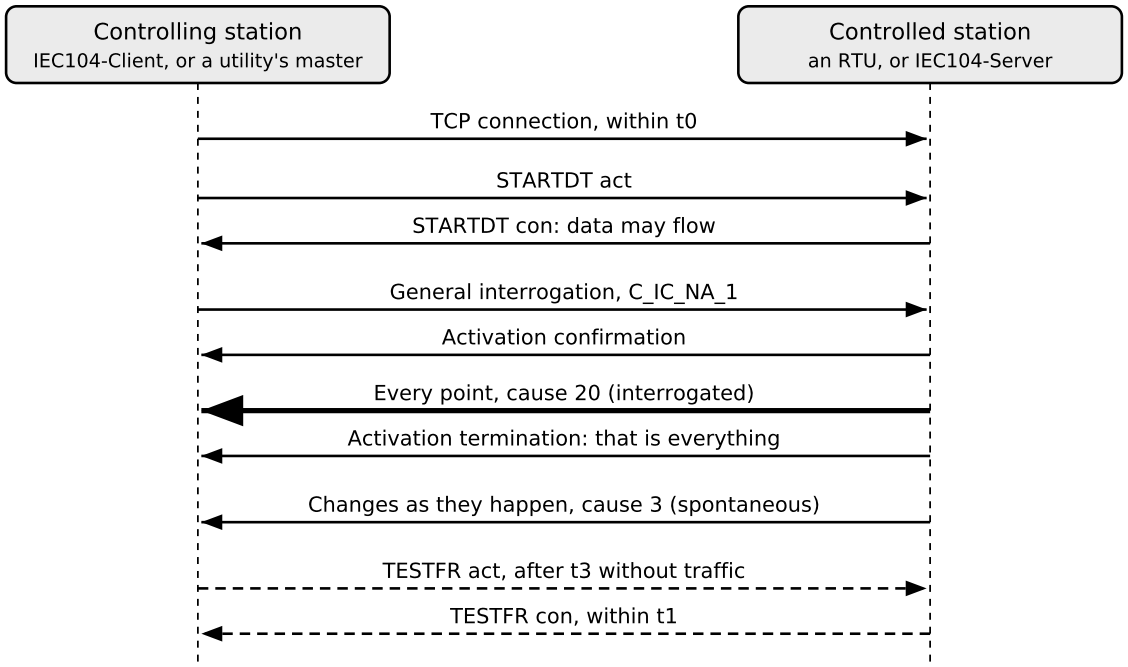
Amtiri

REFERENCE MANUAL

IEC 60870-5-104

Generated on November 25, 2026



IEC 60870-5-104 is the telecontrol protocol of water, power and gas utilities. Two modules speak it. The client is a **controlling station**: it dials an RTU or a substation gateway, interrogates it and sends it commands. The server is a **controlled station**: it answers a utility's SCADA master, and presents the instance's own variables as telecontrol points.

Module	What it does	Typical use
`IEC104-Client` (IEC 60870-5-104 Client)	Takes the points a station reports into variables, and sends variables to the station as commands	Reading and commanding an RTU or a substation gateway
`IEC104-Server` (IEC 60870-5-104 Server)	Reports variables as points to the masters it allows, and accepts commands into writable points	Being read, and commanded, by a utility's SCADA

Enable, **Name** and the **Status and messages** tab work the same way in every module. They are described in Settings every module shares.

> **Caution**
>
> **This protocol has no authentication and no encryption.** Anyone who can reach port 2404 can read every point and operate every writable one. The server's list of allowed masters is the only control either module offers, and it is not a substitute for a network: put a 104 link on a VPN or a dedicated circuit, never on anything reachable from the internet. IEC 62351 security is not implemented.

A session, not a poll

A 104 link is a **session**. It comes up, it is told to start carrying data, and from then on the controlled station speaks when something changes. Nothing is polled, so a point has no schedule of its own: it has an address.

1. The controlling station opens the TCP connection and sends STARTDT.
2. It interrogates the station: "send me everything you have".
3. The station sends every point, then says it has finished.
4. From then on, the station sends each change as it happens.

Until a station has been interrogated, a variable holds whatever it held before the link came up. That is why the client interrogates as soon as the link starts.

Points and types

A point is an **information object address** (IOA), a number from 0 to 16777215 that the utility assigns, and a **type**, which says what the point carries. Every point of a link belongs to one station, named by its **common address**.

A type can be named in three ways, and case does not matter:

- **By family**, such as `M_SP`. A collector named by family accepts every exact type of it: `M_SP` takes both `M_SP_NA_1` and the time-tagged `M_SP_TB_1`, because which of the two a station sends is the station's decision and can change with its firmware.
- **By its exact name**, such as `M_ME_TF_1`, for a point list that arrived written that way.
- **By its exact name without** `_1`, such as `M_ME_TF` or `C_SC_TA`, as most point lists write it.

What is sent, as a command by the client or as a report by the server, is the exact type named. A family name alone sends the form without a time tag, and a name ending in `_TB` or `_TA` sends the time-tagged form.

Family	Exact types	Carries	Variable type	Values
`M_SP`	`M_SP_NA_1`, `M_SP_TB_1`	Single point	boolean	
`M_DP`	`M_DP_NA_1`, `M_DP_TB_1`	Double point	int	0 to 3: 1 off, 2 on, 0 and 3 indeterminate
`M_ST`	`M_ST_NA_1`, `M_ST_TB_1`	Step position	int	-64 to 63
`M_BO`	`M_BO_NA_1`, `M_BO_TB_1`	Bitstring of 32 bits	int	
`M_ME_NA`	`M_ME_NA_1`, `M_ME_TD_1`, `M_ME_ND_1`	Normalised value	float	-1 to just under 1
`M_ME_NB`	`M_ME_NB_1`, `M_ME_TE_1`	Scaled value	int	-32768 to 32767
`M_ME_NC`	`M_ME_NC_1`, `M_ME_TF_1`	Short floating point value	float	
`M_IT`	`M_IT_NA_1`, `M_IT_TB_1`	Integrated total, a counter	int	A signed 32-bit number
`C_SC`	`C_SC_NA_1`, `C_SC_TA_1`	Single command	boolean	
`C_DC`	`C_DC_NA_1`, `C_DC_TA_1`	Double command	int	0 to 3
`C_RC`	`C_RC_NA_1`, `C_RC_TA_1`	Regulating step command	int	0 to 3
`C_SE_NA`	`C_SE_NA_1`, `C_SE_TA_1`	Set-point, normalised	float	
`C_SE_NB`	`C_SE_NB_1`, `C_SE_TB_1`	Set-point, scaled	int	
`C_SE_NC`	`C_SE_NC_1`, `C_SE_TC_1`	Set-point, short floating point	float	
`C_BO`	`C_BO_NA_1`, `C_BO_TA_1`	Bitstring command	int	

The variable type is what the **Type / problem** column of the editors shows, and what the pencil beside **Variable** creates a variable as.

> **Important**
>
> **A value the station marks invalid is not written into its variable.** Every monitored point carries quality bits. `IV` (invalid) or `NT` (not topical) mean the station is saying not to believe the number, so the client leaves the variable as it was and warns. A value that is only blocked, substituted or overflowed is written: each of those is information about a real value.

Link parameters

`k`, `w` and `t0` to `t3` have to match what the far end uses. A utility states all six in its interoperability document, and the defaults are the standard's own. Both modules check them when they open, and refuse to open with values that break a rule below.

Parameter	Default	What it decides	Rule
`k`	12	How many sent I-frames may wait for an acknowledgement. More wait in a queue until the far end acknowledges	1 to 32767
`w`	8	After how many received I-frames an acknowledgement is sent without waiting	1 to `k`
`t0`	30 s	How long the client waits for the TCP connection. The server does not use it	At least 1
`t1`	15 s	How long a sent I-frame or test frame may go unacknowledged before the link is closed	Longer than `t2`
`t2`	10 s	The longest an acknowledgement is held back	Shorter than `t1`
`t3`	20 s	How long the link may be silent before a test frame is sent	Longer than `t2`

Two ends that disagree about `k` produce a link that stalls under load and works perfectly when idle; two that disagree about `t1` produce one that drops every few minutes for no reason either end can see.

Other ways a link ends, as the standard requires: an I-frame with the wrong sequence number, an acknowledgement for a frame that was never sent, or a frame that cannot even be delimited. An ASDU that is merely unreadable is dropped with a warning, and the link stays up.

IEC 60870-5-104 Client

The `IEC104-Client` module keeps one link to one station, for as long as it is open. **Collectors** take the points the station reports into variables. **Forwarders** send a variable to the station as a command whenever the variable changes.

Settings

Key	Setting	Type	Default	Meaning
`host`	Host	string	(empty)	The station's address. Required: without it the module does not open
`port`	Port	int	2404	The TCP port
`commonAddress`	Common address	int	1	The station's common address, sent in every ASDU. An ASDU for another common address is ignored, as a gateway sends the points of several stations down one link; the broadcast address 65535 is accepted
`originatorAddress`	Originator address	int	0	Sent in every ASDU, for a station that serves several masters. 0 when nobody uses it
`interrogationCron`	Interrogation schedule	Quartz cron	(empty)	Repeats the general interrogation on a schedule. Empty: only the one sent each time the link starts
`clockSyncCron`	Clock sync schedule	Quartz cron	(empty)	Offers this host's clock to the station on a schedule. Empty: never
`k`	k	int	12	See Link parameters
`w`	w	int	8	See Link parameters
`t0`	t0 (s)	int, s	30	See Link parameters
`t1`	t1 (s)	int, s	15	See Link parameters
`t2`	t2 (s)	int, s	10	See Link parameters
`t3`	t3 (s)	int, s	20	See Link parameters
`collectors`	Collector	list of rows	empty	The points taken into variables
`forewarders`	Forwarder	list of rows	empty	The commands sent from variables. The key is spelled `forewarders`

Collectors and forwarders

A forwarder row uses every key below. A collector row uses `field`, `ioa` and `typeId`.

Key	Column	Type	Default	Meaning
`field`	Variable	variable code	(none)	Collector: the variable the point's value is written to. Forwarder: the variable whose changes are sent as commands
`ioa`	IOA	int	0	The information object address, 0 to 16777215
`typeId`	Type	string	`M_SP` in a collector, `C_SC` in a forwarder	The point's type, named as Points and types says. A collector takes a monitored type (`M_…`), a forwarder a command (`C_…`)
`selectBeforeOperate`	SBO	boolean	false	Forwarders only: select first, and execute only once the station confirms the selection

The **Type / problem** column is not stored: it shows what is wrong with the row, or the variable type it yields.

Behaviour

- **Opening.** The host, the six link parameters, the two schedules when they are given, and every row's address and type are checked first. A problem stops the module opening, with a message naming it.
- **The link.** The module dials the station, waiting up to `t0`, and sends STARTDT. When the station confirms, the link is up, and a general interrogation of the whole station is sent at once. If the connection fails, or the link ends, the module dials again every 5 seconds, for as long as it is open.
- **A station that stops data transfer** (STOPDT) gets no confirmation: the module closes the connection, reports it, dials again 5 seconds later and sends STARTDT, and the module is back to on once the station confirms. In IEC 60870-5-104 only the controlling station starts and stops data transfer. Before engine 6.18.8 the module confirmed the stop and kept a connection that carried nothing, and after `errorAfterSeconds` it read `Server {server} unreachable` although the station was answering.
- **Values.** A value the station reports is written into the collector with that IOA. Points no collector names are ignored, and so are ASDUs marked as a test. A value of another type family at a collector's address is not written, and the module warns. The station's time tag is not kept: the variable changes when the value arrives.
- **Interrogation.** `interrogationCron` sends the general interrogation again on its schedule. A station that refuses one is reported.
- **Counters are never requested.** The module does not send a counter interrogation. An `M_IT` point is written only when the station sends the counter on its own, spontaneously or periodically; a station that sends counters only when asked leaves those variables unwritten.
- **Clock synchronisation.** `clockSyncCron` offers the station this host's time. It is offered, never imposed, and the reverse never happens: a station's clock does not set this host's. A synchronisation, like an interrogation, is sent only while the link is up.
- **Commands.** Each change of a forwarded variable is sent as a command of the forwarder's type, carrying the variable's value. With **SBO**, the module sends a select, waits for the station to confirm it, and only then sends the execute: a station that never confirms never executes. A negative confirmation is reported.
- **Commands wait for the link.** When the module opens, it sends no command before the link is up. The first time the link starts after the module opens, the current value of each forwarded variable that has one is sent once, after the general interrogation. A link that drops and comes back does not send them again: a command is an action. A change while the link is down is not sent later; the module reports it.

Messages

A problem is written into the module's message variables, and the next ASDU of values the station sends clears it. A warning about one point, a value invalid or of another family, is the exception: it stays while the station keeps reporting that point that way, whatever the other points do, and ends when a good value of that point is written. With several such points, it names the latest one. Before engine 6.18.8 the end of the ASDU that carried it cleared it, so it showed only for a moment. "then `ERROR`" follows `errorAfterSeconds` in Settings every module shares. All texts are in English; the full list is in Module messages. `{type}` is an exact type name; `{cause}` is a cause of transmission, such as `ACTIVATION_CONFIRMATION` or `UNKNOWN_OBJECT_ADDRESS`. `{server}` is the **Host** and **Port**, as `host:port`. `{field}` and `{mapping}` are the variable's code, and in `'{type}' is not a type this module carries`, `{type}` is the type as its row gives it. `{configured}` is the type the point is configured with, and `{quality}` is `invalid` or `not topical`.

When	Level	Text
Opening: no host	`ERROR`	`No host`
Opening: a link parameter breaks its rule	`ERROR`	`k is 1 to 32767, not {k}`, `w is 1 to k ({k}), not {w}`, `t0 is at least one second, not {t0}`, `t2 ({t2}) must be shorter than t1 ({t1})` or `t3 ({t3}) must be longer than t2 ({t2})`
Opening: a schedule does not parse	`ERROR`	`Invalid schedule '{cron}': {reason}`
Opening: an IOA out of range	`ERROR`	`{mapping}: an information object address is 0 to 16777215, not {ioa}`
Opening: a type the module does not carry	`ERROR`	`{mapping}: '{type}' is not a type this module carries`
Opening: a command in a collector	`ERROR`	`{field}: {type} is a command, and a collector reads rather than commands`
Opening: a monitored type in a forwarder	`ERROR`	`{field}: {type} is something a station reports, not a command this module can send`
The link started	`INFO`	`Connected to {server}`
The connection could not be made, or was lost	`WARNING`, then `ERROR`	`Disconnected from {server}`, then `Server {server} unreachable`
The link ended: the station sent STOPDT and the module hung up, the far end closed it, `t1` expired, or the sequence numbers disagreed	`WARNING`, then `ERROR`	`The far end stopped data transfer: {reason}`, then `Server {server} unreachable`
An ASDU could not be read, and was dropped	`WARNING`	`Sequence out of step: {reason}`
The station refused the general interrogation	`WARNING`	`General interrogation did not complete: the station refused it`
A value of another type family at a collector's IOA	`WARNING`	`{ioa} returned {type}, and the point is configured as {configured}`
A value marked invalid or not topical	`WARNING`	`{ioa} reported {quality}`
The station refused a command	`WARNING`	`Command to {ioa} refused: the station returned a negative {cause}`
A variable changed while the link was down	`WARNING`	`Command to {ioa} refused: the link is not up`
A command could not be built from the variable's value	`WARNING`	`Command to {ioa} refused: {reason}`

Example

A pump's run state, an inflow and a volume counter from a station with common address 7, and a pump command sent with select-before-operate:

```
Host                    10.30.0.21
Port                    2404
Common address          7
Interrogation schedule  0 0 * * * ?
Clock sync schedule     0 0 3 * * ?

Collector    P_401_RUN     1001   M_SP
Collector    FIT_401       2001   M_ME_NC
Collector    FQI_401       3001   M_IT
Forwarder    P_401_CMD     5001   C_SC     SBO
```

The station is interrogated when the link starts and again every hour, and is offered this host's clock once a day. `P_401_RUN` and `FIT_401` follow the station from then on. `FQI_401` changes only when the station sends the counter on its own. Writing `P_401_CMD` sends a select to IOA 5001, and the execute follows when the station confirms it.

IEC 60870-5-104 Server

The `IEC104-Server` module listens for a utility's masters, reports the instance's variables to them as points, and accepts their commands into the points marked writable.

Settings

Key	Setting	Type	Default	Meaning
`port`	Port	int	2404	The TCP port. The server listens on every network interface
`commonAddress`	Common address	int	1	This station's common address. A request for another one is answered negatively, with the cause "unknown common address"; the broadcast address 65535 is accepted
`allowedClients`	Allowed masters (comma separated)	list of strings	empty	The IP addresses masters may connect from. **Empty refuses every master**
`k`	k	int	12	See Link parameters
`w`	w	int	8	See Link parameters
`t0`	t0 (s)	int, s	30	Checked, but not used by the server. See Link parameters
`t1`	t1 (s)	int, s	15	See Link parameters
`t2`	t2 (s)	int, s	10	See Link parameters
`t3`	t3 (s)	int, s	20	See Link parameters
`points`	Points	list of rows	empty	The points the server reports and accepts

Points

Key	Column	Type	Default	Meaning
`field`	Variable	variable code	(none)	A monitored point: the variable reported. A command point: the variable a command writes
`ioa`	IOA	int	0	The information object address, 0 to 16777215
`typeId`	Type	string	`M_SP`	The point's type, named as Points and types says
`spontaneous`	Spontaneous	boolean	true in a new row	Report each change without being asked. A point that is not spontaneous is still answered in an interrogation
`deadband`	Deadband	number	(empty)	How far a number must move, from the value last reported spontaneously, to be reported again. Empty or 0: every change
`writable`	Writable	boolean	false	Whether a command may change the variable. A command point must be writable, or the module does not open

Allowed masters

`allowedClients` is **fail-closed**. A master whose address is not in it is disconnected at once, and an empty list refuses everyone: an empty list is far more likely to be an unfinished configuration than an invitation.

- **Exact match only.** A master's IP address, as text such as `10.40.0.5`, must equal one entry. There are no host names, ranges or masks.
- **One link per address.** Several masters can be connected at once. A new connection from an address that already has one replaces it.

What the server answers

- **The link.** The server waits for a master's STARTDT; it never starts data transfer itself.
- **General interrogation** (`C_IC_NA_1`): a confirmation, then every point that is neither a command nor a counter, with cause 20, then a termination. The qualifier is not read: a group interrogation is answered with every point.
- **Counter interrogation** (`C_CI_NA_1`): a confirmation, then every `M_IT` point, with cause 37, then a termination. Counters are sent only in answer to this, and in spontaneous reports.
- **A variable nothing has written yet** is sent marked invalid, rather than as a zero the plant never measured.
- **Time tags.** A point named with a time-tagged type is stamped with the time it is sent.
- **Clock synchronisation** (`C_CS_NA_1`) is confirmed and **not applied**: the plant's clock is not a master's to set. The engine log records the time the master offered.
- **Spontaneous reports.** Each change of a spontaneous point's variable is sent, with cause 3, to every master whose link is started, unless the change is smaller than the point's deadband.
- **Commands.** A command reaches its variable only when a point has that IOA, the point is writable, and the command is of the point's family. The server confirms it, writes the value into the variable, and sends a termination. Anything else is answered negatively and changes nothing: an unknown IOA with the cause "unknown object address", and a point that is not writable, or of another family, with a negative confirmation. A value the variable refuses, such as one that does not fit its declared type, is confirmed and then ends in a negative termination, and the variable keeps what it held. Before engine 6.18.8 a value refused for its type ended in a positive termination, although the variable had not taken it.
- **Select before operate.** A select is confirmed and changes nothing. The server does not require one: an execute alone acts.
- **Anything else** a master sends is answered negatively, with the cause "unknown type".

Messages

All texts are in English; the full list is in Module messages. An executed command, an interrogation answered in full and a confirmed clock synchronisation count as a success, and clear a warning about a refused master or a refused command. Before engine 6.18.8 only an executed command did, so a station whose masters only interrogate kept such a warning until the module reopened. `{field}` and `{mapping}` are the variable's code. A refused command's `{reason}` is `the point is not writable` or `the point is not of that type`.

When	Level	Text
Opening: a link parameter breaks its rule	`ERROR`	As the client says it: `t2 ({t2}) must be shorter than t1 ({t1})`, for example
Opening: an IOA out of range	`ERROR`	`{mapping}: an information object address is 0 to 16777215, not {ioa}`
Opening: a type the module does not carry	`ERROR`	`{mapping}: '{type}' is not a type this module carries`
Opening: a command point that is not writable	`ERROR`	`{field}: a command point that is not writable can never do anything`
Opening: the port cannot be bound	`ERROR`	`Cannot listen on port {port}: {reason}`
The server is listening	`INFO`	`Listening on port {port}`
A master connected	`INFO`	`Client {address} connected`
A master's link ended	`WARNING`	`Client {address} disconnected`, cleared when that master connects again. With several masters gone, it names one still missing
A master not in the allowed list tried to connect	`WARNING`	`Invalid request discarded: {address} is not an allowed client`
An ASDU from a master could not be read	`WARNING`	`Invalid request discarded: {reason}`
A command to a point that is not writable	`WARNING`	`Command to {ioa} refused: {reason}`
A command of another family	`WARNING`	`Command to {ioa} refused: {reason}`
A command's value could not be written, such as one that does not fit the variable's declared type	`WARNING`	`Variable {variable} refused the value: {reason}`

A server that could not open, because another program still holds its port for example, is tried again by the engine.

Example

A pumping station reported to two masters of a utility's SCADA, with a level that is reported only when it moves by 5 cm or more, and a pump command the utility may send:

```
Port              2404
Common address    12
Allowed masters   10.40.0.5, 10.40.0.6

Point   P_401_RUN    1001   M_SP_TB   spontaneous
Point   LIT_401      2001   M_ME_TF   spontaneous   deadband 0.05
Point   FQI_401      3001   M_IT
Point   P_401_CMD    5001   C_SC      writable
```

Each master receives `P_401_RUN` and `LIT_401` as they change, time-tagged, and every point but the counter in a general interrogation. `FQI_401` is sent in answer to a counter interrogation. A single command to IOA 5001 writes `P_401_CMD`, for a formula or another module to act on. A connection from any other address is refused.

The point list as CSV

A utility describes a link as a spreadsheet, often with hundreds of rows. Both editors therefore **import and export the point list as CSV**: the client's **Collector** table, and the server's **Points** table. Both read the same four columns, and each side uses only some of them. The server's **Points** table uses all four:

```
IOA,TYPE,VARIABLE,OPTIONS
1001,M_SP,P_401_RUN,spontaneous
2001,M_ME_NC,FIT_401,"spontaneous,deadband=0.5"
0x1389,C_SC,P_401_CMD,writable
```

The client's **Collector** table uses IOA, TYPE and VARIABLE:

```
IOA,TYPE,VARIABLE
1001,M_SP,P_401_RUN
2001,M_ME_NC,FIT_401
```

- **Columns.** A row holding a cell `IOA` is the header, and the columns are then found by name, in any order. Without a header they are read in the order above. The file is read as UTF-8, up to 4 MB.
- **IOA** is decimal, or hexadecimal written `0x…`.
- **TYPE** is named as Points and types says. The client's collector table takes monitored types only.
- **VARIABLE** is taken as written.
- **OPTIONS** are separated by `,` or `;`. The server uses `spontaneous`, `writable` and `deadband=<number>`; a deadband that is not a number is ignored. The client's collector table does nothing with OPTIONS. `sbo` (or `selectbeforeoperate`) is read, but neither table uses it: select-before-operate belongs to the client's **Forwarder** table, which has no import, so tick **SBO** there.
- **Skipped and counted**: rows whose address cannot be read, rows of a type the table does not take, and addresses already in the table, so importing the same file twice does not double it. The result reads `{n} point(s) imported. Skipped: {n} with an unreadable address, {n} of a type this module does not carry, {n} already in the table.`
- **Export points** shows the table as CSV text in the same place, with its header, ready to copy.

Next steps

- Module catalogue
- Settings every module shares
- Module messages
- Connectivity

This page describes Data Orchester engine 6.18.8.