HomeMatic-Script Dokumentation

Teil 2: Objektmodell

V1.2

Inhaltsverzeichnis

1 Einleitung

Mit der HomeMatic Zentrale ist es möglich, als Reaktion auf ein Ereignis ein Skript auszuführen. Hier wird eine eigene Skriptsprache verwendet, die im Folgenden als HomeMatic-Script bezeichnet wird.

HomeMatic-Script wird bei der HomeMatic Zentrale innerhalb von Programmen verwendet. Mit dieser Skriptsprache kann auf die Logikschicht der HomeMatic Zentrale zugegriffen werden. Dadurch lassen sich z.B. angelernte HomeMatic Komponenten steuern.

Ziel der HomeMatic-Script Dokumentation ist es, dem Anwender einen Einblick in die Programmierung von Skripten zu geben, die mittels Programmen von der HomeMatic Zentrale benutzt werden können.

Die HomeMatic-Script Dokumentation besteht aus den folgenden 4 Dokumenten:

- Teil 1: Sprachbeschreibung

  Beschreibt die Skriptsprache HomeMatic-Script.

- Teil 2: Objektmodell

  Beschreibt auszugsweise das Objektmodell, welches der HomeMatic Zentrale zugrunde liegt.

- Teil 3: Beispiele

  Eine Sammlung von Beispielen, welche den Umgang mit HomeMatic-Script praxisnah illustrieren.

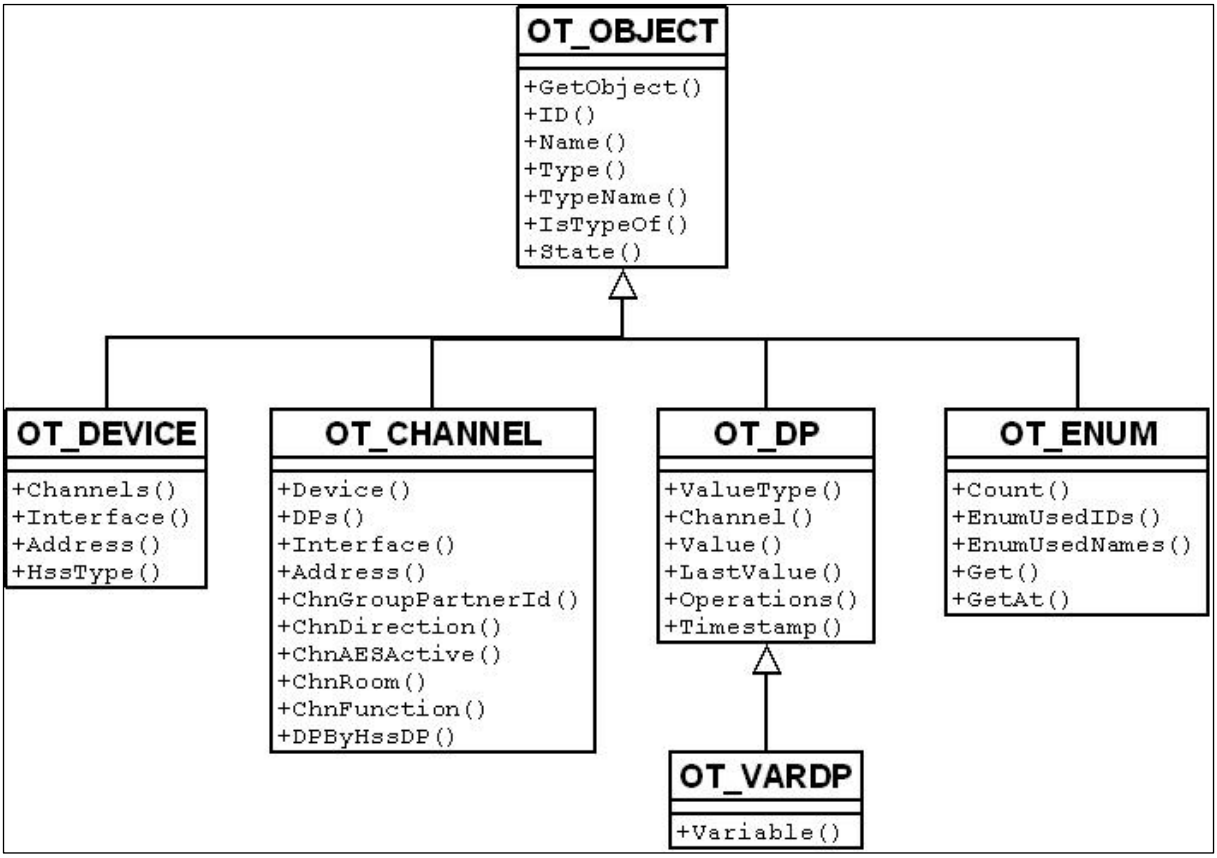
- Teil 4: Datenpunkte

  Gibt einen Überblick über die verfügbaren Datenpunkte der HomeMatic Geräte.

Bei diesem Dokument handelt es sich um Teil 2: das HomeMatic Objektmodell.

Das Dokument beschreibt das Objektmodell, auf das per HomeMatic Script zugegriffen werden kann. Damit ist es möglich, den Zustand von Kanälen und Systemvariablen abzufragen bzw. zu manipulieren. Kenntnisse vom Objektmodell gehören zur Grundlage für eine effiziente Skriptprogrammierung.

2 Übersicht der Objekthierarchie



3 Der Namensraum „system“

Unter dem Namensraum „system“ sind allgemeine Systemfunktionen zusammengefasst. So lässt sich z.B. die aktuelle Uhrzeit abfragen oder prüfen, ob eine Variable existiert.

Funktionsübersicht:

Name	Prototyp	Kurzbeschreibung
System.Date	string system.Date(string format)	Fragt die aktuelle Uhrzeit ab
System.IsVar	boolean system.IsVar(string name)	Prüft, ob eine Variable definiert ist
System.GetVar	var system.GetVar(string name)	Ermittelt den Wert einer Variable

3.1 Uhrzeit Abfragen

string system.Date(string format)

Ermittelt das aktuelle Systemdatum. Die Ausgabe erfolgt als Zeichenkette, deren Aufbau durch den Parameter „format“ festgelegt wird.

3.1.1 Parameter

Der Parameter „format“ legt den Aufbau der Ausgabe fest. Dabei werden Platzhalter für entsprechende Komponenten des Systemdatums verwendet.

Platzhalter	Kurzbeschreibung	Beispiel (24.12.2008 18:30:00)
%%	Das Prozent-Zeichen	„%“
%a	Abgekürzter Wochentagsname	Wed
%A	Vollständiger Wochentagsname	Wednesday
%b	Abgekürzter Monatsname	Dec
%B	Vollständiger Monatsname	December
%c	Datum und Uhrzeit	Wed Dec 18:30:00 2008
%C	Jahrhundert – 1	20
%d	Monatstag	24
%D	Datum %m/%d/%y	12/24/08
%F	Datum %Y-%m-%d	2008-12-24
%h	Abgekürzter Monatsname	Dec
%H	Stunde (24-Stunden-Uhr)	18
%I	Stunde (12-Stunden-Uhr)	06
%j	Nummer des Tages im Jahr	359
%m	Monatsnummer	12

%M	Minute	30
%n	Neue-Zeile-Steuerzeichen	
%p	AM bzw. PM	PM
%r	Uhrzeit (12-Stunden-Uhr)	06:30:00 PM
%S	Sekunde	00
%t	Tabulator-Steuerzeichen	
%T	Uhrzeit (24-Stunden-Uhr)	18:30:00
%u	Wochentag (Montag = 1)	3
%U	Wochennummer (Woche 1 ab dem 1. Sonntag im Januar)	51
%V	Wochennummer (ISO 8601)	52
%w	Wochentag (Sonntag = 0)	3
%W	Wochennummer (Woche 1 ab dem 1. Montag im Januar)	51
%x	Datum	12/24/08
%X	Uhrzeit	18:30:00
%y	Jahreszahl (2 Ziffern)	08
%Y	Jahreszahl (4 Ziffern)	2008
%z	Zeitabstand zu GMT	+0100
%Z	Name der Zeitzone	CET

3.1.2 Rückgabe

Aktuelles Systemdatum.

3.1.3 Beispiele

```
string sDate = system.Date("%d.%m.%Y"); ! sDate = "24.12.2008";
string sTime = system.Date("%H:%M:%S"); ! sTime = "18:30:00";
```

3.2 Existenz einer Variablen prüfen

```
boolean system.IsVar(string name)
```

Prüft, ob eine Variable mit dem angegebenen Namen existiert.

3.2.1 Parameter

Name der Variable als Zeichenkette.

3.2.2 Rückgabewert

„true“, falls eine Variable mit dem angegebenen Namen existiert, ansonsten „false“.

3.2.3 Beispiel

```
var MY_DEFINE = 1;

if (system.IsVar("MY_DEFINE"))
{
  ! MY_DEFINE ist definiert
}
```

3.3 Wert einer Variable ermitteln

```
var system.GetVar(string name)
```

Ermittelt den Wert einer Variablen.

Achtung!

Falls direkt auf eine Variable zugegriffen wird, die nicht definiert ist, wird das komplette Skript **nicht** ausgeführt. Wenn nicht sicher ist, ob eine Variable definiert ist, sollte auf deren Wert immer über „system.GetVar()“ zugegriffen werden.

3.3.1 Parameter

Name der Variable als Zeichenkette.

3.3.2 Rückgabewert

Wert der Variable oder „null“.

3.3.3 Beispiel

Das folgende Skript ist fehlerhaft. Falls „myVar“ nicht definiert ist, wird das Skript nicht ausgeführt.

```
! ACHTUNG: Das folgende Skript wird NICHT ausgeführt
if (system.GetVar("myVar"))
{
  myVar = myVar + 1; ! Fehler: myVar existiert nicht
}
```

Das nachstehende Skript demonstriert die Funktionsweise von „system.GetVar()“. Auf die Variable „myVar“ wird nicht direkt zugegriffen.

```
var myVar = 1;

if (system.IsVar("myVar"))
{
  var x = system.GetVar("myVar");
  x = x + 1;
}

! myVar = 1
! x     = 2
```

4 Allgemeine Objekte

Jedes Objekt ist vom Datentyp „OT_OBJECT“ abgeleitet. Schon dieser Objekttyp definiert eine Menge allgemeiner Methoden, die auch in den abgeleiteten Objekten zur Verfügung stehen.

4.1 Methodenübersicht

Name	Prototyp	Kurzbeschreibung
GetObject	var object.GetObject(integer id) var object GetObject(string name)	Liefert ein Objekt anhand seiner ID bzw. seines Namens
ID	integer object.ID()	Liefert die ID eines Objekts
Name	string object.Name()	Liefert den Namen eines Objekts
Type	integer object.Type()	Liefert die ID des Objekttyps
TypeName	string object.TypeName()	Liefert die Bezeichnung des Objekttyps
IsTypeOf	boolean object.IsTypeOf(integer typeId)	Prüft, ob ein Objekt einen speziellen Typ implementiert
State	var object.State() boolean object.State(boolean newState) boolean object.State(integer newState) boolean object.State(real newState) boolean object.State(time newState) boolean object.State(string newState)	Ermittelt oder setzt den Zustand eines Objekts

4.2 GetObject

```
var object.GetObject(integer id)
var object.GetObject(string name)
```

Die Methode „GetObject“ liefert ein Objekt wahlweise anhand seiner ID oder seines Namens.

4.2.1 Parameter

Die Methode „GetObject“ hat zwei Formen: entweder wird die Objekt-ID als Ganzzahl übergeben oder es wird der Name des Objekts übergeben.

Achtung!

Die HomeMatic Zentrale erlaubt, dass Objekte verschiedener Typen dieselbe Bezeichnung tragen. So kann man ein Gerät mit dem Namen „Funk-Kombisensor“ anlegen und dessen Kanal ebenfalls „Funk-Kombisensor“ nennen.

Die Methode „GetObject“ liefert stets das erste gefundene Objekt des angegebenen Namens zurück. Um Probleme zu vermeiden, sollte man Objekte mit eindeutigen Namen bezeichnen.

4.2.2 Rückgabewert

Der Rückgabewert ist entweder eine Objektreferenz oder „null“ (falls das gesuchte Objekt nicht gefunden wurde).

4.2.3 Beispiel

Grundsätzlich funktioniert die Methode „GetObject“ mit jedem Objekt. Da sich häufig kein Objekt im Zugriff befindet, wird in der Praxis das globale Objekt „dom“ verwendet, um auf Objekte zuzugreifen:

```
var myObject = dom.GetObject("MyObject");
if (myObject)
{
  ! myObject ist gültig
}
```

4.3 ID

```
integer object.ID()
```

Liefert die ID eines Objekts.

Die ID ist für jedes Objekt innerhalb des Objektmodells eindeutig.

4.4 Name

```
string object.Name()
```

Liefert den Namen des Objekts.

Bei Geräten, Kanäle, Systemvariablen, Räumen, Gewerken und Favoriten wird der Name vom Anwender vergeben. Innerhalb eines Objekttyps müssen Namen eindeutig sein, d.h. es dürfen z.B. keine zwei Geräte denselben Namen tragen. Dagegen können Objekte verschiedener Typen durchaus denselben Namen tragen.

4.5 Type

```
integer object.Type()
```

Liefert die ID des Objektstyps.

Eine Liste der definierten Objekttypen ist unter 4.7 bei der Methode „IsTypeOf“ zu finden.

4.6 TypeName

```
string object.TypeName()
```

Liefert den Namen des Objekttyps.

4.7 IsTypeOf

```
boolean object.IsTypeOf(intger typeId)
```

Prüft ob ein Objekt einen bestimmten Typ implementiert.

4.7.1 Parameter

Als Parameter wird die ID des zu prüfenden Objekttyps angegeben. Folgende Typen stehen zur Verfügung:

Objekttyp (Konstante)	Kurzbeschreibung
OT_OBJECT	Objekt
OT_ENUM	Aufzählung
OT_DEVICE	Gerät
OT_CHANNEL	Kanal
OT_DP	Datenpunkt
OT_VARDP	Systemvariable

4.7.2 Rückgabewert

Falls das Objekt den angegebenen Typ implementiert, wird „true“ zurückgegeben, ansonsten „false“.

4.8 State

```
var object.State()
boolean object.State(boolean newState)
boolean object.State(integer newState)
boolean object.State(real newState)
```

```
boolean object.State(time newState)
boolean object.State(string newState)
```

Ermittelt oder setzt den Zustand eines Objekts.

Die Semantik der Methode „State“ hängt stark vom tatsächlichen Objekttyp ab. Häufig wird State bei Datenpunkten ausgeführt, um auf diese Weise Aktionen auszulösen.

4.8.1 Parameter und Rückgabewert

Wird kein Parameter übergeben, liefert State den aktuellen Zustand des Objekts in Form einer Zeichenkette.

Abhängig vom Typ des Objekts und dessen Konfiguration hängt ab, welche Parameter für die State-Methode gültig sind. Wird ein Parameter übergeben, liefert der Rückgabewert Aufschluss darüber, ob der Aufruf erfolgreich war (true) oder nicht (false).

4.8.2 Beispiel

Bei dem Kanal „MySwitch“ handelt es sich um den einzigen Kanal eines Funk-Zwischenstecker-Schaltaktors vom Typ „HM-LC-Sw1-Pl“.

Das folgende Skript schaltet den Kanal ein und liefert in der Variable „state“ den aktuellen Zustand des Aktors:

```
var switch = dom.GetObject("MySwitch");
switch.State(1);
var state  = switch.State();            ! state := 1
```

5 Geräte

Ein Gerät besitzt eine Menge von Kanälen, welche die Funktionalität des Geräts ausmachen. Man kann sich Geräte als eine Art Container für Kanäle vorstellen.

5.1 Methodenübersicht

Name	Prototyp	Kurzbeschreibung
Channels	object device.Channels()	Liefert die Liste der Kanäle in dem Gerät
Interface	integer device.Interface()	Liefert die ID der Schnittstelle, an der das Gerät angeschlossen ist
Address	string device.Address()	Liefert die Seriennummer des Geräts
HssType	string device.HssType()	Liefert die Kurzbezeichnung des HomeMatic Gerätetyps

5.2 Channels

```
object device.Channels()
```

Liefert die Liste aller Kanäle, die das Gerät besitzt. Bei der Liste handelt es sich um eine Aufzählung.

Beispiel:

```
var ccu  = dom.GetObject("HM-CCU System");
var chns = ccu.Channels().EnumUsedNames();

! chns = "Zentralennetzteil\tSabotagekontakt"
```

5.3 Interface

```
integer device.Interface()
```

Liefert die ID der Schnittstelle, über die das Gerät angeschlossen ist.

Die HomeMatic Zentrale verfügt derzeit über drei Schnittstellen:

- BidCos-RF für Funkkomponenten
- BidCos-Wired für drahtgebundene Geräte
- System für interne Geräte

Beispiel:

```
var ccu           = dom.GetObject("HM-CCU System");
var interface     = dom.GetObject(ccu.Interface());
var interfaceName = interface.Name();

! interfaceName = "System"
```

5.4 Address

```
string device.Address()
```

Ermittelt die Seriennummer eines HomeMatic Geräts.

Beispiel:

```
var myDevice = dom.GetObject("MyDevice");
var serial   = myDevice.Address();

! serial = "ABC1234567"
```

5.5 HssType

```
string device.HssType()
```

Ermittelt die Bezeichnung des HomeMatic Gerätetyps.

Beispiel:

```
var ccu    = dom.GetObject("HM-CCU System");
var hmType = ccu.HssType();

! hmType = "HM-CCU-1"
```

6 Kanäle

In Kanälen steckt die eigentliche Funktionalität eines HomeMatic Geräts. Der Zustand eines Kanals kann über die Methode „State“ gelesen und manipuliert werden. Darüber hinaus liefern Kanäle jedoch auch weitere interessante Informationen.

Jeder Kanal besitzt eine Liste von Datenpunkten. Über diese lassen sich differenzierte Steuerungen realisieren bzw. detaillierte Zustandsinformationen abfragen. Welche Datenpunkte von einem konkreten Kanal unterstützt werden und was diese bedeuten ist in großem Maße vom Gerätetyp abhängig.

6.1 Methodenübersicht

Name	Prototyp	Kurzbeschreibung
Device	integer channel.Device()	Liefert die ID des Geräts, in dem der Kanal definiert ist
DPs	object channel.DPs()	Liefert eine Liste der Datenpunkte des Kanals
Interface	integer channel.Interface()	Liefert die ID der Schnittstelle, über die der Kanal angeschlossen ist
Address	string channel.Address()	Liefert die Seriennummer des Kanals
ChnGroupPartnerId	integer channel.ChnGroupPartnerId()	Liefert die ID des Partners in einer Kanalgruppe
ChnDirection	integer channel.ChnDirection()	Ermittelt die Kategorie des Kanals
ChnAESActive	boolean channel.ChnAESActive()	Ermittelt, ob der Kanal AES-verschlüsselt sendet
ChnArchive	Boolean channel.ChnArchive()	Ermittelt, ob der Kanal protokolliert wird
ChnRoom	string channel.ChnRoom()	Ermittelt die Räume, denen der Kanal zugeordnet ist
ChnFunction	string channel.ChnFunction()	Ermittelt die Gewerke, denen der Kanal zugeordnet ist
DPByHssDP	object channel.DPByHssDP(string name)	Ermittelt einen Datenpunkt des Kanals anhand seines Namens

Anmerkung zu den Beispielen

In den nachfolgenden Beispielen wird ein Kanal „MyChannel“ verwendet. Dabei handelt es sich um den ersten Kanal eines 4-Tasten-Funk-Handsenders vom Typ „HM-RC-4“.

6.2 Device

```
integer channel.Device()
```

Ermittelt die ID des Geräts, zu dem der Kanal gehört.

Beispiel:

```
var myChannel = dom.GetObject("MyChannel");
var myDevice  = dom.GetObject(myChannel.Device());  ! myDevice = MyDevice
```

6.3 DPs

```
object channel.DPs()
```

Ermittelt die Datenpunkte, die einen Kanal besitzen.

Beispiel:

```
var myChannel = dom.GetObject("MyChannel");
var dps       = myChannel.DPs().EnumUsedNames();

! dps = "BidCos-RF.ABC1234567:1.PRESS_SHORT\t
!       BidCos-RF.ABC1234567:1.PRESS_LONG"
```

6.4 Interface

```
integer channel.Interface()
```

Liefert die ID der Schnittstelle, an die der Kanal angeschlossen ist.

Beispiel:

```
var myChannel = dom.GetObject("MyChannel");
var iface     = dom.GetObject(myChannel.Interface());

! iface.Name() = "BidCos-RF";
```

6.5 Address

```
string channel.Address()
```

Liefert die Seriennummer des Kanals.

Beispiel:

```
var myChannel = dom.GetObject("MyChannel");
var serial    = myChannel.Address();         ! serial = "ABC1234567:1"
```

6.6 ChnGroupPartnerId

```
integer channel.ChnGroupPartnerId()
```

Liefert die ID des Partners in der Kanalgruppe oder 65535, falls der Kanal keiner Gruppe zugeordnet ist.

Beispiel:

```
var myChannel = dom.GetObject("MyChannel");
var partner   = dom.GetObject(myChannel.ChnGroupPartnerId());
var serial    = partner.Address(); ! serial = "ABC1234567:2"
```

6.7 ChnDirection

```
integer channel.Direction()
```

Liefert die Kategorie des Kanals.

Konstante	Kurzbeschreibung
1	Sender (Sensor)
2	Empfänger (Aktor)
0	Kanal ist nicht verknüpfbar

Beispiel:

```
var myChannel = dom.GetChannel("MyChannel");
var direction = myChannel.Direction();       ! direction = 1
```

6.8 ChnAESActive

```
boolean channel.AESActive()
```

Ermittelt, ob der Kanal AES-verschlüsselt sendet.

Beispiel:

```
var myChannel = dom.GetObject("MyChannel");
var aesActive = myChannel.ChnAESActive();    ! aesActive = true
```

6.9 ChnArchive

```
boolean channel.ChnArchive()
```

Ermittelt, ob der Kanal protokolliert wird.

Beispiel:

```
var myChannel = dom.GetObject("MyChannel");
var archive   = myChannel.ChnArchive();      ! archive = false
```

6.10 ChnRoom

```
string channel.ChnRoom()
```

Ermittelt die Räume, denen der Kanal zugeordnet ist. Der Rückgabewert ist eine Liste von IDs, die von der „foreach“-Schleife verwendet werden kann.

Beispiel:

```
var myChannel = dom.GetObject("MyChannel");
var rooms     = "";

string id;
foreach(id, myChannel.ChnRoom())
{
  var room = dom.GetObject(id);
  rooms = rooms # room.Name() # " ";
}

! rooms = "Terrasse";
```

6.11 ChnFunction

```
string channel.ChnFunction()
```

Ermittelt die Gewerke, denen der Kanal zugeordnet ist. Der Rückgabewert ist eine Liste von IDs, die von der „foreach“-Schleife verwendet werden kann.

Beispiel:

```
var myChannel = dom.GetObject("MyChannel");
var funcs     = "";

string id;
foreach(id, myChannel.ChnFunction())
{
  var func = dom.GetObject(id);
  funcs= funcs # func.Name() # " ";
}

! funcs = "Taster";
```

6.12 DPByHssDP

```
object channel.DPByHssDP(string name)
```

Ermittelt einen Datenpunkt anhand seines Namens.

Beispiel:

```
var myChannel = dom.GetObject("MyChannel");
var dp        = myChannel.DPByHssDP("PRESS_SHORT");
dp.State(1);  ! Kurzer Tastendruck
```

7 Datenpunkte

Datenpunkte modellieren u.a. den Zustand eines Kanals. Über Datenpunkte kann man detaillierte Informationen über einen Kanal einholen bzw. den Zustand des Kanals manipulieren.

7.1 Methodenübersicht

Name	Prototyp	Kurzbezeichnung
ValueType	integer dp.ValueType()	Ermittelt den Datentyp des Wertes, den der Datenpunkt repräsentiert.
Channel	integer dp.Channel()	Liefert die ID des Kanals, zu dem der Datenpunkt gehört
Value	var dp.Value()	Liefert den aktuellen Wert des Datenpunktes
LastValue	var dp.LastValue()	Liefert den Wert des Datenpunktes vor der letzten Aktualisierung
Operations	integer dp.Operations()	Ermittelt, welche Operationen auf dem Datenpunkt ausgeführt werden können
Timestamp	time dp.Timestamp()	Zeitstempel der letzten Aktualisierung

Anmerkung zu den Beispielen

Der Datenpunkt „BidCos-RF.ABC1234567:1.PRESS_SHORT“ wird in den nachfolgenden Beispielen verwendet. Dabei handelt es sich um den kurzen Tastendruck eines Kanals „MyChannel“, erster Kanal eines 4-Tasten Funk-Handsenders vom Typ „HM-RC-4“.

7.2 ValueType

```
integer dp.ValueType()
```

Liefert den Datentyp des Wertes, den der Datenpunkt repräsentiert.

Beispiel:

```
var dp        = dom.GetObject("BidCos-RF.ABC1234567:1.PRESS_SHORT");
var valueType = dp.ValueType(); ! valueType = 2
```

7.3 Channel

```
integer dp.Channel()
```

Liefert die ID des Kanals, zu dem der Datenpunkt gehört.

Beispiel:

```
var dp  = dom.GetObject("BidCos-RF.ABC1234567:1.PRESS_SHORT");
var chn = dom.GetObject(dp.Channel()); ! chn = MyChannel
```

7.4 Value

```
var dp.Value()
```

Liefert den aktuellen Wert des Datenpunktes. Der Wert, der mit „Value()“ gelesen wird, entspricht dem letzten Kenntnisstand der HomeMatic Zentrale. Im Gegensatz dazu fragt die Methode „State()“ den aktuellen Wert direkt bei dem Gerät ab, sofern es sich im Zugriff befindet.

Beispiel:

```
var dp    = dom.GetObject("BidCos-RF.ABC1234567:1.PRESS_SHORT");
var value = dp.Value(); ! value = false
```

7.5 LastValue

```
var dp.LastValue()
```

Liefert den Wert des Datenpunktes vor der letzten Aktualisierung.

Beispiel:

```
var dp    = dom.GetObject("BidCos-RF.ABC1234567:1.PRESS_SHORT");
var value = dp.LastValue(); ! value = true
```

7.6 Operations

```
integer dp.Operations()
```

Ermittelt, welche Operationen auf dem Datenpunkt anwendbar sind. Bei dem Rückgabewert handelt es sich um ein Flag-Feld, d.h. ein Datenpunkt kann mehrere Operationen unterstützen.

Konstante	Kurzbeschreibung
OPERATION_READ	Der Datenpunkt kann gelesen werden
OPERATION_WRITE	Der Datenpunkt kann geschrieben werden
OPERATION_EVENT	Der Datenpunkt löst Ereignisse aus

Kann wenigstens ein Datenpunkt eines Kanals geschrieben werden, so gilt der Kanal als bedienbar.

Beispiel:

```
var myChannel    = dom.GetObject("MyChannel");
boolean writable = false;

string id;
foreach(id, myChannel.DPs().EnumUsedIDs())
{
  var dp = dom.GetObject(id);
  if (OPERATION_WRITE & dp.Operations()) { writable = true; }
}

! writable = true
```

7.7 Timestamp

```
time dp.Timestamp()
```

Liefert den Zeitpunkt der letzten Aktualisierung.

Beispiel:

```
var dp  = dom.GetObject("BidCos-RF.ABC1234567:1.PRESS_SHORT");
var ts  = dp.Timestamp();  ! ts.ToString() = "2008-12-24 18:30:00"
```

8 Systemvariable

Bei Systemvariablen handelt es sich aus technischer Sicht um spezielle Datenpunkte. Dabei müssen Systemvariable jedoch nicht zwingend einem Kanal zugeordnet sein.

8.1 Methodenübersicht

Name	Prototyp	Beschreibung
Variable	string dp.Variable() boolean dp.Variable(var value)	Ermittelt oder setzt den Wert einer Systemvariablen.

8.2 Variable

```
string dp.Variable()
boolean dp.Variable(var value)
```

Ermittelt oder setzt den Wert einer Systemvariablen.

8.2.1 Parameter und Rückgabewert

In der parameterlosen Form ermittelt „dp.Variable()“ den aktuellen Wert einer Systemvariablen. Es gilt zu beachten, dass der Wert immer als Zeichenkette zurückgegeben wird, unabhängig vom tatsächlichen Datentyp der Variable.

Wird dagegen ein Parameter angegeben, setzt „dp.Variable()“ den Wert einer Systemvariablen. Der Rückgabewert gibt Aufschluss darüber, ob das Setzen erfolgreich war (true) oder nicht (false).

Beispiel:

```
var mySysVar = dom.GetObject("MySysVar");
var value;

mySysVar.Variable(1);
value = mySysVar.Variable(); ! value = "1", value.VarType() = 4
```

9 Aufzählungen

Innerhalb des HomeMatic Objektmodells werden verschiedene Aufzählungen verwendet. Eine Aufzählung ist dabei eine Sammlung von Objekten. Die folgenden Aufzählungen werden verwendet:

- Kanäle
  Jedes Gerät enthält eine Liste mit Kanälen.
- Datenpunkte
  Jeder Kanal enthält eine Liste mit Datenpunkten.
- Räume
  Jeder Raum stellt eine Liste von Kanälen dar, die dem Raum zugeordnet sind.
- Gewerke
  Jedes Gewerk stellt eine Liste von Kanälen dar, die dem Gewerk zugeordnet sind.
- Favoriten
  Jeder Favorit stellt eine Liste von Objekten dar. Bei den Objekten kann es sich um Kanäle, Programm, Systemvariable und Separatoren handeln.

9.1 Methodenübersicht

Name	Prototyp	Kurzbeschreibung
Count	integer enum.Count()	Liefert die Anzahl der Objekte in der Aufzählung
EnumUsedIDs	string enum.EnumUsedIDs()	Liefert die IDs der Objekte in der Aufzählung
EnumUsedNames	string enum.EnumUsedNames()	Liefert die Namen der Objekte in der Aufzählung
Get	object enum.Get(string name)	Liefert das Objekt mit dem angegebenen Namen
GetAt	object enum.GetAt(integer index)	Liefert das Objekt mit dem angegebenen Index

Anmerkung zu den Beispielen

Die nachfolgenden Beispiele verwenden einen Raum „Bad“. Dieser beinhaltet zwei Kanäle: „Bad.Deckenleuchte“ und „Bad.Fenster“.

9.2 Count

```
integer enum.Count()
```

Liefert die Anzahl der Objekte in der Aufzählung.

Beispiel:

```
var bad   = dom.GetObject("Bad");
var count = bad.Count()          ! count = 2
```

9.3 EnumUsedIDs

```
string enum.EnumUsedIDs()
```

Liefert die IDs der Objekte in der Aufzählung, so dass sie von der „foreach“-Schleife verwendet werden können.

Beispiel:

```
var bad  = dom.GetObject("Bad");
var chns = bad.EnumUsedIDs();
var list = "";

string id;
foreach(id, chns)
{
  var chn = dom.GetObject(id);
  list = list # chn.Name() # " ";
}

! list = "Bad.Deckenleuchte Bad.Fenster "
```

9.4 EnumUsedNames

```
string enum.EnumUsedNames()
```

Liefert die Namen der Objekte in der Aufzählung, so dass sie von der „foreach“-Schleife verwendet werden können.

Beispiel:

```
var bad  = dom.GetObject("Bad");
var chns = bad.EnumUsedNames();   ! chns = "Bad.Deckenleuchte\tBad.Fenster"
var list = "";

string name;
```

```
foreach(name, chns)
{
  list = list # name # " ";
}

! list = "Bad.Deckenleuchte Bad.Fenster "
```

9.5 Get

```
object enum.Get(string name)
```

Liefert das Objekt mit dem angegebenen Namen oder „null“, falls kein Objekt gefunden wurde.

Beispiel:

```
var bad = dom.GetObject("Bad");
var chn = bad.Get("Bad.Deckenleuchte");  ! chn = Bad.Dechenleuchte
```

9.6 GetAt

```
object enum.GetAt(integer index)
```

Liefert das Objekt mit dem angegebenen Index oder „null“, falls kein Objekt gefunden wurde.

Beispiel:

```
var bad = dom.GetObject("Bad");
var chn = bad.GetAt(0);          ! chn = Bad.Deckenleuchte
```


eQ-3 AG
Maiburger Straße 29
D-26789 Leer
www.eQ-3.com