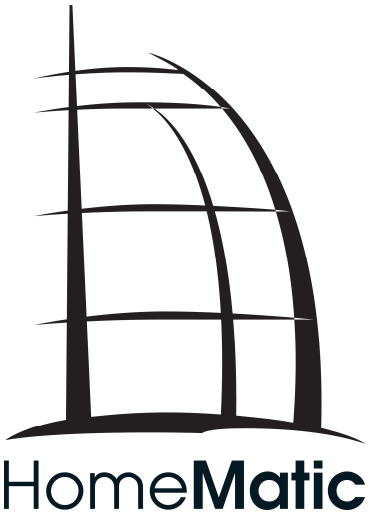

HomeMatic XML-RPC-Schnittstelle

Spezifikation

V1.502

Inhaltsverzeichnis

1 Allgemeines

Dieses Dokument beschreibt eine universelle Programmierschnittstelle zu einem Schnittstellenprozess, über den sich Aktoren und Sensoren in der Hausautomation ansprechen lassen. Es wird davon ausgegangen, dass es einen logikverarbeitenden Prozeß („Logikschicht“) gibt, der sich beliebig vieler Schnittstellenprozesse bedient, um die über unterschiedliche Übertragungssysteme angebundenen Geräte anzusprechen.

Es soll möglich sein, dass der Logikschicht beliebig viele Schnittstellenprozesse bekannt sind. Jeder Schnittstellenprozess wird dazu über eine eindeutige String-Id identifiziert.

Für die Kommunikation zwischen Logikschicht und den Schnittstellenprozessen wird XmlRpc (http://www.xmlrpc.org) verwendet.

Der Schnittstellenprozess kennt nur logische Geräte. Jeder Kanal eines Mehrfachgerätes sowie das Mehrfachgerät selbst werden als logische Geräte mit jeweils eigener Adresse betrachtet. Jedem logischen Gerät sind mehrere von folgenden möglichen Parameter-Sets zugeordnet:

- MASTER
  Dieses Parameter-Set enthält die für das logische Gerät möglichen Konfigurationseinstellungen
- VALUES
  Dieses Parameter-Set enthält die für das logische Gerät gültigen dynamischen Werte (Schaltzustand, Dimmerhelligkeit, Tastendruck-Ereignisse)
- LINK
  Von diesen Parameter-Sets sind meist mehrere vorhanden. Jedes beschreibt die Kommunikationsbeziehung zu einem anderen Gerät am selben Bus.

Beispiel:

Für ein Multifunktionsgerät (Adresse ABC1234567) mit zwei Schaltausgängen und zwei Tastereingängen werden folgende logische Geräte erzeugt:

- „ABC1234567“ für das Gerät. Parameter-Set „MASTER“ ist vorhanden.
- „ABC1234567:0“ für den ersten Schaltausgang. Parameter-Sets „MASTER“ und „VALUES“ sind vorhanden.
- „ABC1234567:1“ für den zweiten Schaltausgang. Parameter-Sets „MASTER“ und „VALUES“ sind vorhanden.
- „ABC1234567:2“ für den ersten Tastereingang. Parameter-Sets „MASTER“, „VALUES“ und „LINK“ sind vorhanden. „LINK“-Parametersets beschreiben jeweils einen durch diesen Tastereingang geschalteten Aktor.
- „ABC1234567:3“ für den zweiten Tastereingang. Parameter-Sets „MASTER“, „VALUES“ und „LINK“ sind vorhanden. „LINK“-Parametersets beschreiben jeweils einen durch diesen Tastereingang geschalteten Aktor.

2 Zugriff auf die Schnittstelle

2.1 HomeMatic Zentrale

Die HomeMatic Zentrale verfügt derzeit über drei Schnittstellen, über die Geräte angeschlossen sein können:

- BidCos-RF für Funk-Komponenten
  Port-Nummer: 2001
- BidCos-Wired für drahtgebundene Geräte
  Port-Nummer: 2000
- System für interne Geräte
  Port-Nummer: 2002

Für jede dieser Schnittstellen stellt die HomeMatic Zentrale einen XML-RPC-Server zur Verfügung. Die entsprechende URL lautet jeweils:

`http://<IP_DER_ZENTRALE>:<PORT_NUMMER>/`

Um Ereignisbenachrichtigungen zu erhalten, kann sich eine Logikschicht bei den Schnittstellenprozessen anmelden. Dies geschieht über die Methode init. In diesem Fall muss die Logikschicht selbst einen XML-RPC-Server und diverse Methoden bereitstellen (siehe Abschitt 4: Methoden der Logikschicht). Des Weiteren muss der von der Logikschicht bereitgestellte XML-RPC-Server die Standardmethode *system.multicall* unterstützen.

3 Datentypen

3.1 ParamsetDescription

ParamsetDescription ist ein Struct, das für jeden Parameter einen Member enthält. Dieser Eintrag ist wiederum ein Struct vom Type ParameterDescription.

3.2 ParameterDescription

ParameterDescription ist ein Struct, der einen einzelnen Parameter eines Parameter-Sets beschreibt. Folgende Member sind in jedem Fall vorhanden:

- TYPE
  Ist ein String, der den Datentypen des Paramters angibt.
- OPERATIONS
  Ist ein Integer, der eine Oder-Verknüpfung der mit diesem Parameter möglichen Operationen ist. Dabei sind folgende Werte möglich: 1=Read, 2=Write, 4=Event.
- FLAGS
  Ist ein Integer, der eine Oder-Verknüpfung von Flags für die UI-Darstellung ist. Folgende Werte haben eine Bedeutung:
  0x01 : Visible-Flag. Dieser Parameter sollte für den Endanwender sichtbar sein.
  0x02 : Internal-Flag. Dieser Parameter wird nur intern verwendet.
  0x04 : Transform-Flag. Änderungen dieses Parameters ändern das Verhalten des entsprechenden Kanals völlig. Darf nur geändert werden, wenn keine Verknüpfungen am entsprechenden Kanal vorhanden sind.
  0x08 : Service-Flag. Dieser Parameter soll als Servicemeldung behandelt werden. Als Datentyp für Servicemeldungen sind nur Boolean und Integer zulässig. Bei `0` bzw. `false` liegt dabei keine Meldung vor, ansonsten liegt eine Meldung vor.
  0x10 : Sticky-Flag. Nur bei Servicemeldungen. Servicemeldung setzt sich nicht selbst zurück, sondern muss von der Oberfläche zurückgesetzt werden.
- DEFAULT
  Gibt den default-Wert des Parameters an. Der Datentyp entspricht dem des entsprechenden Parameters.
- MAX
  Gibt den maximalen Wert des Parameters an. Der Datentyp entspricht dem des entsprechenden Parameters.
- MIN
  Gibt den minimalen Wert des Parameters an. Der Datentyp entspricht dem des entsprechenden Parameters.
- UNIT
  Ist ein String, der die Maßeinheit des Parameters angibt.
- TAB_ORDER
  Ist ein Integer, der die Reihenfolge der Parameter innerhalb eines Parametersets angibt.
- CONTROL

Optional. Ist ein String, der für das Userinterface angibt, welches Control für die Darstellung dieses Wertes verwendet werden sollte. Der Aufbau ist „<Control-Name>.<Variablen-Name>:<Control-Index>“. Der Variablenname ist optional. Er wird nur benötigt für Controls, die auf mehreren Werten operieren. Der Control-Index ist ebenfalls optional. Er wird nur benötigt, um mehrere gleichartige Controls im gleichen Gültigkeitsbereich (z.B. innerhalb eines Kanals) unterscheiden zu können.
Beispiel:
Ein Dimmer hat zwei Werte. Einer ist die Helligkeit vom Typ Float und ein weiterer ist vom Typ „ACTION“ und erlaubt es, den Dimmer auf alte Helligkeit einzuschalten. Die Oberfläche definiert ein Control namens „Dimmer“ um diese beiden Werte dem Benutzer darzustellen.
CONTROL ist dann belegt mit „Dimmer.Level“ für die Helligkeit sowie „Dimmer.OldValue“ für das Einschalten auf alten Wert.

Es können beliebig viele weitere Member vorhanden sein. Diese werden in der entsprechenden Gerätebeschreibung oder unten beim Datentyp definiert und liefern der Benutzeroberfläche weitere Informationen, um das Gerät sinnvoll abbilden zu können.

Folgende Werte von „TYPE“ werden unterstützt:

3.2.1 TYPE=FLOAT

Datentyp des entsprechenden Parameters ist Float.

3.2.1.1.1 Weitere Member in ParameterDescription

- SPECIAL [optional]
  Array diskreter Spezialwerte mit einer speziellen Bedeutung außerhalb des Wertebereichs.
  Die Elemente des Arrays sind vom Typ Struct. Jedes Element hat zwei Felder:
  `String ID` gibt den Bezeichner des Spezialwertes an
  `Float VALUE` gibt den Wert des Spezialwertes an

3.2.2 TYPE=INTEGER

Datentyp des entsprechenden Parameters ist Integer.

- SPECIAL [optional]
  Array diskreter Spezialwerte mit einer speziellen Bedeutung außerhalb des Wertebereichs.
  Die Elemente des Arrays sind vom Typ Struct. Jedes Element hat zwei Felder:
  `String ID` gibt den Bezeichner des Spezialwertes an
  `Integer VALUE` gibt den Wert des Spezialwertes an

3.2.3 TYPE=BOOL

Datentyp des entsprechenden Parameters ist Boolean. Es gibt keine weiteren datentypspezifischen Member in der ParameterDescription.

3.2.4 TYPE=ENUM

Datentyp des entsprechenden Parameters ist Integer. Der Wert des Integers gibt der

Index einer der möglichen Optionen an.

3.2.4.1 Weitere Member in ParameterDescription

- VALUE_LIST
  ist ein Array<String>. Für jeden möglichen Wert des ENUM ist ein Element vorhanden, das den Namen des Wertes angibt. Jedem möglichen (Integer-)Wert des Parameters entspricht dabei eine Position innerhalb des Arrays. Der String an der entsprechenden Position gibt den Symbolischen Namen des Wertes an. Steht an einen Position im Array der leere String, so ist der entsprechende Wert des Parameters nicht definiert und kann nicht angenommen werden.

3.2.5 TYPE=STRING

Datentyp des entsprechenden Parameters ist String. Es gibt keine weiteren datentypspezifischen Member in der ParameterDescription.

3.2.6 TYPE=ACTION

Datentyp des entsprechenden Parameters ist Boolean. Es wird beim Lesen immer FALSE zurückgegeben. Bei einem Event ist der Parameter jedoch immer TRUE. Beim Schreiben auf den Parameter spielt der geschriebene Wert keine Rolle.

Der Typ ACTION wird verwendet, um Vorgänge wie das Drücken einer Fernbedienungstaste abzubilden. In diesem Fall wird beim Drücken der Taste ein Event generiert. Umgekehrt wird beim Schreiben auf diesen Parameter ein Tastendruck simuliert.

3.3 Paramset

Paramset ist ein Struct, das für jeden Parameter einen Member vom entsprechenden Datentyp (siehe ParamsetDescription) enthält.

3.4 DeviceDescription

DeviceDescription ist ein Struct, das die folgenden Member enthält:

- TYPE
  Datentyp String. Typ des Gerätes.
- ADDRESS
  Datentyp String. Adresse des Kanals/Gerätes.
- CHILDREN
  Datentyp Array<String>. Adressen der untergeordneten Kanäle.
- PARENT
  Datentyp String. Adresse des übergeordneten Gerätes. Ist bei Geräten vorhanden aber leer.
- PARENT_TYPE (Nur bei Kanälen)
  Datentyp String. Typ (Kurzbezeichnung) des übergeordneten Gerätes.
- INDEX (Nur bei Kanälen)
  Datentyp Integer. Gibt die Kanalnummer an.

- AES_ACTIVE
  Datentyp Boolean. Gibt an, ob die gesicherte Übertragung für den Kanal aktiviert ist.
- PARAMSETS
  Datentyp Array<String>. Liste der Namen der vorhandenen Parameter-Sets.
- FIRMWARE (Nur bei Geräten, Optional)
  Datentyp String. Firmwareversion des Gerätes.
- AVAILABLE_FIRMWARE (Nur bei Geräten, Optional)
  Datentyp String. Für ein Firmwareupdate verfügbare Firmwareversion.
- VERSION
  Datentyp Integer. Version der Geräte- oder Kanalbeschreibung.
- FLAGS
  Datentyp Integer. Oder-Verknüpfung von Flags für die UI-Darstellung. Folgende Werte haben eine Bedeutung:
  0x01 : Visible-Flag. Dieses Objekt sollte für den Endanwender sichtbar sein.
  0x02 : Internal-Flag. Dieses Objekt wird nur intern verwendet und ist für den Endanwender nicht sichtbar.
  0x08 : Dontdelete-Flag. Dieser Objekt kann nicht gelöscht werden.
- LINK_SOURCE_ROLES (nur bei Kanälen)
  Datentyp String. Durch Leerzeichen getrennte Liste von Rollen, die der Kanal in einer Verknüpfung als Sender annehmen kann. Eine Rolle ist z.B. „SWITCH“ für einen Kanal, der Schaltbefehle senden kann.
- LINK_TARGET_ROLES (nur bei Kanälen)
  Datentyp String. Durch Leerzeichen getrennte Liste von Rollen, die der Kanal in einer Verknüpfung als Empfänger annehmen kann. Eine Rolle ist z.B. „SWITCH“ für einen Kanal, der auf empfangene Schaltbefehle reagieren kann.
- DIRECTION (nur bei Kanälen)
  Datentyp Integer. Gibt die Richtung (Senden oder Empfangen) dieses Kanals in einer direkten Verknüpfung an.
  0 = DIRECTION_NONE (Kanal unterstützt keine direkte Verknüpfung)
  1 = DIRECTION_SENDER
  2 = DIRECTION_RECEIVER
- GROUP (optional, nur bei Kanälen)
  Datentyp String. Nur bei gruppierten Kanälen (Tastenpaaren) vorhanden. Hier wird die Adresse des anderen zur Gruppe gehörenden Kanals angegeben.
- TEAM (optional, nur bei Kanälen mit Team)
  Datentyp String. Nur bei Kanälen mit Team (z.B. Rauchmelder) vorhanden. Gibt die Adresse des virtuellen Teamkanals an.
- TEAM_TAG (optional, nur bei Kanälen und Teams)
  Datentyp String. Nur bei Kanälen mit Team (z.B. Rauchmelder) und bei virtuellen Teamkanälen vorhanden. Für die Auswahl eines Teams an der Oberfläche. Ein Kanal, der einem Team zugeordnet werden soll und der virtuelle Teamkanal (das Team) müssen hier denselben Wert haben.

- TEAM_CHANNELS (optional, nur bei Kanälen, die ein Team darstellen)
  Datentyp Array<String>. Adressen der dem Team zugeordneten Kanäle.
- INTERFACE (optional, nur bei BidCos-RF)
  Datentyp String. Seriennummer des dem Gerät zugeordneten Interfaces.
- ROAMING (optional, nur bei BidCos-RF)
  Datentyp Boolean. Ist true, wenn die Interfacezuordnung des Geräts automatisch den Empfangsverhältnissen angepasst wird.

4 Methoden der Schnittstellenprozesse

In der folgenden Liste mit [optional] gekennzeichneten Methoden müssen von einem konkreten Schnittstellenprozess nicht unbedingt exportiert werden. Bei Verwendung dieser Methoden ist mit einem Fehler zu rechnen, wenn nich vor dem Aufruf mit `system.methodHelp` oder `system.listMethods` die Existenz geprüft wird.

4.1 Methodenübersicht

Methode	*BidCos-RF (Port: 2001)*	*BidCos-Wired (Port: 2000)*	*System (Port: 2002)*
activateLinkParamset	x	-	x
addDevice	x	-	-
addLink	x	x	x
changekey	x	-	-
clearConfigCache	x	x	x
deleteDevice	x	x	-
determineParameter	x	-	-
getDeviceDescription	x	x	x
getInstallMode	x	-	-
getKeyMismatchDevice	x	-	-
getLinkInfo	x	x	-
getLinkPeers	x	x	-
getLinks	x	x	x
getParamset	x	x	x
getParamsetDescription	x	x	x
getParamsetId	x	x	x
getValue	x	x	x
init	x	x	x
listDevices	x	x	x
listTeams	x	-	-
logLevel	x	x	x
putParamset	x	x	x
removeLink	x	x	x
reportValueUsage	x	x	-
restoreConfigToDevice	x	-	-
rssiInfo	x	-	-
searchDevices	-	x	-
setInstallMode	x	-	-
setLinkInfo	x	x	-
setTeam	x	-	-
setTempKey	x	-	-
setValue	x	x	x
system.listMethods	x	x	x
system.methodHelp	x	x	x
system.multicall	x	x	x

updateFirmware	-	x	-
listBidcosInterfaces	x	-	-
setBidcosInterface	x	-	-
getServiceMessages	x	-	-
getMetadata	x	-	-
setMetadata	x	-	-
getAllMetadata	x	-	-

4.2 Allgemeine Methoden

4.2.1 init

```
void init(String url, String interface_id)
```

Mit dieser Methode teilt die Logikschicht dem Schnittstellenprozess mit, dass sie gerade gestartet wurde. Der Schnittstellenprozess wird sich daraufhin selbst initialisieren und z.B. mit listDevices() die der Logikschicht bekannten Geräte abfragen.

Der Parameter `url` gibt die Adresse des XmlRpc-Servers an, unter der die Logikschicht zu erreichen ist.

Der Parameter `interface_id` teilt dem Schnittstellenprozess die Id, mit unter der er sich gegenüber der Logikschicht identifiziert.

Zum Abmelden von der Ereignisbehandlung wird `interface_id` leer gelassen.

4.2.2 listDevices

```
Array<DeviceDescription> listDevices()
```

Diese Methode gibt alle dem Schnittstellenprozess bekannten Geräte in Form von Gerätebeschreibungen zurück.

4.2.3 getDeviceDescription

```
DeviceDescription getDeviceDescription(Sting address)
```

Diese Methode gibt die Gerätebeschreibung des als `address` übergebenen Gerätes zurück.

4.2.4 getParamsetDescription

```
ParamsetDescription getParamsetDescription(String address,
String paramset_type)
```

Mit dieser Methode wird die Beschreibung eines Parameter-Sets ermittelt. Der Parameter `address` ist die Adresse eines logischen Gerätes (z.B. von listDevices zurückgegeben). Der Parameter `paramset_type` ist „MASTER“, „VALUES“ oder „LINK“.

4.2.5 getParamsetId

```
String getParamsetId(String address, String type)
```

Diese Methode gibt die Id eines Parametersets zurück. Diese wird verwendet, um spezialisierte Konfigurationsdialoge (Easymode) den Parametersets zuzuordnen.

4.2.6 getParamset

```
Paramset getParamset(String address, String paramset_key)
```

Mit dieser Methode wird ein komplettes Parameter-Set für ein logisches Gerät gelesen. Der Parameter `address` ist die Addresses eines logischen Gerätes. Der Parameter `paramset_key` ist „MASTER“, „VALUES“ oder die Adresse eines Kommunikationspartners für das entsprechende Link-Parameter-Set (siehe getLinkPeers).

4.2.7 putParamset

```
void putParamset(String address, String paramset_key, Paramset set)
```

Mit dieser Methode wird ein komplettes Parameter-Set für ein logisches Gerät geschrieben. Der Parameter `address` ist die Addresses eines logischen Gerätes. Der Parameter `paramset_key` ist „MASTER“, „VALUES“ oder die Adresse eines Kommunikationspartners für das entsprechende Link-Parameter-Set (siehe getLinkPeers).

Der Parameter `set` ist das zu schreibende Parameter-Set. In `set` nicht vorhandene Member werden einfach nicht geschrieben und behalten ihren alten Wert.

4.2.8 getValue

```
ValueType getValue(String address, String value_key)
```

Mit dieser Methode wird ein einzelner Wert aus dem Parameter-Set „VALUES“ gelesen. Der Parameter `address` ist die Addresse eines logischen Gerätes. Der Parameter `value_key` ist der Name des zu lesenden Wertes. Die möglichen Werte für value_key ergeben sich aus der ParamsetDescription des entsprechenden Parameter-Sets „VALUES“.

4.2.9 setValue

```
void setValue(String address, String value_key, ValueType value)
```

Mit dieser Methode wird ein einzelner Wert aus dem Parameter-Set „VALUES“ geschrieben. Der Parameter `address` ist die Addresse eines logischen Gerätes. Der Parameter `value_key` ist der Name des zu schreibenden Wertes. Die möglichen Werte für `value_key` ergeben sich aus der ParamsetDescription des entsprechenden Parameter-Sets „VALUES“. Der Parameter `value` ist der zu schreibende Wert.

4.2.10 getLinks

```
Array<Struct>getLinks(String address, Integer flags)
```

Diese Methode gibt alle einem logischen Kanal oder Gerät zugeordneten

Kommunikationsbeziehungen zurück.

Der Parameter `address` ist die Kanal- oder Geräteadresse des logischen Objektes, auf das sich die Abfrage bezieht. Bei `address==""` werden alle Kommunikationsbeziehungen des gesamten Schnittstellenprozesses zurückgegeben.

Der Parameter `flags` ist ein bitweises oder folgender Werte:

- 1 = `GL_FLAG_GROUP`
  Wenn `address` einen Kanal bezeichnet, der sich in einer Gruppe befindet, werden die Kommunikationsbeziehungen für alle Kanäle der Gruppe zurückgegeben.
- 2 = `GL_FLAG_SENDER_PARAMSET`
  Das Feld `SENDER_PARAMSET` des Rückgabewertes wird gefüllt.
- 4 = `GL_FLAG_RECEIVER_PARAMSET`
  Das Feld `RECEIVER_PARAMSET` des Rückgabewertes wird gefüllt.

`flags` ist optional. Defaultwert ist 0x00.

Der Rückgabewert ist ein Array von Strukturen. Jede dieser Strukturen enthält die folgenden Felder:

- `SENDER`
  Datentyp String. Adresse des Senders der Kommunikationsbeziehung
- `RECEIVER`
  Datentyp String. Adresse des Empfängers der Kommunikationsbeziehung
- `FLAGS`
  Datentyp Integer. `FLAGS` ist ein bitweises oder folgender Werte:
  - `1=LINK_FLAG_SENDER_BROKEN`
    Diese Verknüpfung ist auf der Senderseite nicht intakt
  - `2=LINK_FLAG_RECEIVER_BROKEN`
    Diese Verknüpfung ist auf der Empfängerseite nicht intakt
- `NAME`
  Datentyp String. Name der Kommunikationsbeziehung
- `DESCRIPTION`
  Datentyp String. Textuelle Beschreibung der Kommunikationsbeziehung
- `SENDER_PARAMSET`
  Datentyp Paramset. Parametersatz dieser Kommunikationsbeziehung für die Senderseite
- `RECEIVER_PARAMSET`
  Datentyp Paramset. Parametersatz dieser Kommunikationsbeziehung für die Empfängerseite

4.2.11 addLink

```
void addLink(String sender, String receiver, String name, String description)
```

Diese Methode erstellt eine Kommunikationsbeziehung zwischen zwei logischen Geräten. Die Parameter `sender` und `receiver` bezeichnen die beiden zu verknüpfenden Partner. Die Parameter `name` und `description` sind optional und beschreiben die Verknüpfung näher.

4.2.12 removeLink

```
void removeLink(String sender, String receiver)
```

Diese Methode löscht eine Kommunikationsbeziehung zwischen zwei Geräten. Die Parameter `sender` und `receiver` bezeichnen die beiden Kommunikationspartner deren Kommunikationszuordnung gelöscht werden soll.

4.3 Optionale Methoden

4.3.1 setLinkInfo

BidCos-RF, BidCos-Wired

```
void setLinkInfo(String sender, String receiver, String name, String description)
```

Diese Methode ändert die beschreibenden Texte einer Kommunikationsbeziehung. Die Parameter sender und receiver bezeichnen die beiden zu verknüpfenden Partner. Die Parameter name und description beschreiben die Verknüpfung textuell.

4.3.2 getLinkInfo

BidCos-RF, BidCos-Wired

```
Array<String> getLinkInfo(String sender_address, String receiver_address)
```

Diese Methode gibt den Namen und die Beschreibung für eine bestehende Kommunikationsbeziehung zurück. Die Parameter `sender_address` und `receiver_address` bezeichnen die beiden verknüpften Partner.

4.3.3 activateLinkParamset

BidCos-RF, System

```
void activateLinkParamset(String address, String peer_address, Boolean long_press)
```

Mit dieser Methode wird ein Link-Parameterset aktiviert. Das logische Gerät verhält sich dann so als ob es direkt von dem entsprechenden zugeordneten Gerät angesteuert worden wäre. Hiermit kann z.B. ein Link-Parameter-Set getestet werden. Der Parameter `address` ist die Addresses des anzusprechenden logischen Gerätes. Der Parameter `peer_address` ist die Addresse des Kommunikationspartners, dessen Link-Parameter-Set aktiviert werden soll.

Der Parameter `long_press` gibt an, ob das Parameterset für den langen Tastendruck aktiviert werden soll.

4.3.4 determineParameter

BidCos-RF

```
void determineParameter(String address, String paramset_key, String parameter_id)
```

Mit dieser Methode wird ein Parameter eines Parameter-Sets automatisch bestimmt. Der Parameter kann bei erfolgreicher Ausführung anschließend sofort über getParamset gelesen werden.

Der Parameter `address` ist die Addresses eines logischen Gerätes.

Der Parameter `paramset_key` ist „MASTER“, „VALUES“ oder die Adresse eines Kommunikationspartners für das entsprechende Link-Parameter-Set (siehe getLinkPeers).

Der Parameter `parameter_id` bestimmt den automatisch zu bestimmenden Parameter.

4.3.5 deleteDevice

BidCos-RF, BidCos-Wired

```
void deleteDevice(String address, Integer flags)
```

Diese Methode löscht ein Gerät aus dem Schnittstellenprozess.

Der Parameter `address` ist die Addresse des zu löschenden Gerätes.

`flags` ist ein bitweises oder folgender Werte:

- `0x01=DELETE_FLAG_RESET`
  Das Gerät wird vor dem Löschen in den Werkszustand zurückgesetzt
- `0x02=DELETE_FLAG_FORCE`
  Das Gerät wird auch gelöscht, wenn es nicht erreichbar ist
- `0x04=DELETE_FLAG_DEFER`
  Wenn das Gerät nicht erreichbar ist, wird es bei nächster Gelegenheit gelöscht

4.3.6 setInstallMode

BidCos-RF

```
void setInstallMode(Boolean on)
```

Diese Methode aktiviert und deaktiviert den Installations-Modus, in dem neue Geräte an der HomeMatic-CCU angemeldet werden können.

Der Parameter `on` bestimmt, ob der Installations-Modus aktiviert oder deaktiviert werden soll.

4.3.7 getInstallMode

BidCos-RF

```
Integer getInstallMode(void)
```

Diese Methode gibt die verbliebene Restzeit in Sekunden im Anlernmodus zurück. Der Wert 0 bedeutet, der Anlernmodus ist nicht aktiv.

4.3.8 getKeyMissmatchDevice

BidCos-RF

```
String getKeyMissmatchDevice(bool reset)
```

Diese Methode gibt die Seriennummer des letzten Gerätes zurück, das aufgrund eines falschen AES-Schlüssels nicht angelernt werden konnte. Mit `reset=true` wird diese Information im Schnittstellenprozess zurückgesetzt.

4.3.9 setTempKey

BidCos-RF

```
void setTempKey(String passphrase)
```

Diese Methode ändert den von der CCU verwendeten temporären AES-Schlüssel. Der temporäre AES-Schlüssel wird verwendet, um ein Gerät anzulernen, in dem ein anderer Schlüssel gespeichert ist als der Schlüssel der CCU.

4.3.10 addDevice

BidCos-RF

```
DeviceDescription addDevice(String serial_number)
```

Diese Methode lernt ein Gerät anhand seiner Seriennummer an die CCU an. Diese Funktion wird nicht von jedem Gerät unterstützt. Rückgabewert ist die DeviceDescription des neu angelernten Geräts.

4.3.11 searchDevices

BidCos-Wired

```
int searchDevices(void)
```

Diese Methode durchsucht den Bus nach neuen Geräten und gibt die Anzahl gefundener Geräte zurück. Die neu gefundenen Geräte werden mit `newDevices` der Logikschicht gemeldet.

4.3.12 changeKey

BidCos-RF

```
void changeKey(String passphrase)
```

Diese Methode ändert den vom Schnittstellenprozess verwendeten AES-Schlüssel. Der Schlüssel wird ebenfalls in allen angelernten Geräten getauscht.

4.3.13 listTeams

BidCos-RF

```
Array<DeviceDescription> listTeams(void)
```

Diese Methode gibt alle dem Schnittstellenprozess bekannten Teams in Form von Gerätebeschreibungen zurück.

4.3.14 setTeam

BidCos-RF

```
void setTeam(String channel_address, String team_address)
```

Diese Methode fügt den Kanal `channel_address` zum Team `team_address` hinzu. Bei `team_address=""` wird der Kanal `channel_address` seinem eigenen Team zugeordnet.

Dabei muss `team_address` entweder leer sein („“) oder eine Seriennummer eines existierenden Teams enthalten.

Teams werden dabei je nach Bedarf erzeugt und gelöscht.

4.3.15 restoreConfigToDevice

BidCos-RF

```
void restoreConfigToDevice(String address)
```

Diese Methode überträgt alle zu einem Gerät in der CCU gespeicherten Konfigurationsdaten erneut an das Gerät.

4.3.16 clearConfigCache

BidCos-RF, BidCos-Wired, System

```
void clearConfigCache(String address)
```

Diese Methode löscht alle zu einem Gerät in der CCU gespeicherten Konfigurationsdaten. Diese werden nicht sofort wieder vom Gerät abgefragt, sondern wenn sie das nächste mal benötigt werden.

4.3.17 rssiInfo

BidCos-RF

```
Struct rssiInfo(void)
```

Gibt ein zweidimensionales assoziatives Array zurück, dessen Schlüssel die Geräteadressen sind. Die Felder des assoziativen Arrays sind Tupel, die die Empfangsfeldstärken zwischen beiden Schlüsselgeräten für beide Richtungen in dbm angeben. ein Wert von 65536 bedeutet, dass keine Informationen vorliegen.

- `Rückgabewert[<Gerät 1>][<Gerät 2>][0]`
  Empfangsfeldstärke an Gerät 1 für Sendungen von Gerät 2
- `Rückgabewert[<Gerät 1>][<Gerät 2>][1]`

Empfangsfeldstärke an Gerät 2 für Sendungen von Gerät 1

4.3.18 updateFirmware

BidCos-Wired

```
Array<Boolean> updateFirmware(Array<String> devices)
```

Diese Methode führt ein Firmware-Update der in `devices` enthaltenen Geräte durch. Die Geräte werden durch Ihre jeweilige Seriennummer spezifiziert.

Der Rückgabewert gibt für jedes Gerät an, ob das Firmware-Update erfolgreich war.

4.3.19 getLinkPeers

BidCos-RF, BidCos-Wired

```
Array<String> getLinkPeer(String address)
```

Diese Methode gibt alle einem logischen Gerät zugeordneten Kommunikationspartner zurück. Die zurückgegebenen Werte können als Parameter `paramset_key` für getParamset() und putParamset() verwendet werden. Der Parameter `address` ist die Adresse eines logischen Gerätes.

4.3.20 logLevel

BidCos-RF, BidCos-Wired, System

```
int logLevel(void);       /* (1) */
int logLevel(int level);  /* (2) */
```

Diese Methode gibt den aktuellen Log-Level zurück (1) bzw. setzt diesen (2). Folgende Werte sind für `level` möglich:

- `6=LOG_FATAL_ERROR:` Nur schwere Fehler loggen.
- `5=LOG_ERROR:` Zusätzlich normale Fehler loggen.
- `4=LOG_WARNING:` Zusätzlich Warnungen loggen.
- `3=LOG_NOTICE:` Zusätzlich Notizmeldungen loggen.
- `2=LOG_INFO:` Zusätzlich Infomeldungen loggen.
- `1=LOG_DEBUG:` Zusätzlich Debugmeldungen loggen.
- `0=LOG_ALL:` Alles wird geloggt.

4.3.21 reportValueUsage

BidCos-RF, BidCos-Wired

```
Boolean reportValueUsage(String address, String value_id,
    Integer ref_counter)
```

Diese Methode teilt dem Interfaceprozess in `ref_counter` mit, wie oft der Wert `value_id` des Kanals `address` innerhalb der Logikschicht (z.B. in Programmen) verwendet wird. Dadurch kann der Interfaceprozess die Verbindung mit der

entsprechenden Komponente herstellen bzw. löschen. Diese Funktion sollte bei jeder Änderung aufgerufen werden.

Der Rückgabewert ist `true`, wenn die Aktion sofort durchgeführt wurde. Er ist `false`, wenn die entsprechende Komponente nicht erreicht werden konnte und vom Benutzer zunächst in den Config-Mode gebracht werden muss. Der Interfaceprozess hat dann aber die neue Einstellung übernommen und wird sie bei nächster Gelegenheit automatisch an die Komponente übertragen.

In diesem Fall ist dann auch der Wert CONFIG_PENDING im Kanal MAINTENANCE der Komponente gesetzt.

4.3.22 setBidcosInterface

BidCos-RF

```
Void setBidcosInterface(String device_address, String interface_address,
    Boolean roaming)
```

Diese Methode setzt das für die Kommunikation mit dem durch `device_address` spezifizierten Gerät verwendete Bidcos-Interface. Die Seriennummer des in Zukunft für die Kommunikation mit diesem Gerät zu verwendenden Interfaces wird in `interface_address` übergeben. Ist der Parameter `roaming` gesetzt, so wird die Interfacezuordnung für das Gerät automatisch in Abhängigkeit von der Empfangsfeldstärke angepasst. Das ist für nicht ortsfeste Geräte wie Fernbedienungen sinnvoll.

4.3.23 listBidcosInterfaces

BidCos-RF

```
Array<Struct>listBidcosInterfaces()
```

Diese Methode gibt eine Liste aller vorhandenen BidCoS-Interfaces in Form eines Arrays von Structs zurück.

Der Rückgabewert ist ein Array von Strukturen. Jede dieser Strukturen enthält die folgenden Felder:

- `ADDRESS`
  Datentyp String. Seriennummer des BidCoS-Interfaces.
- `DESCRIPTION`
  Datentyp String. Textuelle Beschreibung des Interfaces wie in der Konfigurationsdatei für den Schnittstellenprozess angegeben.
- `CONNECTED`
  Datentyp Boolean. Gibt an, ob zum Zeitpunkt der Abfrage eine Kommunikationsverbindung zum Interface besteht.
- `DEFAULT`
  Datentyp Boolean. Gibt an, ob es sich um das Standardinterface handelt. Das Standardinterface wird verwendet, wenn das einem Gerät zugeordnete Interface nicht mehr existiert.

4.3.24 getServiceMessages

BidCos-RF

```
Array<Array> getServiceMessages()
```

Diese Methode gibt eine Liste aller vorhandenen Servicemeldungen in Form eines Arrays zurück.

Der Rückgabewert ist ein Array mit einem Element pro Servicemeldung. Jedes Element ist wiederum ein Array mit

drei Feldern:

- `Rückgabewert[index][0]`
  Datentyp String. Adresse (Seriennummer) des Kanals, der die Servicemeldung generiert hat
- .
- `Rückgabewert[index][1]`
  Datentyp String. ID der Servicemeldung (`CONFIG_PENDING`, `UNREACH`, etc.)
- .

`Rückgabewert[index][2]`
`Datentyp variabel.` Wert der Servicemeldung

4.3.25 setMetadata

BidCos-RF

```
void setMetadata(String object_id, String data_id, Variant value)
```

Diese Methode setzt ein Metadatum zu einem Objekt.

`object_id` ist die Id des Metadaten-Objekts. Üblicherweise ist dies die Seriennummer eines Gerätes oder Kanals.

Durch Übergabe einer beliebigen Id können aber auch eigene Metadaten-Objekte angelegt werden.

`data_id` ist die Id des zu setzenden Metadatums

. Diese Id kann frei gewählt werden.

value ist eine beliebige Variable. Diese wird gespeichert und kann später mittels `getMetadata()` und `getAllMetadata()`wieder abgefragt werden.

4.3.26 getMetadata

BidCos-RF

```
Variant getMetadata(String object_id, String data_id)
```

Diese Methode gibt ein Metadatum zu einem Objekt zurück.

`object_id` ist die Id des Metadaten-Objekts. Üblicherweise ist dies die Seriennummer eines Gerätes oder Kanals.

Durch Übergabe einer beliebigen Id können aber auch eigene Metadaten-Objekte angelegt werden.

`data_id` ist die Id des abzufragenden Metadatums

. Diese Id kann frei gewählt werden.

Der Rückgabewert entspricht in Datentyp und Wert der zuvor an `setMetadata()` als Parameter `value` übergebenen Variablen.

4.3.27 getAllMetadata

BidCos-RF

```
Struct getAllMetadata(String object_id)
```

Diese Methode gibt alle zuvor gesetzten Metadaten zu einem Objekt zurück.

`object_id` ist die Id des Metadaten-Objekts. Üblicherweise ist dies die Seriennummer eines Gerätes oder Kanals.

Durch Übergabe einer beliebigen Id können aber auch eigene Metadaten-Objekte angelegt werden.

Der Rückgabewert ist ein Struct, der zu jedem zuvor gesetzten Metadatum ein Feld enthält. Der Feldname ist der zuvor an `setMetadata()` als Parameter `data_id` übergebene Wert. Der Wert des Feldes entspricht in Datentyp und Wert der zuvor an `setMetadata()` als Parameter `value` übergebenen Variablen.

5 Methoden der Logikschicht

Eine Logikschicht kann ihrerseits über einen XML-RPC-Server verfügen und sich per *init* bei den Schnittstellenprozessen anmelden. In diesem Fall erhält die Logikschicht diverse Informationen direkt, ohne explizit Nachfragen zu müssen, z.B. ob sich die Konfiguration eines Gerätes geändert hat.

Damit der Ablauf reibungslos funktioniert, muss der eingesetzte XML-RPC-Server zwingend die Standardmethode *system.multicall* unterstützen.

5.1 event

```
void event(String interface_id, String address, String value_key, ValueType value)
```

Mit dieser Methode teilt der Schnittstellenprozess der Logikschicht mit, dass sich ein Wert geändert hat oder ein Event (z.B. Tastendruck) empfangen wurde.

Der Parameter `interface_id` gibt die id des Schnittstellenprozesses an, der das Event sendet.

Der Parameter `address` ist die Addresse des logischen Gerätes, zu dem der geänderte Wert / das Event gehört.

Der Parameter `value_key` ist der Name des entsprechenden Wertes. Die möglichen Werte für value_key ergeben sich aus der ParamsetDescription des entsprechenden Parameter-Sets „VALUES“.

Der Parameter `value` gibt den neuen Wert bzw. den dem Event zugeordneten Wert an. Der Datentyp von `value` ergibt sich aus der ParamsetDescription des Values-Parameter-Sets des entsprechenden logischen Gerätes.

5.2 listDevices

```
Array<DeviceDescription> listDevices(String interface_id)
```

Diese Methode gibt alle der Logikschicht bekannten Geräte für den Schnittstellenprozess mit der Id `interface_id` in Form von Gerätebeschreibungen zurück. Damit kann der Schnittstellenprozess durch Aufruf von newDevices() und deleteDevices() einen Abgleich vornehmen.

Damit das funktioniert, muss sich die Logikschicht diese Informationen zumindest teilweise merken. Es ist dabei ausreichend, wenn jeweils die Member ADDRESS und VERSION einer DeviceDescription gesetzt sind.

5.3 newDevices

```
void newDevices(String interface_id , Array<DeviceDescription> dev_descriptions)
```

Mit dieser Methode wird der Logikschicht mitgeteilt, dass neue Geräte gefunden wurden.

Der Parameter `interface_id` gibt die id des Schnittstellenprozesses an, zu dem das

Gerät gehört.

Der Parameter `dev_descriptions` ist ein Array, das die Beschreibungen der neuen Geräte enthält.

Wenn `dev_descriptions` Geräte enthält, die der Logikschicht bereits bekannt sind, dann ist davon auszugehen, dass sich z.B. durch ein Firmwareupdate das Verhalten des Gerätes geändert hat. Die Basisplatform muß dann einen Abgleich mit der neuen Beschreibung durchführen. Dabei sollte die Konfiguration des Gerätes innerhalb der Logikschicht so weit wie möglich erhalten bleiben.

5.4 deleteDevices

```
void deleteDevices(String interface_id, Array<String> addresses)
```

Mit dieser Methode wird der Logikschicht mitgeteilt, dass Geräte im Schnittstellenprozess gelöscht wurden.

Der Parameter `interface_id` gibt die id des Schnittstellenprozesses an, zu dem das Gerät gehört.

Der Parameter `addresses` ist ein Array, das die Adressen der gelöschten Geräte enthält.

5.5 updateDevice

```
void updateDevice(String interface_id, String address, int hint)
```

Mit dieser Methode wird der Logikschicht mitgeteilt, dass sich an einem Gerät etwas geändert hat.

Der Parameter `interface_id` gibt die id des Schnittstellenprozesses an, zu dem das Gerät gehört.

Der Parameter `address` ist die Adresse des Gerätes oder des Kanals, auf das sich die Meldung bezieht.

Der Parameter `hint` spezifiziert die Änderung genauer:

- `UPDATE_HINT_ALL=0`
  Es hat eine nicht weiter spezifizierte Änderung stattgefunden und es sollen daher alle möglichen Änderungen berücksichtigt werden.
- `UPDATE_HINT_LINKS=1`
  Es hat sich die Anzahl der Verknüpfungspartner geändert.

Derzeit werden nur Änderungen an den Verknüpfungspartnern auf diesem Weg mitgeteilt.

6 Fehlercodes

Fehlercode	Bedeutung
-1	Allgemeiner Fehler
-2	Unbekanntes Gerät / unbekannter Kanal
-3	Unbekannter Paramset
-4	Es wurde eine Geräteadresse erwartet
-5	Unbekannter Parameter oder Wert
-6	Operation wird vom Parameter nicht unterstützt

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