IpConv
Protocol Stack

Elcom90Responder

ELCOM−90 Protocol, Responder UE

IPCOMM GmbH

Edition October 2013
Version 2.0

IPCOMM GmbH
Gundstrasse 15
D−91056 Erlangen
Germany

Voice: +49 9131 92076 0
Fax: +49 9131 92076 10
Email: info@ipcomm.de

Table of Contents

Elcom90Responder − General

The module discribed here implements the presentation layer, application layer and the Responder part of the user element (UE) of the ELCOM−90 protocol.

Configuration

To configure the module the following steps have to be performed:

1. Adding at least one communication partner to the **partner** list.
2. Configuration of the Initiator address with the parameter **client_ip**.
3. Configuration of the server IP address (this is also used as Acceptor address).
4. Importing informations objects into the Node configuration.

Import of information configuration from table

Information configuration can be imported from table (e.g. MS Excel). For detailed description of the table file format see "ipConv Functional Specification".

The imported data is stored in the **import** parameter branch, which is automatically generated. The configuration of information objects can be performed manually and imported from table at the same time. In this case following points should be taken into consideration:

- Never mix both configuration methods for the same information object.
- As a rule only those information objects are defined manually which require exceptional treatment e.g. administrative indications. By means of import from table big amount of similar information objects can be imported more effectively.
- On repeated import from table configuration information imported before is removed and replaced by the new one.
- the imported configuration information can be modified manually. These changes will be lost on repeated imports.

Startup

On startup of the module the following actions are executed:

1. The Initiator establishes all required associations, i.e. connection AA to AG.
2. The Initiator creates the groups in the Responder.
3. The Initiator adds information objects to the groups.
4. The Initiator gives permission for unsolicited data transfer to the Responder.
5. The Initiator sends a general scan to the Responder, the Responder sends the initial values back to the Initiator.
6. From now on, the Responder sends data spontaneously or cyclically to the Initiator. It depends on the used association, which functional unit (FU) is used for the data transfer.

Implementation details

For further information about the implementation details, please refer to the ELCOM implementation document (PID).

Elcom90Responder − Normalized Address (NA)

Class	**Direction**	**NA**	**Protocol Specific Group Type**	**Normalized Information Type**	**Description**
Process Information	From Node to Module (to)	inf−{OBJID}	StatusGroup	SI	Single point information
			StatusGroup	DI	Double point information
			MeasureGroup	MFV	Measured short float pointing value
			DiscreteGroup	MSV	Measured scaled value
			DiscreteGroup	MNV	Measured normalized value
	From Module to Node (from)	inf−{OBJID}	BinaryCommandGroup	SC	Single command
			BinaryCommandGroup	DC	Double command
			AnalogueSetpointGroup	SPF	Setpoint command, short floating point
			DigitalSetpointGroup	SP	Setpoint command
Internal Information, connection specific	From Module to Node (from)	con−{PID}−connect		SI	Connection state to a specific partner
Internal Information, module specific	From Node to Module (to)	int−startup		SI	Release of module startup
		int−active		SI	Activate / Deactivate communication

Abbreviations

{PID}	Partner ID (1..255)
{OBJID}	Object Identifier ([−A−Za−z0−9_.]{1,255})

Protocol specific information types

Group type	Description
Measure−group	Floating−point values
Status−group	3−state values (on/off/between)
Discrete−group	Two octet integer
Binary−command−group	2−state commands (on/off)
Analog−setpoint−group	Floating point values
Digital−setpoint−group	Two octet integer command values

Internal module specific information

int−startup

Depending on the configuration this module normally does not start it's operation directly after startup. Rather than it waits for an external release initiated by another protocol module. As soon as this module receives this one−time single indication (SI) from the node with the "APP" value, module operation is released permanently.

int−active

This information is required for redundant operation of two or more protocol converters. If the configuration parameter "active" has the value "FALSE", the module starts up in passive state, i.e. the TCP port isn't bound to any IP address. Receipt of this single indication with the value "APP" activates the module. The module then binds the TCP port and enables communication with the partners. When this single indication is received with the value "DIS" and the module was previously active, it reverts to the passive state, i.e all TCP connections are closed.

Elcom90Responder − Logging

This protocol module distinguishes between three logging levels. The logging levels can be activated at the same time with the OR function.

Logging level	**Description**
0	no logging
1	hexadecimal representation of transport layer telegrams
2	decoded representation of presentation layer frames
4	decoded representation of application layer frames
6	levels 2 and 4
7	levels 1, 2 and 4

Logging level 1

The logging level 1 displays all the telegrams sent and received in hexadecimal presentation. The following syntax is used:

(1): **{D}** [**{PID}**/**{FU}**] <**{DATA}**>

Code	**Value**	**Description**
{D}		Transmission direction
	<<	Send direction
	>>	Receive direction
{PID}	1..255	Partner ID
{FU}	1..7	Functional Unit
{DATA}	00..FF	Telegrams sent and received in hexadecimal presentation

Logging level 2

The logging level 2 displays the PPDUs (Presentation Protocol Data Units) human−readable presentation. The following syntax is used:

(2): **{D}** [**{PID}**/**{FU}**] LEN=**{LEN}** PPDU−ID=**{PPDU}** I−ADDR=**<{ADDR}>** A−ADDR=**<{ADDR}>** DATA=<**{DATA}**>
(2): **{D}** [**{PID}**/**{FU}**] LEN=**{LEN}** PPDU−ID=**{PPDU}** I−ADDR=**<{ADDR}>** A−ADDR=**<{ADDR}>** RESULT=**{RESULT}** DATA=<**{DATA}**>
(2): **{D}** [**{PID}**/**{FU}**] LEN=**{LEN}** PPDU−ID=**{PPDU}** REASON=**{REASON}**
(2): **{D}** [**{PID}**/**{FU}**] LEN=**{LEN}** PPDU−ID=**{PPDU}** RESULT=**{RESULT}**
(2): **{D}** [**{PID}**/**{FU}**] LEN=**{LEN}** PPDU−ID=**{PPDU}** DATA=<**{DATA}**>

Code	Value	Description
{D}		Transmission direction
	<<	Send direction
	>>	Receive direction
{PID}	1..255	Partner ID
{FU}	1..7	Functional Unit
{LEN}	1..32767	Length of the telegram
{PPDU}		PPDU type
	ConnectReq	Connect Request
	ConnectRsp	Connect Response
	ReleaseReq	Release Request
	ReleaseRsp	Release Response
	DataReq	Data Request
{ADDR}		Elcom address[1]
	I	Initiator address
	A	Acceptor address
{REASON}	0..127	Reason code
{RESULT}	0..127	Result code
{DATA}	00..FF	Data field in hex presentation

[1] The following syntax is used for Elcom addresses:

AF=**{AF}** PORT=**{PORT}** IP=**{IP}** SUFFIX=**{SUFFIX}**

Code	Value	Description
{AF}	AF_INET	Address family
{PORT}	1..65535	TCP port number
{IP}	e.g. 172.16.4.1	IP address
{SUFFIX}	AA..AG BA..BG	A−suffix

Logging level 4

The logging level 4 displays the APDUs (Application Protocol Data Units) human−readable presentation. The following syntax is used:

(4): **{D}** [**{PID}**/**{FU}**] LEN=**{LEN}** APDU−ID=**{APDU}** VERSION=**{VERSION}** CLASS=**{CLASS}** RESULT=**{RESULT}** DATA=**{DATA}**
(4): **{D}** [**{PID}**/**{FU}**] APDU−ID=**{APDU}** GTYPE=**{GTYPE}** GNR=**{GNR}** PERSIST=**{PERSIST}** STATIC=**{STATIC}** PRIO=**{PRIO}** GSIZE=**{GSIZE}** OBJLEN=**{OBJLEN}** FUNC=**{FUNC}**
(4): **{D}** [**{PID}**/**{FU}**] APDU−ID=**{APDU}** GTYPE=**{GTYPE}** GNR=**{GNR}** FUNC=**{FUNC}** CF=**{CF}** RESULT=**{RESULT}**
(4): **{D}** [**{PID}**/**{FU}**] APDU−ID=**{APDU}** GTYPE=**{GTYPE}** GNR=**{GNR}** INDEX1=**{INDEX1}** INDEX2=**{INDEX2}** OBJID.(**{N}**) = "**{OBJID}**"
(4): **{D}** [**{PID}**/**{FU}**] APDU−ID=**{APDU}** GTYPE=**{GTYPE}** GNR=**{GNR}** INDEX1=**{INDEX1}** INDEX2=**{INDEX2}** CF=**{CF}** RESULT=**{RESULT}**
(4): **{D}** [**{PID}**/**{FU}**] APDU−ID=**{APDU}** GTYPE=**{GTYPE}** GNR=**{GNR}** INDEX1=**{INDEX1}** INDEX2=**{INDEX2}** T0=**{T0}** DT=**{DT}** TUNIT=**{DT}** PERIODES=**{PERIODES}**
(4): **{D}** [**{PID}**/**{FU}**] APDU−ID=**{APDU}** GTYPE=**{GTYPE}** GNR=**{GNR}** INDEX1=**{INDEX1}** INDEX2=**{INDEX2}** T=**{T}** TRANSMOD=**{TRANSMOD}** MORE−D=**{MORE}** RESULT=**{RESULT}**
(4): **{D}** [**{PID}**/**{FU}**] APDU−ID=**{APDU}** GTYPE=**{GTYPE}** GNR=**{GNR}** INDEX1=**{INDEX1}** INDEX2=**{INDEX2}** T=**{T}** TMODE=**{TMODE}** CTYPE=**{CTYPE}** RESULT=**{RESULT}**

Code	**Value**	**Description**
{D}		Transmission direction
	<<	Send direction
	>>	Receive direction
{PID}	1..255	Partner ID
{FU}	1..7	Functional Unit
{LEN}	0..255	Length of the user data
{APDU}		APDU type
	InitTransfer	Initiate Transfer
	SendData	Send Data
	ConfirmData	Confirm Data
	TestConnReq	Test Connection Request
	TestConnRsp	Test Connection Response
	GroupMgntReq	Group Management Request
	GroupMgntRsp	Group Management Response
	DefineGroupReq	Define Group Request
	DefineGroupRsp	Define Group Response
	GetGroupReq	Get Group Request
	GetGroupRsp	Get Group Response
	SpontMgntReq	Spontaneous Management Request
	SpontMgntRsp	Spontaneous Management Response
	ErrorPDU	Error PDU
	ConnectReq	Connect Request
	ConnectRsp	Connect Response
	CmdTransferReq	Command Transfer Request
	CmdTransferRsp	Command Transfer Response
	SendMixedDataReq	Send Mixed Data Request
	SendMixedDataErrorReq	Send Mixed Data Error Request
{VERSION}	1	Elcom version
{CLASS}	3	Elcom class
{GTYPE}	1..8	Group type
{GNR}	1..32767	Group number
{PERSIST}	0..1	Persistent group
{STATIC}	0..1	Static group
{PRIO}	0..15	Priority
{GSIZE}	1..255	Group size

{OBJLEN}	1..255	Length of object identifiers
{FUNC}	1..4	Function key
{CF}	00..FF	Control field (timestamp)
{INDEX1}	1..32767	Starting Object Index
{INDEX2}	1..32767	Ending Object Index
{N}	0..N	Element number in object identifier string
{OBJID}	[−A−Za−z0−9_.]{1,255}	Object Identifier
{T0}	00..FF	Point of time for oldest group incarnation
{DT}	1..255	Time−slice between two consecutive group incarnations
{TUNIT}	1..7	Time−unit for Dt
{PERIODES}	1..32767	Number of group incarnations requested
{T}	00..FF	Time stamp
{TRANSMOD}	1..2	Mode of transmission
{MORE}	0..1	More data will follow
{TMODE}	0..3	Time mode (interpretation of T)
{CTYPE}	1..6, 252	Command type
{RESULT}	0..127	Result code
{DATA}	00..FF	User data field in hex presentation

Example of a logfile

```
24.10.13 16:29:22 Starting up in active mode ...
24.10.13 16:29:22 port binding succeeded !
24.10.13 16:29:22 Node connection established !
24.10.13 16:29:31 [?]  connection established !
24.10.13 16:29:31 [?]  connection established !
24.10.13 16:29:31 [?]  connection established !
24.10.13 16:29:31 [?]  connection established !
24.10.13 16:29:31 [?]  connection established !
(2): >> 31.109 [?] LEN=48 PPDU−ID=ConnectReq I−ADDR= A−ADDR= DATA=
(4): >> 31.109 [1/1] LEN=4 APDU−ID=ConnectReq VERSION=1 CLASS=3 DATA=
(4): <<31.109 [1/1] LEN=4 APDU−ID=ConnectRsp VERSION=1 CLASS=3 RESULT=0 DATA=
(2):  <<31.109 [1/1] LEN=49 PPDU−ID=ConnectRsp I−ADDR= A−ADDR= RESULT=0 DATA=
(2): >> 31.109 [?] LEN=48 PPDU−ID=ConnectReq I−ADDR= A−ADDR= DATA=
(4): >> 31.109 [1/2] LEN=4 APDU−ID=ConnectReq VERSION=1 CLASS=3 DATA=
(4):  <<31.109 [1/2] LEN=4 APDU−ID=ConnectRsp VERSION=1 CLASS=3 RESULT=0 DATA=
(2):  <<31.109 [1/2] LEN=49 PPDU−ID=ConnectRsp I−ADDR= A−ADDR= RESULT=0 DATA=
(2): >> 31.110 [?] LEN=48 PPDU−ID=ConnectReq I−ADDR= A−ADDR= DATA=
(4): >> 31.110 [1/7] LEN=4 APDU−ID=ConnectReq VERSION=1 CLASS=3 DATA=
(4):  <<31.110 [1/7] LEN=4 APDU−ID=ConnectRsp VERSION=1 CLASS=3 RESULT=0 DATA=
(2):  <<31.110 [1/7] LEN=49 PPDU−ID=ConnectRsp I−ADDR= A−ADDR= RESULT=0 DATA=
(2): >> 31.111 [?] LEN=48 PPDU−ID=ConnectReq I−ADDR= A−ADDR= DATA=
(4): >> 31.111 [1/3] LEN=4 APDU−ID=ConnectReq VERSION=1 CLASS=3 DATA=
24.10.13 16:29:31 Connection to partner 1 established !
(4):  <<31.111 [1/3] LEN=4 APDU−ID=ConnectRsp VERSION=1 CLASS=3 RESULT=0 DATA=
(2):  <<31.111 [1/3] LEN=49 PPDU−ID=ConnectRsp I−ADDR= A−ADDR= RESULT=0 DATA=
(2): >> 31.111 [?] LEN=48 PPDU−ID=ConnectReq I−ADDR= A−ADDR= DATA=
(4): >> 31.111 [1/5] LEN=4 APDU−ID=ConnectReq VERSION=1 CLASS=3 DATA=
(4):  <<31.111 [1/5] LEN=4 APDU−ID=ConnectRsp VERSION=1 CLASS=3 RESULT=0 DATA=
(2):  <<31.111 [1/5] LEN=49 PPDU−ID=ConnectRsp I−ADDR= A−ADDR= RESULT=0 DATA=
(2): >> 31.112 [1/1] LEN=9 PPDU−ID=DataReq DATA=
(4): >> 31.112 [1/1] APDU−ID=GroupMgntReq GTYPE=0 GNR=0 PERSIST=0 STATIC=0 PRIO=0 GSIZE=0 OBJLEN=0 FUNC=4
(4):  <<31.112 [1/1] APDU−ID=GroupMgntRsp GTYPE=0 GNR=0 FUNC=4 CF= RESULT=0
(2):  <<31.112 [1/1] LEN=19 PPDU−ID=DataReq DATA=
(2): >> 31.113 [1/1] LEN=9 PPDU−ID=DataReq DATA=
(4): >> 31.113 [1/1] APDU−ID=GroupMgntReq GTYPE=2 GNR=1 PERSIST=0 STATIC=0 PRIO=0 GSIZE=255 OBJLEN=255 FUNC=1
24.10.13 16:29:31 Group number 1 created !
(4):  <<31.113 [1/1] APDU−ID=GroupMgntRsp GTYPE=2 GNR=1 FUNC=1 CF= RESULT=0
(2):  <<31.113 [1/1] LEN=19 PPDU−ID=DataReq DATA=
(2): >> 31.113 [1/1] LEN=9 PPDU−ID=DataReq DATA=
(4): >> 31.113 [1/1] APDU−ID=GroupMgntReq GTYPE=1 GNR=2 PERSIST=0 STATIC=0 PRIO=1 GSIZE=255 OBJLEN=255 FUNC=1
24.10.13 16:29:31 Group number 2 created !
(4):  <<31.114 [1/1] APDU−ID=GroupMgntRsp GTYPE=1 GNR=2 FUNC=1 CF= RESULT=0
(2):  <<31.114 [1/1] LEN=19 PPDU−ID=DataReq DATA=
(2): >> 31.114 [1/1] LEN=9 PPDU−ID=DataReq DATA=
(4): >> 31.114 [1/1] APDU−ID=GroupMgntReq GTYPE=3 GNR=3 PERSIST=0 STATIC=0 PRIO=1 GSIZE=255 OBJLEN=255 FUNC=1
24.10.13 16:29:31 Group number 3 created !
(4):  <<31.114 [1/1] APDU−ID=GroupMgntRsp GTYPE=3 GNR=3 FUNC=1 CF= RESULT=0
(2):  <<31.114 [1/1] LEN=19 PPDU−ID=DataReq DATA=
(2): >> 31.115 [1/1] LEN=9 PPDU−ID=DataReq DATA=
(4): >> 31.115 [1/1] APDU−ID=GroupMgntReq GTYPE=6 GNR=4 PERSIST=0 STATIC=0 PRIO=0 GSIZE=255 OBJLEN=255 FUNC=1
24.10.13 16:29:31 Group number 4 created !
(4):  <<31.115 [1/1] APDU−ID=GroupMgntRsp GTYPE=6 GNR=4 FUNC=1 CF= RESULT=0
(2):  <<31.115 [1/1] LEN=19 PPDU−ID=DataReq DATA=
(2): >> 31.115 [1/1] LEN=9 PPDU−ID=DataReq DATA=
(4): >> 31.115 [1/1] APDU−ID=GroupMgntReq GTYPE=7 GNR=5 PERSIST=0 STATIC=0 PRIO=0 GSIZE=255 OBJLEN=255 FUNC=1
24.10.13 16:29:31 Group number 5 created !
(4):  <<31.115 [1/1] APDU−ID=GroupMgntRsp GTYPE=7 GNR=5 FUNC=1 CF= RESULT=0
(2):  <<31.116 [1/1] LEN=19 PPDU−ID=DataReq DATA=
(2): >> 31.116 [1/1] LEN=9 PPDU−ID=DataReq DATA=
(4): >> 31.116 [1/1] APDU−ID=GroupMgntReq GTYPE=5 GNR=6 PERSIST=0 STATIC=0 PRIO=0 GSIZE=255 OBJLEN=255 FUNC=1
24.10.13 16:29:31 Group number 6 created !
(4):  <<31.116 [1/1] APDU−ID=GroupMgntRsp GTYPE=5 GNR=6 FUNC=1 CF= RESULT=0
(2):  <<31.116 [1/1] LEN=19 PPDU−ID=DataReq DATA=
(2): >> 31.117 [1/1] LEN=21 PPDU−ID=DataReq DATA=
(4): >> 31.117 [1/1] APDU−ID=DefineGroupReq GTYPE=2 GNR=1 INDEX1=1 INDEX2=1
        OBJID.(0)       = "Breaker_X5";
24.10.13 16:29:31 Object "Breaker_X5" added to group 1 !
(4):  <<31.117 [1/1] APDU−ID=DefineGroupRsp GTYPE=2 GNR=1 INDEX1=1 INDEX2=1 CF= RESULT=
(2):  <<31.117 [1/1] LEN=23 PPDU−ID=DataReq DATA=
(2): >> 31.117 [1/1] LEN=32 PPDU−ID=DataReq DATA=
(4): >> 31.117 [1/1] APDU−ID=DefineGroupReq GTYPE=1 GNR=2 INDEX1=1 INDEX2=2
        OBJID.(0)       = "Current_I2";
        OBJID.(1)       = "Voltage_U1";
24.10.13 16:29:31 Object "Current_I2" added to group 2 !
24.10.13 16:29:31 Object "Voltage_U1" added to group 2 !
(4):  <<31.118 [1/1] APDU−ID=DefineGroupRsp GTYPE=1 GNR=2 INDEX1=1 INDEX2=2 CF= RESULT=
(2):  <<31.118 [1/1] LEN=24 PPDU−ID=DataReq DATA=
(2): >> 31.118 [1/1] LEN=22 PPDU−ID=DataReq DATA=
(4): >> 31.118 [1/1] APDU−ID=DefineGroupReq GTYPE=3 GNR=3 INDEX1=1 INDEX2=1
        OBJID.(0)       = "TAP_POS_T43";
24.10.13 16:29:31 Object "TAP_POS_T43" added to group 3 !
(4):  <<31.118 [1/1] APDU−ID=DefineGroupRsp GTYPE=3 GNR=3 INDEX1=1 INDEX2=1 CF= RESULT=
(2):  <<31.118 [1/1] LEN=23 PPDU−ID=DataReq DATA=
(2): >> 31.119 [1/2] LEN=6 PPDU−ID=DataReq DATA=
(4): >> 31.119 [1/2] APDU−ID=SpontMgntReq GTYPE=2 GNR=1 FUNC=1
(4):  <<31.119 [1/2] APDU−ID=SpontMgntRsp GTYPE=2 GNR=1 FUNC=1 RESULT=0
(2):  <<31.119 [1/2] LEN=7 PPDU−ID=DataReq DATA=
(2): >> 31.119 [1/3] LEN=6 PPDU−ID=DataReq DATA=
(4): >> 31.119 [1/3] APDU−ID=SpontMgntReq GTYPE=1 GNR=2 FUNC=1
(4):  <<31.120 [1/3] APDU−ID=SpontMgntRsp GTYPE=1 GNR=2 FUNC=1 RESULT=0
(2):  <<31.120 [1/3] LEN=7 PPDU−ID=DataReq DATA=
(2): >> 31.120 [1/3] LEN=6 PPDU−ID=DataReq DATA=
(4): >> 31.120 [1/3] APDU−ID=SpontMgntReq GTYPE=3 GNR=3 FUNC=1
(4):  <<31.120 [1/3] APDU−ID=SpontMgntRsp GTYPE=3 GNR=3 FUNC=1 RESULT=0
(2):  <<31.120 [1/3] LEN=7 PPDU−ID=DataReq DATA=
(2): >> 31.121 [1/5] LEN=21 PPDU−ID=DataReq DATA=
(4): >> 31.121 [1/5] APDU−ID=InitTransfer GTYPE=2 GNR=1 INDEX1=0 INDEX2=0 T0= DT=1 TUNIT=6 PERIODES=1
(4):  <<31.121 [1/5] APDU−ID=SendData GTYPE=2 GNR=1 INDEX1=1 INDEX2=1 T= TRANSMOD=1 MORE−D=0 RESULT=0 DATA=
(2):  <<31.121 [1/5] LEN=21 PPDU−ID=DataReq DATA=
(2): >> 31.121 [1/5] LEN=7 PPDU−ID=DataReq DATA=
(4): >> 31.121 [1/5] APDU−ID=ConfirmData GTYPE=2 GNR=1 TRANSMOD=1 RESULT=0
(2): >> 31.122 [1/5] LEN=21 PPDU−ID=DataReq DATA=
(4): >> 31.122 [1/5] APDU−ID=InitTransfer GTYPE=1 GNR=2 INDEX1=0 INDEX2=0 T0= DT=1 TUNIT=6 PERIODES=1
(4):  <<31.122 [1/5] APDU−ID=SendData GTYPE=1 GNR=2 INDEX1=1 INDEX2=2 T= TRANSMOD=1 MORE−D=0 RESULT=0 DATA=
(2):  <<31.122 [1/5] LEN=30 PPDU−ID=DataReq DATA=
(2): >> 31.122 [1/5] LEN=7 PPDU−ID=DataReq DATA=
(4): >> 31.122 [1/5] APDU−ID=ConfirmData GTYPE=1 GNR=2 TRANSMOD=1 RESULT=0
(2): >> 31.123 [1/5] LEN=21 PPDU−ID=DataReq DATA=
(4): >> 31.123 [1/5] APDU−ID=InitTransfer GTYPE=3 GNR=3 INDEX1=0 INDEX2=0 T0= DT=1 TUNIT=6 PERIODES=1
(4):  <<31.123 [1/5] APDU−ID=SendData GTYPE=3 GNR=3 INDEX1=1 INDEX2=1 T= TRANSMOD=1 MORE−D=0 RESULT=0 DATA=
(2):  <<31.123 [1/5] LEN=23 PPDU−ID=DataReq DATA=
(2): >> 31.123 [1/5] LEN=7 PPDU−ID=DataReq DATA=
(4): >> 31.123 [1/5] APDU−ID=ConfirmData GTYPE=3 GNR=3 TRANSMOD=1 RESULT=0
```

Elcom90Responder − Parameter

.name

default: ELC90R

Module name

Partner List

.partner.[{PID}]

{PID}

default: 1 minimum: 1 maximum: 255

Partner ID

Any unique number to identify the connection to a specific communication partner.

.partner.[{PID}].client_ip

pattern: ^[0−9]+\.[0−9]+\.[0−9]+\.[0−9]+$

Client IP address

This parameter defines the IP address of the client. It is required to identify an Elcom−90 Initiator in the Responder part.

General parameters

.server_ip

pattern: ^[0−9]+\.[0−9]+\.[0−9]+\.[0−9]+$

Server IP address

This entry defines the IP address the server is bound to.

.server_port

default: 5997 minimum: 1 maximum: 65535

Server TCP/IP port

This defines the TCP/IP port used by the server.

.bind_on_ip = { TRUE | FALSE }

default: FALSE

bind on a dedicated local IP address (=TRUE) or not (=FALSE)

Determines whether server should bind on a dedicated local IP address (=TRUE) or on any IP address (=FALSE)

.keepalive

unit: [s] default: 5 minimum: 1 maximum: 20

TCP keepalive interval

.timeout

unit: [s] default: 10 minimum: 1 maximum: 65535

Request Timeout

If no response to a request is received within the period of time specified here, the connection to the corresponding communication partner is considered as to be disturbed. All subconnections will be aborted.

.utc_time = { TRUE | FALSE }

default: TRUE

use UTC timezone (=TRUE) or local timezone (=FALSE) for time conversion

If this parameter is set on 'TRUE', timestamps within sent and received telegrams are treated as UTC times. If set on 'FALSE', timestamps are treated as to be local times.

.utc_time = FALSE

.tzsource = { OS | SYSTEM | SPECIFIC | MANUAL }

default: OS

type of timezone conversion mechanism

Defines, which timezone calculation mechanism is used by this protocol stack for conversion of local times into UTC times and vice versa. Following settings may be applied:

VALUE	DESCRIPTION
OS	use operating system builtin time conversion functions, based on global timezone setting
SYSTEM	use ipConv builtin time conversion functions, based on global timezone setting
SPECIFIC	timezone region and zone explicitly used by this protocol stack
MANUAL	timezone string in Posix notation explicitly used by this protocol stack

.tzsource = SPECIFIC

.tzregion = {...}

default: Etc

Region

.tzregion = Africa

.tzzone = {...}

default: Abidjan

Zone

.tzregion = America/Kentucky

.tzzone = { Louisville | Monticello }

default: Louisville

Zone

.tzregion = America/Indiana

.tzzone = { Indianapolis | Knox | Marengo | Petersburg | Tell_City | Vevay | Vincennes | Winamac }

default: Indianapolis

Zone

.tzregion = America/North_Dakota

.tzzone = { Beulah | Center | New_Salem }

default: Center

Zone

.tzregion = America/Argentina

.tzzone = { Buenos_Aires | Catamarca | ComodRivadavia | Cordoba | Jujuy | La_Rioja | Mendoza | Rio_Gallegos | Salta | San_Juan | San_Luis | Tucuman | Ushuaia }

default: Buenos_Aires

Zone

.tzregion = America

.tzzone = {...}

default: Adak

Zone

.tzregion = Antarctica

.tzzone = { Casey | Davis | DumontDUrville | Macquarie | Mawson | McMurdo | Palmer | Rothera | South_Pole | Syowa | Vostok }

default: Casey

Zone

.tzregion = Arctic

.tzzone = { Longyearbyen }

default: Longyearbyen

Zone

.tzregion = Asia

.tzzone = {...}

default: Aden

Zone

.tzregion = Atlantic

.tzzone = { Azores | Bermuda | Canary | Cape_Verde | Faeroe | Faroe | Jan_Mayen | Madeira | Reykjavik | South_Georgia | St_Helena | Stanley }

default: Azores

Zone

.tzregion = Australia

.tzzone = { ACT | Adelaide | Brisbane | Broken_Hill | Canberra | Currie | Darwin | Eucla | Hobart | LHI | Lindeman | Lord_Howe | Melbourne | North | NSW | Perth | Queensland | South | Sydney | Tasmania | Victoria | West | Yancowinna }

default: ACT

Zone

.tzregion = Brazil

.tzzone = { Acre | DeNoronha | East | West }

default: Acre

Zone

.tzregion = Canada

.tzzone = { Atlantic | Central | Eastern | East−Saskatchewan | Mountain | Newfoundland | Pacific | Saskatchewan | Yukon }

default: Atlantic

Zone

.tzregion = Chile

.tzzone = { Continental | EasterIsland }

default: Continental

Zone

.tzregion = Etc

.tzzone = { GMT | GMT0 | GMT−0 | GMT+0 | GMT−1 | GMT+1 | GMT−10 | GMT+10 | GMT−11 | GMT+11 | GMT−12 | GMT+12 | GMT−13 | GMT−14 | GMT−2 | GMT+2 | GMT−3 | GMT+3 | GMT−4 | GMT+4 | GMT−5 | GMT+5 | GMT−6 | GMT+6 | GMT−7 | GMT+7 |

GMT−8 | GMT+8 | GMT−9 | GMT+9 | Greenwich | UCT | Universal | UTC | Zulu }

default: UTC

Zone

.tzregion = Europe

.tzzone = {...}

default: Berlin

Zone

.tzregion = Indian

.tzzone = { Antananarivo | Chagos | Christmas | Cocos | Comoro | Kerguelen | Mahe | Maldives | Mauritius | Mayotte | Reunion }

default: Antananarivo

Zone

.tzregion = Mexico

.tzzone = { BajaNorte | BajaSur | General }

default: BajaNorte

Zone

.tzregion = Pacific

.tzzone = { Apia | Auckland | Chatham | Chuuk | Easter | Efate | Enderbury | Fakaofo | Fiji | Funafuti | Galapagos | Gambier | Guadalcanal | Guam | Honolulu | Johnston | Kiritimati | Kosrae | Kwajalein | Majuro | Marquesas | Midway | Nauru | Niue | Norfolk | Noumea | Pago_Pago | Palau | Pitcairn | Pohnpei | Ponape | Port_Moresby | Rarotonga | Saipan | Samoa | Tahiti | Tarawa | Tongatapu | Truk | Wake | Wallis | Yap }

default: Apia

Zone

.tzregion = US

.tzzone = { Alaska | Aleutian | Arizona | Central | Eastern | East−Indiana | Hawaii | Indiana−Starke | Michigan | Mountain | Pacific | Pacific−New | Samoa }

default: Alaska

Zone

.tzsource = MANUAL

.tzstring

default: CET−1CEST,M3.5.0/2,M10.5.0/3

Timezone string in Posix notation

The value of the TZ string can be in one of two formats. The first format is used when there is no Daylight Saving Time (or summer time) in the local time zone:

std offset

The *std* string specifies the name of the time zone. It must be three or more characters long and must not contain a leading colon, embedded digits, commas, nor plus and minus signs. There is no space character separating the time zone name from the *offset*, so these restrictions are necessary to parse the specification correctly.

The *offset* specifies the time value you must add to the local time to get a Coordinated Universal Time value. It has syntax like [+|−]*hh*[:*mm*[:*ss*]]. This is positive if the local time zone is west of the Prime Meridian and negative if it is east. The hour must be between `0` and `23`, and the minute and seconds between `0` and `59`.

For example, here is how we would specify Eastern Standard Time, but without any Daylight Saving Time alternative:

```
EST+5
```

The second format is used when there is Daylight Saving Time:

std offset dst [*offset*],*start*[/*time*],*end*[/*time*]

The initial *std* and *offset* specify the standard time zone, as described above. The *dst* string and *offset* specify the name and offset for the corresponding Daylight Saving Time zone; if the *offset* is omitted, it defaults to one hour ahead of standard time.

The remainder of the specification describes when Daylight Saving Time is in effect. The *start* field is when Daylight Saving Time goes into effect and the *end* field is when the change is made back to standard time. The following formats are recognized for these fields:

Jn

This specifies the Julian day, with *n* between `1` and `365`. February 29 is never counted, even in leap years.

n

This specifies the Julian day, with *n* between `0` and `365`. February 29 is counted in leap years.

Mm.w.d

This specifies day *d* of week *w* of month *m*. The day *d* must be between `0` (Sunday) and `6`. The week *w* must be between `1` and `5`; week `1` is the first week in which day *d* occurs, and week `5` specifies the *last d* day in the month. The month *m* should be between `1` and `12`.

The *time* fields specify when, in the local time currently in effect, the change to the other time occurs. If omitted, the default is `02:00:00`.

For example, here is how you would specify the Eastern time zone in the United States, including the appropriate Daylight Saving Time and its dates of applicability. The normal offset from UTC is 5 hours; since this is west of the prime meridian, the sign is positive. Summer time begins on the first Sunday in April at 2:00am, and ends on the last Sunday in October at 2:00am.

```
EST+5EDT,M4.1.0/2,M10.5.0/2
```

The schedule of Daylight Saving Time in any particular jurisdiction has changed over the years. To be strictly correct, the conversion of dates and times in the past should be based on the schedule that was in effect then. However, this format has no facilities to let you specify how the schedule has changed from year to year. The most you can do is specify one particular schedule usually the present day schedule and this is used to convert any date, no matter when.

(Source: http://www.gnu.org, TZ Variable, 26.02.2013)

.mixed_data = { TRUE | FALSE }

default: FALSE

use Mixed Data Transfer FU (=TRUE) or Data Transfer FU (=FALSE)

If the parameter is set to "TRUE", the Unsolicited Mixed Data Transfer FU (Functional Unit) is used to transmit spontaneous events. Here several objects from different groups can be sent within the same telegram. However the reception of the data is not confirmed. If this parameter is set to "FALSE", the Unsolicited Data Transfer FU is used. Here only one object is transmitted at a time. The reception has to be confirmed, before the next object is sent.

.sel_timeout

unit: [s] default: 30 minimum: 1 maximum: 65535

timeout for command selection

Defines the command selection monitoring timeout in seconds. If no EXC (execute command) or IHC (inhibit command) request is received within the time specified here, the command is cancelled.

.exec_timeout

unit: [s] default: 30 minimum: 1 maximum: 65535

timeout for command execution

Defines the command execution monitoring timeout in seconds. The initiator can send CBXC (check back before execute command), EXC (execute command), IHC (inhibit command) or IXC (immediate execute command) requests to the responder. The responder sends command(s) to the Node with the qualifier set to select, execute, cancel or direct. If no reply is received from the Node for a particular command within the time specified here, the operation is considered to be failed and the command quality code of the appropriate object is set to "not OK".

.startup = { TRUE | FALSE }

default: FALSE

startup immediately (=TRUE) or wait for startup release from Node (=FALSE)

If the parameter is set on "TRUE", the module starts up immediately after loading. If set on "FALSE", startup is performed after receiving the normalized information "int−startup" from the node. Before this time the module does not respond to any request from masterstation. Standard setting for this parameter is "FALSE".

.active = { TRUE | FALSE }

default: TRUE

startup in active mode (=TRUE) or in passive mode (=FALSE)

This parameter defines the state, in which the module starts up it's operation in general. If set on "TRUE", the module starts up in active state. If set on "FALSE", the module starts up in passive state and can be activated only with the normalized information "int−active" with the "APP" value from the node.

.level = { 0 | 1 | 2 | 4 | 6 | 7 }

default: 0

Logging Level

LEVEL	DESCRIPTION
0	no logging
1	hexadecimal representation of transport layer telegrams
2	decoded representation of presentation layer frames
4	decoded representation of application layer frames
6	levels 2 and 4
7	levels 1, 2 and 4

Extended settings

Priority class to cycle time

.prio2cycle.[{PRIO}]

{PRIO}

default: 0 minimum: 0 maximum: 15

Priority class

.prio2cycle.[{PRIO}].cycle_time

unit: [ms] default: 5000 minimum: 11 maximum: 3600000

Cycle time

Priority class	Default cycle time
0	50 seconds
1	5 seconds
2	8 seconds
3	10 seconds
4	12 seconds
5	15 seconds
6	18 seconds
7	20 seconds
8	25 seconds
9	28 seconds

10	30 seconds
11	32 seconds
12	35 seconds
13	40 seconds
14	45 seconds
15	50 seconds