Network specification for the PC interface

For protocol l -ver s ion 0.2 Dr te it on

l age, June 2011

1. License Agreement

1.1. General
This document method for communication between a network participants (the "Client") and a network service provider (hereinafter "server" called) described. In the following, these methods are referred to as "Protocol". All rights, in particular the copyright of the Protocol are (hereinafter referred to as "ESU") solely in electronic solutions ulm GmbH & Co. KG. © Copyright 2006-2011 ESU.

1.2. Termination of the license
This license text is a letter addressed to anyone offer to conclude a license agreement. The license agreement is concluded on a data processing system by implementing the protocol. The user shall be royalty free non-exlusive, limited and non-transferable contracts.

The implementation of the protocol in a client software for PC systems (eg, home computers or laptops) is allowed without restriction.

1.3. prohibited use
It is strictly forbidden to implement the protocol in a server. In particular, here under the use of the protocol fall within a suitable model for the control of toys device.

The licensee is also prohibited to develop procedures on the basis of the Protocol further communication. This explicitly includes the modification of the existing prototype Koll.

Also prohibited is the implementation of the protocol in software men run on different PC systematically.

1.4. limitations
It is the licensee and prohibited third parties to pass on this document or made publicly available in another form. Content is licensed, not sold.

1.5. Caveat Emptor
We assume no responsibility or liability for the accuracy, content, completeness, reliability, operability, or fitness for a particular purpose of the protocol. In particular, any claims for damages to ESU can be asserted that result from the use of the protocol.

The protocol is provided without warranty of any kind "as is" available. For the record, no support is guaranteed.

1.6. Changes to protocol and license conditions
ESU has the right to change the protocol at any time without notice. In addition, ESU is Untitled calculated to change the terms of this agreement unilaterally.

1.7. choice of law
All legal relations between the parties, including tort law, the law of the Federal Republic of Germany. The jurisdiction is Ulm.

1.8. final provision
If any provision of this contract be invalid or unenforceable invalid or unenforceable after the conclusion, the validity of the remaining provisions will not be affected. such valid and enforceable provision shall be replaced, are the effects of the economic objectives as closely as possible who have been following the Parties with the ineffective beziehungswei- se unenforceable provision to replace the invalid or unenforceable provision.

The above provisions apply in the event that the contract proves to be patchy. § 139 BGB does not apply.

2. connection

TCP socket connection to port 15,471th

Receive buffer is 1024 bytes in size, commands must not be split. but it may be several complete commands in a package. Can not be interpreted a command within the receive buffer, the remaining receive buffer is discarded.

3. objects

Access to the ECoS objects. These objects are ad-ested by a unique ID. There are numerous basic objects with constant IDs that are used for configuration and manage- ment of other objects. In the so-called object managers list objects are managed. The LokManager (base object) with the ID 10 manages eg all locos (list object). About this LokManager the IDs of the list objects can be queried and also new list items are added.

4. Client / participant

The ECoS is a central server. All objects must be that logged on to the ECoS or registered on this server. When connecting a speed controller with ECoS this example is automatically logged in to the ECoS and gets as the locomotives last visited on this hand held controller assigned. Each participant can register in two different ways at one or more objects. A control is necessary to manipulate an object. A view allows monitoring of an object. A handheld controller with a locomotive list of 10 locomotives is, for example, register at all locomotives as view and a locomotive to be controlled additionally as Control. only one user can have a control on an object in the system, ie only one participant may manipulate an object at the same time.

By registering a participant to an object as a view, these participants will be informed automatically about changes to this object via so-called events. A participant who has registered as a controller and as a view of an object is not informed of changes to the object.

5. News

For communication via the PC interface with ASCII protocol is used. Strings such as names are encoded UTF8 and are always enclosed in quotes. Quotes in strings must be sent twice to parse quotation marks in strings better.

Every communication is confirmed by the ECoS with error code (reply). Messages that are triggered by the ECoS (events) must not be confirmed by the PC. When the end of line, a new line (n \, ASCII = 10) used may also, before this new line, a carriage return (\ r, ASCII = 13) when commands are provided.

Each message is always addressed to one, marked by a unique ID, object. There are a number of pre-defined IDs (basic objects) and dynamically assigned IDs (Listenobjek- te). So affect all messages to the ID 10 to LokManager called or ID 1, the basic functions of the ECoS. A list of fixed predetermined IDs based objects found in the appendix. For the IDs of 16 bits are used and IDs are always unique.

5.1. Building messages

Each message from the ECoS to the PC has the form

<HEADER cmd>

TEXT

<END x (str)>

HEADER either REPLY or EVENT.
A REPLY indicates a response to a request from the PC interface, such as the interrogation of the name of a locomotive.

An EVENT is generated by the ECoS message, such as the information to the PC that the speed of an engine changed.

In addition, the command header is repeated with cmd to which the message relates. In an EVENT cmd contains the ID of the object.

END END is always followed by a numeric error code x and in brackets dazugehöri- ge error description st. Error code 0 (str = OK) means that the command could be executed without error.

Between the header and the end (TEXT) may be several lines corresponds hold additional information. Each row in TEXT here has the structure:

id option

In id is the ID of the object and option is available for additional information. Each command from the

PC to the ECoS has the form

command (id [, options [, ...]])

The options options in turn can have optional parameters in square brackets ([]). up to 10 items are allowed per command. The options options apply to all commands to any IDs, whereby all options of each object can not be interpreted.

6. commands

The following is a description of the commands to the ECoS in general terms. The first argument is always an ID on an object of ECoS is addressed. After the ID up to 10 additional arguments (options) may follow.

In describing the objects supported by the respective object options are tert erläu-. The duration option is supported only example of a switching products object. lenaufbau following Tabel- used in the examples for each command:

Command to the ECoS	ECoS response to this command	comment
Description of this example,		

6.1. query objects (id, options, ...)

This command returns a list of objects that belong to the object with the ID id. With the option size only the number of elements is returned instead of the entire list. With the option nr [min, max] one of the list can be requested. Example:

query objects (10, name)	<REPLY query objects (10, name)> 1000 name ["Big Boy"] <END 0 (OK)>	
It is the object 10 (LokManager) requested a list of objects. In this case it is the entire locomotive list of the ECoS. In response to a query command objects, the objects are identified by the IDs of the objects. This example is in the locomotive list of ECoS only a locomotive with the ID 1000. By options in the query objects command the returned list of objects can be made more specific. In the above example is the option name in addition to the object ID nor the name of the object, with output in this case the name of the locomotive.		

Example:

query objects (10, size)	<REPLY query objects (10 size)> 10 size [1] <END 0 (OK)>	
It is the object 10 (LokManager) requested the number of objects. This example is in the locomotive list of ECoS only one locomotive.		

Example:

query objects (10, nr [0.3]) <REPLY query objects (10, nr [0, 3])>	1000 1001 1005 1006 <END 0 (OK)>	
It requested the first four entries in the list of object 10 (LokManager). In this example, are located in the locomotive list of ECoS least 4 locomotives with the ID 1000, 1001, 1005 and 1006th		

6.2. set (id, options, ...)

This command individual properties of an object can be set. Since at least one property is set, and at least two parameters (ID and at least one) are necessary for this command. As a rule, each option will have an additional parameter in square brackets, which contains the new value. Example:

set (1000, addr [3])	<REPLY set (1000, addr [3])> 1000 addr [3] <END 0 (OK)>	
The object with the ID 1000 address 3 is assigned.		

Example:

set (1000, speed [12])	<REPLY set (1000, speed [12])> 1000 speed [12] <END 0 (OK)>	
The speed option is set for the object with the ID 1000 on the 12th If the object with the ID 1000 is a locomotive, it then proceeds are always normalized in the ECoS to 127 speed steps, with a speed 1 means an emergency stop if the decoder supports this with the speed 12 speeds.		

6.3. get (id, options, ...)

This command individual properties of an object can be queried. Example:

get (1000, name, speed)	<REPLY get (1000, name, speed)> 1000 name ["Big Boy"] 1000 speed [12] <END 0 (OK)>	
It is requested in 1000 the name and the speed of the object with the ID.		

6.4. create (id, options, ...)

It is created a new object. Since the object ID is assigned by the ECoS, a new object in the corresponding object manager must be created. A new locomotive is thus applied with an ID 10 (LokManager). then the object ID of the object manager must be used to configure the newly created object. It can thus be configured on an object manager by a participant only a new object. Only after the creation of the new building is completed, a new object can be created again by the same participant. The application of a command object is terminated with a request (id, append). This command assigns a new ID for the object and from this point on is the new object in the list of list of objects in the object manager. Example for applying a driving:

create (10)	<REPLY create (10)> 10 id [1007] <END 0 (OK)>	It is applied an object with the ID 1007th This object is visible to other participants only after append command.
set (1007 addr [5])	<REPLY set (1007, addr [5])> 1,007 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)>	Error: the previously generated locomotive has to be taken into the locomotive list by an append command.
create (10, append)	<REPLY create (10, append)> 10 id [1007] <END 0 (OK)>	
create (10, addr [10], name ["Test"], protocol [DCC 2 8], append)	<REPLY create (10 addr [10], name ["Test"] protocol [DCC28] append) 10 id [1008] <END 0 (OK)>	There is applied a locomotive with the address 10, the locomotive name "Test" and the Protocol DCC28.
create (10, append)	<REPLY create (10, append)> <END 0 (NERROR_OK>	This creates a new locomotive with default values.
Since the CREATE command is addressed to the Lokmanager (ID = 10), a new train is created. Then this locomotive is given the address 5 and the name "Big Boy". For each create command has a append command that receives the new object in the list of objects. Only after this append command other participants have access to this newly created object.		

6.5. delete (id, options, ...)

An existing object is deleted. Objects with fixed IDs can not be deleted. To delete an object, the participant must have logged on to this object as a successful control. Example:

delete (1005)	<REPLY delete (1000)> <END 25 (NERROR_NOCONTROL)>	Error: the user has no control on this object.
request (1 0 0 5, control)	<REPLY request (1005, control)> <END 0 (OK)>	
delete (1005)	<REPLY delete (1005)> <END 0 (OK)>	
The object with the ID 1005 is deleted. The first attempt of the participants had no control to that object. When you delete an object, the Control and possibly a view is automatically deleted to this object then.		

6.6. request (id, options, ...)

A client can register in two ways for an object.

a client registers as a viewer (Option view), the client is automatically informed of changes to the respective object by events.

a client registered as a controller (option control), so the client can make changes to the object. a locomotive is eg on the cab display of ECoS and from there con- trolled (participants traveling screen has the control on this LokObjekt), a client gets back an error message when a request (ID, control). For example, can not be controlled by the ECoS a locomotive, as long as it is under the control of a manual controller. Only when the control is released on the locomotive, a client can retrieve Control in this loco. Each client can only appear once on an object as a controller and / or viewer will register reindeer. Registration as a viewer is always possible as long as the object exists. Registration as a controller is only possible if no other participants signed up (speed controller, Fahrbild- screen on the ECoS client via the PC interface) as controller in this property. The unsuccessful registration as a controller is not saved, that a participant does not automatically get a control if he has once unsuccessfully registered as Control and the object is free. The participant must try to register again as a control again. If a participant is a control to an object-free, a message is sent as an event to all registered viewer.

With the additional option force a locomotive can "steal" from another party advertising to.

Example:

request (1000, view)	<REPLY request (1000 view)> <END 0 (OK)>	
request (1000, control)	<REPLY request (1000, control)> <END 25 (NERROR_NOCONTROL)>	Error: another participant has a Control on this object.
	<EVENT 1000> 1000 speed [40] <END 0 (OK)>	The participant who has the control to that object has, the speed set to the new value 40th These events are automatically sent to all registered participants who have a view on that object.
request (1000, control, force)	<REPLY request (1000, control, force)> <END 0 (OK)>	The participant who had the control on this object, "stolen" this object.
A subscriber first registers with the object with the ID 1000 as a view. Thus he is automatically informed of changes to this object. It then attempts to get a control on this object. This test is acknowledged by an error because another participant (eg driving screen on the ECoS) a Control on this object. The additional force option that participant but "stolen" the Control.		

6.7. release (id, options, ...)

Counterpart to request. The client logs out (option con- trol) as viewer (Option view) or controller. All registered on this object viewer obtained at a log off a station as a control message. Example:

release (1000, view, control)	<REPLY release (1000, view, control)> <END 0 (OK)>	
The user answers entirely on this object. He will no longer be automatically informed by now about changes to this object and the participant is allowed to make no changes to the object more.		

7. Appendix

Listing of the PC interface of the ECoS commands. To each ID the appropriate commands and the supported options are listed.

There are some constant IDs important basic objects. These objects can not be created or deleted. It is also not necessary to register as a Control on a base object. In a base object lists are managed over list objects.

7.1. ECoS base object (id = 1)

request (1, view)	register with the base object ECoS as view.
release (1, view)	logout of View
delete (1005)	<REPLY delete (1005)> <END 0 (OK)>
set (1, stop)	Corresponds to the Stop button on the ECoS
set (1, go)	Corresponds to the Go button on the ECoS
get (1 info)	Information on the ECoS: <get REPLY (1 info)> 1 ECoS 1 Protocol version [0.1] 1 Application Version [1.0.1] one hardware version [1.3] <END 0 (OK)>
get (1, status)	Get the current status information: <REPLY get (1, status)> 1 Status [val] <END 0 (OK)> val = val = STOP GO val = SHUTDOWN

7.2. Base object programming track (id = 5) (planned)

This base object, a participant receives via the PC interface access to the programming track ECoS. The functionality will be similar to the decoder programming in the ECoS setup menu.

7.3. LokManager base object (id = 10)

request (10, view)	Sign of the participant as a view. The participant is automatically informed when something changes on LokManager. If, for example added by another participant a new locomotive, all participants who have registered as view will be informed accordingly.
release (10, view)	Logout of a participant as a view.
query objects (10)	Outputting the complete locomotive list. the IDs of existing engines is only replayed.
query objects (10, name)	outputting the entire locomotive list of IDs and Locomotive name.
query objects (10, addr)	outputting the entire locomotive list of IDs and Address.
query objects (10, protocol)	outputting the entire locomotive list of IDs and Protocol.
query objects (10, addr, name)	Outputting the complete locomotive list with address and name of the locomotive.
get (10 size)	It is issued only the number of existing locomotives.
query objects (10, nr [min, max])	There are the (max-min + 1) locomotives from the min. Lok output.
create (10)	This creates a new locomotive. However, this is not yet included in the locomotive list and can be used by any other party.
create (10, append)	The newly entered locomotive is taken into the locomotive list.
create (10, discard)	The newly entered locomotive is discarded.
create (10 addr [val])	When create command may already address be set for dieneue locomotive.
create (10, name [str])	When create command can also already locomotive name are set for the new locomotive.
create (10, protocol [val])	When create command may already Protocol be set for the new locomotive.
get (10 id)	The ID of the newly created Lok is issued.
create (10, consist)	New consist.

7.4. Base object switching Product Manager (id = 11)

request (11, view)	Sign of the participant as a view. The participant is automatically informed when something changes on switching product manager. If, for example created by another user, a new switching products, all participants who have registered as view will be informed accordingly.
release (11, view)	Logout of a participant as a view.
query objects (11)	Outputting the complete list of existing switching products.
query objects (11, name1)	Outputting the complete list of existing switching products with IDs and name1 (name2 and name3 are also possible).
query objects (11, addr)	Outputting the complete list of existing switching products with IDs and address (AddrExt is also possible).
get (11 size)	It is issued only the number of existing switching products.
query objects (11, nr [min, max])	There are the (max-min + 1) switching products from min. Article switching output.
create (11)	It is created a new switching products.
create (11, append)	The newly created switch item is added to the list of existing switching products.
create (11, discard)	The newly created switch item is discarded.
create (11, route)	New track.

set (11, switch [ProtocolAddrPort]) Direct contact of a port protocol. Protocol is either r or g (for example, set (11, switch [MOT10r])). Exists to this address an object, this is switched according to the programmed address. If at this address no object, this command is issued to the track. If Protocol is omitted, the default Defaultprotoco is used.	MOT DCC Motorola
get (11, switch [ProtocolAddrPort]) reading the current settings of a ports. If at this address no object and was not sent to this address not set command, an error message is returned. If Protocol is omitted, the default default Protocol is used.	

7.5. Base object shuttle train (id = 12) (planned)

query objects (12)	Outputting the complete list of existing shuttle trains.
query objects (12, name)	Outputting the complete list of existing shuttle trains with IDs and names.
query objects (12, size)	It becomes just the number existing Shuttle trains given out.
query objects (12, nr [min, max])	There are the (max-min + 1) shuttle train from the min. Shuttle Train output.
create (12)	This creates a new shuttle route.
create (12, append)	The newly created shuttle route is added to the list before handener shuttle trains.
create (12, discard)	The newly created shuttle route is discarded.

7.6. Base object Device Manager (id = 20) (planned)

request (20, view)	Register as a View.
release (20, view)	logout of view.
query objects (20)	It is all spent on a list ECoSlink connected devices. It will automatically output a brief description of the unit.
query objects (20, size)	Output number of existing devices.
query objects (20, nr [min, max])	Parts list output.

This basic object manages the devices in the system of the ECoS. This will make it possible that a subscriber can configure connected devices via the PC interface. For example, a PC software the locomotive list of connected handhelds influence. When issuing the list next to the ID and a description of the device is sent as there may be different list objects under management here list objects.

7.7. Sniffer list Object (id = 25)

request (25, view)	Register as a View. The participant automatically receives a message via arrived packets at the sniffer.
release (25, view)	logout of view.

This base object access should be made directly to the sniffer. This also is not associated with a locomotive packets can be evaluated at the Sniffer by a subscriber via the PC interface.

This property is managed as a list object in the base object Device Manager.

7.8. Feedback Manager (id = 26)

request (26, view)	register participants as a view.
release (26, view)	logout of view.
query objects (26)	Output list of existing feedback modules.
get (26 size)	Output number of existing feedback modules.
query objects (26, nr [min, max])	Parts list existing Feedback modules output.
query objects (26, ports)	Output of the feedback modules with IDs and number of available ports.
delete (26 del [pos])	S88 module at position pos is deleted. Not available for ECoSDetector.
generate create (26, add [pos, ports])	New S88 module at position pos. The optional parameter ports is the port number for the S88 module. Not available for ECoSDetector.
set (26, size [val])	New S88 module has val ports.

On this object, the feedback modules (s88 and ECoSDetector) to be configured. This Obwalden ject is managed as a list object in the base object Device Manager, but it is again a base object and manages the existing feedback modules in a list. The IDs of the S88 modules depends on the position within the S88 bus. The first module always has the ID 100, the subsequent ID 101, etc. The IDs of ECoSDetector modules depends on the ECoSDetector number. The ECoSDetector with the number 1 has ID 200, with number 2, etc. ID 201

7.9. List object Booster (id = 27) (planned)

set (27, delay [val])	Set the delay for the short-circuit detection for the built-in ECoS booster module.
get (27, delay)	detection reading out the delay of the Kurschluss-.

This list object manages the integrated into the ECoS booster module. This module has as a one-fifth parameter the delay in the short-circuit detection. This property is managed as a list object in the base object Device Manager.

7.10. Base object Control Panel (id = 31) (planned)

This base object access should be possible directly on the control console of the ECoS. Thus, the assignment of the individual tabs for switching items could be a participant in the PC interface eg Center configurable.

7.11. List object Lok (id = dynamic)

request (id, view)	register participants as a view.
request (id, control)	register participants and Control.
request (id, control, force) If another person is already a Control	has to this object can be "stolen" from that user.
release (id, view)	logout of view.
release (id, control)	logout of Control.
get (id, addr)	Outputting the locomotive address.
set (id, addr [val])	Put the locomotive address.
get (id, name)	Outputting the locomotive name.
set (id, name [str])	Setting the locomotive name.
get (id, protocol)	Protocol used to spend.
set (id, protocol [val])	Put the Protocol. val can have the following values: MM14, MM27, MM28, DCC14, DCC28, DCC128, SX32, MMFKT
get (id, profile)	Profile decoder output.
set (id, profile [val])	Decoder profile set. Switching the profile sets CVs on the data stored in the ECoS default values.
get (id, sniffer)	Sniffer address to spend.
set (id, sniffer [addr])	put Sniffer address.
get (id, favorite)	To indicate if the locomotive is one of the favorites.
set (id, favorit [val])	Lok liked this feature.
get (id, speed indicator) The display of the speed is carried out in	km / h. The value of speed indicator thereby indicates the speed in km / h at maximum speed level. If this value is 0, there is the display of speed in speed steps.
set (id, speed indicator [val])	Setting the speed in km / h at maximum speed level.
get (id, cv [nr])	CV nr spend.
set (id, cv [nr, val])	CV nr put on val. The new values are stored only in the ECoS and not programmed to the decoder.
get (id, speed)	Outputting the current speed of the locomotive (normalized to 127 speed steps).
set (id, speed [val])	Setting the speed of the locomotive (normalized to 127 speed steps).
get (id, speedstep)	Outputting the current speed in speed steps depending on the protocol.
set (id, speedstep [val]) set the current speed	Speed levels depending on protocol.
get (id, you)	Outputting the current orientation of the engine (1 = backward).
set (id, you [val])	Setting the current direction of the locomotive (1 = backward).
get (id, func [nr])	Outputting the state of the function nr.
set (id, func [nr, val])	Setting the function number to the value val (at digital outputs can be 0 or 1 eq).
set (id, stop)	Emergency stop.
link (id, id [id2])	Add locomotive with ID id2 to add multi traction or activate the shuttle train with the ID id2 for this locomotive.
unlink (id, id [id2])	Delete locomotive with the id2 from the multi-traction or disable the shuttle train with the ID id2 for this locomotive.
delete (id)	Delete this locomotive.
query objects (id)	Outputting the linked with this locomotive objects (locomotives in a multi traction or shuttle train).
get (id, FUNCDESC [nr])	Query the function description. Return Value: id FUNCDESC [nr, desc [, moment]], with desc out Description List. If the function is a snapshot function, the optional parameter is torque output at.
get (id, funcexists [nr])	Expansion of FUNCDESC. Return Value: id funcexists [nr, desc [, moment]] desc if Not Available -2 if function not assigned -1
set (id, FUNCDESC [nr, desc [, moment]])	Setting the functional description of desc Description List for Function nr. The optional Para Menter moment the function is assigned as a moment function. With desc = -1, the function is turned off.
get (id, locodesc)	Returns locomotive symbol. • id symbol [locotype, image type, image index] with locotype from the values LOCO_TYPE_E, LOCO_ TYPE_DIESEL, LOCO_TYPE_STEAM and LOCO_TYPE_ MISC as a fallback in case can not be used imageindex (see below). image type (from the values SYS IMAGE_TYPE_INT • tembilder) and IMAGE_TYPE_USER (transferred from the user to the device images) imageindex from the Lokbildliste, imageindex • and locotype are firmly linked in the device.
set (id, locodesc [image type, image index])	Sets the locomotive symbol.

12.7. List object switching products (id = dynamic) (planned)

request (id, view)	register participants as a view.
request (id, control)	register participants and Control.
request (id, control, force) If another person is already a Control	has to this object can be "stolen" from that user.

release (id, view)	As view unsubscribe.
release (id, control)	As a control log.
get (id, state)	Returns the current state of the switching item back.
set (id, state [val])	Sets the state of switching article on val.
get (id, addr)	Outputting the address of the circuit article.
get (id, AddrExt)	Advanced Addressing spend. Example: AddrExt [3g, 3r, 4g, 4r] This accessory is connected, and 4 with a drive to address set to a drive at address 3, a drive has to be reversed the address with AddrExt [3r, 3g, 4g, 4r] set become. If an extended addressing programmed to output the extended addressing with a get (id, addr). If no extended addressing is used, the 4 ports are set to the respective default values (addr [3] then gives AddrExt [3g, 3r, 4g, 4r].
set (id, addr [val])	Setting the address.
set (id, AddrExt [Val1, val2, val3, val4]) Upgrade addressing. can be processed	Programmed up to 4 ports of a decoder.
get (id, protocol)	Outputting the protocol used.
set (id, protocol [val])	Set the protocol (val either MM or DCC)
get (id, name1)	Outputting the first line of the name.
set (id, name1 [str])	Put the first line of the name.
get (id, name2)	Outputting the second line of the name.
set (id, name2 [str])	Setting the second line of the name.
get (id, name3)	Outputting the third line of the name.
set (id, name3 [str])	Setting the third line of the name.
get (id, symbol)	Outputting the symbol.
set (id, symbol [nr])	Put the symbol.
get (id, mode)	Outputting whether switching article pulse (PULSE) or switching (SWITCH) is.
set (id, fashion [val])	Setting the switching condition (PULSE or SWITCH).
get (id, duration)	Outputting the switching period.
set (id, duration [val])	Setting the switching period (val either 250, 500, 750, 1000 or 2500).

link (id, id [id2], state [val])	Add switching products with the ID id2 and the condition val on this track. If state is not specified, the current state of the object with the ID id2 is used.
unlink (id, id [id2])	Remove the shift article ID id2 from this path.
delete (id)	delete switching products.
query objects (id)	Outputting the belonging to this track switching products.

7.13. List object feedback module (id = dynamic)

request (id, view)	register participants as a view. Only 100 (s88) / 200 (ECoSDetector).
release (id, view)	logout of view.
get (id, ports)	Port Number spend.
set (id, ports [val])	put port count to 8 or 16th Only s88 modules (IDs 100-163).
delete (id)	delete S88 module. Only s88 modules (IDs 100-163).
get (id, state)	Returns the current state of the Feedback module back.
get (id, railcom [port])	Provides the determined address and the ECoSDetector Aufgleisrichtung of the locomotive on the feedback section (port, address, direction). In section blank or non-RailCom enabled-ECoSDetector inputs becomes 0 returned.

7.14. List object shuttle train (id = dynamic)

get (id, station # [n])	Repeater information output station nr (nr may be 1 or 2).
set (id, station [nr, m: p]) Station number has	S88 module m and S88 port p (nr may be 1 or 2).
get (id, name)	Output names of the shuttle train.
set (id, name [str])	put the name of a shuttle train.
get (id, delay)	Stay spend.
set (id, delay [val])	stay put.
delete (id)	Delete a shuttle train.

8. Value Lists

8.1. Lokbildliste

Type Steam s d e

Diesel Electric m

Misc Index

	grade	description
000	s	symbol
001	d	symbol
002	e	symbol
003	m	symbol
004	d	abj003
005	d	abj004
006	d	bb66000
007	d	br212
008	d	BR218
009	d	br219
010	d	BR232
011	d	br648
012	d	cc40100
013	d	cc72000
014	d	class008
015	d	Desiro
016	d	me026
017	d	picasso
018	d	ram tee
019	d	seemed zeppelin
020	d	svt137
021	d	V060
022	d	V080
023	d	V188
024	d	v200
025	d	V212
026	d	V216
027	d	v300
028	d	v320
029	d	vt008
030	d	vt0115
031	d	vt098
032	d	vt208
033	e	BR103
034	e	br103_red
035	e	BR110
036	e	BR111
037	e	BR143
038	e	BR151
039	e	br151_red
040	d	BR160
041	e	br185
042	e	E044
043	e	E069
044	e	E080
045	e	E091
046	e	E094
047	e	et065
048	e	et087
049	e	ice1
050	e	ice3
051	e	crocodile
052	e	oebb1012
053	e	thalys
054	m	car
055	m	car
056	m	consist_diesel
057	m	consist_e
058	m	consist_s
059	m	controlcar
060	m	crane trucks
061	m	silberling
062	s	bigboy
063	s	BR001
064	s	br003_streamline_black

065	s	br003_streamline_red
066	s	br024
06	s	br038
068	s	br044
069	s	br050
070	s	br051
071	s	br055
072	s	br064
073	s	br075
074	s	br078
075	s	br080
076	s	br085
077	s	br092
078	s	br096
079	s	br098
080	s	glass case
081	d	coef

8.2. Description List

Value	description
0002	function
0003	light
0004	light_0
0005	light_1
0007	sound
0008	music
0009	announce
0010	routing_speed
0011	abv
0032	coupler
0033	steam
0034	panto
0035	high beam
0036	bell
0037	horn
0038	whistle
0039	door_sound
0040	fan
0042	shovel_work_sound
0044	shift
0260	interior_lighting
0261	plate_light
0263	brake sound
0299	crane_raise_lower
0555	hook_up_down
0773	wheel_light
0811	turn
1031	steam-blow
1033	radio_sound
1287	coupler_sound
1543	track_sound
1607	notch_up
1608	notch_down
2055	thunderer_whistle
3847	buffer_sound