j60870 User Guide

Fraunhofer Institute for Solar Energy Systems ISE

openmuc.org

Table of Contents

1. Intro

j60870 is an implementation of the IEC 60870-5-104 protocol standard for client (i.e. master or controlling station) and server (i.e. slave or controlled station) applications.

You can use j60870 to program your individual client or server applications. A simple console client is part of the library. You can execute it using the scripts found in the folder "run-scripts".

2. Distribution

After extracting the distribution tar file the j60870 library can be found in the folder /build/libs.

3. Getting Started

The easiest way to get started is by taking a look at the code of the console client and the sample server which can be found here: cli-app/src/main/java/org/openmuc/j60870/app/ . These applications in combination with the javadoc should satisfy most of your documentation needs.

Here is a short summary of the steps to get a client running:

- Create and configure an instance of ClientConnectionBuilder.
- Connect to the server using ClientConnectionBuilder.build() which returns the connection. The client is now connected to the server via TCP/IP.
- Initialize the data transfer by calling Connection.startDataTransfer.
- Now all incoming ASDUs (except for confirmation messages) will be forwarded to the ASduListener you registered. Every ASDU contains a number of Information Objects. The information objects contain information elements that make up the actual data. You will have to cast the InformationElements of the ASDU to a concrete implementation in order to access the data inside them. Every standardized Information Element is implemented by a class starting with the letters "Ie". The Type Identifier allows you to figure what to cast a particular Information Element to.
- You can use the Connection instance to send commands.

4. Terminology

- **OA** - Originator Address
- **Monitor direction** - direction from server to client
- **Control direction** - direction from client to server
- **CON** - confirmation message
- **COT** - Cause of transmission

- **STARTDT ACT** - Start data tranfer message. Needs to be sent by the client before information messages may be exchanged between client and server.

5. Features

- Server implementation
- Client implementation

The usage and description can be seen in the j60870 javadoc.

5.1. Supported information Elements

Abstract Qualifier of Command	Qualifier of Interrogation
Abstract Quality	Qualifier of Parameter Activation
Ack File or Section Qualifier	Qualifier of Parameter of Measured Values
Binary Counter Reading	Qualifier of Reset Process Command
Binary State Information	Qualifier of Set Point Command
Cause of Initialization	Quality
Checksum	Regulating Step Command
Double Command	Scaled Value
Double Point with Quality	Section Ready Qualifier
File Ready Qualifier	Select and Call Qualifier
File Segment	Short Float
Fixed Test Bit Pattern	Single Command
Last Section or Segment Qualifier	Single Point with Quality
Length of File or Section	Single Protection Event
Name of File	Status and Status Changes
Name of Section	Status of File
Normalized Value	Test Sequence Counter
Protection Output Circuit Information	Time16
Protection Quality	Time24
Protection Start Event	Time56
Qualifier of Counter Interrogation	Value with Transient State

5.2. Supported ASDu types (Client and Server)

ASDu	Number	Description
C_BO_NA_1	51	Bitstring of 32 bits
C_BO_TA_1	64	Bitstring of 32 bit with time tag CP56Time2a
C_CD_NA_1	106	Delay acquisition command
C_CI_NA_1	101	Counter interrogation command
C_CS_NA_1	103	Clock synchronization command
C_DC_NA_1	46	Double command
C_DC_TA_1	59	Double command with time tag CP56Time2a
C_IC_NA_1	100	Interrogation command

ASDu	Number	Description
C_RC_NA_1	47	Regulating step command
C_RC_TA_1	60	Regulating step command with time tag CP56Time2a
C_RD_NA_1	102	Read command
C_RP_NA_1	105	Reset process command
C_SC_NA_1	45	Single command
C_SC_TA_1	58	Single command with time tag CP56Time2a
C_SE_NA_1	48	Set point command, normalized value
C_SE_NB_1	49	Set point command, scaled value
C_SE_NC_1	50	Set point command, short floating point number
C_SE_TA_1	61	Set-point command with time tag CP56Time2a, normalized value
C_SE_TB_1	62	Set-point command with time tag CP56Time2a, scaled value
C_SE_TC_1	63	C_SE_TC_1 Set-point command with time tag CP56Time2a, short floating point number
C_TS_NA_1	104	Test command
C_TS_TA_1	107	Test command with time tag CP56Time2a
F_AF_NA_1	124	Ack file, ack section
F_DR_TA_1	126	Directory
F_FR_NA_1	120	File ready
F_LS_NA_1	123	Last section, last segment
F_SC_NA_1	122	Call directory, select file, call file, call section
F_SC_NB_1	127	QueryLog, request archive file
F_SG_NA_1	125	Segment
F_SR_NA_1	121	Section ready
M_BO_NA_1	7	Bitstring of 32 bit
M_BO_TA_1	8	Bitstring of 32 bit with time tag
M_BO_TB_1	33	Bitstring of 32 bits with time tag CP56Time2a
M_DP_NA_1	3	Double-point information without time tag
M_DP_TA_1	4	Double-point information with time tag
M_DP_TB_1	31	Double-point information with time tag CP56Time2a
M_EI_NA_1	70	End of initialization
M_EP_TA_1	17	Event of protection equipment with time tag
M_EP_TB_1	18	Packed start events of protection equipment with time tag
M_EP_TC_1	19	Packed output circuit information of protection equipment with time tag
M_EP_TD_1	38	Event of protection equipment with time tag CP56Time2a
M_EP_TE_1	39	Packed start events of protection equipment with time tag CP56Time2a
M_EP_TF_1	40	Packed output circuit information of protection equipment with time tag CP56Time2
M_IT_NA_1	15	Integrated totals
M_IT_TA_1	16	Integrated totals with time tag
M_IT_TB_1	37	Integrated totals with time tag CP56Time2a
M_ME_NA_1	9	Measured value, normalized value
M_ME_NB_1	11	Measured value, scaled value
M_ME_NC_1	13	Measured value, short floating point number
M_ME_ND_1	21	Measured value, normalized value without quality descriptor
M_ME_TA_1	10	Measured value, normalized value with time tag
M_ME_TB_1	12	Measured value, scaled value with time tag
M_ME_TC_1	14	Measured value, short floating point number with time tag
M_ME_TD_1	34	Measured value, normalized value with time tag CP56Time2a

ASDu	Number	Description
M_ME_TE_1	35	Measured value, scaled value with time tag CP56Time2a
M_ME_TF_1	36	Measured value, short floating point number with time tag CP56Time2a
M_PS_NA_1	20	Packed single-point information with status change detection
M_SP_NA_1	1	Single-point information without time tag
M_SP_TA_1	2	Single-point information with time tag
M_SP_TB_1	30	Single-point information with time tag CP56Time2a
M_ST_NA_1	5	Step position information
M_ST_TA_1	6	Step position information with time tag
M_ST_TB_1	32	Step position information with time tag CP56Time2a
P_AC_NA_1	113	Parameter activation
P_ME_NA_1	110	Parameter of measured value, normalized value
P_ME_NB_1	111	Parameter of measured value, scaled value
P_ME_NC_1	112	Parameter of measured value, short floating point number

5.3. In client connection implemented commands

Confirmation	clock synchronization command
single Command	test command
single Command with Time Tag	process command
double Command	delay acquisition command
double Command with Time Tag	test command with time tag CP56Time2a
regulating Step Command	parameter of measured values, normalized value
regulating Step Command with Time Tag	parameter of measured values, scaled value
Normalized Value Command	parameter of measured values, short floating point number
set Normalized Value Command with Time Tag	parameter activation
set Scaled Value Command	file Ready
set Scaled Value Command with Time Tag	section Ready
set Short Float Command	call or Select Files
set Short Float Command with Time Tag	last Section or Segment
bit String Command	ack File or Section
bit String Command with Time Tag	Segment
interrogation command	Directory
counter interrogation command	query Log
read command	

6. Develop j60870

We use the Gradle build automation tool. The distribution contains a fully functional gradle build file ("build.gradle"). Thus if you changed code and want to rebuild a library you can do it easily with Gradle. Also if you want to import our software into Eclipse you can easily create Eclipse project files using Gradle. Just follow the instructions on our FAQ site: https://www.openmuc.org/faq/

7. Authors

Developers:

- Dirk Zimmermann

Former Developers:

- Stefan Feuerhahn
- Albrecht Schall