ECoS ESU COMMAND STATION

Netzwerkspezifikation für das PC-Interface

für Protokoll-Version 0.1

Erste Auflage, Juni 2007

Inhalt

0. Lizenzbestimmungen

1. Allgemeines

In diesem Dokument werden Verfahren zur Kommunikation zwischen einem Netzwerkteilnehmer (im Folgenden "Client" genannt) und einem Netzwerk-Dienstanbieter (im Folgenden "Server" genannt) beschrieben. Im Folgenden werden diese Verfahren als "Protokoll" bezeichnet. Sämtliche Rechte, inbesondere das Urheberrecht des Protokolls liegen ausschließlich bei electronic solutions ulm GmbH & Co. KG (im Folgenden "ESU" genannt).

(c) Copyright 2006, 2007 ESU.

2. Abschluss der Lizenz

Dieser Lizenztext stellt ein an jedermann gerichtetes Angebot auf Abschluss eines Lizenzvertrags dar. Der Lizenzvertrag kommt durch Implementierung des Protokolls auf einem datenverarbeitenden System zustande. Dem Anwender werden unentgeltlich nicht-exlusive, eingeschränkte und nicht-übertragbare Nutzungsrechte eingeräumt.

Die Implementierung des Protokolls in einer Client-Software für PC-Systeme (z.B. Homecomputer oder Laptops) ist ohne Einschränkung erlaubt.

3. Verbotene Nutzung

Es ist ausdrücklich verboten, das Protokoll in einem Server zu implementieren. Insbesondere fällt hierunter die Verwendung des Protokolls in einem zur Steuerung von Modellspielwaren geeigneten Gerät.

Dem Lizenznehmer ist es zudem untersagt, auf Grundlage des Protokolls weitere Kommunikationsverfahren zu entwickeln. Dies umfasst ausdrücklich auch die Modifikation des bestehenden Protokolls.

Zudem untersagt ist die Implementierung des Protokolls in Software, die auf anderen als PC-Systemen lauffähig ist.

4. Einschränkungen

Es ist dem Lizenznehmer sowie Dritten untersagt, dieses Dokument weiterzugeben oder in anderer Form öffentlich zugänglich zu machen. Der Inhalt ist lizenziert, nicht verkauft.

5. Gewährleistungsausschluss

Es wird keine Verantwortung oder Haftung für Genauigkeit, Inhalt, Vollständigkeit, Zuverlässigkeit, Funktionsfähigkeit oder Eignung für einen bestimmten Zweck des Protokolls übernommen. Insbesondere können keine Ansprüche für Schäden gegenüber ESU geltend gemacht werden, die durch die Verwendung des Protokolls entstehen.

Das Protokoll wird ohne jegliche Garantie "im vorliegenden Zustand" zur Verfügung gestellt.

Für das Protokoll wird kein Support gewährleistet.

6. Änderungen an Protokoll und Lizenzbedingungen

ESU ist berechtigt das Protokoll jederzeit und ohne Ankündigung zu ändern. Zudem ist ESU berechtigt, die Bestimmungen dieses Lizenzvertrags einseitig zu ändern.

7. Rechtswahl

Auf sämtliche Rechtsbeziehungen zwischen den Parteien, einschließlich des Deliktsrechts, findet das Recht der Bundesrepublik Deutschland Anwendung. Der Gerichtsstand ist Ulm.

8. Schlussbestimmung

Sollten einzelne Bestimmungen dieses Vertrages unwirksam oder undurchführbar sein oder nach Vertragsschluss unwirksam oder undurchführbar werden, so wird dadurch die Wirksamkeit des Vertrages im Übrigen nicht berührt. An die Stelle der unwirksamen oder undurchführbaren Bestimmung soll diejenige wirksame und durchführbare Regelung treten, deren Wirkungen der wirtschaftlichen Zielsetzung möglichst nahe kommen, die die Vertragsparteien mit der unwirksamen beziehungsweise undurchführbaren Bestimmung verfolgt haben. Die vorstehenden Bestimmungen gelten entsprechend für den Fall, dass sich der Vertrag als lückenhaft erweist. § 139 BGB findet keine Anwendung.

1. Verbindung

TCP-Socket Verbindung auf Port 15471.

Empfangsbuffer ist 1024 Byte groß, Befehle dürfen nicht gesplittet werden. Es dürfen sich aber mehrere komplette Befehle in einem Paket befinden. Kann ein Befehl innerhalb des Empfangsbuffers nicht interpretiert werden, wird der restliche Empfangsbuffer verworfen.

2. Objekte

Der Zugriff auf die ECoS erfolgt über Objekte. Diese Objekte werden durch eine eindeutige ID adressiert. Es existieren zahlreiche Basisobjekte mit konstanten IDs, die zur Konfiguration und zur Verwaltung weiterer Objekte verwendet werden. In den so genannten Objektmanagern werden die Listenobjekte verwaltet. Der LokManager (Basisobjekt) mit der ID 10 verwaltet z.B. alle Loks (Listenobjekt). Über diesen LokManager können die IDs der Listenobjekte abgefragt werden und auch neue Listenobjekte hinzugefügt werden.

3. Clients/Teilnehmer

Die ECoS ist ein zentraler Server. Alle Objekte müssen an diesem Server, d.h. an der ECoS angemeldet bzw. registriert sein. Beim Anschließen eines Handreglers an der ECoS wird dieser z.B. automatisch an der ECoS angemeldet und bekommt z.B. die zuletzt auf diesem Handregler aufgerufenen Loks zugewiesen. Jeder Teilnehmer kann sich auf zwei unterschiedliche Arten an einem oder an mehreren Objekten registrieren. Eine Control ist nötig, um ein Objekt zu manipulieren. Eine View erlaubt die Überwachung eines Objekts. Ein Handregler mit einer Lokliste von 10 Loks wird sich z.B. an allen Loks als View registrieren und an einer Lok, die gesteuert werden soll zusätzlich als Control. Im System darf nur ein Teilnehmer eine Control auf ein Objekt haben, d.h. nur ein Teilnehmer darf ein Objekt zur selben Zeit manipulieren.

Durch die Registrierung eines Teilnehmers an einem Objekt als View, wird dieser Teilnehmer automatisch über Änderungen an diesem Objekt über so genannte Events informiert. Ein Teilnehmer der sich als Control und als View an einem Objekt registriert hat, wird nicht über Änderungen am Objekt informiert.

4. Nachrichten

Für die Kommunikation über das PC-Interface wird ein ASCII-Protokoll verwendet. Strings wie z.B. Namen werden UTF8 codiert und sind immer in Anführungszeichen eingeschlossen. Anführungszeichen in Strings müssen doppelt gesendet werden, um Anführungszeichen in Strings besser parsen zu können.

Jede Kommunikation wird durch die ECoS mit Fehlercode bestätigt (reply). Nachrichten, die von der ECoS getriggert werden (events), müssen durch den PC nicht bestätigt werden. Als Zeilenende wird ein new line (\n, ASCII=10) verwendet, bei Befehlen darf auch vor diesem new line ein carriage return (\r, ASCII=13) stehen.

Jede Nachricht ist immer an ein, durch eine eindeutige ID gekennzeichnetes, Objekt gerichtet. Dabei gibt es eine Reihe von fest definierten IDs (Basisobjekte) und dynamisch vergebene IDs (Listenobjekte). So betreffen sämtliche Nachrichten an die ID 10 den so genannten LokManager oder die ID 1 die Basisfunktionen der ECoS. Eine Liste der fest vorgegebenen IDs der Basisobjekte befindet sich im Anhang. Für die IDs werden 16 Bit verwendet und IDs sind immer eindeutig.

4.1. Aufbau der Nachrichten

Jede Nachricht von der ECoS an den PC hat die Form

```
<HEADER cmd>
TEXT
<ENDE x (str)>
```

HEADER ist entweder REPLY oder EVENT.

Ein REPLY kennzeichnet eine Antwort auf eine Anfrage über das PC-Interface, z.B. die Abfrage des Namens einer Lok.

Ein EVENT ist eine durch die ECoS generierte Nachricht, wie z.B. die Information an den PC, dass sich die Geschwindigkeit einer Lok geändert hat.

Im Header wird außerdem mit cmd der Befehl wiederholt, auf den sich die Nachricht bezieht. Bei einem EVENT enthält cmd die ID des Objekts.

ENDE ist immer END gefolgt von einem numerischen Fehlercode x und in Klammer die dazugehörige Fehlerbeschreibung str. Fehlercode 0 (str=OK) bedeutet, dass der Befehl ohne Fehler ausgeführt werden konnte.

Zwischen dem Header und dem Ende (TEXT) können mehrere Zeilen zusätzliche Informationen enthalten sein. Jede Zeile in TEXT hat dabei den Aufbau:

```
id option
```

In id steht die ID des Objektes und option enthält zusätzliche Informationen.

Jeder Befehl vom PC an die ECoS hat die Form

```
befehl(id [, options [, ...]])
```

Die Optionen options können wiederum optionale Parameter in eckigen Klammern ([]) besitzen. Pro Befehl sind bis zu 10 Optionen erlaubt. Die Optionen options gelten für alle Befehle an beliebige IDs, wobei nicht alle Optionen von jedem Objekt interpretiert werden können.

5. Befehle

Es folgt nun eine Beschreibung der Befehle an die ECoS in allgemeiner Form. Das erste Argument ist immer eine ID über die ein Objekt der ECoS adressiert wird. Nach der ID können bis zu 10 weitere Argumente (Optionen) folgen. Bei der Beschreibung der Objekte werden die vom jeweiligen Objekt unterstützten Optionen erläutert. Die Option duration wird z.B. nur von einem Schaltartikel-Objekt unterstützt. Folgender Tabellenaufbau wird bei den Beispielen zu den einzelnen Befehlen verwendet:

Befehl an die ECoS	Antwort der ECoS auf diesen Befehl	Kommentar
Beschreibung zu diesem Beispiel		

5.1. queryObjects(id, options, ...)

Dieser Befehl liefert eine Liste von Objekten, die zum Objekt mit der ID id gehören. Mit der Option size wird anstelle der gesamten Liste nur die Anzahl Elemente zurückgeliefert. Mit der Option nr[min, max] kann ein Bereich der Liste angefordert werden.

Beispiel:

queryObjects(10, name)	<REPLY queryObjects(10, name)> 1000 name[„Big Boy“] <END 0 (OK)>	
Es wird vom Objekt 10 (LokManager) eine Liste der Objekte angefordert. In diesem Fall handelt es sich dann um die gesamte Lokliste der ECoS. In der Antwort auf einen queryObjects-Befehl werden die Objekte durch die IDs der Objekte gekennzeichnet. In diesem Beispiel befindet sich in der Lokliste der ECoS nur eine Lok mit der ID 1000. Durch Optionen im queryObjects-Befehl kann die zurückgelieferte Liste von Objekten genauer spezifiziert werden. In obigem Beispiel wird durch die Option name zusätzlich zur Objekt-ID noch der Name des Objekts, in diesem Fall der Name der Lok, mit ausgegeben.		

Beispiel:

queryObjects(10, size)	<REPLY queryObjects(10, size)> 10 size[1] <END 0 (OK)>	
Es wird vom Objekt 10 (LokManager) die Anzahl Objekte angefordert. In diesem Beispiel befindet sich in der Lokliste der ECoS nur eine Lok.		

Beispiel:

queryObjects(10, nr[0,3])	<REPLY queryObjects(10, nr[0, 3])> 1000 1001 1005 1006 <END 0 (OK)>	
Es werden vom Objekt 10 (LokManager) die ersten 4 Einträge in der Liste angefordert. In diesem Beispiel befinden sich in der Lokliste der ECoS mindestens 4 Loks mit den IDs 1000, 1001, 1005 und 1006.		

5.2. set(id, options, ...)

Mit diesem Befehl können einzelne Eigenschaften eines Objektes gesetzt werden. Da mindestens eine Eigenschaft gesetzt wird, sind bei diesem Befehl auch mindestens zwei Parameter (ID und mindestens eine Option) notwendig. In der Regel wird auch jede Option einen weiteren Parameter in eckigen Klammern besitzen, der den neuen Wert enthält.

Beispiel:

set(1000, addr[3])	<REPLY set(1000, addr[3])> 1000 addr[3] <END 0 (OK)>	
Dem Objekt mit der ID 1000 wird die Adresse 3 zugewiesen.		

Beispiel:

set(1000, speed[12])	<REPLY set(1000, speed[12])> 1000 speed[12] <END 0 (OK)>	
Die Option speed wird für das Objekt mit der ID 1000 auf 12 gesetzt. Wenn das Objekt mit der ID 1000 eine Lok ist, fährt diese anschließend mit der Geschwindigkeit 12. Geschwindigkeiten werden bei der ECoS immer auf 127 Fahrstufen normiert, wobei eine Geschwindigkeit 1 ein Nothalt bedeutet, sofern der Dekoder dies unterstützt.		

5.3. get(id, options, ...)

Mit diesem Befehl werden einzelne Eigenschaften eines Objektes abgefragt.

Beispiel:

get(1000, name, speed)	<REPLY get(1000, name, speed)> 1000 name[„Big Boy"] 1000 speed[12] <END 0 (OK)>	
Es wird der Name und die Geschwindigkeit des Objektes mit der ID 1000 angefordert.		

5.4. create(id, options, ...)

Es wird ein neues Objekt angelegt. Da die Objekt-ID von der ECoS vergeben wird, muss ein neues Objekt beim entsprechenden Objektmanager angelegt werden. Eine neue Lok wird demnach mit einer ID 10 (LokManager) angelegt. Zur Konfiguration des neu angelegten Objekts muss dann auch die Objekt-ID des Objektmanagers verwendet werden. Es kann somit an einem Objektmanager von einem Teilnehmer nur ein neues Objekt konfiguriert werden. Erst wenn das Anlegen des neuen Objektes abgeschlossen ist, kann von diesem Teilnehmer wieder ein neues Objekt angelegt werden. Das Anlegen eines Objektes wird mit einem request(id, append)-Befehl abgeschlossen. Dieser Befehl vergibt eine neue ID für das Objekt und ab diesem Zeitpunkt steht das neue Objekt in der Liste der Listenobjekte des Objektmanagers.

Beispiel zum Anlegen einer Lok:

create(10)	<REPLY create(10)> 10 id[1007] <END 0 (OK)>	Es wird ein Objekt mit der ID 1007 angelegt. Dieses Objekt ist für andere Teilnehmer erst nach einem append-Befehl sichtbar.
set(1007, addr[5])	<REPLY set(1007, addr[5])> 1007 addr[5] <END 0 (OK)>	
set(1007, name[„Big Boy"])	<REPLY set(1007, name[„Big Boy"])> 1007 name[„Big Boy"] <END 0 (OK)>	
create(10)	<REPLY create(10)> <END 35 (NERROR_NOAPPEND)>	Fehler: die zuvor erzeugt Lok muß erst durch einen append-Befehl in die Lokliste aufgenommen werden.
create(10, append)	<REPLY create(10, append)> 10 id[1007] <END 0 (OK)>	
create(10, addr[10], name[„Test"], protocol[DCC28], append)	<REPLY create(10, addr[10], name[„Test"], protocol[DCC28], append) 10 id[1008] <END 0 (OK)>	Es wird eine Lok mit der Adresse 10, dem Lokname „Test" und dem Protocol DCC28 angelegt.
create(10, append)	<REPLY create(10, append)> <END 0 (NERROR_OK>	Es wird eine neue Lok mit Standardwerten angelegt.
Da beim create-Befehl der Lokmanager adressiert wird (ID=10), wird eine neue Lok angelegt. Anschließend wird dieser Lok die Adresse 5 und der Name „Big Boy" gegeben. Zu jedem create-Befehl gehört ein append-Befehl, der das neue Objekt in die Liste der Objekte aufnimmt. Erst nach diesem append-Befehl können andere Teilnehmer auf dieses neu angelegte Objekt zugreifen.		

5.5. delete(id, options, ...)

Ein existierendes Objekt wird gelöscht. Objekte mit festen IDs können nicht gelöscht werden. Um ein Objekt löschen zu können, muss sich der Teilnehmer erfolgreich als Control an diesem Objekt angemeldet haben.

Beispiel:

delete(1005)	<REPLY delete(1000)> <END 25 (NERROR_NOCONTROL)>	Fehler: der Teilnehmer hat keine Control auf dieses Objekt.
request(1005, control)	<REPLY request(1005, control)> <END 0 (OK)>	
delete(1005)	<REPLY delete(1005)> <END 0 (OK)>	
Das Objekt mit der ID 1005 wird gelöscht. Beim ersten Versuch hatte der Teilnehmer noch keine Control auf dieses Objekt. Beim Löschen eines Objektes wird anschließend automatisch die Control und eventuell eine View auf dieses Objekt gelöscht.		

5.6. request(id, options, ...)

Ein Client kann sich auf zwei Arten bei einem Objekt registrieren.

Registriert sich ein Client als Viewer (Option view), so wird der Client automatisch über Änderungen am jeweiligen Objekt durch Events informiert.

Registriert sich ein Client als Controller (Option control), so darf der Client Änderungen am Objekt vornehmen. Befindet sich eine Lok z.B. auf dem Fahrbildschirm der ECoS und wird von dort gesteuert (der Teilnehmer Fahrbildschirm besitzt die Control auf dieses LokObjekt), bekommt ein Client bei einem request(ID, control) eine Fehlermeldung zurück. So kann z.B. eine Lok nicht an der ECoS gesteuert werden, solange sie unter der Kontrolle eines Handreglers ist. Erst wenn die Control auf diese Lok freigegeben wird, kann ein Client sich eine Control auf diese Lok holen.

Jeder Client kann sich jeweils nur einmal an einem Objekt als Controller und/oder Viewer registrieren. Eine Registrierung als Viewer ist immer möglich, solange das Objekt existiert. Eine Registrierung als Controller ist nur dann möglich, wenn sich noch kein anderer Teilnehmer (Handregler, Fahrbildschirm auf der ECoS, Client über das PC-Interface) als Controller an diesem Objekt registriert hat. Die erfolglose Registrierung als Controller wird nicht gespeichert, d.h. ein Teilnehmer bekommt nicht automatisch eine Control, wenn er sich einmal erfolglos als Control registriert hat und das Objekt frei wird. Der Teilnehmer muss erneut versuchen, sich wieder als Control zu registrieren. Wenn ein Teilnehmer eine Control auf ein Objekt frei gibt, wird an alle registrierten Viewer eine Nachricht als Event versendet.

Mit der zusätzlichen option force kann eine Lok von einem anderen Teilnehmer „gestohlen" werden.

Beispiel:

request(1000, view)	<REPLY request(1000, view)> <END 0 (OK)>	
request(1000, control)	<REPLY request(1000, control)> <END 25 (NERROR_NOCONTROL)>	Fehler: ein anderer Teilnehmer hat eine Control auf dieses Objekt.
	<EVENT 1000> 1000 speed[40] <END 0 (OK)>	Der Teilnehmer, der die Control auf dieses Objekt hat, hat die Geschwindigkeit auf den neuen Wert 40 gesetzt. Diese Events werden automatisch an alle registrierten Teilnehmer versendet, die eine View auf dieses Objekt haben.
request(1000, control, force)	<REPLY request(1000, control, force)> <END 0 (OK)>	Dem Teilnehmer, der die Control auf dieses Objekt hatte, wird dieses Objekt „gestohlen".
Ein Teilnehmer registriert sich beim Objekt mit der ID 1000 zunächst als View. Dadurch wird er automatisch über Änderungen an diesem Objekt informiert. Anschließend wird versucht, auch eine Control auf dieses Objekt zu bekommen. Dieser Versuch wird durch einen Fehler quittiert, da ein anderer Teilnehmer (z.B. Fahrbildschirm auf der ECoS) eine Control auf dieses Objekt hat. Durch die zusätzliche Option force wird diesem Teilnehmer jedoch die Control „gestohlen".		

5.7. release(id, options, ...)

Gegenstück zu request. Der Client meldet sich als Viewer (Option view) bzw. Controller (Option control) ab. Alle an diesem Objekt registrierten Viewer erhalten bei einer Abmeldung eines Teilnehmers als Control eine Nachricht.

Beispiel:

release(1000, view, control)	<REPLY release(1000, view, control)> <END 0 (OK)>	
Der Teilnehmer meldet sich vollständig von diesem Objekt ab. Er wird von nun an nicht mehr automatisch über Änderungen an diesem Objekt informiert und der Teilnehmer darf auch keine Änderungen am Objekt mehr vornehmen.		

6. Anhang

Auflistung der Befehle des PC-Interfaces der ECoS. Zu jeder ID werden die entsprechenden Befehle und die unterstützten Optionen aufgelistet.

Es gibt einige konstante IDs wichtiger Basisobjekte. Diese Objekte können auch nicht angelegt oder gelöscht werden. Es ist auch nicht nötig, sich als Control an einem Basisobjekt zu registrieren. In einem Basisobjekt werden Listen über Listenobjekte verwaltet.

6.1. Basisobjekt ECoS (id=1)

request(1, view)	Beim Basisobjekt ECoS als View registrieren
release(1, view)	Als View abmelden
set(1, stop)	Entspricht der Stop-Taste an der ECoS
set(1, go)	Entspricht der Go-Taste an der ECoS
get(1, info)	Informationen über die ECoS: <REPLY get(1, info)> 1 ECoS 1 ProtocolVersion[0.1] 1 ApplicationVersion[1.0.1] 1 HardwareVersion[1.3] <END 0 (OK)>
get(1, status)	Aktuelle Statusinformationen abfragen: <REPLY get(1, status)> 1 Status[val] <END 0 (OK)> val=STOP val=GO val=SHUTDOWN

6.2. Basisobjekt Programmiergleis (id=5) (noch nicht in der ersten Version)

Über dieses Basisobjekt erhält ein Teilnehmer über das PC-Interface Zugriff auf das Programmiergleis der ECoS. Die Funktionalität wird vergleichbar mit der Dekoderprogrammierung im Setup-Menü der ECoS sein.

6.3. Basisobjekt LokManager (id=10)

request(10, view)	Anmelden des Teilnehmers als View. Der Teilnehmer wird automatisch informiert, wenn sich etwas am LokManager ändert. Wird z.B. von einem anderen Teilnehmer eine neue Lok hinzugefügt, so werden alle Teilnehmer, die sich als View registriert haben, hierüber informiert.
release(10, view)	Abmelden eines Teilnehmers als View.
queryObjects(10)	Ausgeben der kompletten Lokliste. Es werden nur die IDs der vorhandenen Loks ausgegeben.
queryObjects(10, name)	Ausgeben der kompletten Lokliste mit IDs und Loknamen.
queryObjects(10, addr)	Ausgeben der kompletten Lokliste mit IDs und Adresse.
queryObjects(10, protocol)	Ausgeben der kompletten Lokliste mit IDs und Protokoll.
queryObjects(10, addr, name)	Ausgeben der kompletten Lokliste mit Adresse und Lokname.
get(10, size)	Es wird nur die Anzahl vorhandener Loks ausgegeben.
queryObjects(10, nr[min, max])	Es werden die (max-min+1) Loks ab der min. Lok ausgegeben.
create(10)	Es wird eine neue Lok angelegt. Diese wird aber noch nicht in die Lokliste aufgenommen und kann von keinem anderen Teilnehmer verwendet werden.
create(10, append)	Die neu angelegte Lok wird in die Lokliste aufgenommen.
create(10, discard)	Die neu angelegte Lok wird verworfen.
create(10, addr[val])	Beim create Befehl kann auch bereits eine Adresse für die neue Lok eingestellt werden.
create(10, name[str])	Beim create Befehl kann auch bereits ein Lokname für die neue Lok vorgegeben werden.
create(10, protocol[val])	Beim create Befehl kann auch bereits das Protokoll für die neue Lok eingestellt werden.
get(10, id)	Die ID der neu anzulegenden Lok wird ausgegeben.
create(10, consist)	Neue Mehrfachtraktion

6.4. Basisobjekt SchaltartikelManager (id=11)

request(11, view)	Anmelden des Teilnehmers als View. Der Teilnehmer wird automatisch informiert, wenn sich etwas am Schaltartikel-Manager ändert. Wird z.B. von einem anderen Teilnehmer ein neuer Schaltartikel angelegt, so werden alle Teilnehmer, die sich als View registriert haben, hierüber informiert.
release(11, view)	Abmelden eines Teilnehmers als View.
queryObjects(11)	Ausgeben der kompletten Liste vorhandener Schaltartikel.
queryObjects(11, name1)	Ausgeben der kompletten Liste vorhandener Schaltartikel mit IDs und Name1 (name2 und name3 sind auch möglich).
queryObjects(11, addr)	Ausgeben der kompletten Liste vorhandener Schaltartikel mit IDs und Adresse (addrext ist auch möglich).
get(11, size)	Es wird nur die Anzahl vorhandener Schaltartikel ausgegeben.
queryObjects(11, nr[min, max])	Es werden die (max-min+1) Schaltartikel ab dem min. Schaltartikel ausgegeben.
create(11)	Es wird ein neuer Schaltartikel angelegt.
create(11, append)	Der neu angelegte Schaltartikel wird in die Liste vorhandener Schaltartikel aufgenommen.
create(11, discard)	Der neu angelegte Schaltartikel wird verworfen.
create(11, route)	Neuer Fahrweg.

set(11, switch[ProtocolAddrPort])	Direktes setzen eines Ports. Protocol ist entweder MOT oder DCC. Port ist entweder r oder g (z.B. set(11, switch[MOT10r])).Existiert zu dieser Adresse ein Objekt, wird dieses entsprechend der programmierten Adressen geschaltet. Existiert zu dieser Adresse kein Objekt, wird dieser Befehl an das Gleis ausgegeben. Wenn Protocol weg gelassen wird, wird das voreingestellte Defaultprotocol verwendet.
get(11, switch[ProtocolAddrPort])	Auslesen der aktuellen Einstellung eines Magnetartikelports. Existiert zu dieser Adresse kein Objekt und wurde an diese Adresse noch kein set-Befehl gesendet, wird eine Fehlermeldung zurückgeliefert. Wenn Protocol weggelassen wird, wird das voreingestellte Defaultprotocol verwendet.

6.5. Basisobjekt Pendelzugsteuerung (id=12) (noch nicht in der ersten Version)

queryObjects(12)	Ausgeben der kompletten Liste vorhandener Pendelzugstrecken.
queryObjects(12, name)	Ausgeben der kompletten Liste vorhandener Pendelzugstrecken mit IDs und Namen.
queryObjects(12, size)	Es wird nur die Anzahl vorhandener Pendelzugstrecken aus gegeben.
queryObjects (12, nr[min, max])	Es werden die (max-min+1) Pendelzugstrecken ab der min. Pendelzugstrecke ausgegeben.
create(12)	Es wird eine neue Pendelzugstrecke angelegt.
create(12, append)	Die neu angelegte Pendelzugstrecke wird in die Liste vorhandener Pendelzugstrecken aufgenommen.
create(12, discard)	Die neu angelegte Pendelzugstrecke wird verworfen.

6.6. Basisobjekt Devicemanager (id=20) (noch nicht in der ersten Version)

request(20, view)	Als View registrieren.
release(20, view)	Als View abmelden.
queryObjects(20)	Es wird eine Liste aller am ECoSlink angeschlossener Geräte ausgegeben. Es wird automatisch auch eine kurze Beschreibung über das Gerät mit ausgegeben.
queryObjects(20, size)	Anzahl vorhandener Geräte ausgeben.
queryObjects(20, nr[min, max])	Teilliste ausgeben.

Dieses Basisobjekt verwaltet die Geräte im System der ECoS. Dadurch wird es möglich sein, dass ein Teilnehmer über das PC-Interface angeschlossene Geräte konfigurieren kann. Z.B. kann eine PC-Software die Lokliste der angeschlossenen Handregler beeinflussen. Bei der Ausgabe der Liste wird neben der ID auch eine Beschreibung des Gerätes gesendet da es sich bei den hier verwalteten Listenobjekte um unterschiedliche Listenobjekte handeln kann.

6.7. Listenobjekt Sniffer (id=25)

request(25, view)	Als View registrieren. Der Teilnehmer bekommt automatisch eine Nachricht über eingetroffene Pakete am Sniffer.
release(25, view)	Als View abmelden.

Über dieses Basisobjekt soll ein Zugriff direkt auf den Sniffer erfolgen. Dadurch können auch die nicht mit einer Lok verknüpften Pakete am Sniffer durch einen Teilnehmer über das PC-Interface ausgewertet werden.

Dieses Objekt wird auch als ein Listenobjekt im Basisobjekt DeviceManager verwaltet.

6.8. S88Manager (id=26)

request(26, view)	Teilnehmer als View registrieren.
release(26, view)	Als View abmelden.
queryObjects(26)	Liste vorhandener S88-Module ausgeben.
get(26, size)	Anzahl vorhandener S88-Module ausgeben.
queryObjects(26, nr[min, max])	Teilliste vorhandener S88-Module ausgeben.
queryObjects(26, ports)	Ausgabe der S88-Module mit IDs und Anzahl verfügbarer Ports.
delete(26, del[pos])	S88-Modul an Position pos wird gelöscht.
create(26, add[pos, ports])	Neues S88-Modul an Position pos erzeugen. Der optionale Parameter ports setzt für das S88-Modul die Portanzahl.
set(26, size[val])	Neues S88-Modul hat val Ports.

Über dieses Objekt werden die S88-Module konfiguriert. Dieses Objekt wird auch als ein Listenobjekt im Basisobjekt DeviceManager verwaltet, es ist aber wiederum ein Basisobjekt und verwaltet die vorhandenen S88-Module in einer Liste. Die IDs der S88-Module hängt von der Position innerhalb des S88-Buses ab. Das erste Modul hat immer die ID 100, das darauffolgende die ID 101 usw.

6.9. Listenobjekt Booster (id=27) (noch nicht in der ersten Version)

set(27, delay[val])	Setzt die Verzögerung für die Kurzschlusserkennung für das in der ECoS integrierte Booster-Modul.
get(27, delay)	Auslesen der Verzögerung der Kurschlusserkennung.

Dieses Listenobjekt verwaltet das in der ECoS integrierte Booster-Modul. Dieses Modul hat als einzigsten Parameter die Verzögerung bei der Kurzschlusserkennung.

Dieses Objekt wird auch als ein Listenobjekt im Basisobjekt DeviceManager verwaltet.

6.10. Basisobjekt Stellpult (id=31) (noch nicht in der ersten Version)

Über dieses Basisobjekt soll ein Zugriff direkt auf das Stellpult der ECoS möglich sein. Dadurch könnte ein Teilnehmer am PC-Interface z.B. die Belegung der einzelnen Tabs für die Schaltartikel konfigurieren.

6.11. Listenobjekt Lok (id=dynamisch)

request(id, view)	Teilnehmer als View registrieren.
request(id, control)	Teilnehmer als Control registrieren.
request(id, control, force)	Wenn ein anderer Teilnehmer bereits eine Control auf dieses Objekt hat, kann von diesem Teilnehmer „gestohlen" werden.
release(id, view)	Als View abmelden.
release(id, control)	Als Control abmelden.
get(id, addr)	Ausgeben der Lok-Adresse.
set(id, addr[val])	Setzen der Lok-Adresse.
get(id, name)	Ausgeben des Loknamens.
set(id, name[str])	Setzen des Loknamens.
get(id, protocol)	Verwendetes Protokoll ausgeben.
set(id, protocol[val])	Setzen des Protokolls. val kann folgende Werte annehmen: MM14, MM27, MM28, DCC14, DCC28, DCC128, SX32, MMFKT
get(id, symbol)	Ausgeben des Symbols der Lok.
set(id, symbol[nr])	Setzen des Loksymbols.
get(id, profile)	Dekoderprofil ausgeben.
set(id, profile[val])	Dekoderprofil setzen. Umschalten des Profils setzt CVs auf die in der ECoS gespeicherten default Werte.
get(id, sniffer)	Snifferadresse ausgeben.
set(id, sniffer[addr])	Snifferadresse setzen.
get(id, favorit)	Ausgeben, ob Lok zu den Favoriten gehört.
set(id, favorit[val])	Lok als Favorit kennzeichnen.
get(id, speedindicator)	Die Anzeige der Geschwindigkeit erfolgt in km/h. Der Wert speedindicator gibt dabei die Geschwindigkeit in km/h bei maximaler Fahrstufe an. Ist dieser Wert 0 erfolgt die Anzeige der Geschwindigkeit in Fahrstufen.
set(id, speedindicator[val])	Setzen der Geschwindigkeit in km/h bei maximaler Fahrstufe.
get(id, cv[nr])	CV nr ausgeben.
set(id, cv[nr, val])	CV nr auf val setzen. Die neuen Werte werden nur in der ECoS gespeichert und nicht auf die Dekoder programmiert.
get(id, speed)	Ausgeben der aktuellen Geschwindigkeit der Lok (normiert auf 127 Fahrstufen).
set(id, speed[val])	Setzen der Geschwindigkeit der Lok (normiert auf 127 Fahrstufen).
get(id, speedstep)	Ausgeben der aktuellen Geschwindigkeit in Fahrstufen je nach Protokoll.
set(id, speedstep[val])	Setzen der aktuellen Geschwindigkeit in Fahrstufen je nach Protokoll.
get(id, dir)	Ausgeben der aktuellen Richtung der Lok (1=rückwärts).
set(id, dir[val])	Setzen der aktuellen Richtung der Lok (1=rückwärts).
get(id, func[nr])	Ausgeben des Zustands der Funktion nr.
set(id, func[nr, val])	Setzen der Funktion nr auf den Wert val (bei digitalen Ausgängen kann val 0 oder 1 sein).
get(id, funcsymbol[nr])	Ausgeben des Funktionssymbols für Funktion nr.
set(id, funcsymbol[nr, val, moment])	Setzen des Funktionssymbols für Funktion nr. Mit dem optionalen Parameter moment wird diese Funktionstaste als Momentfunktion programmiert. Mit val=-1 wird diese Funktion ausgeblendet.
set(id, stop)	Nothalt.
link(id, id[id2])	Füge Lok mit ID id2 zur Multitraktion hinzu oder aktiviere für diese Lok die Pendelzugsteuerung mit der ID id2.
unlink(id, id[id2])	Lösche Lok mit der id2 aus der Multitraktion oder deaktiviere für diese Lok die Pendelzugsteuerung mit der ID id2.
delete(id)	Lösche diese Lok.
queryObjects(id)	Ausgeben der mit dieser Lok verknüpften Objekte (Loks in einer Multitraktion oder Pendelzugsteuerung).

6.12. Listenobjekt Schaltartikel (id=dynamisch) (noch nicht in der ersten Version)

request(id, view)	Teilnehmer als View registrieren.
request(id, control)	Teilnehmer als Control registrieren.
request(id, control, force)	Wenn ein anderer Teilnehmer bereits eine Control auf dieses Objekt hat, kann von diesem Teilnehmer „gestohlen" werden.
release(id, view)	Als view abmelden.
release(id, control)	Als control abmelden.
get(id, state)	Liefert den aktuellen Zustand des Schaltartikels zurück.
set(id, state[val])	Setzt den Zustand des Schaltartikels auf val.
get(id, addr)	Ausgeben der Adresse des Schaltartikels.

get(id, addrext)	Erweiterte Adressierung ausgeben. Beispiel: addrext[3g, 3r, 4g, 4r] Dieser Magnetartikel ist mit einem Antrieb an Adresse 3 angeschlossen und mit einem Antrieb an Adresse 4. Soll ein Antrieb umgekehrt werden muß die Adresse mit addrext[3r, 3g, 4g, 4r] gesetzt werden. Wurde eine erweitert Adressierung programmiert, wird bei einem get(id, addr) die erweiterte Adressierung ausgegeben. Wird keine erweiterte Adressierung verwendet, so werden die 4 Ports auf die entsprechenden default Werte gesetzt (addr[3] ergibt dann addrext[3g, 3r, 4g, 4r].
set(id, addr[val])	Setzen der Adresse.
set(id, addrext [val1, val2, val3, val4])	Erweitere Adressierung. Pro Magnetartikel können bis zu 4 Ports eines Dekoders programmiert werden.
get(id, protocol)	Ausgeben des verwendeten Protokolls.
set(id, protocol[val])	Setzen des Protokolls (val entweder MM oder DCC)
get(id, name1)	Ausgeben der ersten Zeile des Namens.
set(id, name1[str])	Setzen der ersten Zeile des Namens.
get(id, name2)	Ausgeben der zweiten Zeile des Namens.
set(id, name2[str])	Setzen der zweiten Zeile des Namens.
get(id, name3)	Ausgeben der dritten Zeile des Namens.
set(id, name3[str])	Setzen der dritten Zeile des Namens.
get(id, symbol)	Ausgeben des Symbols.
set(id, symbol[nr])	Setzen des Symbols.
get(id, mode)	Ausgeben ob Schaltartikel Impuls (PULSE) oder Umschalten (SWITCH) ist.
set(id, mode[val])	Setzen des Schaltverhaltens (PULSE oder SWITCH).
get(id, duration)	Ausgeben der Schaltdauer.
set(id, duration[val])	Setzen der Schaltdauer (val entweder 250, 500, 750, 1000 oder 2500).
link(id, id[id2], state[val])	Schaltartikel mit der ID id2 und dem Zustand val zu diesem Fahrweg hinzufügen. Wenn state nicht angegeben wird, wird der aktuelle Zustand des Objekts mit der ID id2 verwendet.
unlink(id, id[id2])	Schaltartikel mit der ID id2 aus diesem Fahrweg entfernen.
delete(id)	Schaltartikel löschen.
queryObjects(id)	Ausgeben der zu diesem Fahrweg gehörenden Schaltartikel.

6.13. Listenobjekt S88Modul (id=dynamisch)

request(id, view)	Teilnehmer als View registrieren.
release(id, view)	Als View abmelden.
get(id, ports)	Port-Anzahl ausgeben.
set(id, ports[val])	Port-Anzahl auf 8 oder 16 setzen.
delete(id)	S88-Modul löschen
get(id, state)	Liefert den aktuellen Zustand des S88-Moduls zurück.

6.14. Listenobjekt Pendelzugstrecke (id=dynamisch)

get(id, station[nr])	S88-Information des Bahnhofs nr ausgeben (nr kann 1 oder 2 sein).
set(id, station[nr, m:p])	Bahnhof nr hat S88-Modul m und S88-Port p (nr kann 1 oder 2 sein).
get(id, name)	Namen der Pendelzugstrecke ausgeben.
set(id, name[str])	Namen der Pendelzugstrecke setzen.
get(id, delay)	Aufenthaltsdauer ausgeben.
set(id, delay[val])	Aufenthaltsdauer setzen.
delete(id)	Pendelzugstrecke löschen.