zurückgegeben wird ein Zeiger auf das den ACPLT/KS-Dienst repräsentierende Serviceobjekt oder `NULL`, falls das Erzeugen des Objekts fehlschlägt.

Anmerkungen
Während die Objekte des Serviceparameterobjektbaums und deren Speicher (mit Ausnahme des obersten Objekts) vom Anwender verwaltet werden müssen, werden die Objekte des Antwortobjektbaums automatisch aus dem XDR-Strom der Antwort erzeugt und können mit Hilfe der Funktion `ov_ksclient_service_freeresult()` wieder aus dem Speicher entfernt werden.

2.2.2 ov_ksclient_service_delete()

Funktionsname
ov_ksclient_service_delete()

Aufgabe
Löschen eines bestehenden Serviceobjekts

Syntax

```
#include "ov_ksclient.h"

void OV_DLLFNCEXPORT ov_ksclient_service_delete(
    OV_KSCLIENT_SERVICE   *psvc
);
```

Parameter

`psvc` Zeiger auf das Serviceobjekt, das gelöscht werden soll

Beschreibung
Beim Aufruf der Funktion wird das Serviceobjekt mit dem Zeiger `psvc` gelöscht.
Wird der zugehörige Dienst gerade von einer Verbindung bearbeitet, so wird diese Verbindung geschlossen.
Enthält das Serviceobjekt Referenzen zu Objekten des Antwortobjektbaums, so werden auch diese Objekte mit Hilfe der Funktion `ov_ksclient_service_freeresult()` gelöscht.

Rückgabewert
keiner

Anmerkungen
Da die Objekte des Serviceparameterobjektbaums und deren Speicher (mit Ausnahme des obersten Objekts) nicht von OV verwaltet werden, muß der Anwender selbst dafür Sorge tragen, daß diese Objekte (z.B. vor dem Aufruf von `ov_ksclient_service_freeresult()`) gelöscht werden.

2.2.3 ov_ksclient_service_freeresult()

Funktionsname
ov_ksclient_service_freeresult()

Aufgabe
Löschen des Antwortobjektbaumes eines bestehenden Serviceobjekts

Syntax

```
#include "ov_ksclient.h"

void OV_DLLFNCEXPORT ov_ksclient_service_freeresult(
    OV_KSCLIENT_SERVICE   *psvc
);
```

Parameter

`psvc` Zeiger auf das Serviceobjekt, dessen Antwortobjektbaum gelöscht werden soll

Beschreibung
Beim Aufruf der Funktion wird der Antwortobjektbaum des Serviceobjekts mit dem Zeiger `psvc` gelöscht.

Rückgabewert
keiner

3 ACPLT/KS-Serviceparameter und -antworten

Die ACPLT/KS-Serviceparameter- und -antwortobjekte werden unter ACPLT/OV automatisch mit Hilfe des RPC-Generators `rpcgen` aus der ACPLT/KS-Dienstedefinition `ks_xdr.x` erzeugt. Die erzeugte Headerdatei heißt unter ACPLT/OV `ov_ksclient_xdr.h`.

Für weitere Informationen sei auf die Technologiepapiere von ACPLT/KS verwiesen.

3.1 GetPP

Mit Hilfe des GetPP-Services können die Projected Properties eines oder mehrerer Kommunikationsobjekte ausgelesen werden. Er wurde in der ACPLT/KS-Version 2 durch den GetEP-Service ersetzt.

Serviceparameterobjekte:

KS_GETPP_PAR

```
typedef OV_GETPP_PAR   KS_GETPP_PAR;

typedef struct {
    OV_STRING     path;
    OV_OBJ_TYPE   type_mask;
    OV_STRING     name_mask;
}   OV_GETPP_PAR;
```

Die Parameter werden in einer Struktur vom Typ `KS_GETPP_PAR` übergeben.

`path` Der Pfad des/der abzufragenden Objekte(s), `/` für die Root Domain

`type_mask`
: Bitfeld zur Angabe der gewünschten Objekttypen, z.B. domain oder variable

`name_mask`
: Name des/der abzufragenden Objekte(s), kann reguläre Ausdrücke (*,?,Mengen) enthalten

Serviceantwortobjekte:

KS_GETPP_RES

```
typedef struct {
    KS_RESULT                result;
    KS_UINT                  items_len;      /* only, if result == KS_ERR_OK */
    KS_OBJ_PROJECTED_PROPS   *items_val;     /* only, if result == KS_ERR_OK */
}   KS_GETPP_RES;
```

Das Ergebnis der Serviceanfrage wird in einem `KS_GETPP_RES`-Objekt zurückgegeben.

`result` Fehlercode, `KS_ERR_OK` = kein Fehler

`items_len`
: Anzahl der in `*items_val` zurückgegebenen Datensätze

`*items_val`
: Liste von Projected Properties der auf die Anfrage passenden Objekte

KS_OBJ_PROJECTED_PROPS

```
typedef struct {
    KS_OBJ_TYPE                  objtype;
    union {
        KS_VAR_PROJECTED_PROPS   var_projected_props;
```

```
                                            /* only, if type == KS_OT_VARIABLE */
        KS_LINK_PROJECTED_PROPS   link_projected_props;
                                            /* only, if type == KS_OT_LINK */
    } KS_OBJ_PROJECTED_PROPS_u;
    KS_STRING                   identifier;
    KS_TIME                     creation_time;
    KS_STRING                   comment;
    KS_ACCESS                   access;
}   KS_OBJ_PROJECTED_PROPS;
```

Die Projected Properties eines Objektes enthalten folgende Daten:

`objtype` Typ des Objektes (Domain, Variable, ...)

`KS_OBJ_PROJECTED_PROPS_u`
: Union für die typabhängigen Daten

`identifier`
: Name des Objektes

`creation_time`
: Zeit der Erstellung des Kommunikationsobjektes oder des Objektes, das es repräsentiert

`comment` Optionale Beschreibung des Objektes in "printable ASCII"

`access` Zugriffsmodus des Objektes, z.B. lesender/schreibender Zugriff

Je nach Typ des Objektes werden zusätzliche Daten zurückgegeben:

KS_VAR_PROJECTED_PROPS

```
typedef OV_VAR_PROJECTED_PROPS   KS_VAR_PROJECTED_PROPS;

typedef struct {
    OV_STRING     tech_unit;
    OV_VAR_TYPE   vartype;
}   OV_VAR_PROJECTED_PROPS;
```

`tech_unit`
: Technische/physikalische Einheit, in der der Variablenwert dargestellt wird

`vartype` Typ der Variable (Integer, String usw.)

KS_LINK_PROJECTED_PROPS

```
typedef OV_LINK_PROJECTED_PROPS   KS_LINK_PROJECTED_PROPS;

typedef struct {
    OV_LINK_TYPE   linktype;
    OV_STRING      opposite_role_identifier;
    OV_STRING      association_identifier;
}   OV_LINK_PROJECTED_PROPS;
```

`linktype` Der Typ des Links, entweder "Reference" (die verbundenen Objekte erscheinen auf der nächsten Hierarchiestufe) oder "Detour" (Verweis auf einen anderen Zweigs des Baumes)

`opposite_role_identifier`
: Bei einem bidirektionalen Link gibt er das Part-Objekt des Zielobjektes an, das den Link in Gegenrichtung darstellt

`association_identifier`

3.2 GetEP

Mit Hilfe des GetEP-Services können die Engineered Properties eines oder mehrerer Kommunikationsobjekte ausgelesen werden. Er ersetzt den alten GetPP-Service.

Serviceparameterobjekte:

KS_GETEP_PAR

```
typedef OV_GETEP_PAR   KS_GETEP_PAR;

typedef struct {
    OV_STRING     path;
    OV_OBJ_TYPE   type_mask;
    OV_STRING     name_mask;
    OV_EP_FLAGS   scope_flags;
}   OV_GETEP_PAR;
```

Die Parameter werden in einer Struktur vom Typ `KS_GETEP_PAR` übergeben.

`path` Der Pfad des/der abzufragenden Objekte(s), `/` für die Root Domain

`type_mask`
: Bitfeld zur Angabe der gewünschten Objekttypen, z.B. domain oder variable

`name_mask`
: Name des/der abzufragenden Objekte(s), kann reguläre Ausdrücke (*,?,Mengen) enthalten

`scope_flags`
: Bitfeld, das angibt, ob Parts (`KS_EPF_PARTS`) oder Children (`KS_EPF_CHILDREN`) des/der angegebenen Objekte(s) eingeschlossen werden sollen. `KS_EPF_FLATTEN` gibt an, daß der Server die Parts einer Domain oder Structure rekursiv durchsucht.

Serviceantwortobjekte:

KS_GETEP_RES

```
typedef struct {
    KS_RESULT                 result;
    KS_UINT                   items_len;      /* only, if result == KS_ERR_OK */
    KS_OBJ_ENGINEERED_PROPS   *items_val;     /* only, if result == KS_ERR_OK */
}   KS_GETEP_RES;
```

Das Ergebnis der Serviceanfrage wird in einem `KS_GETEP_RES`-Objekt zurückgegeben.

`result` Fehlercode, `KS_ERR_OK` = kein Fehler

`items_len`
: Anzahl der in `*items_val` zurückgegebenen Datensätze

`*items_val`
: Liste von Engineered Properties der auf die Anfrage passenden Objekte

KS_OBJ_ENGINEERED_PROPS

```
typedef struct {
    KS_OBJ_TYPE                     objtype;
    union {
        KS_DOMAIN_ENGINEERED_PROPS   domain_engineered_props;
        KS_VAR_ENGINEERED_PROPS      var_engineered_props;
        KS_LINK_ENGINEERED_PROPS     link_engineered_props;
    }   KS_OBJ_ENGINEERED_PROPS_u;
    KS_STRING                       identifier;
```

```
    KS_TIME                         creation_time;
    KS_STRING                       comment;
    KS_ACCESS                       access;
    KS_SEMANTIC_FLAGS               semantic_flags;
}   KS_OBJ_ENGINEERED_PROPS;
```

Die Engineered Properties eines Objektes enthalten folgende Daten:

`objtype` Typ des Objektes (Domain, Variable, ...)

`KS_OBJ_ENGINEERED_PROPS_u`
: Union für die typabhängigen Daten

`identifier`
: Name des Objektes

`creation_time`
: Zeit der Erstellung des Kommunikationsobjektes oder des Objektes, das es repräsentiert

`comment` Optionale Beschreibung des Objektes in "printable ASCII"

`access` Zugriffsmodus des Objektes, z.B. lesender/schreibender Zugriff

`semantic_flags`

Je nach Typ des Objektes werden zusätzliche Daten zurückgegeben:

KS_DOMAIN_ENGINEERED_PROPS

```
typedef OV_DOMAIN_ENGINEERED_PROPS   KS_DOMAIN_ENGINEERED_PROPS;

typedef struct {
    OV_STRING   class_identifier;
}   OV_DOMAIN_ENGINEERED_PROPS;
```

`class_identifier`
: Der Bezeichner der Objektklasse, von der das Objekt eine Instanz darstellt, angegeben als absoluter Pfad oder relativ zu `/vendor/classes`

KS_VAR_ENGINEERED_PROPS

```
typedef OV_VAR_ENGINEERED_PROPS   KS_VAR_ENGINEERED_PROPS;

typedef struct {
    OV_STRING     tech_unit;
    OV_VAR_TYPE   vartype;
}   OV_VAR_ENGINEERED_PROPS;
```

`tech_unit`
: Technische/physikalische Einheit, in der der Variablenwert dargestellt wird

`vartype` Typ der Variable (Integer, String usw.)

KS_LINK_ENGINEERED_PROPS

```
typedef OV_LINK_ENGINEERED_PROPS   KS_LINK_ENGINEERED_PROPS;

typedef struct {
    OV_LINK_TYPE   linktype;
    OV_STRING      opposite_role_identifier;
    OV_STRING      association_identifier;
}   OV_LINK_ENGINEERED_PROPS;
```

`linktype` Der Typ des Links, entweder "Reference" (die verbundenen Objekte erscheinen auf der nächsten Hierarchiestufe) oder "Detour" (Verweis auf einen anderen Zweigs des Baumes)

`opposite_role_identifier`
Bei einem bidirektionalen Link gibt er das Part-Objekt des Zielobjektes an, das den Link in Gegenrichtung darstellt

`association_identifier`

3.3 GetVar

Mit Hilfe des GetVar-Services können die Current Properties von Variables, Domains und Structures ausgelesen werden.

Serviceparameterobjekte:

KS_GETVAR_PAR

```
typedef OV_GETVAR_PAR   KS_GETVAR_PAR;

typedef struct {
    OV_UINT     identifiers_len;
    OV_STRING   *identifiers_val;
}   OV_GETVAR_PAR;
```

Die Parameter werden in einer Struktur vom Typ `KS_GETVAR_PAR` übergeben.

`identifiers_len`
Anzahl der in `*identifiers_val` übergebenen Strings

`*identifiers_val`
Liste von Pfadnamen der Objekte, deren Current Properties zurückgeliefert werden sollen

Serviceantwortobjekte:

KS_GETVAR_RES

```
typedef OV_GETVAR_RES   KS_GETVAR_RES;

typedef struct {
    OV_RESULT        result;
    OV_UINT          items_len;      /* only, if result == OV_ERR_OK */
    OV_GETVAR_ITEM   *items_val;     /* only, if result == OV_ERR_OK */
}   OV_GETVAR_RES;
```

Das Ergebnis der Serviceanfrage wird in einem `KS_GETVAR_RES`-Objekt zurückgegeben.

`result` Fehlercode, `KS_ERR_OK` = kein Fehler

`items_len`
Anzahl der in `*items_val` zurückgegebenen Datensätze

`*items_val`
Liste von `KS_GETVAR_ITEM`-Objekten

KS_GETVAR_ITEM

```
typedef OV_GETVAR_ITEM   KS_GETVAR_ITEM;

typedef struct {
    OV_RESULT              result;
```

```
    OV_VAR_CURRENT_PROPS   var_current_props;   /* only, if result == OV_ERR_OK */
}   OV_GETVAR_ITEM;
```

Für jedes Objekt wird ein `KS_GETVAR_ITEM` zurückgeliefert.

`result` Fehlercode, `KS_ERR_OK` = kein Fehler

`var_current_props`
: Current Properties des Objektes

KS_VAR_CURRENT_PROPS

```
typedef OV_VAR_CURRENT_PROPS   KS_VAR_CURRENT_PROPS;
```

?

?

Die Current Properties eines Variable-, Domain- oder Structure-Objektes enthalten folgende Daten:

KS_VAR_VALUE

```
typedef OV_VAR_VALUE   KS_VAR_VALUE;
```

?

?

3.4 SetVar

Mit Hilfe des SetVar-Services können die Current Properties von Variablen neu gesetzt werden.

Serviceparameterobjekte:

KS_SETVAR_PAR

```
typedef OV_SETVAR_PAR   KS_SETVAR_PAR;

typedef struct {
    OV_UINT          items_len;
    OV_SETVAR_ITEM   *items_val;
}   OV_SETVAR_PAR;
```

Die Parameter werden in einer Struktur vom Typ `KS_SETVAR_PAR` übergeben.

`items_len`
: Anzahl der in `*items_val` übergebenen `KS_SETVAR_ITEM`-Objekte

`*items_val`
: Liste von `KS_SETVAR_ITEM`-Objekten, die die zu setzenden Current Properties für jedes Objekt beinhalten

Für jedes Objekt wird ein `KS_SETVAR_ITEM` übergeben.

KS_SETVAR_ITEM

```
typedef OV_SETVAR_ITEM   KS_SETVAR_ITEM;

typedef struct {
    OV_STRING               path_and_name;
    OV_VAR_CURRENT_PROPS    var_current_props;
}   OV_SETVAR_ITEM;
```

`path_and_name`
: Pfad und Name des Objektes

`var_current_props`
: Die neuen Current Properties

KS_VAR_CURRENT_PROPS
siehe Section 3.3 [GetVar], page 18.

KS_VAR_VALUE
siehe Section 3.3 [GetVar], page 18.

Serviceantwortobjekte:

KS_SETVAR_RES

```
typedef OV_SETVAR_RES   KS_SETVAR_RES;

typedef struct {
    OV_RESULT   result;
    OV_UINT     results_len;      /* only, if result == OV_ERR_OK */
    OV_RESULT   *results_val;     /* only, if result == OV_ERR_OK */
}   OV_SETVAR_RES;
```

Das Ergebnis der Serviceanfrage wird in einem `KS_SETVAR_RES`-Objekt zurückgegeben.

`result` Fehlercode, `KS_ERR_OK` = kein Fehler

`results_len`
: Anzahl der in `*results_val` zurückgegebenen Datensätze

`*results_val`
: Liste der Fehlercodes der einzelnen Objekte

3.5 DataExchange

Der DataExchange-Dienst ermöglicht einen synchronisierten Datenaustausch mit Variablenobjekten. Zunächst wird eine Menge von Variablen neu gesetzt und dann innerhalb derselben Anfrage eine andere Menge von Variablen ausgelesen.

Serviceparameterobjekte:

KS_EXGDATA_PAR

```
typedef OV_EXGDATA_PAR   KS_EXGDATA_PAR;

typedef struct {
    OV_UINT          set_vars_len;
    OV_SETVAR_ITEM   *set_vars_val;
    OV_UINT          get_vars_len;
    OV_STRING        *get_vars_val;
}   OV_EXGDATA_PAR;
```

Die Parameter werden in einer Struktur vom Typ `KS_EXGDATA_PAR` übergeben.

`set_vars_len`
: Anzahl der in `*set_vars_val` übergebenen `KS_SETVAR_ITEM`-Objekte

`*set_vars_val`
: Liste von `KS_SETVAR_ITEM`-Objekten, die die zu setzenden Current Properties für jedes Objekt beinhalten

`get_vars_len`
: Anzahl der in `*get_vars_val` übergebenen Strings

`*get_vars_val`
: Liste von Pfadnamen der Objekte, deren Current Properties zurückgeliefert werden sollen

KS_SETVAR_ITEM
siehe Section 3.4 [SetVar], page 19.

Serviceantwortobjekte:

KS_EXGDATA_RES

```
typedef OV_EXGDATA_RES   KS_EXGDATA_RES;

typedef struct {
    OV_RESULT        result;
    OV_UINT          results_len;      /* only, if result == OV_ERR_OK */
    OV_RESULT        *results_val;     /* only, if result == OV_ERR_OK */
    OV_UINT          items_len;        /* only, if result == OV_ERR_OK */
    OV_GETVAR_ITEM   *items_val;       /* only, if result == OV_ERR_OK */
}   OV_EXGDATA_RES;
```

Das Ergebnis der Serviceanfrage wird in einem `KS_GETPP_RES`-Objekt zurückgegeben.

`results` Fehlercode, `KS_ERR_OK` = kein Fehler

`results_len`
: Anzahl der in `*results_val` zurückgegebenen Datensätze

`*results_val`
: Liste der Fehlercodes der einzelnen geänderten Variablenobjekte

`items_len`
: Anzahl der in `*items_val` zurückgegebenen Datensätze

`*items_val`
: Liste von `KS_GETVAR_ITEM`-Objekten

KS_GETVAR_ITEM
siehe Section 3.3 [GetVar], page 18.

3.6 CreateObject

Der CreateObject-Dienst erlaubt die Erzeugung neuer Kommunikationsobjekte durch einen Klienten. Dabei wird i.a. kein einzelnes Objekt erzeugt, sondern eine Instanz einer vorhandenen Objektklasse.

Serviceparameterobjekte:

KS_CREATEOBJECT_PAR

```
typedef OV_CREATEOBJECT_PAR   KS_CREATEOBJECT_PAR;

typedef struct {
```

```
    OV_UINT             items_len;
    OV_CREATEOBJ_ITEM   *items_val;
}   OV_CREATEOBJECT_PAR;
```

Die Parameter werden in einer Struktur vom Typ `KS_CREATEOBJECT_PAR` übergeben.

`items_len`
: Anzahl der in `*items_val` übergebenen Datensätze

`*items_val`
: Liste von `KS_CREATEOBJ_ITEM`-Strukturen, die die zu erzeugenden Objekte beschreiben

KS_CREATEOBJ_ITEM

```
typedef OV_CREATEOBJ_ITEM   KS_CREATEOBJ_ITEM;

typedef struct {
    OV_STRING        factory_path;
    OV_STRING        new_path;
    OV_PLACEMENT     place;
    OV_UINT          parameters_len;
    OV_SETVAR_ITEM   *parameters_val;
    OV_UINT          links_len;
    OV_LINK_ITEM     *links_val;
}   OV_CREATEOBJ_ITEM;
```

Für jedes zu erzeugende Objekt wird ein `KS_CREATEOBJ_ITEM` übergeben.

`factory_path`
: Vollständiger Pfad eines instantiierbaren "Factory"-Kommunikationsobjektes (Flag `KS_AC_INSTANTIABLE` als Zugriffsmodus gesetzt)

`new_path` Vollständiger Pfad des zu erzeugenden Objektes

`place` Falls der ACPLT/KS-Server Mengen von Kindobjekten als geordnete Mengen verwaltet, kann hier ein Wunsch angegeben werden, wo das neue Objekt plaziert werden soll

`parameters_len`
: Anzahl der in `*parameters_val` übergebenen Datensätze

`*parameters_val`
: Liste von Part-Namen und deren Initialwerten, dient zum Setzen von Read-only-Parametern während der Erzeugung des neuen Objekts

`links_len`
: Anzahl der in `*links_val` übergebenen Datensätze

`*links_val`
: Liste von Links, die während der Erzeugung des Objekts initialisiert werden sollen

KS_PLACEMENT
siehe Section 3.9 [Link], page 25.

KS_SETVAR_ITEM
siehe Section 3.4 [SetVar], page 19.

KS_LINK_ITEM
siehe Section 3.9 [Link], page 25.

Serviceantwortobjekte:

KS_CREATEOBJECT_RES

```
typedef OV_CREATEOBJECT_RES   KS_CREATEOBJECT_RES;

typedef struct {
    OV_RESULT                 result;
    OV_UINT                   obj_results_len;   /* only, if result == OV_ERR_OK */
    OV_CREATEOBJECTITEM_RES   *obj_results_val;  /* only, if result == OV_ERR_OK */
}   OV_CREATEOBJECT_RES;
```

Das Ergebnis der Serviceanfrage wird in einem `KS_GETEP_RES`-Objekt zurückgegeben.

`result` Fehlercode, `KS_ERR_OK` = kein Fehler

`obj_results_len`
: Anzahl der in `*obj_results_val` zurückgegebenen Datensätze

`*obj_results_val`
: Liste von `OV_CREATEOBJECTITEM_RES`-Objekten

KS_CREATEOBJECTITEM_RES

```
typedef OV_CREATEOBJECTITEM_RES   KS_CREATEOBJECTITEM_RES;

typedef struct {
    OV_RESULT   result;
    OV_UINT     params_results_len;   /* only, if result == OV_ERR_BADINITPARAM */
    OV_RESULT   *params_results_val;  /* only, if result == OV_ERR_BADINITPARAM */
    OV_UINT     link_results_len;     /* only, if result == OV_ERR_BADINITPARAM */
    OV_RESULT   *link_results_val;    /* only, if result == OV_ERR_BADINITPARAM */
}   OV_CREATEOBJECTITEM_RES;
```

Für jedes zu erzeugende Objekt wird ein `KS_CREATEOBJECTITEM_RES`-Objekt zurückgeliefert.

`result` Fehlercode des einzelnen Objekts

`params_result_len`
: Anzahl der in `*params_results_val` zurückgegebenen Datensätze

`*params_result_val`
: Falls `result = OV_ERR_BADINITPARAM`: Liste der Fehlercodes der zu initialisierenden Parts

`link_result_len`
: Anzahl der in `*link_results_val` zurückgegebenen Datensätze

`*link_result_val`
: Falls `result = OV_ERR_BADINITPARAM`: Liste der Fehlercodes der zu initialisierenden Links

3.7 DeleteObject

Der DeleteObject-Dienst löscht einzelne oder Gruppen von Kommunikationsobjekten.

Serviceparameterobjekte:

KS_DELETEOBJECT_PAR

```
typedef OV_DELETEOBJECT_PAR   KS_DELETEOBJECT_PAR;

typedef struct {
    OV_UINT     paths_len;
    OV_STRING   *paths_val;
}   OV_DELETEOBJECT_PAR;
```

Die Parameter werden in einer Struktur vom Typ `KS_DELETEOBJECT_PAR` übergeben.

`paths_len`
: Anzahl der in `*paths_val` übergebenen Strings

`*paths_val`
: Liste der Pfadnamen der zu löschenden Objekte

Serviceantwortobjekte:

KS_DELETEOBJECT_RES

```
typedef OV_DELETEOBJECT_RES   KS_DELETEOBJECT_RES;

typedef struct {
    OV_RESULT   result;
    OV_UINT     results_len;      /* only, if result == OV_ERR_OK */
    OV_RESULT   *results_val;     /* only, if result == OV_ERR_OK */
}   OV_DELETEOBJECT_RES;
```

Das Ergebnis der Serviceanfrage wird in einem `KS_DELETEOBJECT_RES`-Objekt zurückgegeben.

`result` Fehlercode, `KS_ERR_OK` = kein Fehler

`results_len`
: Anzahl der in `*results_val` zurückgegebenen Datensätze

`*results_val`
: Liste der Fehlercodes der einzelnen Objekte

3.8 RenameObject

Der RenameObject-Dienst benennt Objekte um bzw. verschiebt sie innerhalb des Objektbaums.

Serviceparameterobjekte:

KS_RENAMEOBJECT_PAR

```
typedef OV_RENAMEOBJECT_PAR   KS_RENAMEOBJECT_PAR;

typedef struct {
    OV_UINT                items_len;
    OV_RENAMEOBJECT_ITEM   *items_val;
}   OV_RENAMEOBJECT_PAR;
```

Die Parameter werden in einer Struktur vom Typ `KS_RENAMEOBJECT_PAR` übergeben.

`items_len`
: Anzahl der in `*paths_val` übergebenen `OV_RENAMEOBJECT_ITEM`-Objekte

`*items_val`
: Liste von `OV_RENAMEOBJECT_ITEM`-Objekten

KS_RENAMEOBJECT_ITEM

```
typedef OV_RENAMEOBJECT_ITEM   KS_RENAMEOBJECT_ITEM;

typedef struct {
    OV_STRING      old_path;
    OV_STRING      new_path;
    OV_PLACEMENT   place;
}   OV_RENAMEOBJECT_ITEM;
```

Für jedes umzubenennende Objekt wird ein `KS_RENAMEOBJECT_ITEM`-Objekt übergeben.

`old_path` Alter Pfadname des Objekts

`new_path` Neuer Pfadname des Objekts

`place` Plazierungswunsch für das umbenannte Objekt

KS_PLACEMENT
siehe Section 3.9 [Link], page 25.

Serviceantwortobjekte:

KS_RENAMEOBJECT_RES

```
typedef OV_RENAMEOBJECT_RES   KS_RENAMEOBJECT_RES;

typedef struct {
    OV_RESULT   result;
    OV_UINT     results_len;      /* only, if result == OV_ERR_OK */
    OV_RESULT   *results_val;     /* only, if result == OV_ERR_OK */
}   OV_RENAMEOBJECT_RES;
```

Das Ergebnis der Serviceanfrage wird in einem `KS_RENAMEOBJECT_RES`-Objekt zurückgegeben.

`result` Fehlercode, `KS_ERR_OK` = kein Fehler

`results_len`
Anzahl der in `*results_val` zurückgegebenen Datensätze

`*results_val`
Liste der Fehlercodes der einzelnen Objekte

3.9 Link

Serviceparameterobjekte:

KS_LINK_PAR

```
typedef OV_LINK_PAR   KS_LINK_PAR;

typedef struct {
    OV_UINT        items_len;
    OV_LINK_ITEM   *items_val;
}   OV_LINK_PAR;
```

Die Parameter werden in einer Struktur vom Typ `KS_LINK_PAR` übergeben.

`items_len`

`*items_val`

KS_LINK_ITEM

```
typedef OV_LINK_ITEM   KS_LINK_ITEM;

typedef struct {
    OV_STRING      link_path;
    OV_STRING      element_path;
    OV_PLACEMENT   place;
    OV_PLACEMENT   opposite_place;
}   OV_LINK_ITEM;
```

`link_path`

`element_path`

`place`

opposite_place

KS_PLACEMENT

```
typedef OV_PLACEMENT   KS_PLACEMENT;

typedef struct {
    OV_PLACEMENT_HINT   hint;
    OV_STRING           place_path;     /* only, if hint == OV_PMH_BEFORE/AFTER */
}   OV_PLACEMENT;
```

hint

place_path

Serviceantwortobjekte:

KS_LINK_RES

```
typedef OV_LINK_RES   KS_LINK_RES;

typedef struct {
    OV_RESULT   result;
    OV_UINT     results_len;      /* only, if result == OV_ERR_OK */
    OV_RESULT   *results_val;     /* only, if result == OV_ERR_OK */
}   OV_LINK_RES;
```

Das Ergebnis der Serviceanfrage wird in einem KS_LINK_RES-Objekt zurückgegeben.

result

results_len

*results_val

3.10 Unlink

Serviceparameterobjekte:

KS_UNLINK_PAR

```
typedef OV_UNLINK_PAR   KS_UNLINK_PAR;

typedef struct {
    OV_UINT          items_len;
    OV_UNLINK_ITEM   *items_val;
}   OV_UNLINK_PAR;
```

Die Parameter werden in einer Struktur vom Typ KS_UNLINK_PAR übergeben.

items_len

*items_val

KS_UNLINK_ITEM

```
typedef OV_UNLINK_ITEM   KS_UNLINK_ITEM;

typedef struct {

    OV_STRING   link_path;
    OV_STRING   element_path;
}   OV_UNLINK_ITEM;
```

link_path

`element_path`

Serviceantwortobjekte:

KS_UNLINK_RES

```
typedef OV_UNLINK_RES   KS_UNLINK_RES;

typedef struct {
    OV_RESULT   result;
    OV_UINT     results_len;      /* only, if result == OV_ERR_OK */
    OV_RESULT   *results_val;     /* only, if result == OV_ERR_OK */
}   OV_UNLINK_RES;
```

Das Ergebnis der Serviceanfrage wird in einem `KS_UNLINK_RES`-Objekt zurückgegeben.

`result`

`results_len`

`*results_val`

3.11 GetCanonicalPath

Das Objektmodell von ACPLT/KS ermöglicht es, mehrere Sichten der Informationen in einem Server zur Verfügung zu stellen. Eine Sicht entspricht jeweils einem Pfad, durch den das Objekt adressierbar ist. Server können eine bestimmte, kanonische Sicht eines Objektes definieren. Der GetCanonicalPath-Service liefert den kanonischen Pfad des übergebenen Objektes zurück. Falls keine kanonische Sicht definiert ist, so wird der angegebene Pfad zurückgeliefert.

Serviceparameterobjekte:

KS_GETCANONICALPATH_PAR

```
typedef OV_GETCANONICALPATH_PAR   KS_GETCANONICALPATH_PAR;

typedef struct {
    OV_UINT     paths_len;
    OV_STRING   *paths_val;
}   OV_GETCANONICALPATH_PAR;
```

Die Parameter werden in einer Struktur vom Typ `KS_GETCANONICALPATH_PAR` übergeben.

`paths_len`
: Die Anzahl der in `*paths_val` übergebenen Datensätze

`*paths_val`
: Eine Liste mit den Pfadnamen von Objekten, die zu kanonischen Namen aufgelöst werden sollen

Serviceantwortobjekte:

KS_GETCANONICALPATH_RES

```
typedef OV_GETCANONICALPATH_RES   KS_GETCANONICALPATH_RES;

typedef struct {
    OV_RESULT                   result;
    OV_UINT                     results_len;   /* only, if result == OV_ERR_OK */
    OV_GETCANONICALPATHITEM_RES *results_val;  /* only, if result == OV_ERR_OK */
}   OV_GETCANONICALPATH_RES;
```

Das Ergebnis der Serviceanfrage wird in einem `KS_GETCANONICALPATH_RES`-Objekt zurückgegeben.

`result` Der Fehlercode, `KS_ERR_OK` = kein Fehler

`results_len`
Anzahl der in `*results_val` zurückgegebenen Datensätze

`*results_val`
Liste mit den kanonischen Namen der Objekte; Format s.u.

KS_GETCANONICALPATHITEM_RES

```
typedef OV_GETCANONICALPATHITEM_RES   KS_GETCANONICALPATHITEM_RES;

typedef struct {
    OV_RESULT   result;
    OV_STRING   canonical_path;     /* only, if result == OV_ERR_OK */
}   OV_GETCANONICALPATHITEM_RES;
```

Für jeden Pfadnamen wird ein `KS_GETCANONICALPATHITEM_RES`-Objekt zurückgeliefert.

`result` Fehlercode für das einzelne Objekt, `KS_ERR_OK` oder `KS_ERR_BADNAME`

`canonical_path`
Der kanonische Pfad des Objektes

3.12 GetHist

Der GetHist-Service

Serviceparameterobjekte:

KS_GETHIST_PAR

```
typedef OV_GETHIST_PAR   KS_GETHIST_PAR;

typedef struct {
    OV_UINT           paths_len;
    OV_STRING         *paths_val;
    OV_UINT           max_answers;
    OV_UINT           items_len;
    OV_GETHIST_ITEM   *items_val;
}   OV_GETHIST_PAR;
```

Die Parameter werden in einer Struktur vom Typ `KS_GETHIST_PAR` übergeben.

`paths_len`

`*paths_val`

`max_answers`

`items_len`

`*items_val`

KS_GETHIST_ITEM

```
typedef OV_GETHIST_ITEM   KS_GETHIST_ITEM;

typedef struct {
    OV_STRING         part;
    OV_HISTSELECTOR   selector;
}   OV_GETHIST_ITEM;
```

`part`

selector

KS_HISTSELECTOR

```
#define KS_HISTSELECTOR_u   OV_HISTSELECTOR_u

typedef OV_HISTSELECTOR   KS_HISTSELECTOR;

typedef struct {
    OV_HSEL_TYPE                hseltype;
    union {
        OV_TIMEHISTSELECTOR     ths;
        OV_STRINGHISTSELECTOR   shs;
    }   OV_HISTSELECTOR_u;
}   OV_HISTSELECTOR;
```

`hseltype` Der Typ des History Selectors: `KS_HSELT_NONE`, `KS_HSELT_TIME` oder `KS_HSELT_STRING`

`ths` Falls `hseltype = KS_HSELT_TIME`: ein `KS_TIMEHISTSELECTOR`-Objekt (s.u.)

`shs` Falls `hseltype = KS_HSELT_STRING`: ein `KS_STRINGHISTSELECTOR`-Objekt (s.u.)

KS_TIMEHISTSELECTOR

```
typedef OV_TIMEHISTSELECTOR   KS_TIMEHISTSELECTOR;

typedef struct {
    OV_INTERPOLATION_MODE   ip_mode;
    OV_ABSRELTIME           from;
    OV_ABSRELTIME           to;
    OV_TIME_SPAN            delta;
}   OV_TIMEHISTSELECTOR;
```

`ip_mode` Gewünschter Interpolationsmodus für die Daten: `KS_IPM_NONE`,`KS_IPM_LINEAR`, `KS_IPM_MIN`, `KS_IPM_MAX` oder `KS_IPM_HOLD`

`from` Startzeitpunkt (absolut oder relativ) der History

`to` Endzeitpunkt (absolut oder relativ) der History

`delta`

KS_STRINGHISTSELECTOR

```
typedef OV_STRINGHISTSELECTOR   KS_STRINGHISTSELECTOR;

typedef struct {
    OV_STRING   mask;
}   OV_STRINGHISTSELECTOR;
```

`mask`

KS_ABSRELTIME

```
#define KS_ABSRELTIME_u   OV_ABSRELTIME_u

typedef OV_ABSRELTIME   KS_ABSRELTIME;

typedef struct {
    OV_TIME_TYPE       timetype;
    union {
```

```
        OV_TIME        abstime;
        OV_TIME_SPAN   reltime;
    }   OV_ABSRELTIME_u;
}   OV_ABSRELTIME;
```

`timetype` Typ der Zeitangabe: `KS_TT_ABS` oder `KS_TT_REL`

`abstime` Falls `timetype = KS_TT_ABS`: absolute Zeitangabe

`reltime` Falls `timetype = KS_TT_REL`: relative Zeitangabe

Serviceantwortobjekte:

KS_GETHIST_RES

```
typedef OV_GETHIST_RES   KS_GETHIST_RES;

typedef struct {
    OV_RESULT                result;
    OV_UINT                  results_len;     /* only, if result == OV_ERR_OK */
    OV_GETHISTSINGLERESULT   *results_val;    /* only, if result == OV_ERR_OK */
}   OV_GETHIST_RES;
```

Das Ergebnis der Serviceanfrage wird in einem `KS_GETHIST_RES`-Objekt zurückgegeben.

`result` Der Fehlercode, `KS_ERR_OK` = kein Fehler

`results_len`
Anzahl der in `*results_val` zurückgegebenen Datensätze

`*results_val`
Liste von `KS_GETHISTSINGLERESULT`-Objekten

KS_GETHISTSINGLERESULT

```
typedef OV_GETHISTSINGLERESULT   KS_GETHISTSINGLERESULT;

typedef struct {
    OV_RESULT               result;
    OV_UINT                 items_len;     /* only, if result == OV_ERR_OK */
    OV_GETHISTRESULT_ITEM   *items_val;    /* only, if result == OV_ERR_OK */
}   OV_GETHISTSINGLERESULT;
```

`result` Der Fehlercode, `KS_ERR_OK` = kein Fehler

`items_len`

`*items_val`

KS_GETHISTRESULT_ITEM

```
typedef OV_GETHISTRESULT_ITEM   KS_GETHISTRESULT_ITEM;

typedef struct {
    OV_RESULT      result;
    OV_VAR_VALUE   value;     /* only, if result == OV_ERR_OK */
}   OV_GETHISTRESULT_ITEM;
```

`result`

`value`

4 Das API des ACPLT/KS-Servers unter OV

4.1 Funktionstypen

4.1.1 OV_FNC_SIGHANDLER

Funktionstyp
OV_FNC_SIGHANDLER

Syntax

```
#include "ov_ksserver.h"

typedef void OV_DLLFNCEXPORT OV_FNC_SIGHANDLER(
    int   signal
);
```

Parameter

`signal`

Beschreibung
Signal handler function prototype for server shutdown

Rückgabewert
keiner

Siehe auch
?

4.2 Funktionen

4.2.1 ov_ksserver_create()

Funktion
ov_ksserver_create()

Aufgabe
Erzeugen des ACPLT/KS-Servers für ACPLT/OV

Syntax

```
#include "ov_ksserver.h"

OV_RESULT OV_DLLFNCEXPORT ov_ksserver_create(
    OV_STRING           servername,
    int                 port,
    OV_FNC_SIGHANDLER   *sighandler
);
```

Parameter

`servername`
: Name des Servers. Unter diesem Namen registriert sich der Server beim ACPLT/KS-Manager.

`port`
: Portadresse des Servers oder `KS_ANYPORT`

`sighandler`

Beschreibung
Beim Aufruf der Funktion wird ein neues ACPLT/KS-Serverobjekt mit dem angegebenen Servernamen und der angegebenen Portadresse erzeugt, das Zugriff auf die OV-Datenbasis gestattet. Dies ist nur möglich, sofern noch kein Serverobjekt existiert (es kann nur einen Server in einem Prozeß geben). Bei erfolgreichem Erzeugen des Serverobjekts wird der Server gestartet und beim ACPLT/KS-Manager registriert.

Rückgabewert
Der Boolsche Rückgabewert gibt an, ob das Serverobjekt erzeugt und der Server erfolgreich gestartet werden konnte (`TRUE`) oder nicht (`FALSE`).

4.2.2 ov_ksserver_delete()

Funktion
ov_ksserver_delete()

Aufgabe
Löschen des ACPLT/KS-Servers für ACPLT/OV

Syntax

```
#include "ov_ksserver.h"

void OV_DLLFNCEXPORT ov_ksserver_delete(void);
```

Parameter
keine

Beschreibung
?
?

Rückgabewert
keiner

4.2.3 ov_ksserver_start()

Funktion
ov_ksserver_start()

Aufgabe
Starten des ACPLT/KS-Servers für ACPLT/OV

Syntax

```
#include "ov_ksserver.h"

void OV_DLLFNCEXPORT ov_ksserver_start(void);
```

Parameter
keine

Beschreibung
?
?

Rückgabewert
keiner

4.2.4 ov_ksserver_terminate()

Funktion
ov_ksserver_terminate()
Aufgabe
Beenden eines ACPLT/KS-Servers (nur RMOS)
Syntax

```
#include "ov_ksserver.h"

#if OV_SYSTEM_RMOS
OV_RESULT OV_DLLFNCEXPORT ov_ksserver_terminate(
    OV_UINT   taskid
);
#endif
```

Parameter

`taskid`

Beschreibung
Rückgabewert

4.2.5 ov_ksserver_stop()

Funktion
ov_ksserver_stop()
Aufgabe
Anhalten des ACPLT/KS-Servers für ACPLT/OV
Syntax

```
#include "ov_ksserver.h"

void OV_DLLFNCEXPORT ov_ksserver_stop(void);
```

Parameter
keine
Beschreibung
Beim Aufruf der Funktion wird die Bearbeitung des aktiven ACPLT/KS-Servers beendet, der Server beim ACPLT/KS-Manager deregistriert und das Serverobjekt gelöscht.
Rückgabewert
keiner

4.2.6 ov_ksserver_run()

Funktion
ov_ksserver_run()
Aufgabe
Ausführen der Hauptschleife des ACPLT/KS-Servers für ACPLT/OV
Syntax

```
#include "ov_ksserver.h"

void ov_ksserver_run(void);
```

Parameter
keine

Beschreibung
Beim Aufruf der Funktion wird die Hauptschleife des ACPLT/KS-Servers von OV ausgeführt bis das Programm von außen ein Signal (CTRL-C) erhält. In dieser Hauptschleife werden mit höchster Priorität ACPLT/KS-Dienste bearbeitet (Diensterequests an den Server sowie Dienste über Klientenverbindungen, siehe Funktion `ov_ksserver_servependingevents()`). Solange keine ACPLT/KS-Dienste bearbeitet werden müssen, werden die beim OV-Scheduler registrierten aktiven Objekte bearbeitet (siehe Funktion `ov_scheduler_schedulenextevent()` der OV-Basisbibliothek). Ist nichts zu tun, schläft der Prozeß.

Rückgabewert
keiner

Anmerkungen
Die Funktion `ov_ksserver_run()` entspricht dem folgenden Quelltext:

```
OV_TIME_SPAN   *ptimeout;
while(!ov_ksserver_isgoingdown()) {
    ptimeout = ov_scheduler_schedulenextevent();
    ov_ksserver_servependingevents(ptimeout);
}
```

4.2.7 ov_ksserver_servependingevents()

Funktion
ov_ksserver_servependingevents()

Aufgabe
Wartende Services des ACPLT/KS-Servers für ACPLT/OV bearbeiten

Syntax

```
#include "ov_ksserver.h"

OV_BOOL OV_DLLFNCEXPORT ov_ksserver_servependingevents(
    OV_TIME_SPAN   *ptimeout
);
```

Parameter

`ptimeout` Zeiger auf die Zeitstruktur, die diejenige Zeitdauer angibt, für die sich der Prozeß schlafenlegen soll, wenn keine ACPLT/KS-Dienste anstehen

Beschreibung
Beim Aufruf der Funktion werden alle anstehenden ACPLT/KS-Dienste (Server- und Klientendienste) bearbeitet. Stehen keine Dienste an, so legt sich der Prozeß schlafen, bis vom Netzwerk neue ACPLT/KS-Dienste anstehen oder die durch `ptimeout` gegebene Zeitdauer abgelaufen ist.

Rückgabewert
Der Rückgabewert gibt an, ob ACPLT/KS-Dienste bearbeitet worden sind (`TRUE`) oder nicht (`FALSE`).

Anmerkungen
Die Bearbeitung der Dienste wird nach Ablauf der durch `ptimeout` gegebenen Zeitdauer *nicht* unterbrochen, so daß das Programm ggf. entsprechend *später* aus dem Funktionsaufruf zurückkehrt. Andererseits kann das Programm schon *früher* aus dem Funktionsaufruf zurückkehren, falls ACPLT/KS-Dienste bearbeitet worden sind.

4.2.8 ov_ksserver_haspendingevents()

Funktion
ov_ksserver_haspendingevents()

Aufgabe
Abfrage, ob beim ACPLT/KS-Server für ACPLT/OV wartende Services vorhanden sind
Syntax

```
#include "ov_ksserver.h"

OV_BOOL OV_DLLFNCEXPORT ov_ksserver_haspendingevents(void);
```

Parameter
keine
Beschreibung
Beim Aufruf der Funktion wird abgefragt, ob beim ACPLT/KS-Server von OV wartende Services vorhanden sind.
Rückgabewert
Der Rückgabewert gibt an, ob beim ACPLT/KS-Server von OV wartende Services vorhanden sind (`TRUE`) oder nicht (`FALSE`).

4.2.9 ov_ksserver_downserver()

Funktion
ov_ksserver_downserver()
Aufgabe
Herunterfahren des ACPLT/KS-Servers für ACPLT/OV
Syntax

```
#include "ov_ksserver.h"

void OV_DLLFNCEXPORT ov_ksserver_downserver(void);
```

Parameter
keine
Beschreibung
?
?
Rückgabewert
keiner

4.2.10 ov_ksserver_isgoingdown()

Funktion
ov_ksserver_isgoingdown()
Aufgabe
Abfrage, ob der ACPLT/KS-Server für ACPLT/OV heruntergefahren wird
Syntax

```
#include "ov_ksserver.h"

OV_BOOL OV_DLLFNCEXPORT ov_ksserver_isgoingdown(void);
```

Parameter
keine
Beschreibung
Beim Aufruf der Funktion wird abgefragt, ob der ACPLT/KS-Server für ACPLT/OV heruntergefahren wird, weil ein externes Signal (CTRL-C) empfangen wurde.
Rückgabewert
Der Rückgabewert gibt an, ob der ACPLT/KS-Server für ACPLT/OV heruntergefahren wird (`TRUE`) oder nicht (`FALSE`).

Table of Contents