



**DRIVECOM**

# Gerätebeschreibung

Version: 1.1, 28. Mai 2002

Herausgeber: DRIVECOM Nutzergruppe e.V.  
Postfach 1102, D-32817 Blomberg  
Telefon : 0 52 35 / 3-4 18 64  
Fax : 0 52 35 / 3-4 18 62  
Internet: <http://www.drivecom.org/>

Alle Rechte, auch die der Übersetzung, vorbehalten. Kein Teil dieser Information darf in irgendeiner Form (Druck, Fotokopie, Mikrofilm oder einem anderen Verfahren) ohne schriftliche Genehmigung der DRIVECOM Nutzergruppe e.V., reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Änderungen vorbehalten

**Verfasser:**

Krumsiek	Phoenix Contact	Blomberg
Hadlich	ifak system GmbH	Barleben
Heines	Phoenix Contact	Blomberg
Lange	Softing GmbH	München
Dr. Leurs	Rexroth Indramat	Lohr / Main
Mirbach	Lenze	Hameln
Pollmeier	ESR Pollmeier	Ober-Ramstadt
Riedl	ifak Magdeburg e.V.	Barleben
Schäfer	Lenze	Hameln
Schleicher	Indramat Refu	Metzingen
Schnurbusch	Lenze	Hameln
Schwarz	Mannesmann Dematic	Wetter
Senneka	Stöber Antriebstechnik	
Teipen	Hanning Elektro-Werke	Oerlinghausen
Vothknecht	Phoenix Contact	Blomberg
Ziegler	SEW-EURODRIVE	Bruchsal

**Historie:**

Bearbeiter	Version, Änderungen
Riedl	07.07.2000 V0.0.2 Entwurf
Riedl	17.07.2000 V0.0.46 Entwurf
Riedl	V0.2 Entwurf Referenzierungen an XML-Links angelehnt, Redefine hinzugefügt, domain-Attribut bei Sprachdatei, Interface für Scriptzugriff auf DriveServer
Riedl	27.11.2000 V0.8a Harmonisierung mit FDCML 1.0
Mirbach	V1.0 Einarbeitung der bei der Implementierung des Generischen DriveServers gewonnenen Ergebnisse
Mirbach	V1.1 Hinzufügen des Datentyps ARRAY_OF_VT_UI1

Anmerkungen und Kommentare sind an Herrn Stefan Pollmeier [gl@esr-pollmeier.de](mailto:gl@esr-pollmeier.de) zu richten.



## Inhalt

1.	Einleitung .....	5
1.1.	Übersicht .....	5
1.2.	Referenzen .....	5
1.3.	Abkürzungen .....	5
2.	Konzept .....	6
2.1.	Übersicht .....	6
2.2.	Architektur .....	6
2.3.	Quellen und Profile .....	6
3.	Basiselemente der Sprache .....	7
3.1.	Übersicht .....	7
3.2.	AIP .....	7
3.3.	device .....	8
3.4.	DeviceIdentityObject .....	8
3.5.	DeviceManagerObject .....	9
3.5.1.	ProcessDataItemList .....	10
3.5.2.	ParameterItemList .....	11
3.6.	DeviceFunctionObject .....	12
3.6.1.	DictionaryList .....	12
3.6.2.	HelpFileList .....	13
3.6.3.	containerList, componentList .....	13
3.6.4.	varTemplateList .....	14
3.6.5.	Variable .....	17
3.6.6.	Aufzählungen .....	18
3.6.7.	Menu .....	19
3.6.8.	Methoden .....	19
3.6.9.	Neudefinitionen .....	19
4.	Benutzte Property-IDs .....	20
5.	Wörterbuch (DTD der Sprachdatei) .....	21
6.	Hilfdateien .....	21
7.	Interface .....	21
7.1.	Überblick .....	21
7.2.	Funktionsbeschreibung .....	22
7.3.	IDL .....	23
8.	Syntax der Sprachen-DTD .....	24

# 1. Einleitung

## 1.1. Übersicht

Dieses Dokument beschreibt die Methode für die elektronisch auswertbare Beschreibung von Antriebsparametern und Antriebsfunktionalität. Die Gerätebeschreibung wird für die Konfigurierung des DriveServers benutzt. Sie kann weiterhin für die Produktbeschreibung in weiteren Bereichen benutzt werden. Die hier vorgestellte DTD umfaßt folgende Aspekte:

- Beschreibung der Antriebsparameter
- Unterstützung der Parameterabhängigkeiten
- Logische Gruppierung der unterstützten Antriebsparameter

Für diese Gerätebeschreibung wird eine Document Type Definition (DTD) entsprechend des XML-Standards definiert.

## 1.2. Referenzen

- [1] OLE for Process Control, Data Access Custom Interface Standard, Version 2.0, <http://www.opcfoundation.org>
- [2] DRIVECOM, DriveServer V 1.0
- [3] XML Version 1.0, <http://www.w3c.org/xml>
- [4] Language Specification of Electronic Device Description
- [5] Field DeviceConfiguration Markup Language 1.0

## 1.3. Abkürzungen

DOM	<b>D</b> ocumnet <b>O</b> bject <b>M</b> odell
DTD	<b>D</b> ocument <b>T</b> ype <b>D</b> efinition
FDCML	<b>F</b> ield <b>D</b> evice <b>C</b> onfiguration <b>M</b> arkup <b>L</b> anguage
OPC	<b>O</b> LE for <b>P</b> rocess <b>C</b> ontrol
W3C	<b>W</b> orld <b>W</b> ide <b>W</b> eb <b>C</b> onsortium
XML	<b>e</b> Xtensible <b>M</b> arkup <b>L</b> anguage

## 2. Konzept

### 2.1. Übersicht

Die Gerätebeschreibung der DRIVECOM ist für die herstellerunabhängige Beschreibung der Antriebseigenschaften entworfen worden. Mit Hilfe der Gerätebeschreibung kann der DriveServer herstellerunabhängig konfiguriert werden. Die Art und Weise der Identifizierung eines Antriebs am Bus ist nicht Gegenstand dieser Spezifikation und erfolgt in der Regel nach herstellerspezifischen Regeln.

Die Gerätebeschreibung ist lesbarer Text, der vom Antriebshersteller geschrieben wird, um all die Informationen zu beschreiben, die zur Konfiguration und Bedienung des Antriebs notwendig sind.

### 2.2. Architektur

Die Systemarchitektur besteht aus der Sprachdefinition in Form der DTD, der Definition des Wörterbuches in Form einer DTD, einer Vielzahl von Gerätebeschreibungen (XML-Dateien), die in den DriveServer geladen werden und den landesspezifischen Sprachdateien. Der DriveServer kann zusätzliche Layout-Informationen in Form von Stylesheets hinzuziehen. Die Daten können intern z.B. in vordefinierten Objekten (DOM) oder in spezifischen Strukturen abgelegt werden. Die Stylesheets müssen vom DriveServer selbst nicht ausgewertet werden. Sie können jedoch ggf. an die Clients weitergegeben werden.

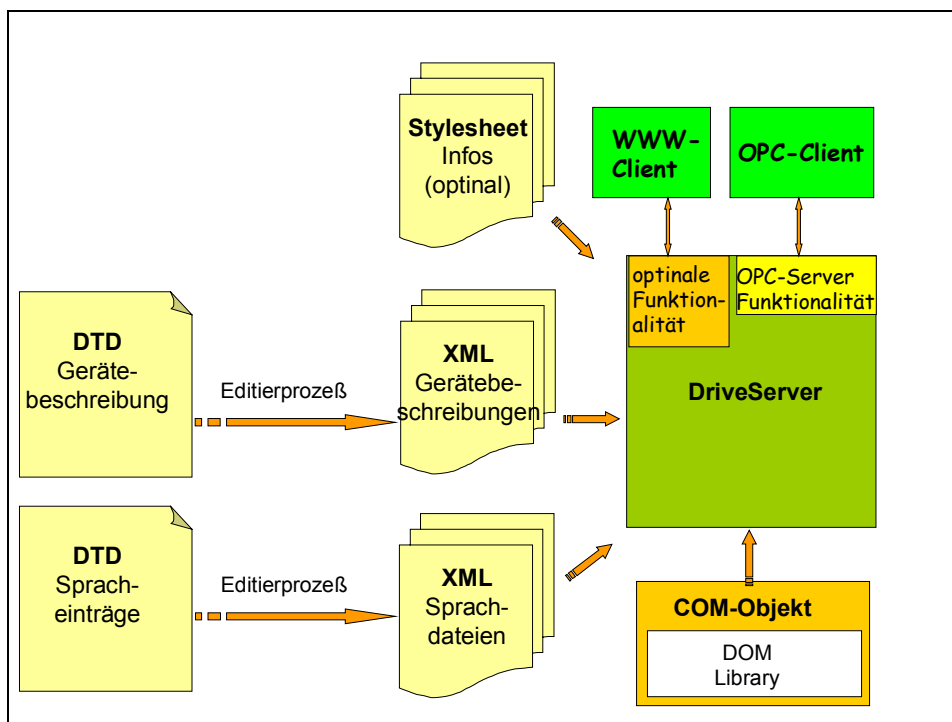


Abbildung 1: Integration der Gerätebeschreibung in DriveServer-Konzept

### 2.3. Quellen und Profile

Die Gerätebeschreibung enthält alle benötigten Informationen, um den Antrieb zu beschreiben. Dazu können die Beschreibungen in Standard- und gerätespezifische Teile gegliedert sein. Standardbeschreibungen werden von der eigentlichen Beschreibung importiert und können von Herstellergruppen spezifiziert werden. Der Hersteller muß dann lediglich die zusätzlichen Teile beschreiben.

## 3. Basiselemente der Sprache

### 3.1. Übersicht

Die DRIVECOM-Gerätebeschreibung basiert auf der XML-Technologie, die vom W3C definiert wurde, und auf der Spezifikation FDCML.

Die Semantik der einzelnen Elemente wird in dieser Spezifikation bzw. in [5] beschrieben.

Die DRIVECOM-Gerätebeschreibung benutzt aus der FDCML folgende Elemente:

```
AIP
device
DeviceIdentityObject
DeviceManagerObject
communicationEntity
DeviceFunctionObject
```

### 3.2. AIP

Das Element 'AIP' dient in der FDCML zur Definition der 'Application Integrated Profiles' (siehe [5]). Dieses Element dient bei der ausschließlichen Nutzung der XML-Gerätebeschreibung für die Konfiguration des DriveServers als ein Einsprungpunkt, um Verweise auf weitere XML-Dateien zu deren Import zu legen, und um das Gerät zu beschreiben.

Syntax:

```
<!ELEMENT AIP
                                (importList?,
                                (INTERBUSBasedDevice|
                                device|
                                devices))>
```

Die Elemente 'INTERBUSBasedDevice' und 'devices' werden hier nicht weiter betrachtet.

Im Element 'device' werden die weiteren Informationen der DRIVECOM (gerätespezifische Parameter und Funktionen) codiert.

In dieser Spezifikation ist insbesondere das Element 'DeviceFunctionObject' beschrieben. Dieses Element ist in der ursprünglichen Fassung der FDCML leer.

Syntax:

```
<!ELEMENT importList          (file+)>
<!ELEMENT file                 EMPTY>
<!ATTLIST file                 xml:link (simple) #IMPLIED
                                href CDATA #REQUIRED>
```

Unter 'importList' können Referenzen auf zu importierende XML-Dateien gelegt werden, um Profilbeschreibungen bzw. herstellerspezifische Standardbeschreibungen benutzen zu können. Die Imports müssen vom DriveServer zuvor, auch rekursiv, geladen werden. Wird der Importabschnitt benutzt, muß mindestens eine Datei angegeben werden. Für die Implementierung ist es offen, die Imports ggf. in einem Preprozessor aufzulösen und mit der so vorverarbeiteten Datei den DriveServer zu konfigurieren.

href: String mit URL der einzubindenden Datei

### 3.3. device

Das Element 'device' dient zur Beschreibung aller gerätespezifischen Funktionen (siehe [5]). Diese Spezifikation nutzt die mandatory Attribute und lediglich die Kind-Elemente 'DeviceIdentityObject', 'DeviceManagerObject' und 'DeviceFunctionObject'. Die zusätzlichen Kind-Elemente müssen für DRIVECOM-Zwecke nicht ausgefüllt werden.

Syntax:

```
<!ELEMENT device (DeviceIdentityObject,
DeviceManagerObject,
DeviceFunctionObject*,
ApplicationProcessObject*) >
<!ATTLIST device
formatName NMTOKEN #FIXED "FDCML"
formatVersion NMTOKEN #FIXED "1.0"
fileName CDATA #REQUIRED
fileCreator CDATA #REQUIRED
fileCreationDate CDATA #REQUIRED
fileModificationDate CDATA #REQUIRED
fileVersion CDATA #REQUIRED
%id.unique.opt;>
```

fileName:	Dateiname der XML-Datei
fileCreator:	spezifischer String des Erzeugers der Datei
fileCreationDate:	Erstellungsdatum der Datei
fileModificationDate:	Datum der letzten Änderung an der Datei
fileVersion:	herstellerspezifische Version der Datei

### 3.4. DeviceIdentityObject

In diesem Element werden alle netzwerk- oder busunabhängigen Eigenschaften beschrieben, die zur Identifikation des Gerätes dienen (siehe [5]).

Syntax:

```
<!ELEMENT DeviceIdentityObject (vendor,
vendorText?,
family,
deviceType,
designation,
deviceText?,
orderNumber,
version) >
<!ELEMENT vendor (const+|edit+|labelRef) >
<!ATTLIST vendor
vendorID CDATA #IMPLIED>
```



vendor:	String mit Name des Antriebsherstellers, zusätzlich können auch noch Referenzen auf Homepages oder auf externe Strings gelegt werden
vendorID:	optionale ID eines Herstellers, die eine Organisation vergeben kann
	<code>&lt;!ELEMENT vendorText (const+ edit+ labelRef) &gt;</code>
vendorText:	Zusätzliche Informationen zum Hersteller, zusätzlich können auch noch Referenzen auf externe Strings etc. gelegt werden
	<code>&lt;!ELEMENT family (const+ edit+ labelRef) &gt;</code>
family:	Herstellerspezifische Informationen zur Gerätereihe, zusätzlich können auch noch Referenzen auf externe Strings etc. gelegt werden
	<code>&lt;!ELEMENT deviceType (const+ edit+ labelRef) &gt;</code>
deviceType:	String mit konkreter, herstellerspezifischer Bezeichnung einer Antriebsreihe oder Gerätetyps, zusätzlich können auch noch Referenzen auf externe Strings etc. gelegt werden
	<code>&lt;!ELEMENT designation (const+ edit+ labelRef) &gt;</code> <code>&lt;!ATTLIST designation deviceID CDATA #IMPLIED&gt;</code>
designation:	String mit konkreter, herstellerspezifischer Bezeichnung des Antriebs, zusätzlich können auch noch Referenzen auf externe Strings etc. gelegt werden
deviceID:	herstellerspezifische GeräteID (optimal)
	<code>&lt;!ELEMENT deviceText (const+ edit+ labelRef) &gt;</code>
deviceText:	String mit ausführlicher Beschreibung einer Antriebsreihe oder Gerätetyps, zusätzlich können auch noch Referenzen auf externe Strings etc. gelegt werden
	<code>&lt;!ELEMENT orderNumber (const+ edit+ labelRef) &gt;</code>
orderNumber:	String mit herstellerspezifischer Bestellnummer einer Antriebsreihe oder Gerätetyps, zusätzlich können auch noch Referenzen auf externe Strings etc. gelegt werden
	<code>&lt;!ELEMENT version (const+ edit+ labelRef) &gt;</code>
version:	String mit Version/Revisionsnummer des Gerätes, zusätzlich können auch noch Referenzen auf externe Strings etc. gelegt werden

### 3.5. DeviceManagerObject

Im Element 'DeviceManagerObject' werden alle netzwerkspezifischen bzw. kommunikations-spezifischen Eigenschaften des Gerätes vermerkt.

Syntax:

```
<!ELEMENT DeviceManagerObject (dictionaryList?,  
helpFileList?,  
toolList?,  
pictureList?,  
physicalConnectionPointList?,
```

```

    localDataItemList?,
    additionalItemList*,
    communicationEntity+) >
  
```

Für die DRIVECOM ist hier nur der Eintrag 'communicationEntity' von Bedeutung. Hier werden die Beziehungen zwischen den DriveServer-OPC-Items zu den Kommunikationsobjekten hergestellt.

Syntax:

```

<!ELEMENT communicationEntity      (%naming.opt;;
                                     helpFileList?,
                                     toolList?,
                                     pictureList?,
                                     cfgItemList?,
                                     additionalItemList*,
                                     processDataItemList?,
                                     parameterItemList?,
                                     logicalConnectionPointList?,
                                     MAUUsageList?,
                                     internalConnectionPointList?,
                                     slotUsageList?) >

<!ATTLIST communicationEntity      %id.unique.opt;
                                     protocol CDATA #REQUIRED
                                     communicator (NO|YES) 'YES'
                                     communicationEntityType (SLAVE|
                                                                MASTER|
                                                                CLIENT|
                                                                SERVER|
                                                                INTERCONNECTION|
                                                                PEER|
                                                                MASTER_SLAVE) 'SLAVE'
                                     %enabled.def;>
  
```

Hierin sind alle Einträge optional. (Anmerkung: in der Original-FDCML sind einige Einträge Pflicht!).

Für die DRIVECOM sind die Einträge 'processDataItemList' und 'parameterItemList' von Interesse.

### 3.5.1. ProcessDataItemList

Dieses Element dient zur Beschreibung der vorhandenen Prozeßdatenobjekte.

Syntax:

```

<!ELEMENT processDataItemList      (%naming.opt;;
                                     (processDataCategory|
                                     processDataItem) +) >
  
```

Die optionale Bezeichnung '%naming.opt;' der Liste ist nicht erforderlich. Es werden lediglich Einträge für die Prozeßdaten als 'processDataItem' gefordert.

Syntax:

```

<!ELEMENT processDataItem      (%naming;,
                                pictureList?,
                                (accessPath|(bytePos,bitPos?)),
                                datatype,
                                specificProperty*,
                                uses?,
                                processDataItem*,
                                instances?)>

<!ATTLIST processDataItem     %id.unique.req;
                                direction (I|Q) #REQUIRED
                                signalType (D|A) #REQUIRED
                                processDataItemType CDATA #IMPLIED
                                %enabled.def;>
  
```

naming:	String mit Bezeichnung des Prozeßdatenobjektes für Anzeigezwecke, wird bei DRIVECOM nicht ausgewertet
pictureList:	bei DRIVECOM nicht benutzt
accessPath, bytePos, bitPos:	String mit Syntax des Zugriffspfades zu einem Datenobjekt entsprechend der DriveServer Spezifikation
dataType:	String mit Datentyp
specificProperty:	bei DRIVECOM nicht benutzt
uses:	bei DRIVECOM nicht benutzt
processDatItem:	dient dem Zusammensetzen eines Datenobjektes aus mehreren Prozeßdatenobjekten, bei DRIVECOM zunächst nicht benutzt
%id.unique.req;:	eindeutige ID zum Prozeßdatenobjekt
direction:	Angabe der Übertragungsrichtung
signalType:	analog oder digital
processDataItemType:	bei DRIVECOM nicht benutzt
%enabled.def;:	Aktivierung des Objektes, default: YES

### 3.5.2. ParameterItemList

Dieses Element dient zur Beschreibung der vorhandenen Parameterdatenobjekte.

Syntax:

```

<!ELEMENT parameterItemList  (%naming.opt;,
                                (parameterCategory|
                                parameterItem|
                                parameterGroup)+)>
  
```

Die optionale Bezeichnung '%naming.opt;' der Liste ist nicht erforderlich. Es werden lediglich Einträge für die Parameterdaten als 'parameterItem' gefordert.

```
<!ELEMENT parameterItem      (%naming.opt; ,
                               pictureList?,
                               (accessPath|
                                (bytePos,bitPos?)),
                               datatype,
                               specificProperty*,
                               (%values;)?,
                               parameterItem*,
                               instances?)>
<!ATTLIST parameterItem      %id.unique.req;
                               %access.req;
                               parameterItemType CDATA #IMPLIED
                               %enabled.def;>
```

Beschreibung der enthaltenen Elemente siehe 3.5.1 ProcessDataItem.

### 3.6. DeviceFunctionObject

Element 'DeviceFunctionObject' ist in der FDCML-Spezifikation nicht ausgefüllt und steht für die Verwendung innerhalb der DRIVECOM zur Verfügung. Hier werden die DriveServer-spezifischen Objekte beschrieben

Syntax:

```
<!ELEMENT DeviceFunctionObject (dictionaryList?,
                                 helpFileList?,
                                 containerList?,
                                 componentList?,
                                 varTemplateList?,
                                 varList?,
                                 varEnumList?,
                                 menuList?,
                                 methodList?,
                                 redefineList?)>
```

#### 3.6.1. DictionaryList

Eine Liste von Wörterbüchern kann an mehreren Stellen in der XML-Datei angelegt werden, jedoch ist nur diese im 'DeviceFunctionObject' für den DriveServer von Interesse und wird ausgewertet.

Syntax:

```
<!ELEMENT dictionaryList     (dictionary+)>
<!ELEMENT dictionary         (file)>
<!ATTLIST dictionary         %lang.req;
                               dictID CDATA #IMPLIED>
```

Jedes Wörterbuch setzt sich aus einem Verweis auf eine Datei und die von dieser Datei unterstützte Sprache sowie eine 'dictID' zusammen, mit deren Hilfe von anderen Teilen der XML-Datei auf das Wörterbuch referenziert werden kann.

### 3.6.2. HelpFileList

Eine Liste von Hilfedateien kann an mehreren Stellen in der XML-Datei angelegt werden, jedoch ist nur diese im 'DeviceFunctionObject' für den DriveServer von Interesse und wird ausgewertet.

Syntax:

```

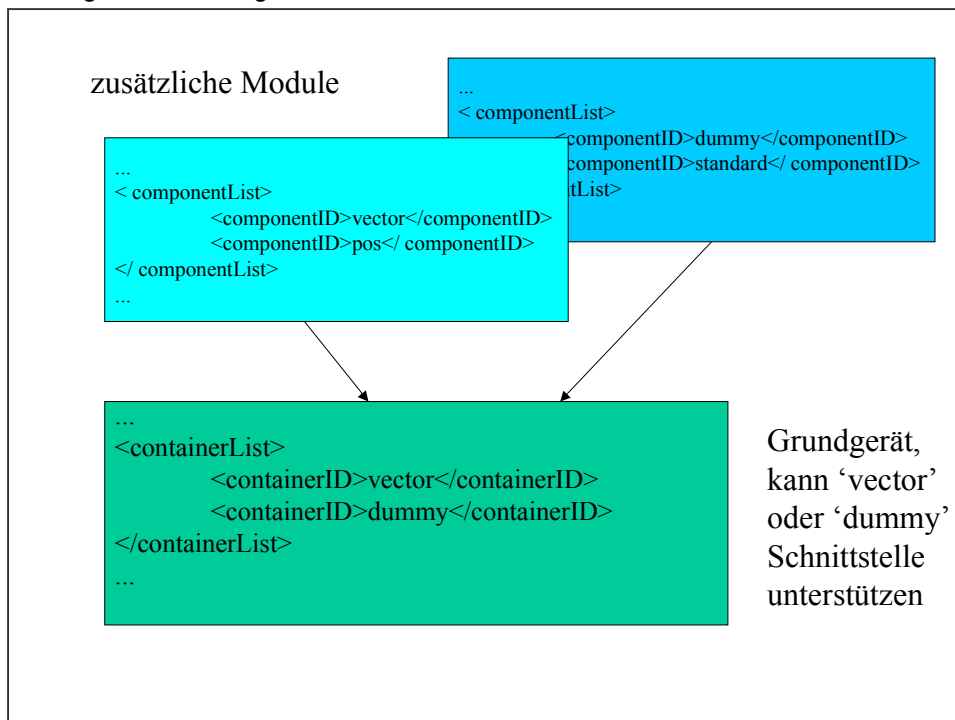
<!ELEMENT helpFileList          (helpFile+)>
<!ELEMENT helpFile              (file)>
<!ATTLIST helpFile              %lang.req;
                                helpFileID CDATA #IMPLIED>

```

Jeder Eintrag setzt sich aus einem Verweis auf eine Datei und die von dieser Datei unterstützte Sprache sowie eine 'helpFileID' zusammen, mit deren Hilfe von anderen Teilen der XML-Datei auf die Hilfedatei referenziert werden kann.

### 3.6.3. containerList, componentList

Die Einträge in 'containerList' und 'componentList' dienen der Zuordnung von Grundgeräten zu möglichen Modulen und vice versa.



**Abbildung 2: Beziehung Grundgerät/Modul**

Abbildung 2 skizziert die Beziehung zwischen Grundgerät und dazu passenden Modultypen. So gibt das Grundgerät beispielsweise bekannt, die (abstrakten) Schnittstellen 'vector' und 'dummy' zu unterstützen. Dies bedeutet, daß Module, die sich mit einer dieser Schnittstellen verbinden können (z.B. durch physisches Aufstecken), damit potentiell in der Lage sind, auf dem Grundgerät 'gesteckt' zu sein. Diese Information ist für den DriveServer während der Identifikation der Antriebe notwendig, so daß ggf. zusätzliche Algorithmen für die Identifizierung der Module ausgeführt werden.



Ein Modul seinerseits kann nun auch mehrere Schnittstellen unterstützen und damit auf mehreren Grundgerätetypen eingesetzt werden

Syntax:

```
<!ELEMENT containerList      (containerID+)>
<!ELEMENT containerID       (#PCDATA)>
<!ELEMENT componentList     (componentID+)>
<!ELEMENT componentID      (#PCDATA)>
```

**containerID:** Containerfähigkeit der Antriebskomponente, herstellerspezifische Kennung der Schnittstelleneigenschaft des Containers. Der Container gibt hiermit seinen Typ bekannt. Mittels des herstellerspezifischen Algorithmus zur Zuordnung Antrieb/Beschreibung kann damit ggf. nach weiteren Komponenten am Antrieb gesucht und diese identifiziert werden.

**componentID:** Komponentenfähigkeit der Antriebskomponente, herstellerspezifische Kennung der Schnittstellenfähigkeit der Komponente. Benutzung siehe oben.

### 3.6.4. varTemplateList

In diesem Abschnitt können Templates definiert werden, von denen die Variable/Parameter bis auf den Namen alle Eigenschaften erben können. Templatenamen müssen in der XML-Datei (und in deren Importen) eindeutig sein.

Weiterhin verfügt jedes Template-Element über das Attribut propID (hier nicht mit dargestellt, siehe 4 Benutzte Property-IDs), das den Property-Wert für das spätere Item-Attribut definiert. Der DriveServer kann diese Definitionen übernehmen und in die Liste der verfügbaren Properties aufnehmen.

Syntax:

```
<!ELEMENT varTemplateList    (varTemplate+)>
<!ELEMENT varTemplate       (classList,
                             type,
                             DFOAccess,
                             limits?,
                             readTimeout?,
                             writeTimeout?,
                             unit?,
                             (help*|helpRef|helpFileRef)?,
                             formatstring?,
                             defaultvalue?,
                             preReadFunc?,
                             postReadFunc?,
                             preWriteFunc?,
                             postWriteFunc?,
                             scalingFactor?,
                             paramsetList?,
```



	<pre>visible?&gt;</pre>
<pre>&lt;!ATTLIST varTemplate</pre>	<pre>%id.unique.req;&gt;</pre>
<b>varTemplate:</b>	Spezifischer Vorlagetyp mit eindeutigem Namen, über den referenziert wird.
<pre>&lt;!ELEMENT classList</pre>	<pre>(class+)&gt;</pre>
<pre>&lt;!ELEMENT class</pre>	<pre>EMPTY&gt;</pre>
<pre>&lt;!ATTLIST class</pre>	<pre>c (DYNAMIC  </pre>
	<pre>LOCAL  </pre>
	<pre>OPERATE  </pre>
	<pre>ALARM) #REQUIRED&gt;</pre>
<b>classList, class:</b>	Hier wird definiert, wie die Variable vom DriveServer benutzt werden kann. Es können mehrere Attribute gesetzt werden. <b>OPERATE:</b> Normaler Parameter, der der Manipulation der Antriebs dient <b>DYNAMIC:</b> Die Variable kann dynamisch vom Gerät aus geändert werden. Der DriveServer kann ggf. diese Werte dann zyklisch vom BusServer abfragen. <b>LOCAL:</b> Variable wird nur local im DriveServer benutzt und wird nicht zum Antrieb kommuniziert <b>ALARM:</b> Variable dient der Alarmsignalisierung
<pre>&lt;!ELEMENT type</pre>	<pre>EMPTY&gt;</pre>
<pre>&lt;!ATTLIST type</pre>	<pre>t (VT_BOOL  </pre>
	<pre>VT_UI1  </pre>
	<pre>VT_I2  </pre>
	<pre>VT_I4  </pre>
	<pre>VT_BSTR  </pre>
	<pre>VT_DATE  </pre>
	<pre>VT_CY  </pre>
	<pre>VT_R4  </pre>
	<pre>VT_R8  </pre>
	<pre>ARRAY_OF_VT_UI1) #REQUIRED</pre>
	<pre>enum (no  </pre>
	<pre>enumerated  </pre>
	<pre>bit_enumerated) "no"</pre>
	<pre>enum_ref IDREF #IMPLIED&gt;</pre>
<b>type:</b>	OPC-Datentyp des Datenpunktes, handelt es sich um einen enum, wird noch die Referenz auf eine Enum-Struktur erwartet. Der type kann in der Variable noch überschrieben werden, um z.B. eine Enumeration hinzuzufügen.
<pre>&lt;!ELEMENT DFOAccess</pre>	<pre>EMPTY&gt;</pre>
<pre>&lt;!ATTLIST DFOAccess</pre>	<pre>%access.req;&gt;</pre>



DFOAccess:	Zugriffsrecht (Lesen ("RO", Schreiben ("WO"), Lesen&Schreiben ("RW"))
<!ELEMENT limits	(minval, maxval)?>
<!ATTLIST limits	funcRef IDREF #IMPLIED>
<!ELEMENT minval	EMPTY>
<!ATTLIST minval	val CDATA #REQUIRED>
<!ELEMENT maxval	EMPTY>
<!ATTLIST maxval	val CDATA #REQUIRED>
limits:	Grenzen der Vorlage/Variable, die entweder eine Konstante des entsprechenden Datentyps ist oder eine Referenz auf eine Methode, die der DriveServer bei der Bereichsüberprüfung aufzurufen hat. Sind keine Bereichsgrenzen angegeben, gelten die Wertebereiche des zugrundeliegenden Datentyps.
<!ELEMENT readTimeout	(#PCDATA) >
<!ELEMENT writeTimeout	(#PCDATA) >
read/writeTimeout:	Zeit in ms, bis ein synchrones Read/Write erfolgreich abgeschlossen sein muß. Bei Remote-Zugriffen muß der DriveServer ggf. eine zusätzliche Berechnung der Verzögerung durchführen.
<!ELEMENT unit	(#PCDATA) >
<!ATTLIST unit	kind (string  proc  varRef) "string" >
unit:	Einheit zur Vorlage/Variable. Die Einheit kann eine Stringkonstante, ein Verweis auf eine Methode oder auf eine Variable sein (nur bei Benutzung des Attributes innerhalb einer Variable).
<!ELEMENT defaultvalue	(#PCDATA) >
defaultvalue:	Wert zur Vorlage/Variable, auf den der DriveServer alle Variablen dieses Typs nach der Konfigurierung setzt.
helpRef:	Referenz auf eine im dictionaryList angegebene Sprachdatei, die über den helpID angesprochen wird. Der Eintrag textID bestimmt den anzuzeigenden Text.
helpFileRef:	Referenz auf eine im helpFileList angegebene Sprachdatei, die über den helpFileID angesprochen wird. Der Eintrag helpID ist ein Einsprungpunkt innerhalb der Hilfe.
<!ELEMENT preReadFunc	EMPTY>
<!ATTLIST preReadFunc ref	IDREF #REQUIRED>
<!ELEMENT postReadFunc	EMPTY>
<!ATTLIST postReadFunc	ref IDREF #REQUIRED>



	<pre>&lt;!ELEMENT preWriteFunc          EMPTY&gt; &lt;!ATTLIST preWriteFunc          ref IDREF #REQUIRED&gt;</pre>
	<pre>&lt;!ELEMENT postWriteFunc         EMPTY&gt; &lt;!ATTLIST postWriteFunc         ref IDREF #REQUIRED&gt;</pre>
X_YFunc:	Referenz auf eine im Abschnitt <code>methodList</code> definierte Methode, die zum angegebenen Zeitpunkt (vor/nach Lesen/Schreiben der Variable durch den DriveServer ausgeführt wird.
	<pre>&lt;!ELEMENT scalingFactor         (#PCDATA) &gt;</pre>
scalingFactor:	Faktor, mit dem der Wert vom Bus Server beim Lesen multipliziert wird, bzw. beim Schreiben dividiert wird.
	<pre>&lt;!ELEMENT formatstring         EMPTY&gt; &lt;!ATTLIST formatstring         str CDATA #REQUIRED</pre>
formatstring:	Formatierung eines Wertes entsprechend der C-Konventionen (z.B. bei printf), die Formatierung kann vom OPC-Client als Property abgefragt werden. Der formatierte String kann auch im DriveServer gebildet werden und kann dann über ein Property abgefragt werden.
	<pre>&lt;!ELEMENT paramsetList         (set*) &gt; &lt;!ELEMENT set                  (#PCDATA) &gt;</pre>
paramsetList, set:	Angabe einer ganzen Zahl, in welchem Datensatz die Variable enthalten ist, paramsetList ist die Aufzählung dieser Parametersätze
	<pre>&lt;!ELEMENT visible              (#PCDATA) &gt;</pre>
visible:	Das Attribut enthält eine ganze Zahl. In Kontextabhängigkeit können damit einzelne Variable ein/ausgeblendet werden. Defaulteinstellung für alle Variable ist der Wert ,0', d.h. die Variable ist im-mer sichtbar. Sichtbar werden Variable, wenn der Kontext in DriveServer einen Wert <code>&lt;= visible</code> hat.

### 3.6.5. Variable

In diesem Abschnitt können alle Variablen/Parameter des Antriebs aufgeführt werden. Die Variablen enthalten stets einen Verweis auf ein `varTemplate`. Von dieser Vorlage werden alle Attribute geerbt. Die einzelnen Attribute (Beschreibung siehe 3.6.4 `varTemplateList`) können dann pro Variable explizit überschrieben werden und gelten dann auch nur für diese.

```
<!ELEMENT varList              (var*) >
<!ELEMENT var                  (%labels;,
                                classList?,
                                type?,
                                uses?,
                                DFOAccess?,
                                limits?,
                                readTimeout?,
```

	<pre>writeTimeout?, unit?, (help* helpRef helpFileRef)?, defaultvalue?, preReadFunc?, postReadFunc?, preWriteFunc?, postWriteFunc?, scalingFactor?, paramsetList?, visible?)&gt; &lt;!ATTLIST var name ID #REQUIRED varTemplate IDREF #REQUIRED&gt;</pre>
var:	Der Eintrag entspricht einem Geräteparameter. Das Attribut <code>name</code> ist die eindeutige Bezeichnung innerhalb der XML-Dateien und im DriveServer. <code>varTemplate</code> ist die Referenz auf die Vorlage.
labels:	anzuweisender Bezeichner der Variable, dies können entweder ein oder mehrere konstante Strings in den verschiedenen Sprachen oder ein Verweis auf einen externen Wörterbucheintrag sein.
uses:	Verweis auf ein Prozeß- oder Parameterdatenobjekt aus dem Abschnitt 'communicationEntity' mit dem Namen "DRIVECOM" (siehe 3.5 DeviceManagerObject).

### 3.6.6. Aufzählungen

Aufzählungen dienen der Zuordnung von Zahlenwerten zu Texten. Eine Aufzählung kann von mehreren Variablen referenziert werden. Die Aufgabe des DriveServers ist es, diese Referenzen aufzulösen. Die Einträge der Aufzählung werden als Properties zu den Items mitgeliefert. Aufgabe des Clients ist es, anstatt des Item-Wertes den entsprechenden Text anzuzeigen.

	<pre>&lt;!ELEMENT varEnumList (enum+)&gt; &lt;!ELEMENT enum (enum_entry+)&gt; &lt;!ATTLIST enum name ID #REQUIRED&gt;</pre>
enum:	Eine Aufzählung mit eindeutigen Namen, über den referenziert wird.
	<pre>&lt;!ELEMENT enumEntry EMPTY&gt; &lt;!ATTLIST enumEntry value CDATA #REQUIRED %labels;&gt;</pre>
enumEntry:	Einzelner Eintrag in einer Aufzählung. Der Text kann entweder als ein oder mehrere konstante Strings in den verschiedenen Sprachen oder als Referenz auf einen Wörterbucheintrag angegeben werden. Je nach Kontext ist <code>value</code> der Zahlenwert der Variable oder die entsprechende Bitposition.

### 3.6.7. Menu

Die Menustruktur dient dem Aufbau des Namensraumes des DriveServers, d.h. die Informationen, die man beim Browsen erhält. Die Menüs können hierarchisch ineinander geschachtelt werden. An den Blättern werden die Variable eingetragen, wenn sie zur Laufzeit/Browsezeit sichtbar sind.

```
<!ELEMENT menuList          (menu+)>
<!ELEMENT menu              (%labels;,
                             visible?,
                             (help*|helpRef|helpFileRef)?,
                             m_entry+)>
<ATTLIST menu                %id.unique.req;>
```

menu:

Ein Menu hat einen eindeutigen Namen innerhalb der XML-Datei. Zusätzlich wird über labels definiert, welcher Text zur Anzeige kommt. Ein Menü kann den Sichtbarkeitsregeln unterzogen werden. Ist dieser Wert nicht angegeben, hat es den Sichtbarkeitswert ,0' und ist immer sichtbar. Ein Verweis auf einen Hilfeeintrag ist optional (erst ab OPC V2.03 einsetzbar).

```
<!ELEMENT m_entry           EMPTY>
<ATTLIST m_entry            kind (var|menu) #REQUIRED
                             ref IDREF #REQUIRED>
```

m\_entry:

Ein einzelner Menu-Eintrag kann entweder eine Variablen- oder eine Menu- Referenz sein.

### 3.6.8. Methoden

Der Abschnitt 'methodList' umfaßt die Methodendefinitionen, auf die aus anderen Bereichen der XML-Datei referenziert wird. Die Methoden sind in Skriptsprachen zu notieren und können vom DriveServer zu den entsprechenden Zeitpunkten ausgeführt werden.

```
<!ELEMENT methodList       (method+)>
<!ELEMENT method           (#PCDATA)>
<ATTLIST method            %id.unique.req;
                             lang (VBSCRIPT|JSCRIPT) "JSCRIPT">
```

method:

Eine Methode wird über ihren Namen eindeutig beschrieben. Zusätzlich kann eine der beiden angegebenen Skriptsprachen spezifiziert werden, in der die Methode geschrieben ist. Die Methode selbst wird in der jeweiligen Skriptsprache in den Elementtext (allerdings als CDATA-Cast) hinterlegt. Jede Methode, die durch Referenzen direkt aufgerufen werden, müssen einen Parameter haben. Dieser Parameter wird zur Laufzeit mit dem Namen der zugehörigen Variable gefüllt.

### 3.6.9. Neudefinitionen

Die Neudefinitionen erlauben die Deklaration von Elementen (Datentypen, Variablen, Aufzählungen, Methoden, Definitionen siehe oben), die bereits im XML-Dokument, auch in eingebundenen Dateien, definiert worden sind. Diese Elemente werden gelöscht und neu angelegt.

```
<!ELEMENT redefineList     (varTemplateList?,
                             varList?,
```

```

enumList?,
menuList?,
methodList?) >
  
```

## 4. Benutzte Property-IDs

In diesem Abschnitt werden alle zusätzlich zur OPC- Spezifikation definierten Property-IDs aufgeführt. Die IDs der DRIVECOM beginnen bei 6000.

ID-Offset	Datentyp des zurückkommenden VARIANT	Beschreibung
0	VT_BSTR	Klasse der Variable ("DYNAMIC", "LOCAL", "OPERATE", "ALARM")
1	VT_ARRAY VT_BSTR	Hilfe zum Parameter bzw. Menueintrag beim Browse, der erste Array-Eintrag ist die URL auf die Datei, der zweite der Einsprungpunkt
2	VT_ARRAY VT_BSTR	Enumeration, wenn der Parameter über eine Aufzählungsdefinition verfügt, steht dieses Array bereit. Es hat so viele Einträge, wie in der Aufzählung definiert. Die Trennung zwischen Wert/Text erfolgt über die Position im String. Der Text beginnt ab der 10. Stelle. Die Zeichen davor bilden den Zahlenwert und können ggf. numerisch gewandelt werden.
3	VT_ARRAY VT_BSTR	Bitenumeration, wenn der Parameter über eine bitorientierte Aufzählungsdefinition verfügt, steht dieses Array bereit. Es hat so viele Einträge, wie in der Aufzählung definiert. Die Trennung zwischen Bitposition/Text erfolgt über die Position im String. Der Text beginnt ab der 10. Stelle. Die Zeichen davor bilden den Zahlenwert und können ggf. numerisch gewandelt werden.
4	VT_I4	Read-Timeout in ms
5	VT_I4	Write-Timeout in ms
6	variiert	Defaultvalue, der Datentyp variiert entsprechend der Definition, entspricht dem kanonischen Datentyp, der unter Property-ID '1' abgefragt werden kann
7	VT_R4	Skalierungsfaktor
8	VT_BSTR	Formatierungsstring entsprechend der "C-printf" Formatierung
9	VT_BSTR	Formatierter aktueller Wert des Parameters entsprechend des Formatierungsstrings
10	VT_I4	Sichtbarkeitswert, ist nur erreichbar, wenn variable auch wirklich sichtbar - und damit im Namensraum definiert - ist
11	VT_BSTR	Zugriffspfad des Parameters im Antrieb, nicht zu verwechseln mit dem kompletten Zugriffspfad eines Items im DriveServer (diesem sind Rounting- und Hierarchieinformationen vorangesetzt). Dieser String wird im DriveServer zur Bildung des Item-Zugriffspfades benutzt.



12	VT_BSTR	Datentyp des Parameters im Antrieb entsprechend FDCML
13	VT_I4	Anzahl der zu übertragenden Bytes

## 5. Wörterbuch (DTD der Sprachdatei)

Die einzelnen Wörterbücher werden sprachbezogen angelegt und können durch die Hersteller frei erstellt werden. Der DriveServer ersetzt beim Laden der Gerätebeschreibungen die Textreferenzen mit den Inhalten aus den Sprachdateien. Ob und welches Wörterbuch zur Laufzeit benutzt wird, hängt von den Einstellungen des DriveServers ab. Die Referenzen auf die Wörterbücher sind in der DriveServer-XML-Konfigurationsdatei (siehe 3.6.1 DictionaryList) hinterlegt.

Die Sprachdateien können nach folgender Syntax angelegt werden:

```
<!ELEMENT DRIVECOM_LANG_V01      (text_entry*)>
<!ATTLIST DRIVECOM_LANG_V01      xml:lang CDATA #REQUIRED>
<!ELEMENT text_entry              EMPTY>
<!ATTLIST text_entry              t_id ID #REQUIRED
                                   text CDATA #REQUIRED>
```

DRIVECOM\_LANG\_V01 xml:lang: Kennzeichnung der Sprache, die das Wörterbuch unterstützt

text\_entry t\_id: String mit ID, auf die in den Gerätebeschreibungen referenziert wird

text\_entry text: String mit dem eigentlichen Textinhalt

## 6. Hilfedateien

Das Hilfesystem wird in diesem Rahmen nicht spezifiziert. Es wird lediglich vorausgesetzt, daß die referenzierten Dateien (\*.hlp, \*.html, \*.pdf, etc.) erreichbar sind. Die eigentlichen Aufrufe muß dann der Client durchführen. Dieser erhält vom DriveServer über die Property-Anfrage zu einem Item die Informationen zur Datei und den Einsprungpunkt.

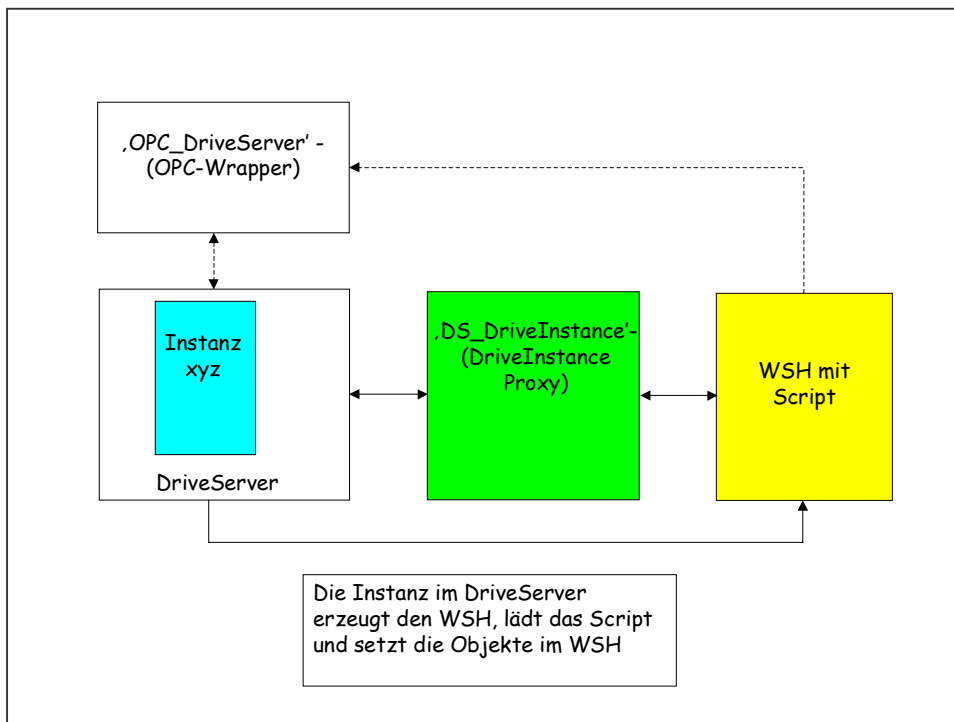
## 7. Interface

### 7.1. Überblick

Der DriveServer führt Skripte in Zusammenarbeit mit der Schnittstelle DS\_DriveInstance aus. Dazu werden vor dem eigentlichen Start des Scriptes zwei Objekte in der Skript- Umgebung vom DriveServer gesetzt. Diese beiden Objekte können im Skript unter folgenden Namen angesprochen werden:

'OPC\_DriveServer': Eine Instanz des OPC-Wrapper-Servers, auf die bereits ein 'Connect' im DriveServer durchgeführt wurde.

'DS\_DriveInstance': Ein Proxy zum Zugriff auf das DS\_DriveInstance Interface.



**Abbildung 3: Datenfluß zwischen Instanz und Skript**

Diese beiden Objekte werden auf Skript- Level Basis gesetzt, so daß sie während der gesamten Ausführung zur Verfügung stehen. Nach der Skriptbearbeitung werden diese Objekte vom DriveServer gelöscht. Weiterhin wird durch den DriveServer in der Skriptumgebung die Variable 'strPath2Drive' gesetzt.

Die Schnittstelle `,'DS_DriveInstance'` ist notwendig, um auch schreibend auf die Properties der Parameter zugreifen zu können. Die OPC- Schnittstelle erlaubt nur das Lesen von diesen Werten.

## 7.2. Funktionsbeschreibung

Die in den Funktionen übergebenen Property-IDs entsprechen der OPC-Spezifikation und den Festlegungen in der Konfigurations-DTD `,'DRIVECOM_EDD_V01'`.

### **GetPropertyValue(varVariableName, varPropId) As Variant**

[in] VARIANT varVariableName: eindeutiger Name der Variable

[in] VARIANT varPropId: Nummer der Property

Die Funktion `,'GetPropertyValue'` liefert den Property-Wert zur angegebenen Variable und zur geforderten Property-ID. Der Variablenname entspricht der eindeutigen Bezeichnung im XML-Dokument.

### **SetPropertyValue(varVariableName, varPropId, pvarPropValue) As Boolean**

[in] VARIANT varVariableName: eindeutiger Name der Variable

[in] VARIANT varPropId: Nummer der Property

[in] VARIANT varPropValue: neuer Wert des Properties

Die Funktion `,'SetPropertyValue'` setzt den Property-Wert zur angegebenen Variable und zur geforderten Property-ID. Der Variablenname entspricht der eindeutigen Bezeichnung im XML-Dokument. Die Funktion kehrt mit TRUE zurück, wenn der neue Wert von varPropValue übernommen wurde.

### 7.3.IDL

```
[
    object,
    uuid(F7A15FA1-9F82-11d4-8DA3-00E07D815C6E),
    dual,
    helpstring("IDS_DriveInterface-Schnittstelle"),
    pointer_default(unique),
    oleautomation
]
interface IDS_DriveInterface : IDispatch
{
    [id(1), helpstring("Methode GetPropertyValue")]
    HRESULT GetPropertyValue ([in] VARIANT varVariableName,
                              [in] VARIANT varPropId,
                              [out, retval] VARIANT* pvarPropValue);

    [id(2), helpstring("Methode SetPropertyValue")]
    HRESULT SetPropertyValue ([in] VARIANT varVariableName,
                              [in] VARIANT varPropId,
                              [in] VARIANT varPropValue,
                              [out, retval] VARIANT_BOOL* pbResult);
}
```

## 8. Syntax der Sprachen-DTD

```
<!-- DRIVECOM Sprachdatei -->
<!ELEMENT DRIVECOM_LANG_V01 (text_entry*)>
<!ATTLIST DRIVECOM_LANG_V01 xml:lang CDATA #REQUIRED>

<!-- Texteintrag -->
<!ELEMENT text_entry EMPTY>
<!-- Text-ID zum Auffinden des Texteintrags, referenziert aus XML-Datei -->
<!ATTLIST text_entry t_id ID #REQUIRED
                    text CDATA #REQUIRED>
```