Simple Railroad Command Protocol

Version 0.8.3

Edited by

Matthias Trute

2000, 2006 (25.7.2006)

SRCP is an internet protocol for controlling and programming (digital) model railway systems.

1. Introduction

Operating digital model railway kits is to this day strongly influenced by the used hard ware. There is an agreement between some manufacturers in order to reach mutual compatibility on the level of devices. This agreement is not generally accepted though. There is no common operating language, every system uses its own. To provide high-performance programs software has to be adjusted and tested separately for all used digital systems. In addition a multiuser operation is restricted if present at all. The following equally important goals are pursued with the SRCP:

Consideration of different digital systems with a common operation language in multiuser operation over a (computer) network.
Further scaling capability in the range of spontaneously built model railway systems, large stationary constructions and operation at model train conventions where individually created modules are to be put together.
Paying attention to different safety strategies, from none to my own one.
Software shall also be feasable by the interested layman.
A SRCP server abstracts the control of the construction. All informations are provided either due to inquiring the plant directly whereever possible or by the command history.

1.1. Terms

In this text the conventions are used according to RFC 1122. Itemized these are MUST, SHALL, RECOMMENDED, CAN and OPTIONAL.

MUST (respectively REQUIRED or the negation MUST NOT) marks a compulsively redeemed commandment. SHALL and RECOMMENDED stand for a pragmatic recommendation. It can also be ignored after mature deliberation. CAN and OPTIONAL stand for features that can be left out with a clear conscience.

2. Overview

2.1. The Big Picture

SRCP procures client server communication for operating a model railway. The server is the component that is connected with the plant. A client contacts the server in order to send commands to the plant and to receive informations from there. The client can be run on the same computer as the server. It can also be run via the internet from the other end of the world. The used model railway system is of minor importance to the client. It is the server's task to translate the commands of SRCP into action.

A SRCP server has got no knowledge about the concrete railway system. Neither it knows about the topology nor (generally) is it informed about the installed equipment. It conveys the commands intuitly. Its task however is to collect all available information as precisely as possible. Doing that it handles its knowledge about the possibilities and limits of the controlled plant. Its data have always to be understood as the best knowledge about the state of the railway system.

SRCP does not fulfill any formal real time demands.

2.2. SRCP Server And Central Units

A SRCP server SHALL represent exactly one central unit in the usual sense (hardware or emulated in software). Every CU disposes of at least one bus for addressing devices. Over that bus information is read and/or commands are executed. In case a CU has several of these command channels to the model railway plant it is valid in SRCP and leads to several busses.

If a SRCP server can manage and provide several CU it is valid. Separating the CUs is made by different bus numbers. The SRCP servers have to be prosecuted on different TCP ports if there are several CUs connected to a computer but without a common SRCP server. Exactly one of these servers HAS TO provide the standard port.

2.3. Network Connection

The SRPC is based on the data transfer technologies of TCP. One port is occupied. At present the port 12345 SHALL be used. Registering with IANA can change this for the future and give the port a name, too. The temporary name is srcp 12345/tcp.

2.4. Lexical Units

All character codes are described in their decimal form as # followed by the numeric value of the character.

The entire communication consists of the characters of the 7bit ASCII character set in the range from #32 to #127 (including the borders). In addition the characters #9(TAB) for space, #10 and #13 (CR, LF) for end of line are valid. All further characters (esp. with a code value >127) are ignored in incoming data and removed. In outgoing data they can be included, but not used within SRCP. The characters #9 and #13 are seen as white space and count as one white space also if repeated several times (also mixed).

Numbers are processed at least as signed 32bit integers. This range of values can be extended in special cases. Zeros in the leading position are not significant. Floating point numbers are NOT used.

2.5. Commands

Commands are sent in the command mode and in hand shake from client to server. The SRCP server processes the command and generates a respond that is to be sent to the client.

Commands consist of a command word, followed by a command parameter list separated by white space. The end of a command is the end of line. It is not valid to resume a command in the following line. If the list consists of several elements they need to be separated by white space. Elements containing white space are not valid.

If a parameter list contains more elements than speified for the command the supernumerary elements HAVE TO be ignored, the command list has to be shortenened and then processed. If the parameter list contains too few elements an error message needs to be generated. If there are several parameter lists defines for one command the list is always to be shortened if in doubt.

The end of line always consists of the character #10(\n, LF). A prefixed #13(\r, CR) is valid. The character #13 is ignored. A single line including the end of the line MUST NOT exceed over a length of 1000 characters. Information sent by the server is subject to the same terms. The character #13 CAN be sent immediately before #10.

All words are case-sensitive. Commands and replies of the SRCP are always written in uppercase letters. The only exceptions are explaining texts on error messages and informations sent by the devices.

2.6. Replies

A server HAS TO reply to every command of a client. That is to be sent to the client on the same connection. A reply SHALL be single-line. With the sign \(Backslash, #92) immediately before the LF(resp CRLF) it is marked that the reply contains another line. In this case the character sequence \LF(resp. \CRLF) is equated with the white space. That way a reply can be spread over several lines. A single line including the end of the line MUST NOT exceed the length of 1000 characters.

A reply is either an acknowledgement of receipt, an error message or the result of a command. If the reply is an error message the command MUST NOT be executed.

A reply is complete with the effective end of line. Processing more than one answer in one line is not valid.

Replies are always preceeded by a time stamp. The time stamp is generated from a unique time source. It’s format is seconds.milliseconds. The following sources can be used:

TIME Device

    Information from the TIME device on bus 0 (preferred time source).

System Time

    System time of the server (e.g. since 01/01/1970 00:00:00 (POSIX seconds)).

Simple Time

    The value of a regularly increasing counter (for embedded systems).

After the server has sent its reply it processes the next command. That way commands are handled by a stricly two-way communication between the client and the server in a changing order of commands and reactions. The server HAS TO process the commands in the chronological order of receiving them.

If the result of a command can be ascertained due to communication with connected model rilway devices this opportunity SHALL be used. In case there are insufficient results for a task the answer is “416 ERROR no data”.

3. State Of Server

A SRCP server is always in one of three possible states towards the client: hand shake, command mode or info mode.

On connection the hand shake mode is activated. Further operation parameters are determined by the client in this mode. The connection changes from the hand shake mode to either the command mode or the info mode according to the client's demand. From all three modes the connection can be closed any time.

```
                          open
                            |
                            V
                      +-----------+
                      | Handshake |
                      +-----------+
                            |
      +---------------------+---------------------+
      |                     |                     |
      V                     |                     V
+-----------+               |               +-----------+
| Command - |               |               | info mode |
| mode      |               |               +-----------+
+-----------+               |                     |
      |                     |                     |
      V                     V                     V
    close                 close                 close
```

4. Commands

Subsequently the SRCP commands and their corresponding replies are described in detail. All have to be implemented by an SRCP server.

Commands are processed in hand shake and command mode only. They arenot valid during info mode and MUST be ignored.

Commands MUST be processed by the server in the order of receiving them. This involves especially the order while conveying the devices of the plant.

Unknown or unrecognized commands MUST be replied with the message
"410 ERROR unknown command".

4.1. Establishing A Connection

A client establishes a TCP/IP connection with the server. The server sends a single line welcome string. Then the communication parameters and the desired operation mode are negotiated between client and server. The operation mode starts wih the final command GO sent by the client and it has to be confirmed by the server.

The server MUST differ between the client sessions internally. Once a mark(Kennzeichen) is placed it MUST NOT be used for a second time during the entire server runtime.

4.2. Welcome

After establishing a connection between server and client the server sends a text line to the client: The welcome. It consist of information about the server separated by the character ; (semicolon, #59). Every of these informations must be seen as a key - value pair. The order of these keys is undetermined. It is also generally possible to specify keys withdifferent values repeatedly. For every key exactly one value is valid. One value consists of one or more words separated by white space. The end of line counts as the end of the value. A semicolon CAN be missing.

The welcome CAN be created as an error message when not enough resources. In this case the server MUST shut down the connection with the client after sending the welcome message and MUST NOT accept and execute commands by the client.

The following keys MUST be determined during normal welcome:

SRCP <version>: supported SRCP version by the server. It MUST correspond exactly with one of the released version numbers. This data MUST be given exactly once. A repeated specification is not valid.

The following keys CAN be given to show definite informations. A server is free to use further keys. Keys whose identifiers start with the string SRCP MUST NOT be used as long as they are not defined in the SRCP.

SRCPother <version>: SRCP version additionally supported by the server. It must correspond exactly with a released version number and be different from the specification in the key SRCP. A client SHALL ignore this data.

4.3. Hand Shake

After establishing a connection and sending the welcome message the server waits for a command from the client. The server executes it and sends a single line reply to the client. Then the server waits for the next command. That way informations and commands are being exchanged.

The following commands are defined. If not executed the determined default setting is valid.During the hand shake phase no other commands than listed are valid.

SET PROTOCOL SRCP <VERSION> The desired protocol that is to be used is arranged. If this command is missing the SRCP version given during welcome is accepted. Every released version number from and including 0.8 is valid as <VERSION>. The server sends as reply either

201 OK PROTOCOL SRCP or the error message 400 ERROR unsupported protocol. Please note that not every SRCP compliant server supports all versions.

SET CONNECTIONMODE SRCP <MODE> The type of connection is agreed upon. Valid are INFO for the unidirectional information mode and COMMAND for the bidirectional command mode. The server send either 202 OK or the error message 401 ERROR unsupported connection mode to the client. If this command is missing the COMMAND is set as connection mode.

GO By this command the hand shake phase is completed and the chosen operation mode is activated. The server send 200 OK <ID> immediately before that to the client. If the command cannot be executed the server sends the error message 402 ERROR unsufficient data and remains in the hand shake mode. This is intended for future extensions and embedding SRCP in other protocols. The field <ID> marks the numerical session ID given by the server. This is clear to the server and as long as the server is running it will never be given twice. It MUST NOT be identical to zero.

Instead of a reply or the welcome the hand shake can be cancelled by the error message 500 ERROR out of ressources in addition to closing the connection. Any further communication is not possible then.

After completing the hand shake the agreed attributes cannot be changed and are valid for the entire connection.

The following messages are valid during hand shake:

200 OK <ID>

Command accepted and executed. The server leaves hand shake and changes to the selected connection mode. The session id is generated.

201 OK PROTOCOL SRCP

Protocol selection accepted and set.

202 OK CONNECTIONMODEOK

Connection mode accepted and set.

400 ERROR unsupported protocol

The protocol is not supported

401 ERROR unsupported connection mode

the connection mode is not supported.

402 ERROR unsufficient data

Command executed.

500 ERROR out of ressources

no ressources left.

To cancel hand shake the TCP connection needs to be closed without further comments.

5. Command Mode

During command mode the server waits for commands from the client and executes these. As the result of the execution the server always creates a reply due to be sent to the client in the same TCP session. Subsequently the next command is executed. An overlapping or otherwise divergent execution of commands is not valid. The following commands are defined:

GET

request of values

SET

setting values

CHECK

checks a command

WAIT

waiting for values

INIT

Initializing of elements

TERM

Quits elements initialized by INIT

RESET

reinitializes an element

VERIFY

verifying the adjustment of an element

The server‘s replies to the client in the command mode always consist of a time stamp at the beginnig, a numerical answer code and further data. The numerical answer code is structured in groups. 100-199 marks informations and results, 200-299 includes receipts confirming the processing according to the rules, 400-499 marks error while executing commands, 500-599 errors by the server itself, 600-699 specific error codes in implementation.

1xx INFO: Informations, results
2xx OK: Command is accepted and brought to execution. Attention: This is no confirmation for actually executing the command.
4xx/5xx ERROR: An error condition has occured. The command is ignored and not executed.
6xx ERROR: A server specific error has occured. The command in question is not executed.

The significance of the time stamp arises from the kind of reply: For quittances (OK, ERROR)i t specifies the point in time when the command is processed and the receipt OK or ERROR is generated. During INFO the time stamp defines the point in time when the SRCP server first gets to know about the reported result or when the reported ongoing takes place if the server can detect that. If the subject is only a partial change of parameters in a parameter list this is the point in time of the last change. In this text the specification of the time stamp is generally assumed but for reasons of clarity not indicated if it is not necessary for explaining an issue.

While starting the server program internally commands CAN already be executed without the definite assignment from the client. In any case all changes from the determined default state by this norm have to be made available during INFO mode.

The end of the server program is ensured by the command "Section 5.4.10.3". A receipt ("200 OK") follows in this case, too but before its execution. Subsequently the connection to the server is closed. This is valid for all connected clients. During INFO mode SESSIONS are informed at least one second before about the subsequent program end.

5.1. Commands

Valid commands MUST always be executed. Verifying a command with the current knowledge of the server in order to suppress commands that i.e. would not make a change is not valid(prohibition of optimization). The command INIT generally serves for initializing and configuring the indicated device or group of device. After the INIT the indicated element is in default state. During INFO mode a message in the form of 101 INFO INIT <devicegroup> <addr> is sent for the affected element.

By the command TERM the initialization is cancelled and the effected element is in default state. Before using it again the affected element has to be reinitialized. During INFO mode a message in the form of 102 INFO <devicegroup> <addr> is sent for the affected element.

After INIT any write access to the affected device by SET or INIT for all active sessions is invalid until reading access is processed by the commands GET or VERIFY(depending on the device group) or TERM. This includes the session executing the INIT.

The command SET changes certain parameters of the element in question. By the SET command the element CAN be initialized implicitly. The reply to SET is a "200 OK" if successful. The command CHECK corresponds with the command SET in all aspects except for that the actual execution of the command is suppressed. The aim of this command is to support the debugging of clients.

The command GET detects the current parameters. This includes programmed parameters as well that can only be set by INIT. The result is 100 INFO or a n appropriate error message. The informations represent the best knowledge. If a server can find out about the data by requesting the plant components it SHALL do so. If the current situation or the system based upon does not permit that the server CAN go back to own informations i.e. resulting from the last SET. In no case the informations can be seen as completely identical to the actual state of the plant.

The command WAIT awaits the arrival of a certain match in the state of a device for a determined time span. Waiting blocks the client session. If the time span is exceeded the error message "417 ERROR timeout" is generated. If the expected state happens within the determined time an "200 OK" is generated.

The command RESET is for reinitializing an element. This command SHALL be a shortened order of TERM/INIT.The result is either a "200 OK" in the case of success or an error message. The affected element MUST be in the default state subsequently.

The command VERIFY is for checking whether an element state corresponds with a determined sample. The reply is either a "200 OK" in the case of success or an error message. This has to be applied especially with programmable devices if the affected parameters cannot be influenced by SET.

5.2. Response Messages

Response messages are generated by the server as the reply to commands from the client.

Response messages always start with the time stamp like all replies, followed by the response code and the remaining words according to the response code. One of the mesages listed in the following overview MUST be used.

100 INFO <more data>

The server generates information about a device.

101 INFO <more data>

The server generates information about the initialisation of a device.

102 INFO <more data>

The server generates information about the termination of a device.

200 OK

The command accepted and brought to execution. It does not mean, that the command was executed at all, it may be delayed for varios reason!

410 ERROR unknown command

The command part of the line could be understood by the server.

411 ERROR unknown value

The actual value is unknown.

412 ERROR wrong value

a parameter is outside the valid range.

The command cannot execute currently. please try again later.

414 ERROR device locked

The device is currently locked by another sessionid. The session id of this session can only perform read operations.

415 ERROR forbidden

the command is categorically forbidden.

416 ERROR no data

no information at all is available.

417 ERROR timeout

a time limit is exceeded.

418 ERROR list too long

the parameter list is too long.

419 ERROR list too short

the parameter list is too short.

420 ERROR unsupported device protocol

The device does not support the given protocol.

421 ERROR unsupported device

This device is not supported on the specified bus.

422 ERROR unsupported device group

The device group is not supported on the specified bus.

423 ERROR unsupported operation

the requested action is not supported by this device.

424 ERROR device reinitialized

The affected device is reinitialized. The command GET has to be processed first.

425 ERROR not supported

The operation is not supported.

499 ERROR unspecified error

an error has occured but cannot be specified. This error message SHALL be avoided.

Using further error messages SHALL be avoided. In any case the defined message text is to be used when the named error situation occurs. If there are further data about the determined error these can be attached to the message. A client CAN ignore these.

5.3. Addressing And Parameters

Parameters are to be given as a list. Single parameters are separated by white space.The position of the parameter within the list indicates its significance(position correlated parametrization). This is valid for all groups of devices. A single parameter MUST NOT contain a white space. For single commands there can be defined different parameter lists that differ by the number of parameters and semantics.

Addressing elements of a construction is based on the following elements: the bus the group of devices and the device address if several devices are possible.

A bus is categorically the abstraction of a concrete hard ware communication string. A bus is generally represented by a central unit. A bus SHALL be used to map parallel address spaces on a real system.

A bus is determined and initialized by the configuration of the server. While running changing the busses is only possible by resetting a server.

A bus is named by a running number starting with zero and counted upwards without a break. Bus 0 is reserved for the server itself. There always MUST be at least one additional bus connected with the CU. It is not relevant whether the CU is implemented by hard or soft ware.

In all commands the number of the bus must be given right after the command word as the first parameter and is used during all replies and informations.

Device groups are the topic of the next paragraph.

The aim of addressing are the devices: decoders and encoders(in the following it is spoken of decoders only; encoders are applied by analogy). Every decoder is an element of at least one bus and a device group, the affiliation of a decoder with several buses at one time is valid but SHALL be avoided. If the bus affiliation (i.e. with locomotives) changes the data are not adopted automatically. It is the client‘s task to carry out this transfer. Devices belonging to several device groups at the same time are also valid.

If devices with an identical hard ware address but different control protocols coexist on one bus theSRCP server MUST make sure that the concerned device acts comprehensibly. If necessary several busses have to be provided to support addressing.

For an SRCP server all devices count as known that have been addressed before and are not deleted by TERM. A device in the default state CAN also be known. In some hard ware systems broad casts are mapped by certain attributes of the processed data. An SRCP server CAN use these to put the defined wild card behaviour into action. In systems not supporting this the server MUST emulate this function. An SRCP server is NOT ALLOWED to accept possible parameters of this kind from a client. This particularly concerns especially reserved addresses.

5.4. Device Groups

Device groups are sets of similar devices. The following device groups are defined:

DESCRIPTION

Devices describing other devices or busses.

DESCRIPTION devices can describe other devices with their attributes. Replies by a server CAN be formulated on several lines. It is distinguished between the DESCRIPTION of busses and the DESCRIPTION of device groups.

FB

Feedback sensors.

Feedback devices FB are encoders that signal an occurence on the construction. They have exactly one way in that is identified by an address and has at least two distinguishable states. If single encoders are pooled in one device group it has no impact on the identification. The group not valid under the bus number 0.

GA

Generic Accessoire.

The device group GA is provided for decoders that make one or more ports with two or more possible values at a time available under one address. Usually these are stationary construction elements like switches or uncoupler. The group GA is not valid in the bus number 0.

GL

Generic Loco.

The device group GL indicates engine decoders in the engine address room of the hard ware. The group GL is not valid in the bus number 0.

LOCK

Devices locking other devices.

The LOCK devices are devices providing a lock over another device.They are optional. If present the server MUST contact the LOCK devices before any actions on the lockable devices. They are ALLOWED to exist on particular busses but not on all busses. Providing lock devices for all busses of a server should be aspired.

POWER

Energy Supply.

POWER signifies the state of the energy supply of the particular bus. POWER is invalid on bus number 0. Activating the energy supply on one bus can activate other busses implicitly. This MUST be considered in INFO mode.

SERVER

SRCP Server.

SERVER together with the bus 0 signifies activities concerning the server(esp. TERM and RESET). The use with other busses is not valid. The SRCP server is the only element of this device group.

SESSION

SRCP Client Session.

SESSION together with the bus 0 signifies the amount of active client connections. Using other busses than 0 is not valid.

SM

Service Mode.

The service mode SM effects decoders used in the program mode. Therefore it CAN be necessary to displace the decoder on another bus (programming track) or that a bus has to be suspended for other commands. On this bus commands CAN be refused with the error message temporarily prohibited or forbidden. This group is unvalid on the bus 0. Technical implementation can include direct access on the used digital systems.

TIME

Time Norm.

The time normal TIME is only valid with the bus number 0. It is for providing a uniform model time. This is counted faster or slower than real time going out from a starting point but exactly like real time monotone with model second precision. During INFO mode the model time is processed at every full model minute. The time normal is optional. Only one device is valid in this device group.

Some device groups correspond with the concerned plant elements. They are initialized by default on the lowest common denominator in the case of different implementations. Using further features requires an initialization where information about the hard ware is necessary. An overview about what hard ware can (resp. must) be initialized and how it is supposed to be done can be found in the separate document SRCP Devices . The data given there are as binding as the SRCP itself.

Every SRCP server MUST support the device groups SERVER, SESSION and DESCRIPTION in bus 0. The device groups POWER and DESCRIPTION MUST be available for every further bus. All other device groups are optional. If they are supported it MUST be specified in the DESCRIPTION of the bus. Supporting DESCRIPTION for device groups is optional and can be supported separately for every device group.

The following overview is supposed to show the allocation of device groups and commands on the busses.

```
           |  SET  GET  WAIT  INIT TERM  RESET VERIFY
           | CHECK
        ---+------------------------------------------
           |
        GL |  1..  1..  --   1..  1..   1..   --
           |
        SM |  1..  1..  --   1..  1..   1..   1..
           |
        GA |  1..  1..  --   1..  1..   1..   --
           |
        FB |  1..  1..  1..  1..  1..   1..   --
           |
      TIME |  0    0    0    0    0     0     --
           |
     POWER |  1..  1..  --   1..  1..   --    --
           |
    SERVER |  --   0    --   --   0     0     --
           |
   SESSION |  0    0    --   --   0     --    --
           |
      LOCK |  0..  0..  --   0..  0..   --    --
           |
DESCRIPTION |  0..  0..  --   --   --    --    --
```

Entries signed with the - tag denotes combinations that MUST NOT be used.

5.4.1. FB: Feed back

Feedback sensors have an address where exactly one current value is recorded. In its simplest case a feedback is a binary signal. A default state that can be influenced by the server is not present.

5.4.1.1. GET

GET <bus> FB <addr>

It supplies the current value of a single feedback sensor as a “*100 INFO <bus> FB <addr> <value>*” If no data is available, the error message “416 ERROR no data” is generated.

5.4.1.2. SET

SET <bus> FB <addr> <value>

Sets the value for the FB specified.

5.4.1.3. INIT

INIT <bus> FB <optional parameters for initialization>

An already existing feedback bus is initialized. The exact parameters are known to the server only from its configuration.

5.4.1.4. TERM

TERM <bus> FB

By TERM the concerned bus is taken out of the runing communication without deleting it from the list of present busses. The attached hard ware SHALL be turned off if all busses taking hold of it are terminated. Running WAITs MUST be closed by time out as long as the WAIT requirement is not met.

5.4.1.5. WAIT

WAIT <bus> FB <addr> <value> <timeout>

Waiting until <timeout> seconds(real time) for reaching the specified value. In case the value is already there its eventuation is reported instantly without waiting.

The session blocks further operation until either an error message (417 ERROR timeout) or the value specified occures (*100 INFO <bus> FB <addr> <value>*).

5.4.1.6. INFO

100 INFO <bus> FB <addr> <value>

Value of feedback sensor <addr> on the specified bus.

101 INFO <bus> FB

Initialisation of the FB devices of the specified bus.

102 INFO <bus> FB

Termination of the FB devices of the specified bus.

5.4.2. GA: Generic Accessoire

A Generic Accessoire generally indicates a decoder that can serve one or more ports under one address. Often these are switch decoders or signal decoders working as impulse decoders. One must note that there are restrictions during simultaneous activation and/or deactivation of ports. These are to be taken from the description of the decoder if necessary. An SRCP server cannot always indicate and report these features itself.

The default state of a device is indicated by zero on all ports.

5.4.2.1. GET

GET <bus> GA <addr> <port>

The server sends all available information about the current state of the switch decoder specified by <addr> to the client as "*100 INFO <bus> GA <addr> <port> <value>*" If no data is available, the error message "416 ERROR no data" is generated. Other error message may occur if the parameters given (addr, port) are out of range.

5.4.2.2. INIT

INIT <bus> GA <addr> <protocol> <optional further parameters>

This command initializes a GA at address addr in the server. The following parameters are permitted:

M

Maerklin/Motorola Format

valid addresses are from 1 to 324, valid port are 0 or 1, valid values are: 0 and 1

N

NMRA-DCC Format

valid addresses are from 1 to 511, valid ports are 0 or 1, valid values are: 0 and 1

S

Seletrix Format

valid addresses are from 0 to 111, valid ports are 1 to 8, valid values are: 0 and 1

P

Protocol by server

The server uses the initialisation parameters from its own configuration. The parameter list MAY give some parameters. Addr, port and value are not limited.

5.4.2.3. SET

SET <bus> GA <addr> <port> <value> <delay>

The port <port> of the decoder with the address <addr> is set on the value <value> for <delay> milliseconds. After the expiry of time the server automatically sends the value 0 to the decoder. In case the delay is -1 the automatic deactivation remains undone and the port remains active until it is deactivated by a further SET command. An SRCP server CAN execute momentary deactivations and reactivations of the port during this time. If applicable this is a demand of the hardware that does not permit certain maximum power durations to be exceeded. A delay of 0 is not valid.

Meaning of the arguments

- `addr` must be valid or the sign * (asterisk, #42), which mean “ALL known GA”.
- `port` must be valid. In case an unsupported port is given, an error message “412 ERROR wrong value” is created.
- `value` must be valid (0=>deactivated, >0=>activated) If an unvalid value is given an error message “412 ERROR wrong value” is created.
- `delay` value in milliseconds (1.000th seconds). It indicates after what time the server shall deactivate an active port automatically (i.e. set port value to 0). If `delay` is -1 the port is not automatically deactivated. If `value`=0 is given (deactivation) the `delay` parameter is ignored but it must be given (useful value is 1). A port counts as active if its state is not equal zero. A delay of 0 is not valid. An error message “412 ERROR wrong value” is created.

5.4.2.4. INFO

100 INFO <bus> GA <addr> <port> <value>

  Value of GA port at addr on the specified bus.

101 INFO <bus> GA <addr> <device protocol> ....

  Initialisation of a GA device of the specified bus.

102 INFO <bus> GA <addr>

  Termination of a GA device of the specified bus.

5.4.3. GL: Generic loco

A generic loco generally indicates all engine decoders. The defaultstate is zero with all parameters.

5.4.3.1. GET

GET <bus> GL <addr>

The reply is a complete “*100 INFO <bus> GL <addr> <drivemode> <V> <V_max> <f1> . . <fn>*” line.

5.4.3.2. INIT

INIT <bus> GL <addr> <protocol> <optional further parameters>

By INIT the kind of data transfers and fundamental features of the decoder is reported to the server. Changing the decoder features on programmable decoders is not permitted (for this the device group SM is to be called upon).Exception: If decoders are programmable by a sequence of normal commands the device group SM MUST NOT be used.

- `addr` must be valid.
- `protocol` must be one of the following. In case an unsupported port is given, an error message "412 ERROR wrong value" is created.

A

analog operation.

addr: 0, no functions and no speed steps

F

reserved for native Fleischmann.

L

reserved for Loconet.

M protocolversion decoderspeedsteps numberofdecoderfunctions

reserved for Mï¿½klin/Motorola MM.

- `protocolversion` The protocol version is 1 for the old MM protocol (e.g. delta decoder), 2 for the new protocol (additionally four functions).
- `decoderspeedsteps` The decoder speed steps comply with those supported by the decoder (if applicable under attention for the number of the decoder functions).
- `numberofdecoderfunctions` The number of decoder features results in the number of the addressable functions without attention for a possibly unused function. They are always to be seen consecutively without interception.

N <protocolversion> <decoderspeed steps> <numberofdecoderfunctions>

reserved for NMRA/DCC.

- `protocolversion`The protocol version is 1 for the short address format and 2 for the long address format.
- `decoderspeedsteps`The decoder speed steps comply with those supported by the decoder (if applicable under attention for the number of the decoder functions).

- `numberofdecoderfunctions`The number of decoder features results in the number of the addressable functions without attention for a possibly unused function. They are always to be seen consecutively without interception.

P

protocol by server.

any address, any number of functions (additional information: any number of speed steps). The server specifies the type of decoder by itself.

S

reserved for native Selectrix.

Z

Z reserved for native Zimo.

A server CAN NOT support one or more of these specifications. Specifications that are not given MUST NOT be used. These specifications will be put in concrete terms in later revisions of the SRCP.

Hint: In some protocols there are several scopes for functions. Usually there is one depending on the riding direction and several not depending on the riding direction. A server MUST provide the first as the first function and the others as second, third etc. function.

5.4.3.3. SET

SET <bus> GL <addr> <drivemode> <V> <V_max> <f1> . . <fn>

The given values are allocated to the decoder. The default state is indicated by the value 0 for all parameters(apart from the address). The meanings of the parameters are as follows:

- `addr` must be valid or the sign * (asterisk, #42), which mean "ALL known GL".
- `drivemode` must be one of the following values or an error message "412 ERROR wrong value" will be sent:
  = unchanged
  0 backward
  1 forward
  2 emergency stop
- `V, V_max` speed level and maximum speed level. 0 is standstill. A value >0 means movement of the engine (i.e. a real speed level >0). V<0 is not valid as well as V>V_max. The speed raises from 0 to V_max so that a speed V2>V1 causes the engine not to go slower on V2 than on V1. In all error

situations an error message “412 ERROR wrong value” is created. Using the wild card = is only valid with both parameters at a time.

- `f1 . . fn` =(=unchanged), 0(=off), 1(on). If applicable F1 is dependent on the direction. It is valid to give other values than =, 0 or 1. The SRCP server rejects invalid values with the error message “412 ERROR wrong value” is created.

Example:

INIT 1 GL 1 N 1 128 5 SET 1 GL 1 1 4 100 1 0 1 0 =

The engine 1 (NMRA decoder, 128 decoder speed steps, 1+4 functions) addressed via bus 1 goes forward with speed step 4 (of 100 possible). From the five functions there are only FUNCTION and F2 active. The value of F4 is set from the knowledge of the server about the current state.

5.4.3.4. TERM

TERM <bus> GL <addr>

Sets the engine onto default mode and removes the engine from the knowledge of the server.

5.4.3.5. INFO

An INFO covers the reaction of the GET as well as the output in INFO mode as a reaction to the SET and INIT commands. It MUST include all current values no matter whether they are currently changed or not. The format is as follows:

100 INFO <bus> GL <addr> <drivemode> <V> <V_max> <f1> . . <fn>

    parameters of GL addr on the specified bus.

101 INFO <bus> GL <addr> <protocol> <optional further parameters>

    Initialisation of a GL device of the specified bus.

102 INFO <bus> GL <addr>

    Termination of a GL device of the specified bus.

Here the parameter V is not the speed step transmitted by the client, but the real speed step sent to the engine by the server obmissing possible special indications: The speed step 0 indicates standstill. The speed steps from 1 to conclusively V_max indicate the movement of the engine in the given <drivemode>

5.4.4. SM: Service Mode

The devices of the service mode are provided for the permanent change of devices. They are an optional component. If they are not supported an error message “425 ERROR not supported” must be created. They are intended to be used for decoder programming. A deafault state is not specified.

5.4.4.1. GET

GET MUST consult the decoder directly. A reply MUST not be detected from information on the server. Using wild cards is not valid.

GET <bus> SM <decoderaddress> <type> <1 or more values>

The meaning of the parameter is as with SET. The result is an 100 INFO, that renders the particular value. A GET results in the error message “416 ERROR no data”, if there is no possibility to read out the decoder.

5.4.4.2. INIT

INIT <bus> SM <protocol>

It initializes the concerned bus for programming decoders in the given protocol. Consequently the bus CAN be suspended for other tasks. Valid protocols are:

NMRA

Decoder for NMRA standard.

5.4.4.3. SET

SET <bus> SM <decoderaddress> <type> <1or more values>

The following <more values> are defined:

CV

Configuration Variable: The first parameter of the remainder list indicates the address of an NMRA decoder variable (<cvaddress>). It receives the value described in the second parameter <value>. Limitations in the value area of the parameters arise from the specifications of the decoder. A server cannot check decoder specific value areas.

CVBIT

Configuration Variable Bit: 3 parameter: <cvaddress> <bit> <value> The bit <bit> of the address <cvaddress> is set on the value <value>. <bit> goes from 0 to 7. <value> is 0 or 1.

REG

Register

5.4.4.4. TERM

Quits the program mode on the concerned bus. Possibly incomplete programming cycles are suited for quitting.

TERM <bus> SM

"*100 INFO <bus> SM <decoderaddress> <type> <1 or more values>*".

5.4.4.5. VERIFY

VERIFY MUST consult the decoder directly. A reply MUST not be detected from information on the server. Using wild cards is not valid.

VERIFY <bus> SM <decoderaddress> <type> <1 or more values>

As with GET the decoder must be consulted directly. The meaning of the parameters is as with SET. The result is either a 200 OK if the parameter in the decoder complies with the given one or the error message " 412 ERROR wrong value".

5.4.4.6. INFO

An INFO covers the reaction of the GET as well as the output in INFO mode as a reaction to the SET and INIT commands. It MUST include all current values no matter whether they are currently changed or not. The format is as follows:

100 INFO <bus> SM <decoderaddress> <type> <1 or more values>

parameters of SM decoderaddr on the specified bus.

101 INFO <bus> GL <addr> <protocol> <optional further parameters>

Initialisation of a GL device of the specified bus.

102 INFO <bus> GL <addr>

Termination of a GL device of the specified bus.

5.4.5. LOCK: write lock

By locking a client can suspend a device exclusively for a current session. This is also valid for physically identical devices addressable in several device groups: A LOCK on a decoder in the device group GL also prevents write access to the same decoder in the device group SM! Other sessions, also by the same client, are reported the error message “414 ERROR device locked” during write access(SET, INIT,TERM). Executing these commands is allowed to the client session holding the lock only . Read access(GET) is still accepted from all clientsand is functionable without limitations. Locking a device from the LOCK group, also of another bus, is not valid. A bus cannot be locked as a whole.

The default state is not locked.

For the devices of the group GL an emergency stop is ALWAYS valid for all clients. A lock by another session is without effect in this case. Here the engine MUST execute the emergency stop only, all other parameters, in case they are given, MUST remain unchanged. Particularly the LOCK remains.

A lock is automatically cancelled during the end of the client connection and run-out of the lock time.

The locks are an optional part. If they are unsupperted an error message “425 ERROR not supported” MUST be created.

5.4.5.1. GET

GET <bus> LOCK <devicegroup> <addr>

Here the session ID is the only clear identification of the session containing the lock.

This command reports the current lock status of a device as 100 INFO line. If the given device is not locked the session ID 0 is reported.

5.4.5.2. SET

SET <bus> LOCK <device group> <addr> <duration>

Puts a lock on the addressed device. One must pay attention that LOCK devices do not have an own address. They are identified via the device lockd by themselves only.

Other clients still have access for reading and writing in areas relevant for safety only. The LOCK is held for <duration> seconds. After this time span runs out the LOCK is automatically quitted by the server. The time starts with the last SET command. A duration of 0 means an unlimited time span. A repeated stop of the SET command restarts the timer each time.

5.4.5.3. TERM

TERM <bus> LOCK <devicegroup> <addr>

A lock on the device <devicegroup> <addr> of the bus <bus> is removed.

5.4.5.4. INFO

100 INFO <bus> LOCK <devicegroup> <addr> <duration> < sessionid>

parameters of GL addr on the specified bus.

102 INFO <bus> LOCK < devicegroup> <addr>

Initialisation of a LOCK on a device of the specified bus.

102 INFO <bus> LOCK < devicegroup> <addr>

Termination of a LOCK in a device of the specified bus.

5.4.6. POWER: Energy Supply

The energy supply is enabled for every bus separately. The default state is OFF.

If the server learns about changes in the energy supply from the construction itself (e.g. recognizing a short circuit), it means the same like a corresponding command and MUST be reported in info mode.

5.4.6.1. GET

GET <bus> POWER

Reports the current state of the energy supply as 100 INFO.

5.4.6.2. INIT

INIT <bus> POWER

It initailizes the power supply but it is not turned on!

5.4.6.3. SET

SET <bus> POWER ON/OFF [freetext]

Activates(ON) or deactivates(OFF) the energy supply on the affected bus. The <freetext> is an optional addition allowing up to 100 characters. It is reported to other clients in INFO. An SRCP server does not evaluate this parameter.

5.4.6.4. TERM

TERM <bus> POWER

Quits the power supply and turns it off.

Informs about the current state of the power supply and possible additional data.

5.4.6.5. INFO

100 INFO <bus> POWER ON/OFF <freetext>

parameters of GL addr on the specified bus.

101 INFO <bus> POWER

Initialisation of POWER on the specified bus.

102 INFO <bus> POWER

Termination of POWER of the specified bus.

5.4.7. TIME: Time Normal

The model time is opposed to the normal time a time distorted by a constant factor. The time distortion is specified by a multiplier and a diviser to this normal time. The timer is only allowed on bus 0. There is only one timer.

In the default state the timer is stopped.

The timer is an optional part. If it is not supported an error message “425 ERROR not supported” MUST be created at all commands.

Time coverage consists of the data: day, hour, minute and second. A minute is 60 seconds, an hour is 60 minutes and a day is 24 hours. The days are counted consecitively. A separation into larger time intervals (calendar function) is the client‘s task.

If a full model minute is reached the current model time is given in INFO mode. The commands WAIT and GET additionally valuate the model seconds.

The model time is generated as follows:

(Delta) model time = (delta) real time * FX/FY

Examples:

FX=10 FY=1 -> In every real minute 10 model minutes are generated(one every six seconds)
FX=1 FY=10 -> Every 10 real minutes are one model minute.
FX=1 FY=1 -> In every real minute there is one model minute.

5.4.7.1. GET

GET 0 TIME

Reports the current model time as “*100 INFO 0 TIME <JulDay> <Hour> <Minute> <Second>*”.

5.4.7.2. INIT

INIT 0 TIME <fx> <fy>

It initializes the timer with the given distortion. The timer is not started automatically. The parameters must be positive numbers different from zero. A server CAN limit the value range to a bounded scope (e.g. 1 .. 10). These values MUST be given in the documentation.

5.4.7.3. SET

SET 0 TIME <JulDay> <Hour> <Minute> <Second>

It sets the timer on the defined point of time and starts it. With a running timer the new time starts when the running model second ends so that the new beginning model second corresponds with the new time.

5.4.7.4. TERM

TERM 0 TIME

Stops and removes the timer. All running WAITs a cancelled by TIME OUT with
“417 ERROR timeout”.

5.4.7.5. WAIT

WAIT 0 TIME <JulDay> <Hour> <Minute> <Second>

It waits until the model time reaches or outruns the given point and reports an INFO string with the then active model time.

If the timer is not running the error message “416 ERROR no data” is generated. If the current model time is later than the given time already the condition is fulfilled without further waiting time. Obviously wrong time data are reported to the calling client by “412 ERROR wrong value” or ignored. The WAIT MUST always evaluate the currently valid model time that can be changed by SET if applicable.

5.4.7.6. INFO

100 INFO 0 TIME <JulDay> <Hour> <Minute> <Second>

Reports all parameters and current data from the timer.

101 INFO 0 TIME fx fy

Initialisation of TIME.

102 INFO 0 TIME

Termination of TIME.

5.4.8. SESSION

Sessions are active connections from a client to a server. A client CAN have multiple sessions with a server. For the server these are independent of each other. A session is distinguished from others by a session ID. This session ID is generated by the server during hand shake and is explicit for the uptime of the server. A session ID given out once cannot be used a second time.

Sessions are only valid on the bus 0. Ther is no defined default state.

5.4.8.1. TERM

GET 0 SESSION <SESSIONID>

Reports data about the session as 100 INFO.

5.4.8.2. INFO

100 INFO 0 SESSION <SESSIONID> [further optional parameters]

The data for the session are transmitted. Further parameters can consist of the IP address, the port number and other details.

101 INFO 0 SESSION <SESSIONID>

Initialisation of TIME.

101 INFO 0 SESSION <SESSIONID>

Termination of SESSION.

5.4.9. DESCRIPTION

The DESCRIPTION device groups describes other device groups and busses. They inform the client about the fundamental features.

5.4.9.1. GET

This command occurs in two versions: with a parameterlist and without. If the parameter list is left out the server informs the client about the concerned bus. With a parameter list the concerned device is described regarding its initialization.

5.4.9.1.1. Bus description

GET <bus> DESCRIPTION

--> INFO <bus> DESCRIPTION <list of devicegroups>

The INFO covers all device groups supported by the concerned server in one line. Example:

```
GET 0 DESCRIPTION - -> INFO 0 DESCRIPTION SERVER SESSION TIME
GET 1 DESCRIPTION - -> INFO 1 DESCRIPTION FB
```

The bus 0 supports the device groups SERVER, SESSION and TIME. The bus 1 only supports checkbackers.

In the second variant the address of the concerned device and as reply the parameters defined during initialization are given.

5.4.9.1.2. Device description

GET <bus> DESCRIPTION <devicegroup> [<address>]

The address is only ALLOWED to be left out with device groups without an address. The parameterlist is identical with the one from the INIT command in this case. Example:

GET 1 DESCRIPTION GL 1 --> INFO 1 DESCRIPTION GL 1 N 128 5

i.e. the engine with the address 1 is an NMRA compatible decoder with 128 real speed steps and five functions.

5.4.10. SERVER

The server as a whole. This device group is only defined on bus 0. Valid are the commands TERM and RESET. The default state is RUNNING.

5.4.10.1. GET

GET 0 SERVER

Informs the current state of the server. The following messages are defined:

The last two messages areparticularly important in INFO mode.

5.4.10.2. RESET

RESET 0 SERVER

Re-initializes the server. All devices are set in the default state. A RESET during a RESET MUST be refused (“413 ERROR temporarily prohibited”). Existing client sessions remain untouched also during hand shake.

5.4.10.3. TERM

TERM 0 SERVER

Quits the running server and closes all connections. The connected busses SHALL be turned off and the active devices set in the default mode. For the calling client the quitting “200 OK”) is transmitted. Sessions in INFO mode are informed about the running TERM. A TERM MUST NOT be cancelled.

5.4.10.4. INFO

100 INFO 0 SERVER <status info>

RUNNING: server in normal commission
RESETTING: server executes a reset
TERMINATING: server is terminated.

101 INFO 0 SERVER

Initialisation of TIME.

102 INFO 0 SERVER

Termination of SERVER.

6. Info Mode

In info mode all changes of all devices are sent unidirectionally to the connected client. It is PROHIBITED to the client to send commands or other to the server. The server must ensure that data sent anyhow do not have consequences. The server is free to quit the connection.

The info mode sends all data as they become known on the installation. Like with data given during command mode(see replies to GET commands) they are always supplemented with a time stamp. INFO data are sent only. Especially results of commands are sent only after they have been validated by the actor(decoder) or when they have benn transmitted to the construction successfully(in case there cannot be a validation from the construction). There are no changes or interpretations to the data. This particularly concerns the train engine speed that is vitually sent by the client but converted according to the real conditions of the decoder.

When activating the info mode in a session the DESCRIPTIONS for all busses and all active devices known to the server in their current state are transmitted. Subsequently all changes are sent. If there is a clear default state for some device groups, all devices in this default state SHALL not be transmitted. Optional device groups are only ALLOWED to be given if they are supported.

Specifics of the device groups during activation of the info mode:

FB

All devices with the status 0 are not transmitted. All others are in form of "*100 INFO <bus> FB <addr> <value>*".

GA

The individual ports of the devices known to the server are separated and transmitted each with their last activity in form of "*100 INFO <bus> GA <addr> <port> <value>*".

GL,SM

All engines moving or standing with an active function are reported. Engines in standstill MUST be reported (as "*100 INFO <bus> GL <addr> <drivemode> <V> <V_max> <f1> . . <fn>*" or "*100 INFO <bus> SM <decoderaddress> <type> <1 or more values>*".) if they have been addressed at least once and not removed by TERM. Likewise changes are reported by INIT. During SM all programming activities are reported. A concluded programming action MUST NOT be sent with new connections.

POWER

The state of the energy supply MUST be given as "*100 INFO <bus> POWER ON/OFF <freetext>*".

TIME

The current model time MUST be transmitted (as "*100 INFO 0 TIME <JulDay> <Hour> <Minute> <Second>*" and "*101 INFO 0 TIME fx fy*".) as long as this feature is supported and the timer is running.

SESSION

The identification of all currently active client sessions is sent "*100 INFO 0 SESSION <SESSIONID> [further optional parameters]*". All further informations MUST be detected in command mode.

LOCK

All devices currently lockes are reported as "*100 INFO <bus> LOCK <devicegroup> <addr> <duration> < sessionid>*".

From the initial transmission of all states all changes on the installation and the server MUST be transmitted also those into the default state. An exception is the device group TIME: It reports every beginning of a new model minute but not of model seconds.

These information contain only the defined INFO messages (response code 100..199).

7. Glossary

The terms itemized here form interfaces to other documents.

Julian Date The Julian Date is a simple day count without any further structures(like weeks, months, years). According to the arbitrary origin (beginning of the year, fixed point of time in the past) there are different possibilities to convert into other calendars (basis: 1.1.4713 BC 12-00 GMT: see Collected Algorithms from CACM #199)

XRCP A supplement to the SRCP in order to form a higher level of abstraction. Apart from connecting the clients to the server it is there for the communication of the clients among each other. An XRCP system is created as middle ware between clients and SRCP servers.

8. Thanks go to...

The following people did major or minor contributions:

Torsten Vogt
Martin Ostermann
Kurt Harders
Olaf Schlachter
Tobias Schlottke
Edbert van Eimeren
Stefan Bormann
Michael Reukauff
Martin Wolf
Matthias Peick
Martin Schï¿½beck