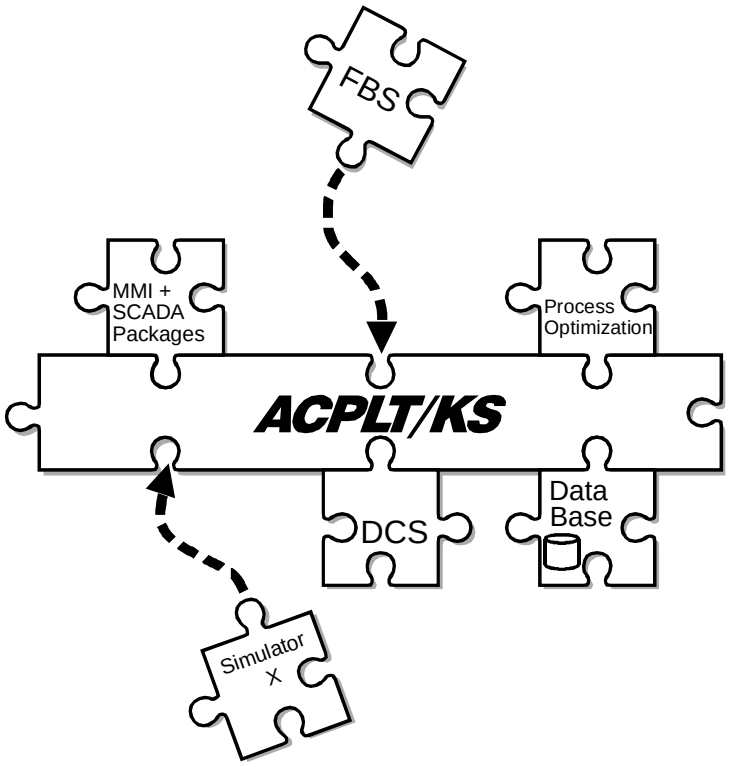
ACPLT/KS

Technologiepapier Nr. 4:
Das Objektmodell



Inhalt

Technologiepapier Nr. 4 – Ausgabe 09.97.2
ersetzt: Ausgabe 04.97.1

Ansprechpartner: H. Albrecht
Lehrstuhl für Prozeßleittechnik der RWTH Aachen
D-52056 Aachen
Tel.: +49 241 80-7703
Fax.: +49 241 8888-238
eMail: ks@plt.rwth-aachen.de
WWW: http://www.plt.rwth-aachen.de/ks

1 Einleitung

Das Kommunikationssystem ACPLT/KS (im folgenden auch „KS“ genannt) verbindet im Bereich von Anwendungen der Prozeß- und Betriebsführung rechnergestützte Werkzeuge untereinander und mit Prozeßleitsystemen. Dieses Technologiepapier beschreibt das ACPLT/KS zugrundeliegende Objektmodell und den Zugriff darauf mittels spezieller KS-Services.

2 Das KS-Objektmodell

Die Informationen von Servern werden bei ACPLT/KS in einer baumartigen Struktur von sogenannten KS-Kommunikationsobjekten (oder kurz: Objekte) abgelegt, die beliebig tief sein darf. Damit wird gegenüber der bei MMS streng dreischichtigen Struktur eine deutlich flexiblere Abbildung von Anwendungssystemen auf Kommunikationsobjekte erzielt. Zur Zeit kennt ACPLT/KS drei Typen von Kommunikationobjekten:

- „Domains“, die als Container für eine beliebige Zahl von „Kindobjekten“ fungieren. Ein Kindobjekt darf dabei jedes beliebige KS-Objekt sein – auch ein Domain-Objekt, so daß eine rekursive Schachtelung von Domains möglich ist (siehe auch Abbildung 2.1).
- „Variablen“, die beispielsweise aktuelle Zustandswerte eines Prozesses oder Prozeßleitsystems widerspiegeln.
- „Histories“, über die auf Zustandarchive (oder ähnliches) zugegriffen werden kann. Histories sind Gegenstand eines eigenen Technologiepapiers.

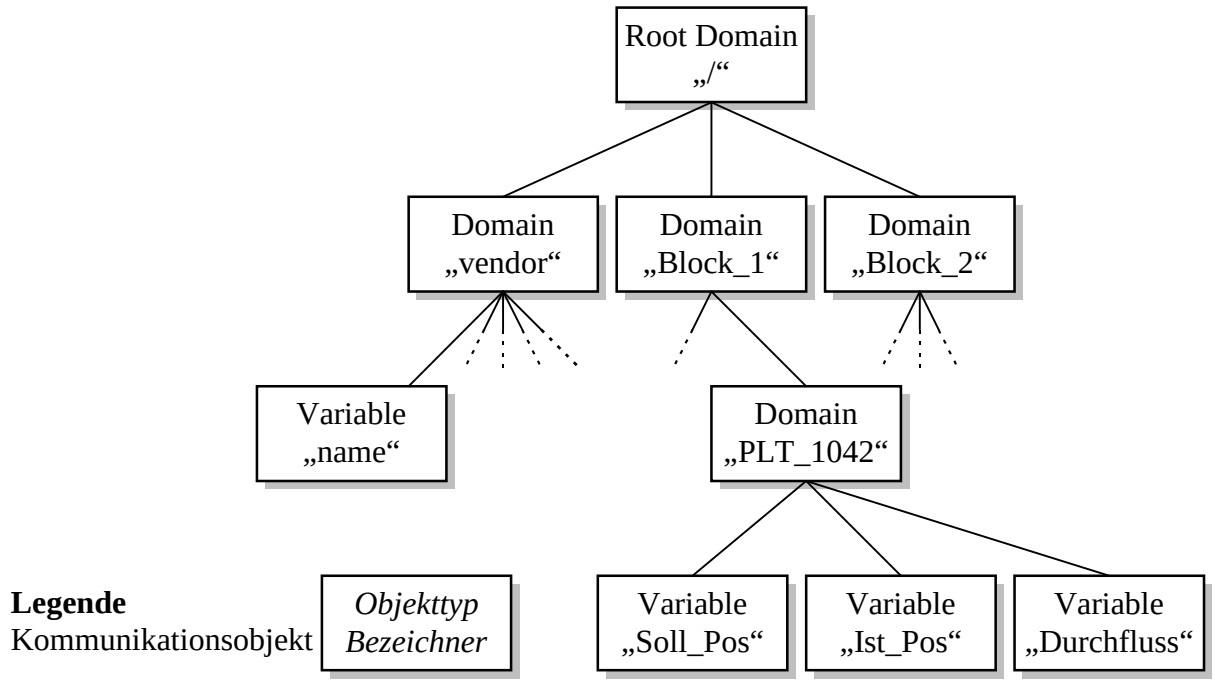


Abbildung 2.1: *Das KS-Objektmodell erlaubt eine hierarchische Strukturierung der Kommunikationsobjekte.*

Jedes KS-Objekt besitzt einen Bezeichner (Namen), der bis zu 255 Zeichen lang sein darf. Die Bezeichner dürfen nur aus den Groß- und Kleinbuchstaben „A“ bis „Z“ und „a“ bis „z“, dem Unterstrich „_“ sowie den Ziffern „0“ bis „9“ gebildet werden. Andere Zeichen sind nicht zulässig. Zwischen Groß- und Kleinschreibung wird im Normalfall unterschieden, es sei denn, daß das einem KS-Server unterlagerte Prozeßleitsystem (oder Simulator oder ähnliches) keine Unterscheidung zwischen Groß- und Kleinschreibung kennt. In diesem Fall muß der KS-Server in den Anfragen von Klienten die Groß- und Kleinschreibung ignorieren und in seinen

Antworten die Bezeichner in derjenigen Form zurückliefern, in der sie vom darunterliegenden (Prozeßleit-) system benutzt werden.

Ein KS-Objekt wird immer über seinen vollständigen Namen adressiert, der sich aus den mit einem Schrägstrich „/“ verbundenen Bezeichnern aller übergeordneten Domains und dem eigenen Bezeichner zusammensetzt. Der vollständige Name der Variable „Durchfluss“ der PLT-Stelle „PLT_1042“ aus Abbildung 2.1 lautet daher „/Block_1/PLT_1042/Durchfluss“. Der erste Schrägstrich im Namen darf hierbei nicht weggelassen werden, da er die oberste Domain (auch „root domain“ genannt) bezeichnet. Innerhalb aller Kinder eines Domain-Objektes darf kein Bezeichner mehrfach vorkommen.

2.1 Objekteigenschaften

Entsprechend seines Typs besitzt ein KS-Kommunikationsobjekt verschiedene sogenannte „Eigenschaften“ („properties“). Eine Eigenschaft kann beispielsweise der Erzeugungszeitpunkt des Objekts oder auch der aktuelle Variablenwert sein. ACPLT/KS unterscheidet zusätzlich zwischen „projektierten“ und „aktuellen“ Eigenschaften (siehe Abbildung 2.2). Die projektierten Eigenschaften werden bei der Erzeugung eines Objektes festgelegt und danach im Allgemeinen nicht mehr verändert.

Die aktuellen Eigenschaften eines Kommunikationsobjektes stammen zumeist direkt aus dem dem KS-Server unterlagerten Anwendersystem (Prozeßleitsystem oder Simulator). Sie verändern sich im Gegensatz zu den projektierten Eigenschaften im Laufe der Zeit mehr oder weniger häufig. Beispiele für aktuelle Eigenschaften sind der aktuelle Variablenwert (Meßwert) oder die Zuverlässigkeit (Vertrauenswürdigkeit) eines Meßwertes.

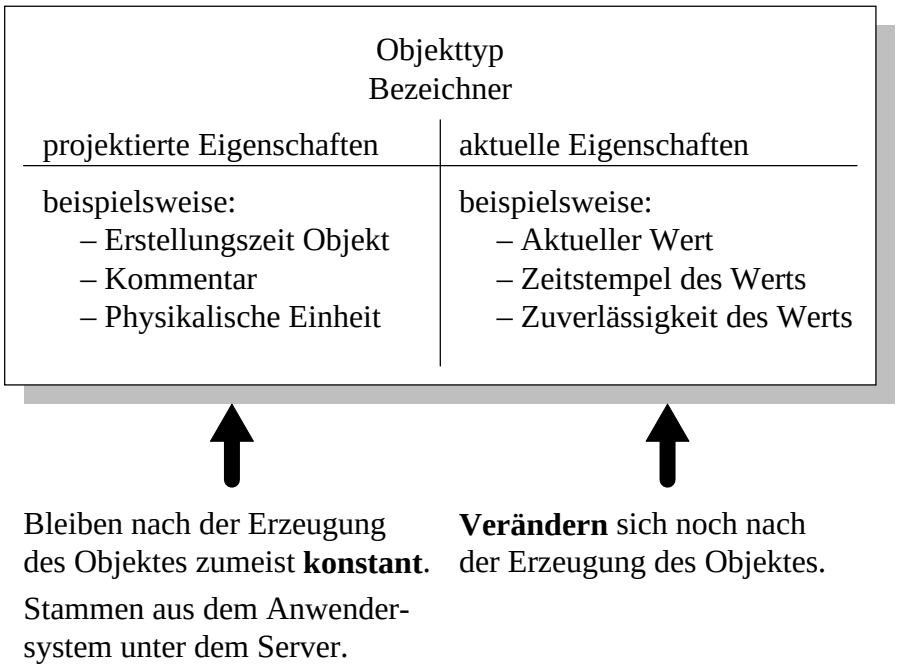


Abbildung 2.2: *Jedes KS-Kommunikationsobjekt besitzt projektierte und aktuelle Eigenschaften.*

Die jeweiligen projektierten und aktuellen Eigenschaften von KS-Kommunikationsobjekten sind in Abbildung 2.3 aufgeführt. History-Objekte werden in einem eigenen Technologiepapier behandelt, daher wird hier nicht weiter auf sie eingegangen. Alle Kommunikationsobjekte in der Abbildung erben die projektierten und aktuellen Eigenschaften der jeweiligen Elternklasse, so daß diese Eigenschaften nicht erneut in der abgeleiteten Klasse aufgelistet werden.

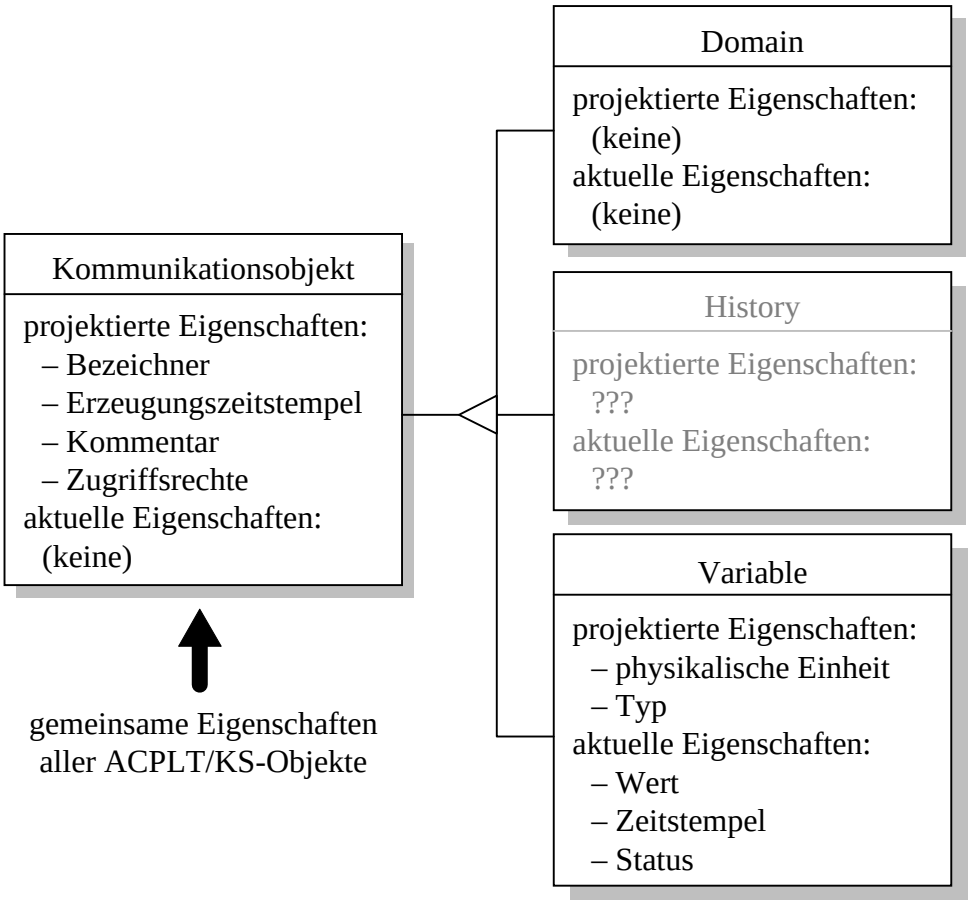


Abbildung 2.3: *Projektierte und aktuelle Eigenschaften der verschiedenen KS-Objektklassen.*

2.2 Kodierung von Objekteigenschaften

Für den Nachrichtenaustausch zwischen KS-Servern und KS-Klienten müssen die Objekteigenschaften kodiert (verpackt) werden. Die Beschreibung der Eigenschaften erfolgt über die beiden Datenstrukturen `KS_OBJ_PROJECTED_PROPS` und `KS_OBJ_CURRENT_PROPS` getrennt nach projektierten und aktuellen Eigenschaften.

Die Struktur `KS_OBJ_PROJECTED_PROPS` beschreibt die projektierten Eigenschaften eines KS-Kommunikationsobjektes.

```
const KS_NAME_MAXLEN    =  255;
const KS_COMMENT_MAXLEN = 4095;

struct KS_OBJ_PROJECTED_PROPS {
    KS_OBJ_VARIANT_PROJECTED_PROPS  variant;
    string                          identifier<KS_NAME_MAXLEN>;
    KS_TIME                         creation_time;
    string                          comment<KS_COMMENT_MAXLEN>;
    KS_ACCESS                       access_mode;
};
```

Die Variable `identifier` in der oben aufgeführten Struktur enthält den Bezeichner des betrachteten KS-Kommunikationsobjekts. Die Länge eines Bezeichners ist auf 255 Zeichen (Konstante `KS_NAME_MAXLEN`) begrenzt. Zulässig in Bezeichnern sind aus dem ASCII-Zeichensatz die Groß- und Kleinbuchstaben „A“ bis „Z“, „a“ bis „z“, der Unterstrich „_“ sowie die Ziffern „0“ bis „9“. Nationale Sonderzeichen sind nicht erlaubt. Der Zeitstempel in `creation_time` gibt den Zeitpunkt der Objekterstellung (lokale Server-Zeit) an. Daneben kann jedes KS-Objekt mit einem Kommentar versehen werden (abzulegen in `comment`). Die Länge

von Kommentaren ist auf 4 kByte begrenzt, um die Ressourcen von Servern und Klienten nicht über Gebühr zu strapazieren. In `access_mode` schließlich sind die (aktuellen) Zugriffsrechte auf das Kommunikationsobjekt abgelegt. Die Zugriffsrechte können einen der folgenden Werte annehmen:

```
enum KS_ACCESS {
    KS_AC_NONE      = 0, /* Kein Zugriff                        */
    KS_AC_READ      = 1, /* Nur lesender Zugriff                */
    KS_AC_WRITE     = 2, /* Nur schreibender Zugriff            */
    KS_AC_READWRITE = 3  /* Lesender und schreibender Zugriff */
};
```

Die Zugriffsrechte können (je nach KS-Server) in Abhängigkeit von der Authentifizierung des Klienten wechseln. So kann ein und dasselbe Objekt je nach Authentifizierung entweder nur lesbar oder auch beschreibbar sein.

Die Struktur `KS_OBJ_CURRENT_PROPS` beschreibt die aktuellen Eigenschaften eines KS-Kommunikationsobjektes. In der aktuellen Spezifikation ACPLT/KS 1.0 sind keine gemeinsamen aktuellen Objekteigenschaften definiert, es existieren aber objekttyp-abhängige aktuelle Eigenschaften.

```
struct KS_OBJ_CURRENT_PROPS {
    KS_OBJ_VARIANT_CURRENT_PROPS  variant;
};
```

Die je nach Objektklasse (Domain, Variable, History) unterschiedlichen projektierten und aktuellen Eigenschaften werden zusammenfassend wie folgt kodiert:

```
union KS_OBJ_VARIANT_PROJECTED_PROPS switch (KS_OBJTYPE type) {
    case KS_OT_DOMAIN:
        void;                   /* Keine neuen Eigenschaften */
    case KS_OT_VARIABLE:
        KS_VAR_PROJECTED_PROPS   var_projected_props;
    case KS_OT_HISTORY:
        void;                   /* Wird spaeter definiert    */
    default:
        void;
};

union KS_OBJ_VARIANT_CURRENT_PROPS switch (KS_OBJTYPE type) {
    case KS_OT_DOMAIN:
        void;                   /* Keine neuen Eigenschaften */
    case KS_OT_VARIABLE:
        KS_VAR_CURRENT_PROPS  var_current_props;
    case KS_OT_HISTORY:
        void;                   /* Wird spaeter definiert    */
    default:
        void;
};
```

Die in ACPLT/KS 1.0 verfügbaren Typen von Kommunikationsobjekten sind in der Aufzählung `KS_OBJTYPE` aufgeführt. Je nach Einsatzzweck können die dort aufgelisteten Bezeichner auch zu einer Bitmaske kombiniert werden (siehe auch den `KS_GETPP`-Service).

```
enum KS_OBJTYPE {
    KS_OT_DOMAIN   = 0x0001,
    KS_OT_VARIABLE = 0x0002,
    KS_OT_HISTORY  = 0x0004,
    KS_OF_ANY      = 0x0003
};
```

2.2.1 Eigenschaften von Variablenobjekten

Variablenobjekte besitzen neben den für alle Kommunikationsobjekte gemeinsamen, projektierten Eigenschaften noch weitere, neue projektierte und aktuelle Eigenschaften. Diese neuen Eigenschaften werden in den folgenden Datenstrukturen kodiert:

```
const KS_TECHNUNIT_MAXLEN = 63;

struct KS_VAR_PROJECTED_PROPS {
    string       tech_unit<KS_TECHUNIT_MAXLEN>;
    KS_VAR_TYPE  type;
};

struct KS_VAR_CURRENT_PROPS {
    KS_VAR_VALUE  value;
    KS_TIME       time;
    KS_STATE      state;
};
```

Die statische Eigenschaft `tech_unit` in der Datenstruktur `KS_VAR_PROJECTED_PROPS` ist eine Zeichenkette, die die physikalische Einheit der Variable enthält. Zulässig sind hierbei alle Zeichen des (amerikanischen) ASCII-Zeichensatzes mit Zeichencodes von 32 bis 126. Nationale Sonderzeichen (wie beispielsweise griechische Buchstaben) sind ebenso wie Zeilenvorschübe (CR/LF) nicht zulässig. Der Variablentyp in `type` wird durch eine (unter Umständen rekursive) Datenstruktur vom Typ `KS_VAR_TYPE` dargestellt, die weiter unten näher beschrieben wird.

Die Datenstruktur `KS_VAR_CURRENT_PROPS` enthält die aktuellen Eigenschaften, die bei Variablenobjekten gegenüber den gemeinsamen Eigenschaften neu hinzukommen. Dabei beinhaltet `value` den aktuellen Wert der Variablen (für eine genaue Erläuterung der dazu verwendeten Datenstruktur siehe unten). Die Variable `time` zeigt den (Erfassungs-) Zeitpunkt des Wertes an, der nicht notwendigerweise identisch mit demjenigen Zeitpunkt ist, an dem dieser Wert in das Variablenobjekt geschrieben wurde. Die Zeit wird von ACPLT/KS mit einer maximalen Auflösung von einer Mikrosekunde dargestellt. Bei Zeitpunkten gibt die Struktur `KS_TIME` die Anzahl von Sekunden und Mikrosekunden seit dem 1. Januar 1970 00:00 Uhr (UTC) an.

```
struct KS_TIME {
    u_long secs;
    u_long usecs;
};
```

Weiterhin kann jeder Variablen über `state` ein Status zugeordnet werden, der die Qualität des Variablenwertes beschreibt (vertrauenswürdig, fehlerhaft, und so weiter...).

```
enum KS_STATE {
    KS_ST_NOTSUPPORTED = 0, /* Kein Status verfuegbar            */
    KS_ST_UNKNOWN      = 1, /* Status zur Zeit unbekannt         */
    KS_ST_BAD          = 2, /* Information ist fehlerhaft        */
    KS_ST_QUESTIONABLE = 3, /* Information ist fragwuerdig       */
    KS_ST_GOOD         = 4  /* Information ist vertrauenswuerdig */
};
```

ACPLT/KS kennt die nachfolgend aufgelisteten (generischen) Datentypen beziehungsweise Felder von Datentypen:

```
enum KS_VAR_TYPE {
    KS_VT_VOID        = 0x00, /* kein Wert                    */
    KS_VT_BOOL        = 0x02, /* Boolean                      */
    KS_VT_INT         = 0x10, /* Integer 32 Bit breit         */
    KS_VT_UINT        = 0x11, /* dto.                         */
    KS_VT_SINGLE      = 0x20, /* 32 Bit IEEE-Darstellung      */
    KS_VT_DOUBLE      = 0x21, /* 64 Bit IEEE-Darstellung      */
    KS_VT_STRING      = 0x30, /* beliebig lange Zeichenkette  */
    KS_VT_TIME        = 0x31, /* Zeitstempel oder Zeitspanne  */
    KS_VT_BYTE_VEC    = 0x81, /* beliebig langes Bytefeld     */
    KS_VT_BOOL_VEC    = 0x82, /* beliebig langes Boolean-Feld */
    KS_VT_INT_VEC     = 0x90  /* beliebig langes Int-Feld     */
    KS_VT_UINT_VEC    = 0x91, /* usw. pp. (Felder)            */
    KS_VT_SINGLE_VEC  = 0xA0,
    KS_VT_DOUBLE_VEC  = 0xA1,
    KS_VT_STRING_VEC  = 0xB0,
    KS_VT_TIME_VEC    = 0xB1,
};
```

Um das Nachrichtenprotokoll nicht unnötigerweise zu verkomplizieren, kennt ACPLT/KS bei Variablenobjekten weder mehrdimensionale Felder noch zusammengesetzte Datenstrukturen. Mehrdimensionale Felder lassen sich jedoch immer auf eindimensionale Felder durch Umrechnung der Indizes abbilden. Ebenso bildet man zusammengesetzte Datenstrukturen zweckmäßig auf eine Domain mit entsprechenden Variablenobjekten ab. Die Variablenobjekte entsprechen dann den einzelnen Feldern in der Datenstruktur.

```
typedef string KS_STRING<>;

union KS_VAR_VALUE switch (KS_VAR_TYPE type) {
    case KS_VT_VOID:        void;
    case KS_VT_BOOL         KS_BOOL    val_bool;
    case KS_VT_INT:         long       val_int;
    case KS_VT_UINT:        u_long     val_uint;
    case KS_VT_SINGLE:      float      val_single;
    case KS_VT_DOUBLE:      double     val_double;
    case KS_VT_STRING:      KS_STRING  val_string<>;
    case KS_VT_TIME:        KS_TIME    val_time;
    case KS_VT_BYTE_VEC:    opaque     val_opaque<>;
    case KS_VT_BOOL_VEC:    KS_BOOL    val_bool_vec<>;
    case KS_VT_INT_VEC:     long       val_int_vec<>;
    case KS_VT_UINT_VEC:    u_long     val_uint_vec<>;
    case KS_VT_SINGLE_VEC:  float      val_single_vec<>;
    case KS_VT_DOUBLE_VEC:  double     val_double_vec<>;
    case KS_VT_STRING_VEC:  KS_STRING  val_string_vec<>;
    case KS_VT_TIME_VEC:    KS_TIME    val_time_vec<>;
};
```

Der Wert eines KS-Variablenobjektes wird durch die Datenstruktur `KS_VAR_VALUE` beschrieben. In dieser wird nicht nur der Wert vermerkt, sondern auch (implizit) der Variablentyp. Dadurch wird bei ACPLT/KS zwischen Servern und Klienten nicht nur der reine Variablenwert ausgetauscht, sondern auch zugleich der Typ mitgeschickt.

2.2.2 Eigenschaften von Domainobjekten

Domainobjekte besitzen in ACPLT/KS 1.0 nur die gemeinsamen Eigenschaften aller KS-Kommunikationsobjekte.

3 Die Objekt-Services

Der Service KS_GETPP dient dazu, die statischen Informationen von Kommunikationsobjekten zu erfragen. Zugleich kann mit Hilfe dieses Services der Objektbaum eines Servers Domain-weise durchsucht werden. Dazu ist der KS_GETPP-Service mit der folgenden Datenstruktur aufzurufen (nähere Informationen über die Authentifizierung mittels auth siehe [1]).

```
struct KS_GETPP_REQ {
    KS_AUTHREQ  auth;
    string      path<>;
    KS_OBJTYPE  type_mask;
    string      name_mask<>;
};
```

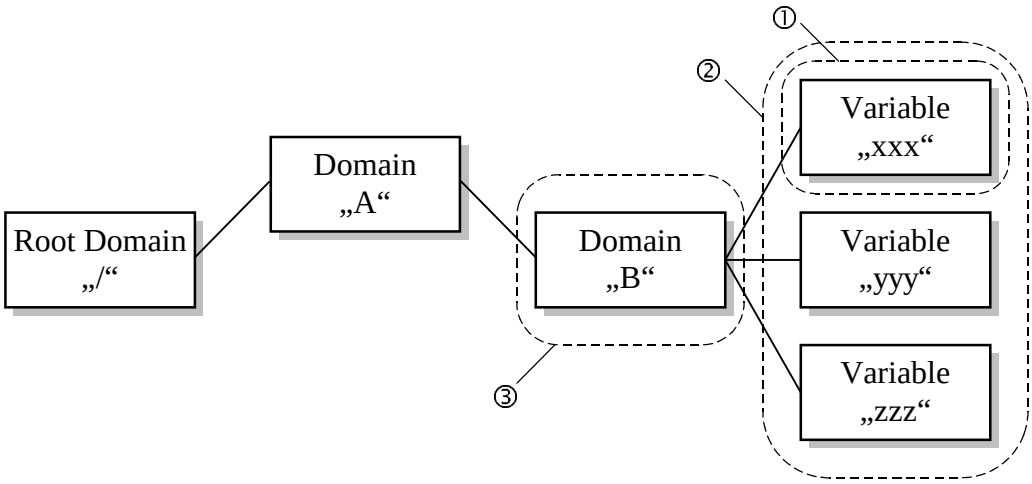


Abbildung 3.1: *Adressierung von Objekten beim* GetPP-*Service.*

In der Variablen path der Struktur KS_GETPP_REQ ist der vollständige Pfad der abzufragenden Domain oder Kommunikationsobjekte anzugeben. Dieser Pfad darf kein abschließendes "/"-Zeichen aufweisen (die einzige Ausnahme ist der Pfad auf die root domain "/"). In name_mask ist ein (Objekt-) Name anzugeben, der gegebenenfalls auch Jokerzeichen enthalten oder ein sogenannter „regulärer Ausdruck“ sein darf. Dabei sind die beiden folgenden Fälle zu unterscheiden (siehe auch Abbildung 3.1):

- Es soll ein einzelnes Kommunikationsobjekt – eine Variable oder eine History – abgefragt werden (①). Dann muß path den Pfad zum Objekt enthalten und als name_mask der Bezeichner des abzufragenden Objekts angegeben werden. Im betrachteten Beispiel ist dann path = "/A/B" und name_mask = "xxx" zu setzen.
- Es sollen alle Kindobjekte einer Domain abgefragt werden (②). Hierfür muß als path der Pfad der betrachteten Domain (im Beispiel wäre dies "/A/B") und als name_mask die gewünschte Maske angegeben werden (beispielsweise "*", um alle Kindobjekte zu erhalten).
- Soll eine Domain selbst abgefragt werden (Beispiel ③), so gibt es dafür zwei Möglichkeiten. Im ersten Fall gibt man als path den vollständigen Bezeichner der abzufragenden Domain an ("/A/B") und als name_mask einfach nur eine leere Zeichenkette ("") an. Im zweiten Fall betrachtet man die abzufragende Domain als Kind der darüberliegenden Domain und gibt damit im Beispiel ③ als path den Pfad "/A" an, sowie bei die name_mask den Namen der Domain selbst, also "B". Der zweite Weg kann jedoch nicht benutzt werden, um die Root Domain mittels GetPP abzufragen.

Die Namensmaske in name_mask kann einen regulären Ausdruck enthalten. Dieser ist wie folgt definiert:

- Die Jokerzeichen "?" und "*" passen auf jeweils ein einzelnes oder beliebig viele beliebige Zeichen.
- Mittels "[*Menge*]" wird ein (nicht unbedingt zusammenhängender) Bereich festgelegt, aus dem ein einzelnes Zeichen stammen muß. *Menge* kann entweder einfach aus einer Aufzählung von Zeichen bestehen oder aus einem Zeichenbereich der Form "*Zeichen-Zeichen*". Die Angabe "[^*Menge*]" paßt hingegen auf alle Zeichen, die nicht in *Menge* enthalten sind.
- Das auf einen rückwärtigen Schrägstrich ("\" = „backslash“) folgende Zeichen verliert eine eventuelle besondere Interpretation: "\?" entspricht somit einem einfachen "?" ohne besondere Bedeutung.

Über die Variable type_mask der Datenstruktur KS_GETPP_REQ läßt sich festgelegen, welche Kommunikationsobjekttypen von der Suche erfaßt werden. Will man alle Objekttypen erfassen, so ist KS_OT_ANY anzugeben, andernfalls müssen die entsprechenden Konstanten KS_OT_*xxx* über die ODER-Funktion miteinander verbunden werden (wie beispielsweise: "KS_OT_DOMAIN | KS_OT_VARIABLE").

Als Antwort des KS_GETPP Services erhält man die projektierten Eigenschaften aller Kommunikationsobjekte zurück, auf die die angegebene Maske paßt.

```
union KS_GETPP_RET switch (KS_RESULT result) {
    case KS_ERR_OK:
        KS_OBJ_PROJECTED_PROPS  items<>;
    default:
        void;
};

struct KS_GETPP_REPLY {
    KS_AUTH_REPLY  auth;
    KS_GETPP_RET   ret;
};
```

Der KS_GETPP-Service besitzt die folgende RPC-Prozedurnummer:

```
KS_GETPP_REPLY KS_GETPP(KS_GETPP_REQ) = 0x0101;
```

Der Service kann die folgenden Fehlermeldungen zurückliefern:

KS_ERR_OK
: Der Service KS_GETPP wurde erfolgreich ausgeführt.

KS_ERR_GENERIC
: Es trat bei der Ausführung des Service ein nicht näher definierter Fehler auf.

KS_ERR_BADPATH
: Der in path angegebene Pfad existiert nicht.

KS_ERR_BADMASK
: Der in name_mask angegebene reguläre Ausdruck ist ungültig.

KS_ERR_UNKNOWNAUTH
: Der Klient benutzt in seiner Serviceanforderung ein A/V-Schema, das vom Server nicht unterstützt wird.

KS_ERR_BADAUTH
: Der Klient konnte sich nicht authentifizieren.

4 Die Variablen-Services

Für den lesenden und schreibenden Zugriff auf die aktuellen Eigenschaften von Variablenobjekten stellt ACPLT/KS die Services KS_GETVAR und KS_SETVAR zur Verfügung.

4.1 Lesender Variablenzugriff

Nachfolgend wird zuerst der Dienst zum Lesen von Variablenobjekten beschrieben.

```
union KS_GETVAR_ITEM switch (KS_RESULT result) {
    case KS_ERR_OK:
        KS_OBJ_CURRENT_PROPS  item;
    default:
        void;
};
```

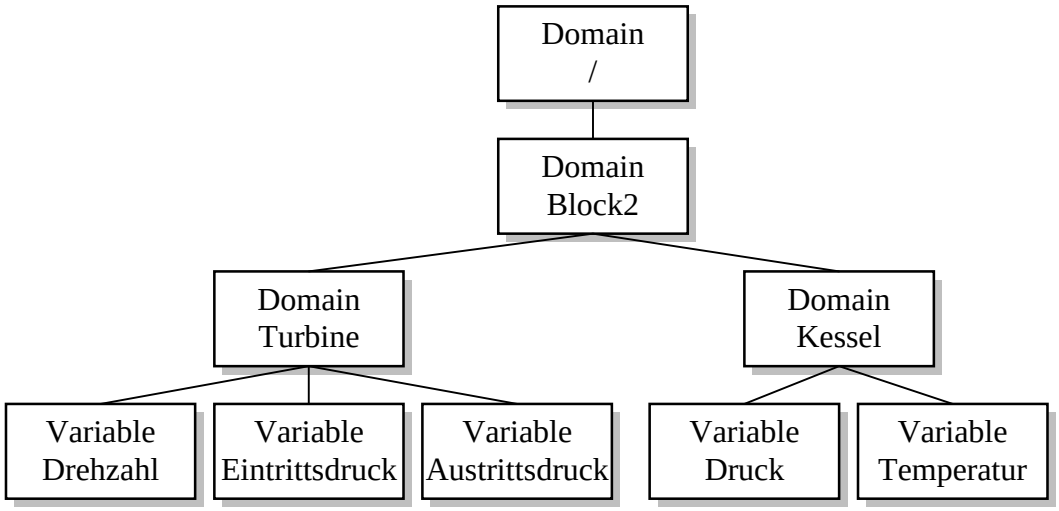
In der Datenstruktur KS_GETVAR_ITEM dürfen nur die aktuellen Eigenschaften von Variablen zurückgeschickt werden, aktuelle Eigenschaften anderer Kommunikationsobjekttypen sind nicht zulässig.

```
struct KS_GETVAR_REQ {
    KS_AUTH_REQ  auth;
    KS_STRING    identifiers<>;
};

union KS_GETVAR_RET switch (KS_RESULT result) {
    case KS_ERR_OK:
        KS_GETVAR_ITEM  items<>;
    default:
        void;
};

struct KS_GETVAR_REPLY {
    KS_AUTH_REPLY  auth;
    KS_GETVAR_RET  ret;
};
```

Dem KS_GETVAR-Service übergibt man eine Datenstruktur vom Typ KS_GETVAR_REQ, in der die Namen der zu lesenden Objekte aufgelistet sind. Im einfachsten Fall bestehen diese Namen aus dem Bezeichner inklusive des vollständigen Pfades – also zum Beispiel (siehe Abbildung 4.1): "/Block2/Turbine/Dampfdruck". ACPLT/KS erlaubt daneben auch die Adressierung der Variablenobjekte über relative Pfade.



Absolute Adressierung

```
/Block2/Turbine/Drehzahl
/Block2/Turbine/Eintrittsdruck
/Block2/Turbine/Austrittsdruck
/Block2/Kessel/Druck
/Block2/Kessel/Temperatur
```

Relative Adressierung

```
/Block2/Turbine/Drehzahl
./Eintrittsdruck
./Austrittsdruck
../Kessel/Druck
./Temperatur
```

Abbildung 4.1: *Vergleich der absoluten mit der relativen Adressierung von Variablenobjekten.*

Die relative Adressierung ist nur innerhalb einer einzelnen Serviceanfrage möglich, nicht aber über zwei oder mehrere Serviceanfragen hinweg. Bei jedem Variablenzugriff vermerkt sich der KS-Server die zum angesprochen Variablenobjekt zugehörige Eltern-Domain. Der darauffolgende Variablenzugriff (noch innerhalb der selben Serviceanfrage) kann sich dann auf die vermerkte Domain beziehen, indem man statt des vollständigen, absoluten Pfades nun relativ zu `"."` adressiert. Greift man zuerst auf die Variable `"/Block2/Kessel/Druck"` zu (siehe den Objektbaum in Abbildung 4.1), so merkt sich der Server die zu diesem Variablenobjekt zugehörige Elterndomain `"/Block2/Kessel"`. Beim darauffolgenden Zugriff beispielsweise auf `"/Block2/Kessel/Temperatur"` kann man dann das Variablenobjekt auch gleichwertig über `"./Temperatur"` relativ adressieren. Zusätzlich kann die Angabe `"./"` auch entfallen, so daß `"Temperatur"` gleichwertig mit `"./Temperatur"` ist.

Bei der relativen Adressierung ist weiterhin auch die Angabe von `".."` zulässig, womit die jeweils nächsthöhere Eltern-Domain angesprochen wird. Eine mehrfache Angabe von `".."` ist zulässig, wodurch man immer die Eltern-Domain der Eltern-Domain (und so weiter) adressiert.

Das Auslesen der Variablen erfolgt in der Reihenfolge, in der diese in der Service-Anforderung aufgezählt sind. Tritt beim Zugriff auf eine einzelne Variable ein Fehler auf, dann arbeitet der Server trotzdem die noch verbleibenden abzufragenden Variablen in der Liste ab (sofern der aufgetretene Fehler dieses nicht grundsätzlich verhindert). In jedem Fall liefert der Server zu jeder abzufragenden Variablen einen (gültigen) Fehlercode im Feld `items` in der Antwortdatenstruktur `KS_GETVAR_REPLY` zurück. Bei der relativen wie auch bei der absoluten Adressierung ist zu beachten, daß beim Zugriff auf eine nicht existierende Variable alle folgenden Variablenzugriffe mit relativer Adressierung einen Folgefehler (`KS_ERR_BADNAME`) verursachen. Die Werte zusammen mit eventuellen Fehlercodes sind in der Serviceantwort `KS_GETVAR_REPLY` in `items` in der selben Reihenfolge abgelegt, in der die Angabe der Variablennamen in der Serviceanforderung `KS_GETVAR_REQ` erfolgte.

Die ACPLT/KS-Spezifikation garantiert nicht, daß das Auslesen aller Variablen einer einzelner Serviceanforderung atomar erfolgt.

Der Service KS_GETVAR kann die folgenden Fehlermeldungen zurückliefern (in result):

KS_ERR_OK
: Der Service KS_GETVAR wurde erfolgreich ausgeführt.

KS_ERR_GENERIC
: Es trat bei der Ausführung des Service ein nicht näher definierter Fehler auf.

KS_ERR_UNKNOWNAUTH
: Der Klient benutzt in seiner Serviceanforderung ein A/V-Schema, das vom Server nicht unterstützt wird.

KS_ERR_BADAUTH
: Der Klient konnte sich nicht authentifizieren.

Daneben können auf Variablenbasis die folgenden Fehlermeldungen auftreten:

KS_ERR_OK
: Der Lesezugriff auf die betroffene Variable wurde erfolgreich ausgeführt.

KS_ERR_BADNAME
: Der Pfad und/oder Variablenname ist ungültig oder existiert nicht. Wird mit ".." versucht, die Eltern-Domain der Root-Domain anzusprechen, tritt dieser Fehler ebenfalls auf. Dieser Fehlercode tritt dann als Folgefehler auf, wenn die betroffene Variable relativ adressiert wird und bereits bei der Adressierung der (innerhalb der Serviceanforderung) vorangegangenen Variable ein Fehler auftrat.

KS_ERR_NOACCESS
: Auf das betroffene Variablenobjekt darf nicht lesend zugegriffen werden.

4.2 Schreibender Variablenzugriff

Für das Schreiben von Werten in Variablenobjekte ist der KS_SETVAR-Service zuständig. Hiermit lassen sich grundsätzlich nur die aktuellen Eigenschaften eines Variablenobjektes verändern. Der Service erlaubt es, mehrere Variablen in einem einzigen Request zu beschreiben. Treten dabei Fehler auf, so werden diese auf Variablenbasis zurückgemeldet. Es gelten die gleichen Annahmen über die Abarbeitung der Serviceanforderung, wie sie zuvor bereits beim KS_GETVAR-Service beschrieben wurden: Es ist garantiert, daß alle innerhalb einer einziger Serviceanforderung angesprochenen Variablen in derjenigen Reihenfolge beschrieben werden, in der sie in der Anforderung aufgeführt sind. Weiterhin gelten die Angaben über absolute und relative Adressierung. Der Service KS_SETVAR ist nicht atomar.

Jede einzelne zu setzende Variable innerhalb der Serviceanforderung KS_SETVAR wird durch ihren Namen inklusive des vollständigen Pfades (Variable path_and_name in der Datenstruktur KS_SETVAR_ITEM) adressiert. Die neu zu schreibenden aktuellen Eigenschaften sind jeweils in current_props der Datenstruktur KS_SETVAR_ITEM angegeben.

```
struct KS_SETVAR_ITEM {
    string                path_and_name<>;
    KS_OBJ_CURRENT_PROPS  current_props;
};
```

In der Datenstruktur KS_SETVAR_ITEM dürfen nur die aktuellen Eigenschaften von Variablen an den Server gesendet werden, aktuelle Eigenschaften anderer Kommunikationsobjekttypen sind nicht zulässig (und führen dazu, daß im Allgemeinen die Bearbeitung der gesamten Dienstanforderung mit einem generischen Fehler abgebrochen werden sollte).

```
struct KS_SETVAR_REQ {
    KS_AUTH_REQ     auth;
    KS_SETVAR_ITEM  items<>;
};
```

Der Service liefert neben einer auf die gesamte Serviceanforderung bezogenen Fehlermeldung (in `result`) noch eine Liste von Fehlermeldungen auf Variablenbasis (in `results`) zurück. Der Server antwortet jedoch nur dann mit dieser Liste in `results`, wenn kein Fehler bei der Serviceanforderung auftrat (`result = KS_ERR_NONE`), der die Abarbeitung der einzelnen Schreibzugriffe verhinderte. Die Reihenfolge der Fehlercodes in der Diensteantwort entspricht derjenigen der Diensteanfrage, der erste Fehlercode ist der ersten zu schreibenden Variable zugeordnet, und so weiter.

```
union KS_SETVAR_RET switch(KS_RESULT result) {
    case KS_OK:
        KS_RESULT  results<>;
    default:
        void;
};

struct KS_SETVAR_REPLY {
    KS_AUTH_REPLY  auth;
    KS_SETVAR_RET  ret;
};
```

Der Service kann die folgenden Fehlermeldungen zurückliefern (in `result`):

`KS_ERR_OK`
: Der Service `KS_SETVAR` wurde erfolgreich ausgeführt.

`KS_ERR_GENERIC`
: Es trat bei der Ausführung des Service ein nicht näher definierter Fehler auf.

`KS_ERR_UNKNOWNAUTH`
: Der Klient benutzt in seiner Serviceanforderung ein A/V-Schema, das vom Server nicht unterstützt wird.

`KS_ERR_BADAUTH`
: Der Klient konnte sich nicht authentifizieren.

Daneben können auf Variablenbasis die folgenden Fehlermeldungen auftreten:

`KS_ERR_OK`
: Der Schreibzugriff auf die betroffene Variable wurde erfolgreich ausgeführt.

`KS_ERR_BADNAME`
: Der Pfad und/oder Variablenname ist ungültig oder existiert nicht. Eine Adressierung mittels `".."` über die Root-Domain hinaus, wird ebenfalls mit diesem Fehlercode angezeigt. Dieser Fehlercode tritt dann als Folgefehler auf, wenn die betroffene Variable relativ adressiert wird und bereits bei der Adressierung der (innerhalb der Serviceanforderung) vorangegangenen Variable ein Fehler auftrat.

`KS_ERR_NOACCESS`
: Auf das betroffene Variablenobjekt darf nicht schreibend zugegriffen werden.

`KS_ERR_BADTYPE`
: Der Typ der zu schreibenden Daten stimmt nicht mit dem Datentyp der Variablen überein.

4.3 Synchronisierter Datenaustausch

Neben dem „einfachen“ Schreiben oder Lesen der Werte von Variablenobjekten bietet ACPLT/KS einen synchronisierten Datenaustausch mit Variablenobjekten an. Hierbei übermittelt ein Klient einen Satz neuer Variablenwerte an den KS-Server und erfragt innerhalb der gleichen Serviceanfrage die Werte anderer Variablen (Abbildung 4.2).

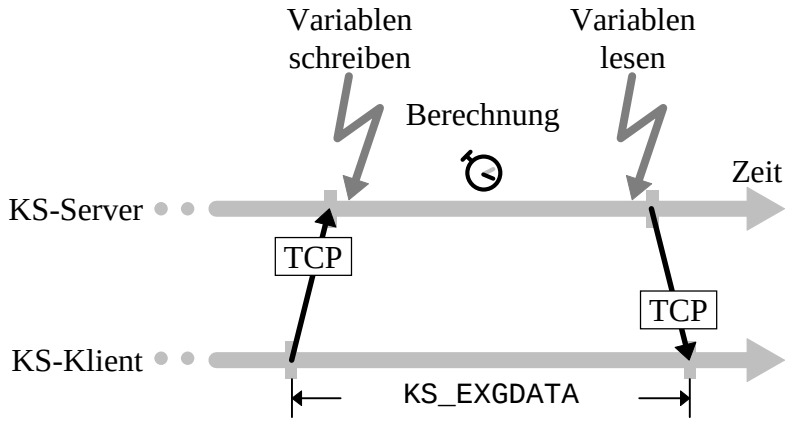


Abbildung 4.2: *Der Service* `KS_EXGDATA` *erlaubt ein Schreiben von Variablenwerten mit anschließendem, synchronisierten Lesen von Variablen.*

Durch die Schreibzugriffe auf Variablenobjekte wird im Server beispielsweise eine spezielle Berechnung auf Basis der soeben gesetzten Variablen ausgelöst. Sobald dann die Daten aus dieser Berechnung für den Klienten bereitstehen, bekommt dieser die Werte vom Server zurückgeliefert. Der Server stellt somit bei diesem `KS_EXGDATA`-Service sicher, daß das Auslesen der angeforderten Variablen erst dann geschieht, nachdem die dazugehörigen Ergebnisse zur Verfügung stehen (Synchronisation des Auslesens).

Die dem Service `KS_EXGDATA` zu übergebenden Parameter entsprechen den Parametern, die bereits für das Schreiben und Lesen von Variablen im `KS_SETVAR`- beziehungsweise `KS_GETVAR`-Service zum Einsatz kommen. Lediglich die Authentifizierungsinformationen werden nur einmal zu Beginn der Serviceanforderung übermittelt.

```
struct KS_EXGDATA_REQ {
    KS_AUTH_REQ     auth;
    KS_SETVAR_ITEM  set_vars<>;
    KS_STRING       get_vars<>;
};
```

Die Antwort des Servers entspricht im Fall des `KS_EXGDATA`-Service ebenso denjenigen Antworten, die von `KS_SETVAR` sowie `KS_GETVAR` zurückgeliefert werden. Das Feld `results` enthält dabei die Fehlercodes für alle schreibenden Variablenzugriffe und das Feld `items` enthält die gelesenen Variablenwerte. Auch hier werden die Authentifzierungsinformationen nur einmal zu Beginn der Serviceanforderung übertragen.

```
struct KS_EXGDATA_ITEMS {
    KS_RESULT       results<>;
    KS_GETVAR_ITEM  items<>;
};

union KS_EXGDATA_RET switch (KS_RESULT result) {
    case KS_OK:
        KS_EXGDATA_ITEMS items;
    default:
        void;
};

struct KS_EXGDATA_REPLY {
    KS_AUTH_REPLY   auth;
    KS_EXGDATA_RET  ret;
};
```

Der Service kann die folgenden Fehlermeldungen zurückliefern (in `result`):

`KS_ERR_OK`
: Der Service `KS_EXGDATA` wurde erfolgreich ausgeführt.

`KS_ERR_NOTIMPLEMENTED`
: Dieser Service ist nicht implementiert.

`KS_ERR_GENERIC`
: Es trat bei der Ausführung des Service ein nicht näher definierter Fehler auf.

`KS_ERR_UNKNOWNAUTH`
: Der Klient benutzt in seiner Serviceanforderung ein A/V-Schema, das vom Server nicht unterstützt wird.

`KS_ERR_BADAUTH`
: Der Klient konnte sich nicht authentifizieren.

`KS_ERR_CANTSYNC`
: Der angegebene Satz von zu schreibenden und lesenden Variablenobjekten kann nicht synchronisiert abgearbeitet werden.

Daneben können auf Variablenbasis die folgenden Fehlermeldungen auftreten:

`KS_ERR_OK`
: Der Schreib-/Lesezugriff auf die betroffene Variable wurde erfolgreich ausgeführt.

`KS_ERR_BADNAME`
: Der Pfad und/oder Variablenname ist ungültig oder existiert nicht. Eine Adressierung mittels `".."` über die Root-Domain hinaus, wird ebenfalls mit diesem Fehlercode angezeigt. Dieser Fehlercode tritt dann als Folgefehler auf, wenn die betroffene Variable relativ adressiert wird und bereits bei der Adressierung der (innerhalb der Serviceanforderung) vorangegangenen Variable ein Fehler auftrat.

`KS_ERR_NOACCESS`
: Auf das betroffene Variablenobjekt darf nicht schreibend/lesend zugegriffen werden.

`KS_ERR_BADTYPE`
: Der Typ der zu schreibenden Daten stimmt nicht mit dem Datentyp der Variablen überein.

4.4 Service-Prozedurnummern

Die Variablen-Services besitzen die folgenden RPC-Prozedurnummern:

```
KS_GETVAR_REPLY KS_GETVAR(KS_GETVAR_REQ)    = 0x0101;
KS_SETVAR_REPLY KS_SETVAR(KS_SETVAR_REQ)    = 0x0102;
KS_EXGDATA_REPLY KS_EXGDATA(KS_EXGDATA_REQ) = 0x0103;
```

5 AuA – Abkürzungen und Akronyme

ACPLT/KS	Kommunikationssystem des Lehrstuhls für Prozeßleittechnik der RWTH Aachen
IP	Internet Protocol
ISO	International Standards Organization
MMS	Manufacturing Message Specification
ONC	Open Network Computing
OSI	Open Systems Interconnect
RPC	Remote Procedure Call/Calling
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
XDR	External Data Representation

6 Literaturverzeichnis

[1] ACPLT/KS Group:
Technologiepapier #7: A/V-Module.
Lehrstuhl für Prozeßleittechnik, RWTH Aachen, 1996