

SoftWindingLink Manual

Customer orders down to the cutting machines

Product	SoftWindingLink (customer orders to the cutting machines)
Document version	Version 1.1
Date	October 2026

Contents

1. Introduction 4
1.1 What SoftWindingLink is 4
1.2 What it does and what it does not do 4
1.3 Unattended process: what the Link does not check 4
1.4 How it fits 5
1.5 Conventions 6
2. Installation 7
2.1 Requirements 7
2.2 Step 1. Database schema and user 7
2.3 Step 2. Configuration 7
2.4 Step 3. The Windows service 8
2.5 Step 4. License 8
2.6 Upgrading an earlier installation 8
2.7 Security 9
3. The administration window 10
3.1 Opening it 10
3.2 Top bar and engine in the window 10
3.3 Status 11
3.4 Machines 11
3.5 Orders 13
3.6 Clients 14
3.7 Printers 15
3.8 Events 17
3.9 Log 17
3.10 PLC profiles 18
3.11 Settings 19
4. Defining a machine 21
4.1 Cut confirmation 22
4.2 Cut mode R: the PLC counts the cut and stores its length 23
4.3 Omron NX1 with NB screen 24
4.4 Lenze C430, Siemens and other PLCs without FINS (SFW) 24
4.5 First commissioning of a real machine 24
4.6 Optional machine functions 25
5. Zebra labels 26
6. Connecting a customer 27

6.1 Choosing the channel 27
6.2 Quick guide for the developer 27
6.3 Database mailbox 28
6.4 Files: CSV, XML and Excel 31
6.5 REST / JSON 35
6.6 TCP: AHP protocol 37
6.7 MQTT / VDA 5050 37
6.8 Events and cursor 41
6.9 Functions per channel 42
7. How the Link works 43
7.1 The path of an order through the database 43
7.2 Stock control and products (optional) 44
7.3 Claiming a line 46
7.4 Sending to the PLC 46
7.5 Confirming and closing 46
7.6 Machine orders 46
7.7 Recovery 47
7.8 Machine safety 47
8. The "Link live" web report 48
9. Operation and troubleshooting 50
9.1 What to check daily 50
9.2 Common problems 50
9.3 Error codes 51
9.4 Where the logs are 52
9.5 Maintenance 52
10. Simulators 53
10.1 The simulated machine 53
10.2 The simulated customer system 54
Annex A. Configuration (SoftWindingLink.ini) 57
Annex B. Command line 58
Annex C. OMRON_CP and OMRON_NX profiles 59
Annex D. sw_link_* tables 60

1. Introduction

1.1 What SoftWindingLink is

SoftWindingLink is the program that carries the orders of customer systems (an ERP, a WMS, an MES) down to the cutting machines and returns the result, without SoftWinding or SoftWinding Lite running at the station. It talks to the machine PLC, prints the label of every cut on a Zebra printer and exchanges data with the customer system through a database mailbox, files (CSV, XML or Excel), REST/JSON, TCP or MQTT with the VDA 5050 standard (logistics robots, AWS IoT Core).

It consists of three parts that share the same database:

- **SoftWindingLinkSvc.exe**: the Windows service (or a console) that does the work. One thread per machine.
- **SoftWindingLink.exe**: the administration window, where machines, clients and printers are defined and what is happening can be watched (chapter 3).
- The **"Link live"** report of SoftWindingWeb, to watch the status from any browser (chapter 8).

1.2 What it does and what it does not do

1. It reads from the SoftWinding database the pending order lines of every machine it manages.
2. It claims one line (atomically, even with other machines or a classic SoftWinding around) and sends it to the PLC with the same sequence SoftWinding uses.
3. It watches the counter. When the cut is done (by counter, by the HMI screen or by an external confirmation) it queues the label, updates the line and the batch, notifies the customer and, if stock control is on (section 7.2), writes the warehouse movement and deducts the reel.
4. After a crash it recovers what it left half-way, without duplicating movements or labels.

> **Important.** Starting the machine and cutting remain the operator's responsibility. `AutoStart` is off on every machine and the Link never triggers the hydraulic cut. It does not cancel a measurement in progress and it does not modify the SoftWinding schema; it only adds its own `sw_link_*` tables.

1.3 Unattended process: what the Link does not check

SoftWindingLink is an unattended process: it receives the order from another system and carries it to the machine without anyone at a computer confirming it. A SoftWinding station works differently: the operator is there, chooses or scans the reel, sees the product and corrects by hand. That is why the Link does not make the checks SoftWinding makes with a person in front of it:

Table 1. Checks of a SoftWinding station the Link does not have.

Check	On a SoftWinding station	In SoftWindingLink
Mother reel	In the scanner flow, the station asks to scan the source reel and checks that it matches the product and reel of the order.	The Link does not know which reel is loaded on the machine. `ReelID` is part of the order data: it is recorded, printed and returned. With the machine option "Check the reel against the product" it only checks in the database that the reel belongs to that product (error 4005), not that it is the reel on the shaft.
Cable being measured (product)	The operator sees the product and description on screen and decides.	The Link cannot check which cable runs through the machine; the product in the order is taken as correct.
Scanner and containers	A scanner can be used to validate reels, containers and locations.	There is no scanner, and containers, locations and labels are not validated.
Stock	The operator sees the stock of the reel and adjusts it by hand.	With stock control on it deducts what was cut from the reel of the order, but it does not stop for lack of stock (it can go negative; a warning goes to the Log) and does not check before cutting that the reel has enough meters.
Errors and warnings	Windows and messages shown to the operator as they happen.	They are written to the Log and the customer is notified with `line.error`; nobody is alerted on screen, so the Log has to be reviewed.

Responsibility. The system that sends the orders and the operator who loads the machine are responsible for the mother reel and the cable matching the order. If an installation needs these checks (mother reel traceability, scanner validation), they must be done on a SoftWinding station, or in the customer's own system before the order is sent and when the machine is loaded.

1.4 How it fits

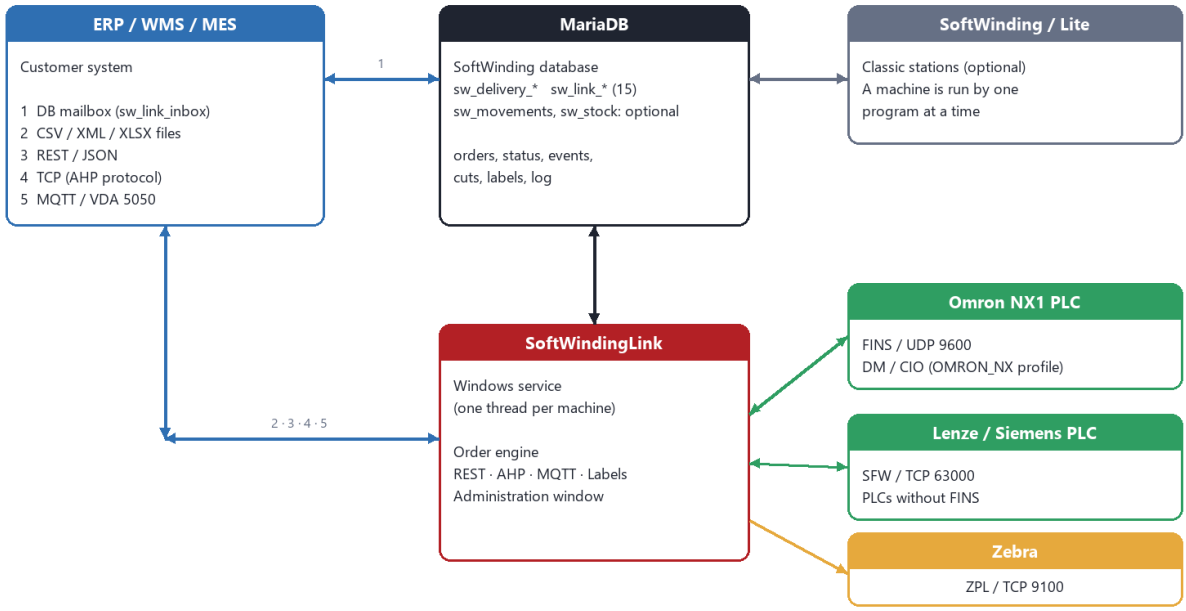


Figure 1. Architecture: the customer, the database, SoftWindingLink, the machines and the printers.

Omron NX1 PLCs are driven through FINS/UDP, which is the current SoftWinding standard. All other PLCs (Lenze C430, Siemens and other brands without FINS) are driven through the SFW protocol over TCP, a

LOIMEX in-house standard. The Omron address profiles (`OMRON_CP` for the CP series PLCs and `OMRON_NX` for the NX1) are stored in the database and edited from the window.

1.5 Conventions

- `Text like this` is a table, field, file, button or command name.
- Lengths in the database are in meters with three decimals; over AHP they travel as whole millimeters.
- A "line" is one position of an order: a product, a length in meters and the number of times a machine must cut it.
- The administration window and the two simulators speak six languages: Spanish (default), English, German, French, Italian and Portuguese (section 3.1). This manual only exists in Spanish and English.

2. Installation

2.1 Requirements

- Windows (the SoftWinding server or any computer with access to the database and to the machines).
- MariaDB with the SoftWinding database (schema 172) and the 64-bit client library `libmariadb.dll`, which ships next to the executables.
- Network: UDP 9600 to the Omron PLCs, TCP 63000 to the SFW PLCs, TCP 9100 to the Zebra printers and, if used, the AHP (3001) and REST (8090) listening ports.

The files are in the `dist` folder: `SoftWindingLinkSvc.exe` (service), `SoftWindingLink.exe` (window), `libmariadb.dll`, the `labels` folder with the label templates and the simulators for testing without a machine (chapter 10). Copy the folder to the server, for example to `C:\SoftWindingLink`.

2.2 Step 1. Database schema and user

Do this once, with a MariaDB administrator account. No SoftWinding table is modified.

1. Open the window (`SoftWindingLink.exe`), **Settings** tab, fill in the server, port and database and press **Install or update the Link schema…** It asks for an administrator account and creates the 16 `sw_link_*` tables, the `OMRON_CP` and `OMRON_NX` profiles and the query views for database clients (`sw_link_v_*`). It is safe to repeat: it only creates what is missing.
2. Alternatively, run `sql\create_sw_link.sql` with the MariaDB client.
3. Create the Link user: edit the password in `sql\create_sw_link_user.sql` and run it. It gives the service only what it needs.
4. For each customer that will use the database mailbox, create its user with `sql\create_sw_link_client_user.sql` (section 6.3): it can only write to the mailbox and read its results.
5. Only if you will turn stock control on (section 7.2): also run `sql\create_sw_link_user_stock.sql`, which gives the Link user permission to create products, reels and stock rows and to deduct stock. With the control off (the default) it is not needed and the Link cannot touch those tables. The **Install or update the Link schema…** button runs it when the control is on in Settings and a user is created.

2.3 Step 2. Configuration

The `SoftWindingLink.ini` file sits next to the executables and is shared by the service and the window. The first time it starts, the service creates a default one. The connection data are entered in the **Settings** tab (section 3.11) and saved with **Save**.

The database password is stored in clear text only the first time: the Link replaces it with a DPAPI-encrypted version (`PassEnc=`), which can only be decrypted on that computer and with that Windows account. Machines, printers and clients are not in the ini: they are in the database and are managed from the window.

Decide whether the Link keeps the stock. By default it only exchanges orders with the machines. If you want SoftWinding to also keep the cable warehouse (movements, reel deduction, product creation), turn on **Stock control** in the **SoftWinding stock and products** group of the **Settings** tab (or `[Stock] Control=1` in the ini) and grant the permissions of step 1. Read section 1.3 (what the Link does not check) and 7.2 first.

2.4 Step 3. The Windows service

In the **Status** tab there are the **Install service**, **Uninstall service**, **Start** and **Stop** buttons. Windows asks for administrator confirmation (UAC). The service is called `SoftWindingLink` and starts automatically with Windows. From the command line:

```
SoftWindingLinkSvc.exe --install
SoftWindingLinkSvc.exe --uninstall
sc start SoftWindingLink
sc stop SoftWindingLink
SoftWindingLinkSvc.exe --console     (console)
```

After installing, check that the Status tab shows "Windows service: Running".

2.5 Step 4. License

SoftWindingLink is a paid product and uses the same license system as SoftWinding. Without a license it starts in DEMO mode: it works the same, without any restriction, but it warns in the window title, in the "License" row of Status, in the log and in the `licensed` field of `/api/v1/health`.

1. Open **Settings > License** (or run `SoftWindingLinkSvc.exe --license`): it shows the machine ID, the volume serial number of drive C: in four groups of digits (for example `225.239.852.0`). The **Copy** button puts it on the clipboard.
2. Send LOIMEX the ID, the company name as it should appear and the purchase order number.
3. LOIMEX returns an unlock code (four numbers separated by hyphens). Type the company name in **Licensed to**, the code in **Unlock code** and press **Activate**.
4. It is saved in `SoftWindingLink.ini`, section `[License]` (`Licensed` and `UnLock`), shared by the service and the window. The service reads it at start: restart it so that it stops warning.

> **Note.** The code is valid for that computer only: if the system disk changes, the ID changes and a new code must be requested. LOIMEX can also issue a company code, valid on any computer for that name. Each product has its own key: the SoftWindingLink key does not unlock SoftWinding or SoftWindingWeb, nor the other way round.

2.6 Upgrading an earlier installation

To move from an earlier version (schema 3, 4 or 5) to the current one (schema 6):

1. Stop the service and the window, and replace the executables and the `sql` folder.
2. With a MariaDB administrator account, run `sql\create_sw_link.sql`, `sql\create_sw_link_views.sql` and `sql\update_sw_link_permisos.sql` in this order. All three can be repeated safely. The first one creates what is missing (the `sw_link_vda` and `sw_link_waste` tables, the `LiveInfo` and `CableMm` columns, the new profile rows) and sets the schema version to 6; the second creates the four views; the third gives the Link user permission on the new tables. The **Install or update the Link schema…** button of the window does the first two.
3. Start the service. The Link keeps working with schemas 3 to 6, but with one older than 6 some functions are missing: the MQTT channel (schema 4), wastes (schema 5), and the live data and per-line cable diameter (schema 6). The log warns about what is missing.
4. Review the profile of each Omron machine (next notice) and, if there are customer database users created with the old script, remove their access to the internal tables: `REVOKE SELECT ON softwinding.sw_link_machines FROM 'acme'@'%';` and the same for `sw_link_orders` (section 6.3).

> **Note.** Profiles `OMRON_CP` and `OMRON_NX`. The profile that used to be called `OMRON_NX` was actually the map of the CP series PLCs: the script renames it `OMRON_CP` and creates `OMRON_NX` from it, with the block of order texts and panel (DM 2400 to DM 2639) shifted to DM 6000 (annex C). Machines that already used `OMRON_NX` keep `OMRON_NX`. If one of them is a CP, or if its PLC does not expose the DM 6000 area onwards over FINS (when an order is sent the PLC answers with FINS code 1103 and the machine goes to the "PLC refused the line" error), change its profile to `OMRON_CP` in the machine form.

2.7 Security

- The Link database user has minimal permissions; do not use `root`.
- REST tokens are not stored: only their SHA-256 fingerprint. They are shown once when the client is created and copied to the clipboard.
- Each client has a list of allowed IPs and allowed machines. After 8 authentication failures from one IP it is blocked for 5 minutes; limit of 600 requests per minute and client.
- The REST and AHP ports are off by default. Enable them only if clients use them, and restrict access with the firewall.

3. The administration window

3.1 Opening it

Run `SoftWindingLink.exe`. It reads the same `SoftWindingLink.ini` as the service and works directly on the database: the service does not need to be running to define machines, clients or printers. It accepts these options:

Table 2. Window options.

Option	Effect
`--ini=path`	Uses another configuration file.
`--lang=es|en|de|fr|it|pt`	Window language: Spanish (default), English, German, French, Italian or Portuguese. It can also be chosen in Settings > General and is saved in `[Window] Lang` of the ini.
`--tab=N`	Opens tab N (0 = Status … 8 = Settings).
`--cfgtab=N`	With `--tab=8`, opens sub-tab N of Settings (0 General … 4 Stock).
`--autostart`	Starts the engine inside the window on opening.
`--width=N --height=N`	Window size.

3.2 Top bar and engine in the window

The top bar shows whether the engine runs in this window and which database it is connected to. The **Start the engine here** button starts the same engine as the service inside the window, useful for testing or for a commissioning with the technician in front of the machine. Do not use it at the same time as the service on the same machines: each machine is controlled by a single instance (the one that writes `Owner` and its heartbeat); the second one waits for the ownership time to expire.

The **blue header** of the window gathers the essentials: the title, the state of the engine in this window (green dot if it runs here, grey if not) with its start/stop button, the database state (green, or red with the full error in the help balloon), the license (green if licensed, amber in DEMO mode) and the **Help** button, which opens a quick guide. At the bottom, the **help bar** explains the button or field under the mouse (and the same explanation appears as a balloon). Every button and tab has an icon. The language is chosen in Settings > General (or with `--lang`): Spanish, English, German, French, Italian and Portuguese.

The tables are colored to be read at a glance: **machines** by state (free, prepared, measuring, waiting for confirmation, error, offline), **lines** (amber on the machine, green done, red with error, grey pending), **events** (green `line.done`, red `line.error`, orange `line.canceled`, yellow `line.waste`) and the **log** (yellow warnings, red errors).

3.3 Status

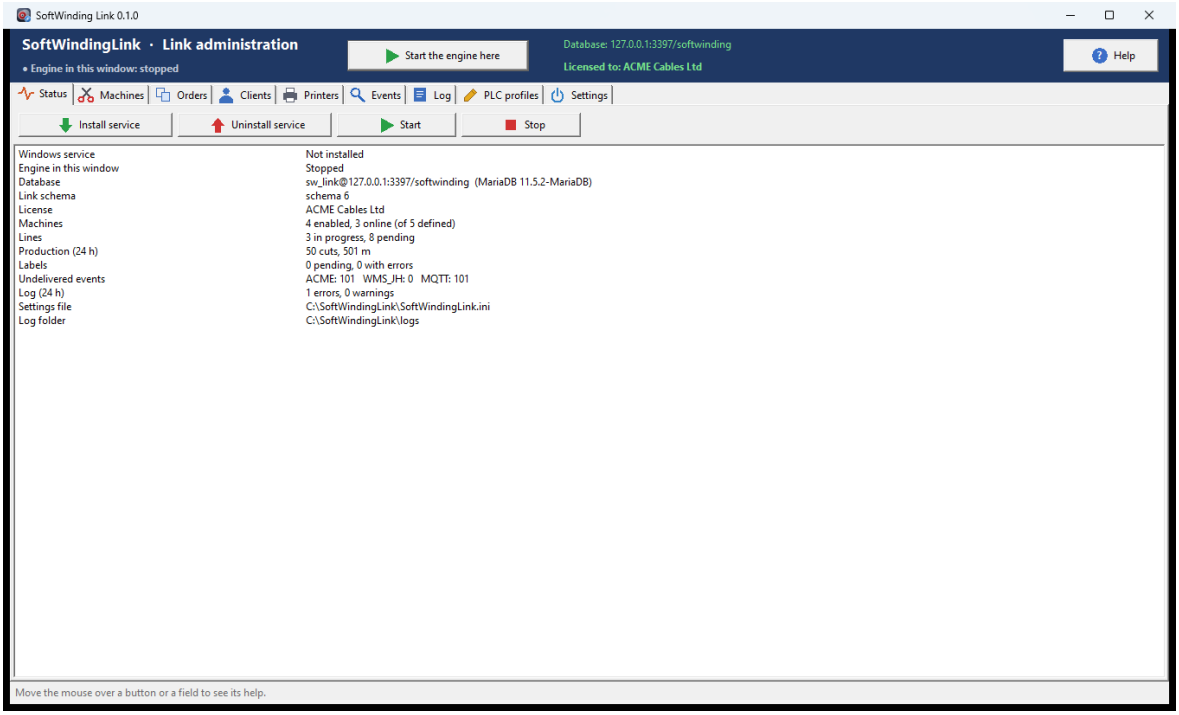


Figure 2. Status tab: service, database, machines, production, labels and warnings.

It summarizes on one screen whether everything is fine: status of the Windows service, of the window engine and of the database (with the Link schema version), enabled and connected machines, lines in progress and pending, production of the last 24 hours, pending or failed labels, undelivered events per client, and errors and warnings from the log. The service buttons (section 2) are here too.

3.4 Machines

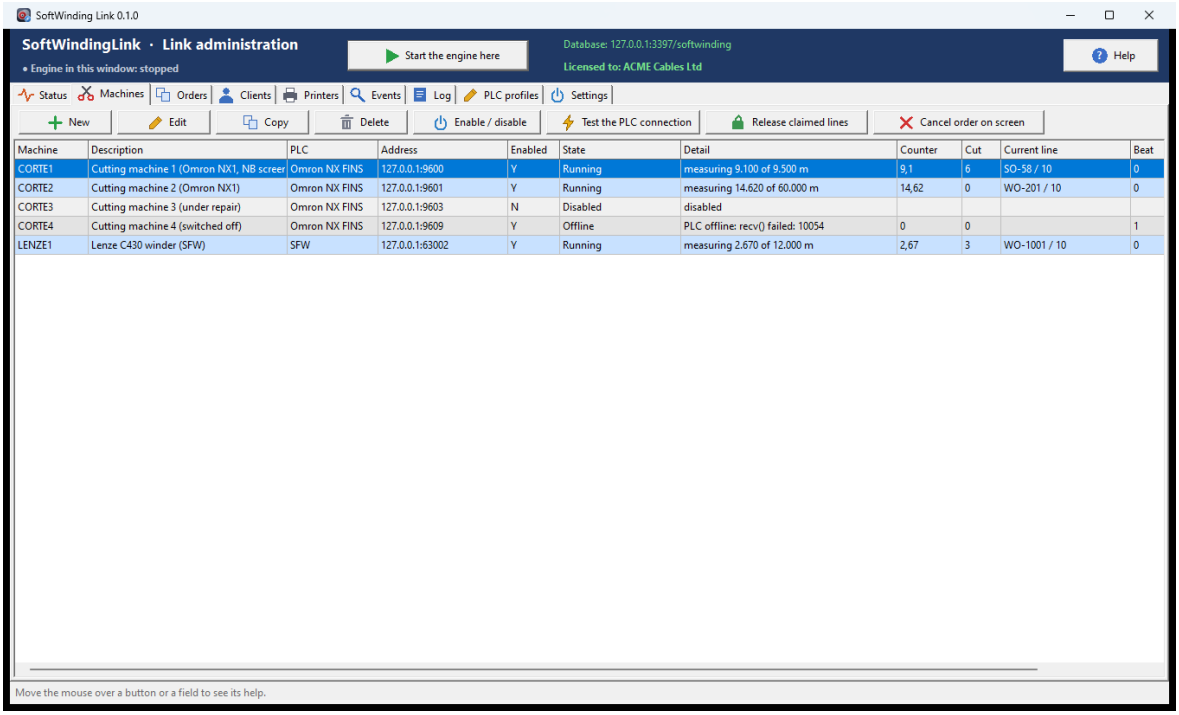


Figure 3. Machines tab. Rows are colored by state.

One row per machine with its PLC, address, live state, detail, counter, cut number and the line in progress. It refreshes by itself every 2 seconds. Buttons:

Table 3. Buttons of the Machines tab.

Button	What it does
New / Edit	Opens the machine form (below).
Copy	Creates a new machine with the same data and leaves it disabled, so you only change the IP.
Delete	Deletes the machine. Refused if it has claimed lines.
Enable / disable	Sets `Enabled` to Y or N. The service applies it in under 10 s.
Test the PLC connection	Connects to the PLC, reads version, counter and run state and shows the result. It writes nothing to the PLC.
Release claimed lines	Returns to pending the lines claimed by a machine whose service is no longer running (the heartbeat has expired). Refused if the Link is controlling it right now.
Cancel order on screen	Cancels the order the selected machine has on its screen, even if the line has already measured or cut, and leaves the PLC clean (preselection 0, cuts and texts blank). It asks for confirmation. Refused if the machine is running (error 4014: stop it first). **It needs the engine running** (the service or **Start the engine here**): the order is served by the machine thread, which answers within a few seconds. It is the same function as `DELETE /machines/{id}/order` over REST (section 6.5).

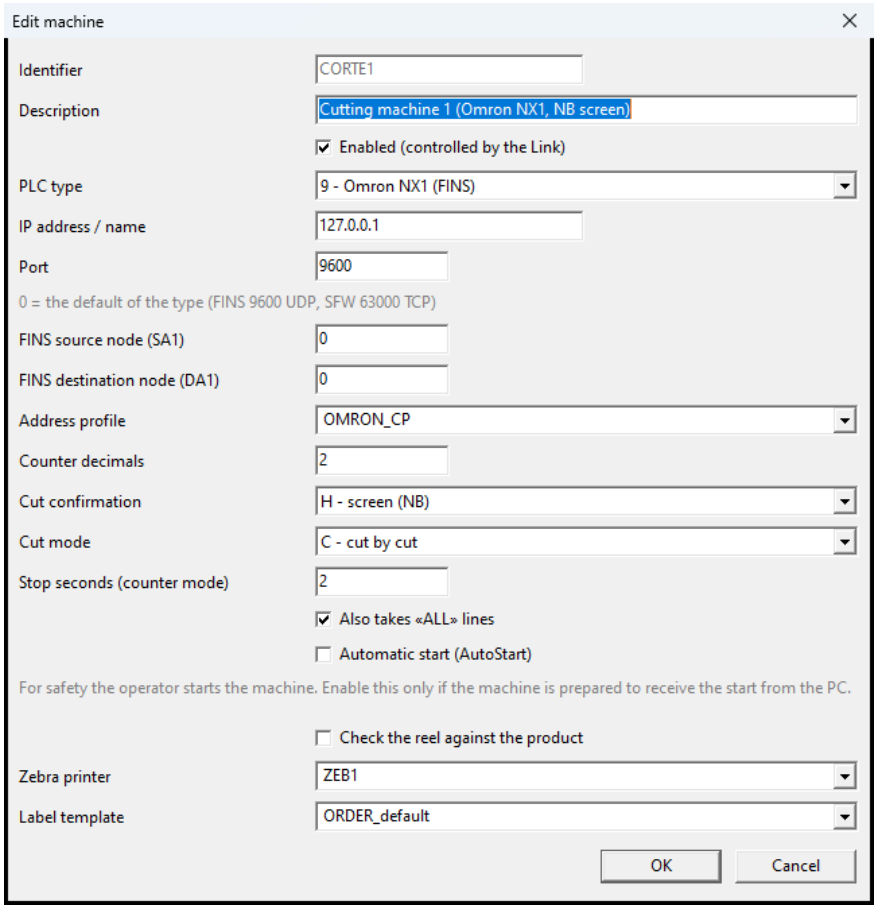


Figure 4. Machine form.

The form fields are explained in section 4.

3.5 Orders

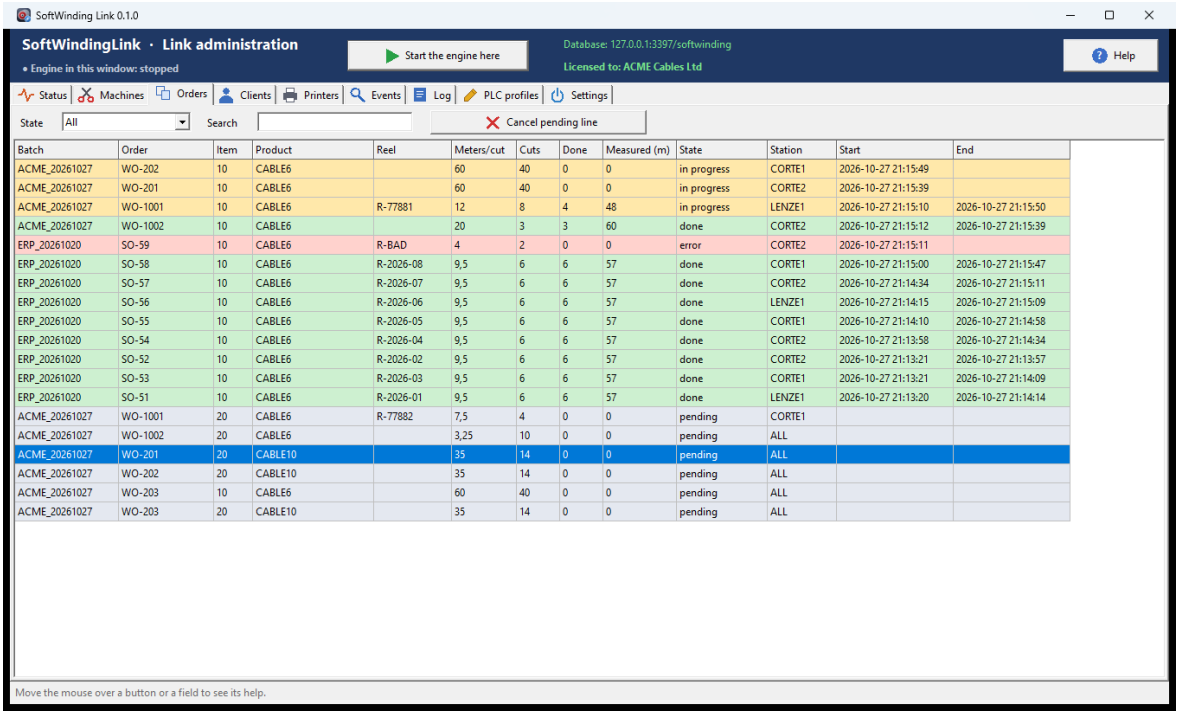


Figure 5. Orders tab: all SoftWinding lines, wherever they come from.

Lists the `sw_delivery_items` lines with their batch, order, product, reel, meters per cut, cuts, cuts done, measured, status, machine and dates. Filter by status and search by text. **Cancel pending line** deletes a line that has not started yet (and notifies the client with `line.canceled`); a line in progress or done cannot be canceled.

> **Note.** Canceling a line makes way for the next one. If the machine has **Also takes "ALL" lines** ticked, it takes any pending line of the database with `StationID = ALL` (for example, the orders imported by SoftWinding), not only the ones you send. When you cancel the line you have just sent, the Link offers the machine the next one in the queue, and so on. For a machine to work only with what arrives from a customer, untick that box and send each line with its `machineId` (chapter 4).

3.6 Clients

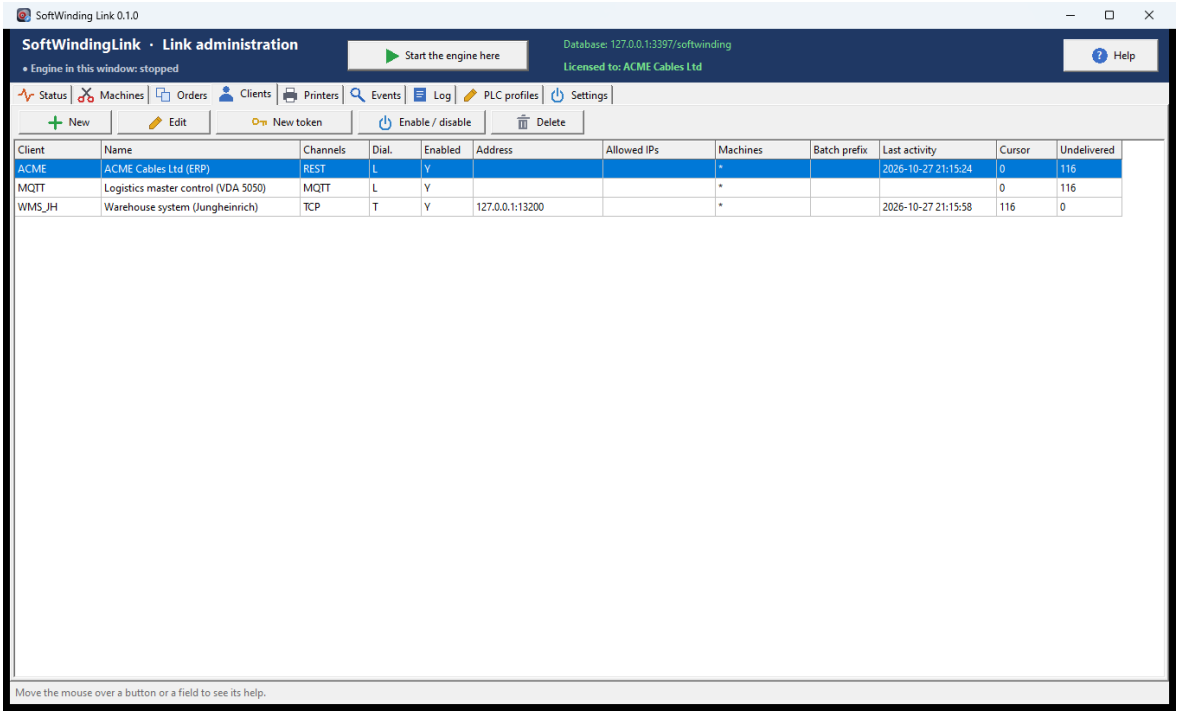


Figure 6. Clients tab.

A client is an external system that sends orders through the database mailbox or files (`DB` channel), over REST, TCP or MQTT. Channels are entered as a comma-separated list, for example `DB,REST`; the `DB` channel needs no token. For each one the window shows its name, channels, dialect, whether it is active, its TCP address, allowed IPs and machines, batch prefix, last activity and its event cursor (the last delivered event).

- **New**: asks for the data (next figure). For the REST channel it generates a random token and shows it once; it is copied to the clipboard. A new client does not receive the event history: its cursor starts at the last one.
- **New token**: invalidates the previous one and generates another (REST).
- **Enable / disable** and **Delete**.

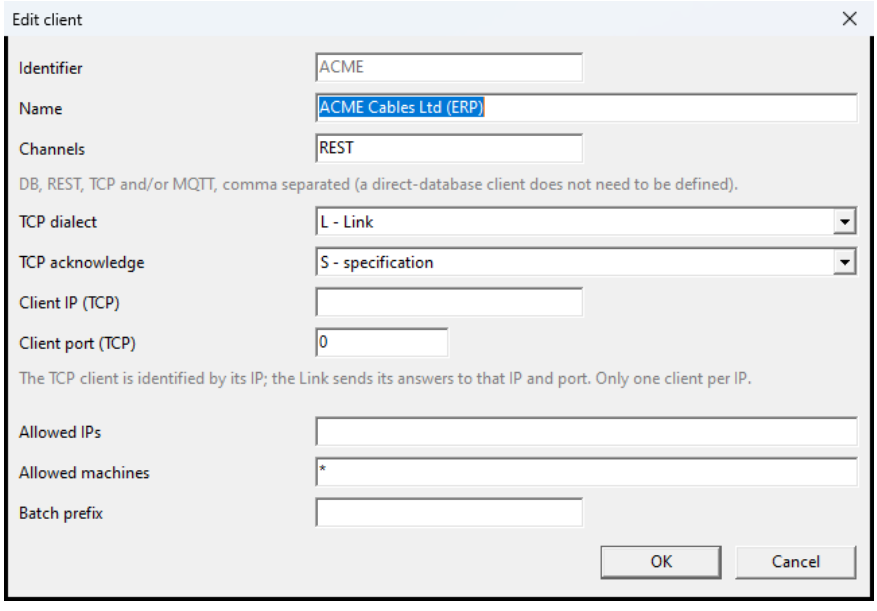


Figure 7. Client form.

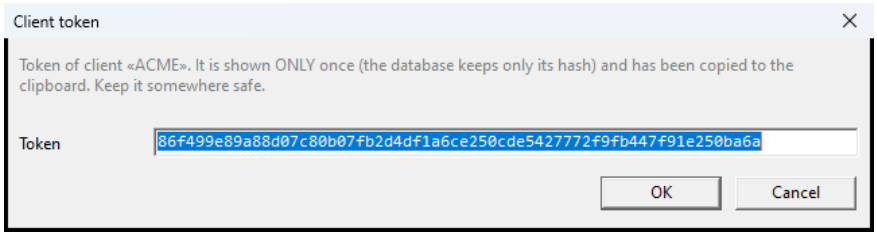


Figure 8. The token is shown only once.

3.7 Printers

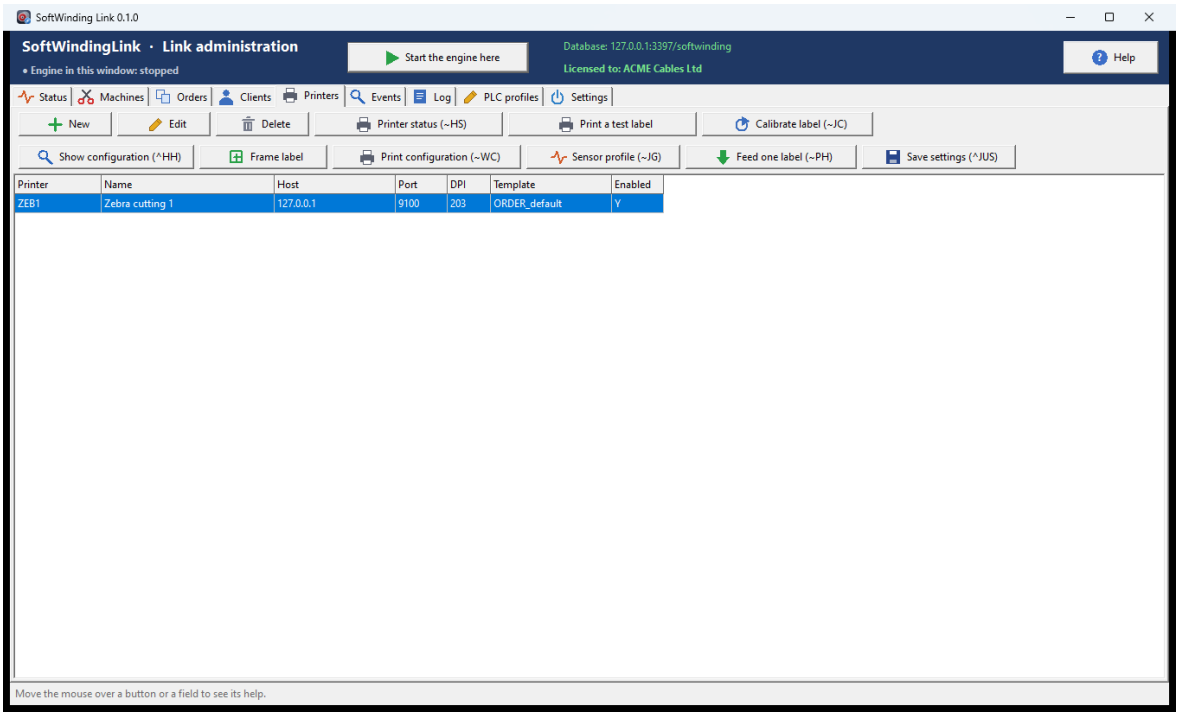


Figure 9. Printers tab.

The Zebra printers, over TCP/IP (port 9100). **Printer status (~HS)** asks the Zebra whether it has paper and whether it is paused; **Print test label** sends a label with the printer name and the time (asks for confirmation).

The tab is also used to **calibrate and test** the printer, all through ZPL to the selected printer. The commands that print or move paper use the same sending as the labels: they ask `~HS` first and **are not sent if the printer is out of paper or paused**.

Table 4. Calibrating and testing the printer.

Button	What it does
Calibrate label (~JC)	Asks for confirmation (labels loaded, cover closed), reads the configuration (`^HH`), sends `~JC`, waits up to 40 s for the printer to answer again and shows the label length before and after (in dots and mm) and the print width.
Show configuration (^HH)	Shows what the printer returns: width, label length, darkness, speed, media type and sensor.
Frame label	Prints a label with a frame on the edges, a central cross and marks every 100 dots. The frame must be fully visible: if it is cut or a blank strip remains, calibrate.
Print configuration (~WC)	The printer prints its configuration label.
Sensor profile (~JG)	Prints the sensor graph, to see whether it tells the label from the backing.
Feed one label (~PH)	Feeds one blank label, to see where the cut falls.
Save settings (^JUS)	Asks for confirmation and saves the current settings, calibration included, in the printer memory.

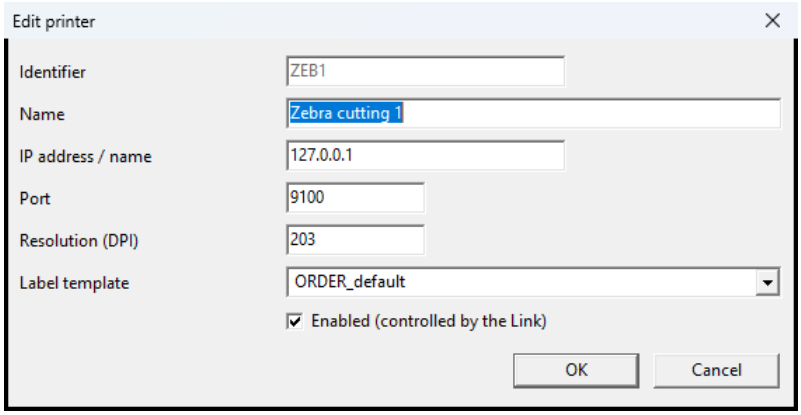


Figure 10. Printer form.

3.8 Events

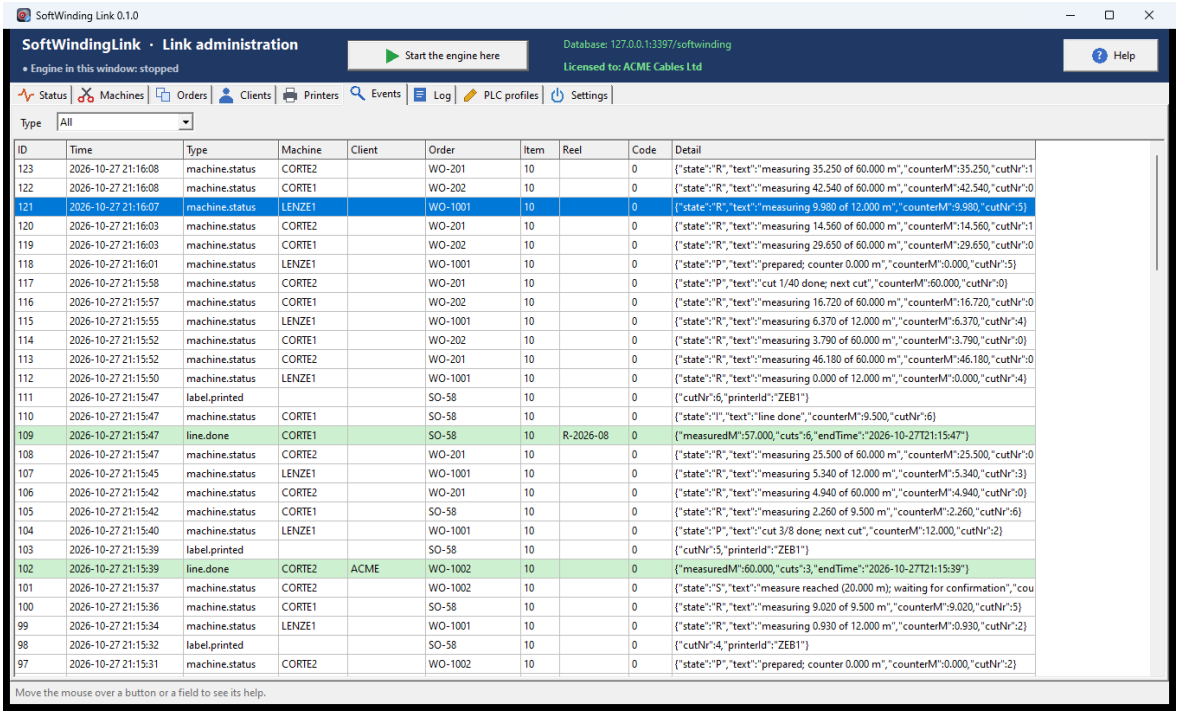


Figure 11. Events tab: the outgoing queue to the clients.

Shows the `sw_link_events` queue: `line.done`, `line.error`, `line.canceled`, `line.waste`, `machine.status` and `label.printed`, with the machine, the client that owns the order, the order, the position, the reel, the error code and the content (JSON). Filter by type. This is what clients will see over REST (`GET /events`) and TCP.

3.9 Log

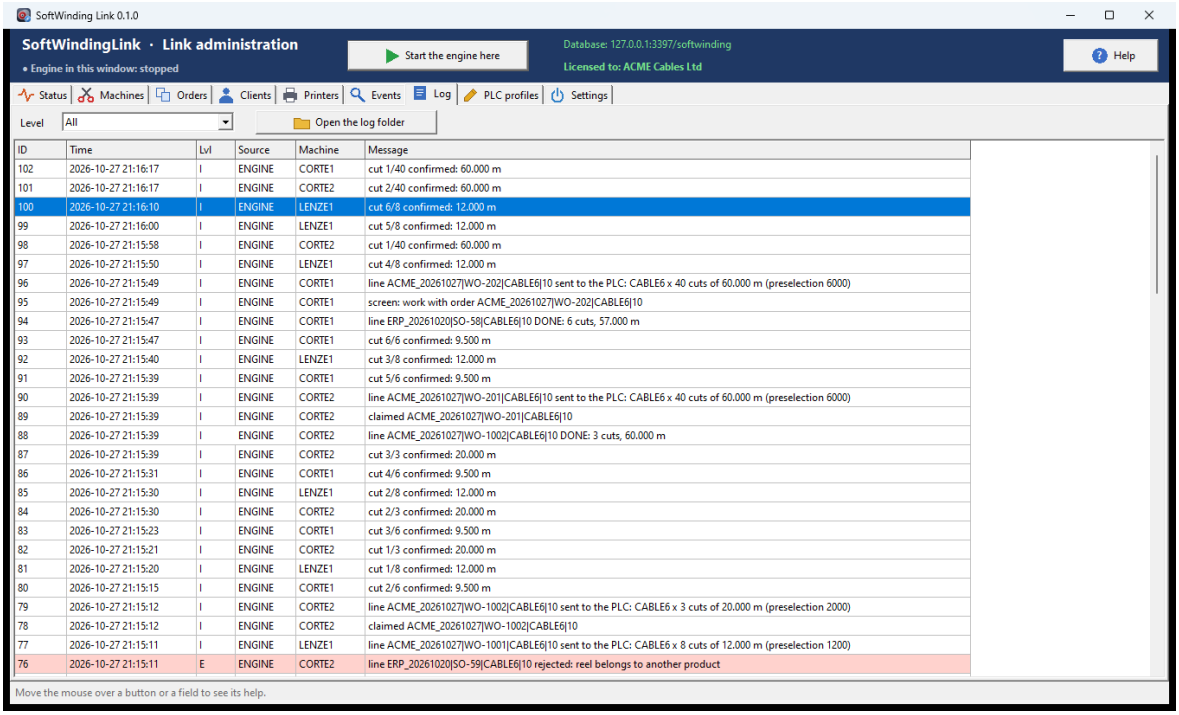


Figure 12. Log tab.

Shows `sw_link_log`: every order sent to the PLC, every claimed line, every confirmed cut, PLC connections and disconnections, telegrams, errors and actions made from this window (source `ADMIN`). Filter by level (`D` debug, `I` information, `W` warning, `E` error). **Open log folder** opens the `logs` folder, which also holds a daily `link_YYYYMMDD.log` file.

3.10 PLC profiles

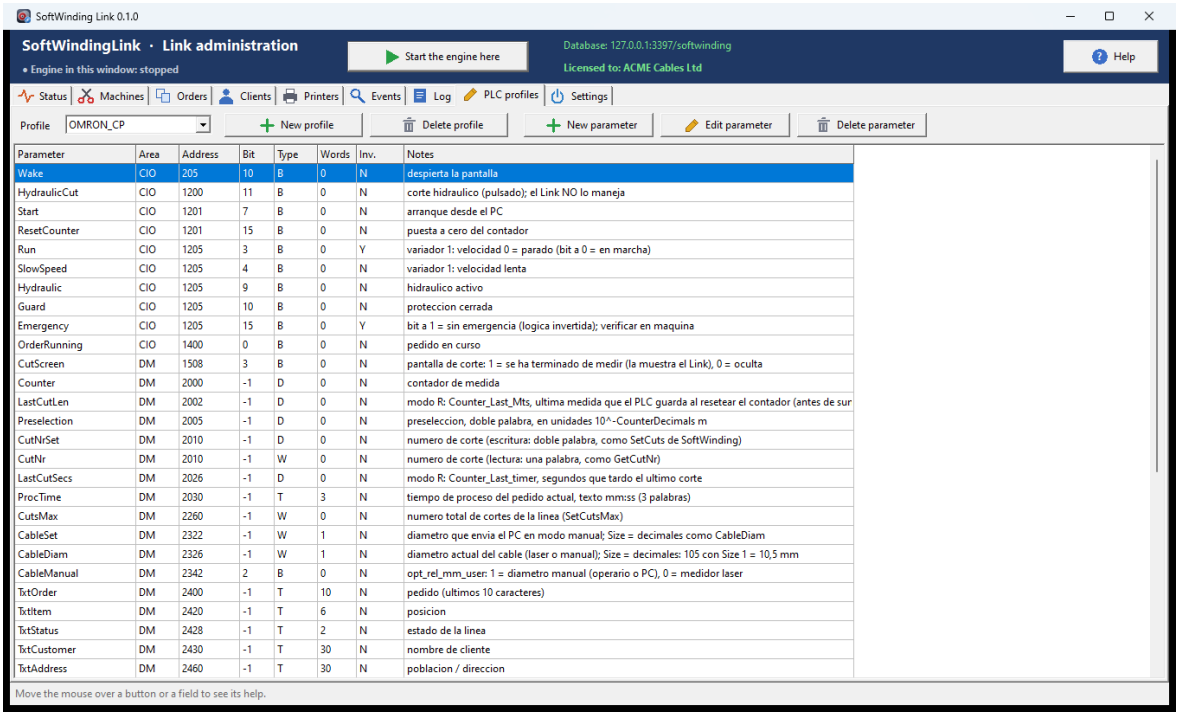


Figure 13. PLC profiles tab: the OMRON_NX memory map.

A profile is the memory map of a type of Omron PLC: for every parameter, the area (`CIO` or `DM`), the address, the bit, the data type (`W` word, `D` double word, `B` bit, `T` text), the size in words of text fields and whether the logic is inverted. The `OMRON_NX` profile ships preloaded with the addresses SoftWinding uses; if the PLC program changes an address, correct it here without recompiling anything. **New profile** copies the current one to create another (for example, for another PLC family); a profile in use by a machine cannot be deleted.

There are two Omron profiles: `OMRON_CP`, for CP series PLCs (the block of order texts and panel is at DM 2400 to DM 2639), and `OMRON_NX`, for the NX1, which is the same except that this block starts at DM 6000 (annex C). The profile is chosen in the **Address profile** field of the machine form. A new row in `OMRON_CP` reaches `OMRON_NX` by itself the next time `create_sw_link.sql` is run.

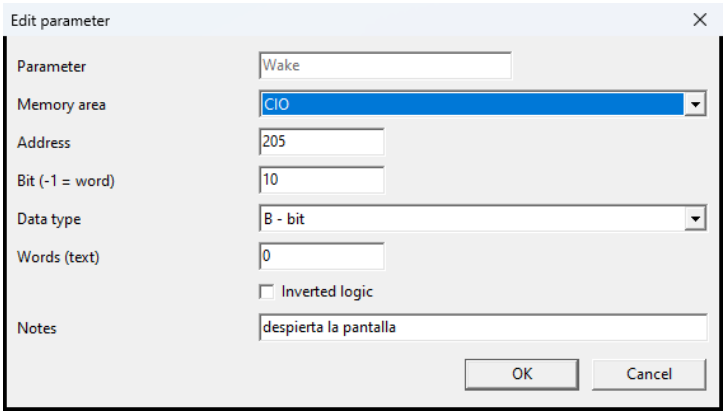


Figure 14. Editing a parameter.

The complete profile is in annex C. SFW PLCs (type 8) do not use a profile: their functions are fixed by the protocol.

3.11 Settings

The tab has five sub-tabs: **General** (database, engine and others, folders and license), **TCP (AHP)**, **REST / JSON**, **MQTT / VDA 5050** and **Stock**. A fixed bar below, with **Save**, **Open the manual** and the result message, is visible in all of them. A sub-tab can be opened directly with `--tab=8 --cfgtab=N`.

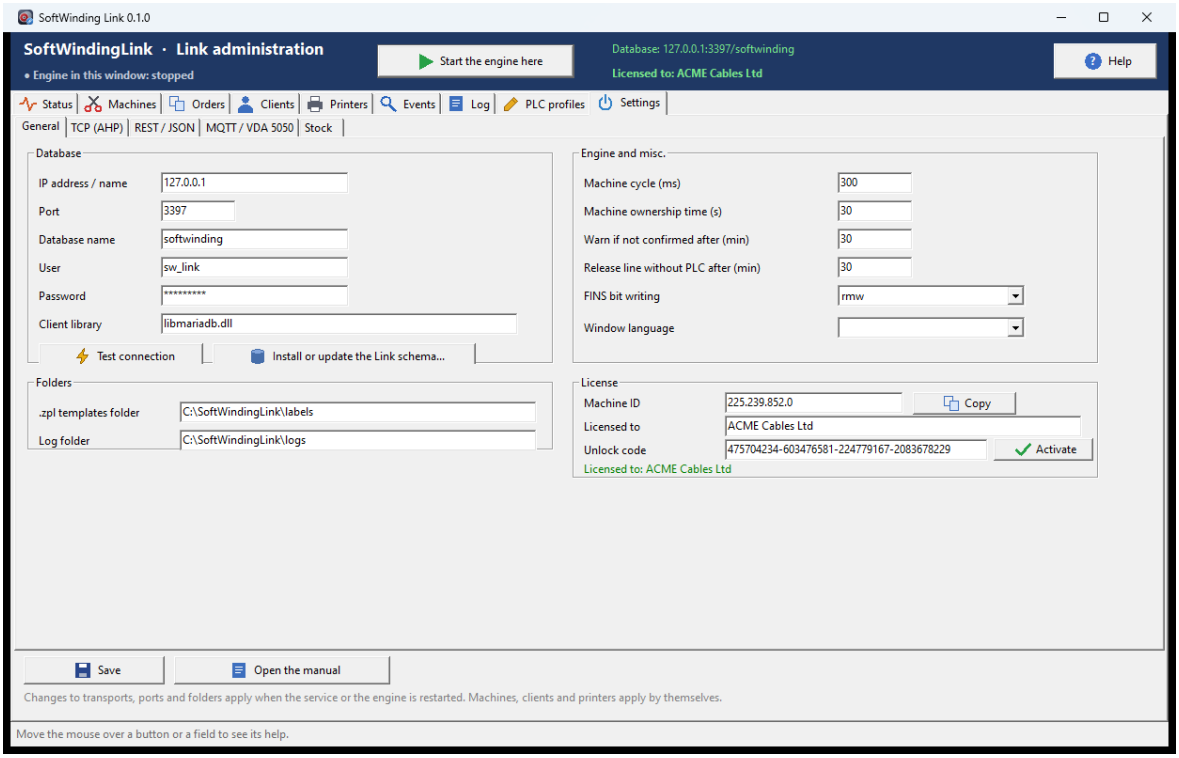


Figure 15. Settings, General sub-tab.

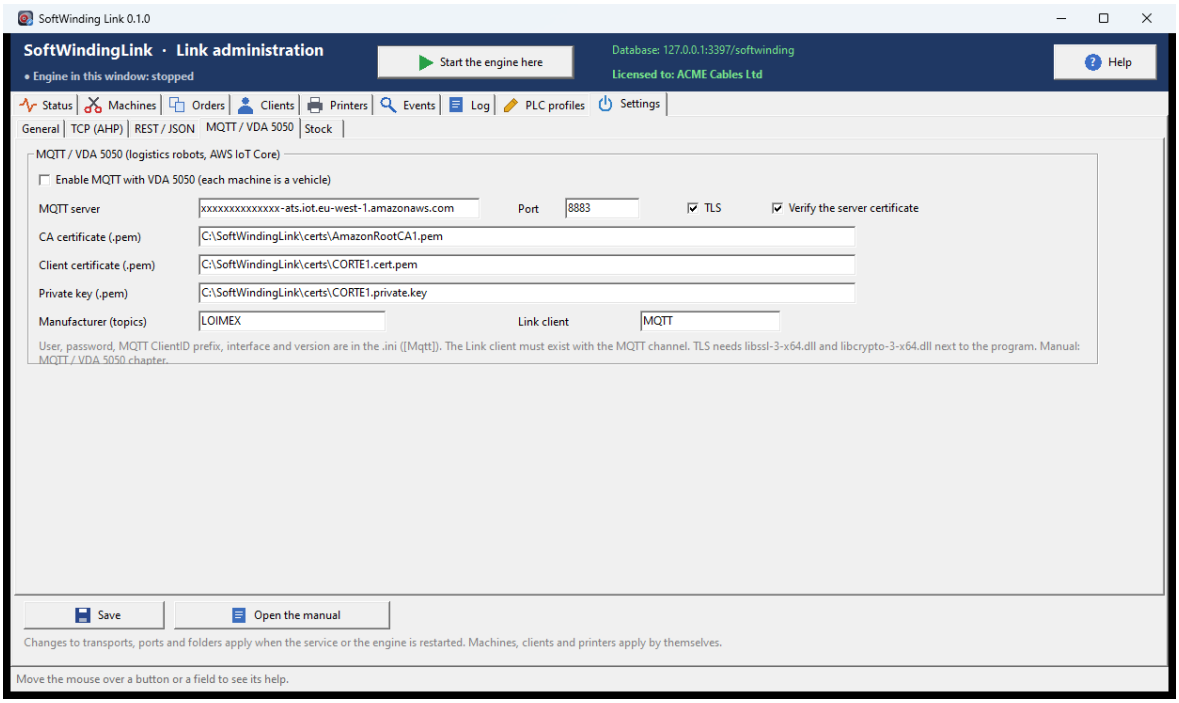


Figure 16. Settings, MQTT / VDA 5050 sub-tab.

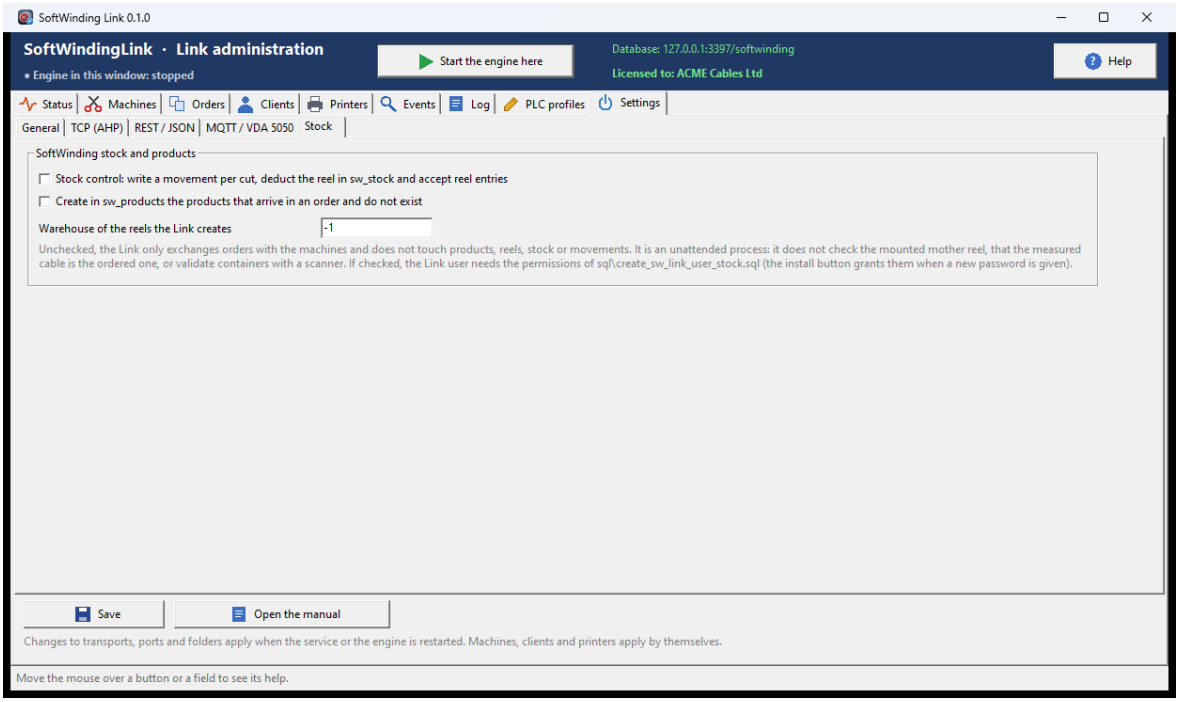


Figure 17. Settings, Stock sub-tab.

Table 5. Groups of the Settings tab.

Group	What is configured
Database	Server, port, database, user, password and client library. **Test connection** checks the data before saving; **Install or update the Link schema…** creates the tables (section 2).
Transports for clients	Enable TCP (AHP) and/or REST/JSON, with their listening IP and port. Nothing needs enabling for direct database access.
MQTT / VDA 5050	Enable the channel, server, port, TLS, certificate files, manufacturer and Link client (section 6.7). User, password and other settings are in the ini.
Engine and others	Cycle of every machine (300 ms), machine ownership time (30 s), warning without confirmation (30 min), release line without PLC (30 min), FINS bit writing (`rmw`: read-modify-write the word, like SoftWinding; `bit`: bit command) and window language (Spanish, English, German, French, Italian or Portuguese).
Folders	`.zpl` label templates and logs.
License	Machine ID (with **Copy**), **Licensed to**, **Unlock code** and the **Activate** button (section 2, step 4). At the top right, the window shows "Licensed to: …" or "DEMO: program without license".
SoftWinding stock and products	**Stock control** (movements, reel deduction and reel entries), **create missing products** and warehouse of new reels. Unchecked, the Link only exchanges orders (section 7.2).

Save writes the ini. Changes of transports, ports, folders and stock control apply when the service or the engine restarts; machines, clients and printers apply by themselves. **Open the manual** opens this document (`docs\Manual_SoftWindingLink_es.pdf` or `_en.pdf`).

4. Defining a machine

In the **Machines** tab, press **New**. A machine from SoftWinding Lite or classic becomes a "Link machine" when it is defined here with `Enabled` ticked; until then the Link does not touch it.

Table 6. Fields of the machine form.

Field	Meaning
Identifier	The `StationID` written into the lines (`sw_delivery_items`). Max 30 ASCII characters, no spaces. It cannot be changed afterwards.
Enabled	The Link controls the machine. Unticked, it neither claims lines nor touches it.
PLC type	`9` Omron NX1 (FINS), `6` Omron FINS (classic CJ/CP), `8` SFW (Lenze, Siemens… PLCs without FINS).
IP address / name, Port	Where the PLC is. Port `0` = the one of the type (FINS 9600 UDP, SFW 63000 TCP).
FINS source / destination node	SA1 and DA1 of the FINS frame (Omron only). Check the PLC network configuration.
Address profile	The Omron memory map: `OMRON_CP` (CP series PLC) or `OMRON_NX` (NX1); the difference is the DM area of the order texts (annex C). Empty for SFW.
Counter decimals	The PLC counter is in units of 10^-n m. With 2, the counter counts hundredths of a meter.
Cut confirmation	How the cut is known to be done (next table).
Cut mode	`C` cut by cut: every cut is confirmed and the next one prepared. `L` the PLC counts the cuts of the line: it ends when `CutNr` = cuts of the line. `R` the PLC adds the cut when the counter is reset and stores its length: every cut is recorded with the real measure (next section).
Stop seconds	With confirmation by counter: seconds the machine must stay stopped for the cut to count as done.
Also takes "ALL" lines	Besides lines with its identifier, it takes those with `StationID = ALL`, which are almost all the ones SoftWinding imports on its own. For the machine to work only with what a customer sends it with its `machineId`, untick it.
Automatic start (AutoStart)	Whether the Link sends the run command after preparing the line. Off for safety; enable it only if the machine is prepared to receive the start from the PC and the guards are correct.
Check the reel against the product	If the line carries a `ReelID` and the reel exists in `sw_reels` with another product, it is rejected (error 4005). If the reel does not exist, it is accepted.
Zebra printer and Label template	Where and how the label of each cut is printed (chapter 5). Empty = no printing.

4.1 Cut confirmation

Table 7. Confirmation modes (ConfirmMode).

Mode	The cut counts as done when…	Use it with…
C	the counter reaches the preselection and the machine has been stopped for `StopSeconds` seconds.	PLCs without HMI; SFW (mandatory).
H	screen mode: the Link does not activate orders by itself; the NB screen chooses the order (previous, next, work with it) and sends the "finish" command (`HmiCommand` = 100), like the "next" button of SoftWinding. Zeroing the counter is the operator's decision after cutting (below).	Omron with NB screen (recommended).
X	the customer sends an external confirmation (`POST …/confirm` over REST).	When the customer system is in charge.
P	the PLC's own "done" flag. Reserved: it currently behaves like `C`.	—

If the confirmation does not arrive within 30 minutes (configurable) the line does not close by itself: the machine goes to "stopped waiting for confirmation", `line.error` 4009 is emitted and the window shows a warning.

4.1.1 Screen mode (H): choosing the order from the machine

With `ConfirmMode = H` the Link **does not activate orders by itself**: the operator chooses the order on the machine screen, as SoftWinding does with screen navigation. It is the default mode of the machines created from the window. The operator drives the screen with the `HmiCommand` order (`SW_HMI_Action`, DM 2699): the Link reads it, **clears it to 0** and acts.

Table 8. Screen commands (DM 2699).

HmiCommand	Command	What the Link does
`10`	Previous order	Shows the previous line of the batch (stops at the first).
`20`	Next order	Shows the next line of the batch (stops at the last).
`90`	Work with the order	Claims the line shown and prepares it: preselection, cuts and order in progress (CIO 1400.0 = 1). From then on the machine works on that order and 10, 20 and 90 are ignored.
`100`	Finish	Counts the cut as done (confirmation of mode H).

- **What is shown:** the pending lines of the machine (`StationID` = the machine, or `ALL` if it also takes those) of the **first batch** by priority, the one automatic mode would take, ordered by priority, position, order and position (up to 200). For the chosen line the Link writes the texts and the panel (order, position, status, customer, product, description, "to cut" and "already measured") and wakes the screen, but **without preselection** and without an order in progress. With no pending lines the screen is left blank.
- **On finishing** (100 or the closing mode), **canceling** (`DELETE /machines/{id}/order`) or **releasing** the line, the screen goes back to showing the first of the pending ones.
- **When the engine starts** nothing is activated. A line claimed in the previous session that the PLC no longer has running goes back to the queue; if the PLC does have it running (the Link crashed in the middle of the job) it is resumed without sending it again.
- Modes `C`, `X` and `P` (automatic order, no navigation) do not change.

4.2 Cut mode R: the PLC counts the cut and stores its length

Some Omron machines count the cuts in the PLC program itself: when the counter is reset and the counter is greater than or equal to the preselection, the PLC adds 1 to the cut number (`CutNr`). Since the reset clears the counter, the value the cut was made with would be lost. Mode `R` solves it: the PLC copies the counter into a register (`LastCutLen`) just before adding the cut, and the Link reads that register and records every cut with its real measure, as SoftWinding does.

Logic the PLC program must have when the counter is reset (the order matters):

```
IF  counter reset:
    IF  Preselection > 0  AND  Counter >= Preselection  THEN
        LastCutLen  := Counter;          // 1st the length
        LastCutSecs := Seconds;          //     and how long it took (optional)
        CutNr       := CutNr + 1;        // 2nd the cut
    END IF
    Counter := 0;
END IF
```

- `LastCutLen` (`Counter_Last_Mts` in the PLC program) is a double word at `DM 2002`, in the same units as the counter (with 2 decimals, 57 m = 5700). Optionally `LastCutSecs` (`Counter_Last_timer`, `DM 2026`, double word) stores the seconds the cut took; the Link only notes it in the Log ("cut 2/3 counted by the PLC: 5.070 m in 4 s"). If you use other addresses, change them in the **PLC profiles** tab.
- The Link reads `CutNr` first and `LastCutLen` afterwards; that is why the PLC must write the length before adding the cut. When `CutNr` goes up, the Link records the cut (`sw_link_cuts`), the movement, the label and the line progress with that length.
- The Link neither zeroes the counter nor writes `CutNr` while the line is in progress: the PLC does. The cut confirmation (`C`, `H`, `X`) is not used in this mode.
- If the Link misses a cut (for example after a network outage) and `CutNr` goes up by more than one, the missing cuts are recorded with the preselection length and a warning is left in the Log; the last one always carries the real measure.
- The line ends when `CutNr` reaches the cuts of the line. If the profile has no `LastCutLen` or the PLC is of type SFW, the machine goes to error and no line is sent to it.
- The PLC may be much faster than the Link polling: the Link **does not need to see the counter at the preselection**, only `CutNr` going up. The PLC stores only the last measure: if the Link sees more than one new cut between two reads, only the last one carries its real measure (above). With real cuts, several seconds long, it should not happen.

4.2.1 Wastes: operator reset before the preselection

If the operator resets the counter before reaching the preselection (for example, 18 m measured out of 20 m), it is not a valid cut: the Link records it as an **invalid measure (waste)** and it is the customer system (the ERP) that decides what to do. Detection needs a flag from the PLC, `ShortReset`. The logic the PLC must have on the reset (the order matters):

```
IF  counter reset:
    IF  Preselection > 0  AND  Counter >= Preselection  THEN          // valid cut
        LastCutLen := Counter;  LastCutSecs := Seconds;
        CutNr      := CutNr + 1;
    ELSE IF  Preselection > 0  AND  Counter > 0  THEN                 // waste
        LastCutLen := Counter;  LastCutSecs := Seconds;               // 1st the measure
        ShortReset := TRUE;                                           // 2nd the flag
    END IF
    Counter := 0;
END IF
```

- `ShortReset` is a profile bit (type `B`, area `CIO` or `DM` with bit) **with no default row on purpose**: reading a wrong bit of a real PLC would give false wastes. It is added in the **PLC profiles** tab when the PLC has it; without that parameter the function is off and everything works as before.
- The `Counter > 0` and `Preselection > 0` conditions are essential: the Link also zeroes the counter (when preparing a line and when closing it in modes `C`, `H` and `X`) and then the flag must not be raised. The measure and the seconds are stored **before** raising the flag.
- The Link records the waste in `sw_link_waste`, emits the `line.waste` event (`{"valid":false,"reason":"reset","waste":1,"cutNr":2,"measuredM":18.000,"presetM":20.000,"shortM":2.000,"seconds":14}`), leaves a warning in the Log and **lowers the flag** (it also lowers it when preparing each line).
- The waste is **not a cut**: it does not enter `sw_link_cuts`, nor movements and stock (even if the cable was physically used: the ERP resolves it), nor labels, and it adds neither to `Qty_Measured` nor to `Qty_Cutted`. In mode `L` it is subtracted from what has accumulated.
- It reaches the customer over REST (`GET /events`) and through the DB mailbox (`sw_link_events` table and `sw_link_v_waste` view), not over AHP or MQTT / VDA 5050. If the Link is down when it happens, the flag stays raised and the waste is recorded on return; a waste with the machine having no claimed line is not picked up by the Link.

4.3 Omron NX1 with NB screen

The NB screen is an Omron HMI programmed with NB Designer, the same tool used on the Lenze C430. Typical machine configuration:

```
PLC type        9 - Omron NX1 (FINS)
Address         192.168.1.50      Port 9600 (or 0)
Profile         OMRON_NX  (OMRON_CP for a CP)
Confirmation    H - screen (NB)
Cut mode        C - cut by cut
AutoStart       off
```

4.4 Lenze C430, Siemens and other PLCs without FINS (SFW)

Type `8`. No profile and no FINS nodes; confirmation is by counter (`C`). The Link waits 200 ms between writes because the PLC loses those arriving less than ~150 ms apart. One connection only: the Link and SoftWinding cannot be connected to the same PLC at once.

4.5 First commissioning of a real machine

1. With the operator present and AutoStart off, define the machine disabled and use **Test the PLC connection**: it must read the version, the counter and the run state.
2. On an Omron, compare the addresses of the PLC program with the machine profile, `OMRON_CP` or `OMRON_NX` (annex C) and correct them in the **PLC profiles** tab if they differ.
3. Create one test line with little length (for example 1 m and 2 cuts) and enable the machine.
4. Check on the PLC and on the screen that preselection, cuts and texts arrive. Start by hand, cut and confirm.
5. Check the recorded cut (`sw_link_cuts`), the label, the line status and the `line.done` event; with stock control on, also the movement (`sw_movements`, type `O`, `UserID = LINK`) and the deduction in `sw_stock`.
6. Test the emergency stop: while active nothing must be claimed or sent.

4.6 Optional machine functions

Besides the order cycle, the Omron profile carries optional parameters: if the PLC has them, the Link offers the functions; if they are missing from the profile, the function is simply not available (error 501 or 4015 when requested).

Table 9. Optional functions and their profile parameters.

Function	Profile parameters	What it does
Cut screen	`CutScreen (DM 1508.03)`	The Link sets the bit to 1 when the counter reaches the preselection (modes `C`, `R` and `L`), and the cut screen appears on the machine; it goes back to 0 when the counter drops, on confirming, finishing, canceling or releasing the line. With confirmation by counter it is shown only `StopSeconds` seconds; with external or screen confirmation, until confirmed. The customer can show or hide it over REST (`POST …/cut-screen`).
Cable diameter	`CableDiam (DM 2326), CableSet (DM 2322), CableManual (DM 2342.02)`	`CableDiam` is the current diameter (from the laser gauge or manual); the `CableManual` bit says which (1 = manual, 0 = laser); `CableSet` is where the PC writes the diameter in manual mode. The `Size` field of `CableDiam` and `CableSet` is the number of **decimals**: with `Size = 1`, the PLC value 105 is 10.5 mm. It is read and sent over REST (`/machines/{id}/cable`) or MQTT, and can travel with the line (`cableDiameterMm`): the Link applies it before preparing the PLC.
Process time and speed	`ProcTime (DM 2030, T 3), Speed (DM 5491)`	`ProcTime` is the winding time as `mm:ss` text (3 words); `Speed` is the working speed in m/min (`Size` = decimals, 0 = integers). They are read over REST (`GET /machines/{id}/process`) and appear in the MQTT / VDA 5050 `state`.

4.6.1 The "SoftWinding connected" flag and the order in progress

`HmiActive` (DM 2698, "SoftWinding order connect" in the PLC) tells the PLC whether an order program is connected. The Link handles it as SoftWinding does:

Table 10. HmiActive values.

Value	Meaning	When
`0`	`DISCONNECTED`	The Link is closed (or does not own the machine).
`1`	`CONNECTED`	Connected with no order data on the screen. It is the value when the engine starts and when a line finishes, is canceled or released.
`2`	`TRANSFERRING`	There are orders on the screen: the lines being shown (screen mode), the order being sent and also while the order in progress is being worked.

`OrderRunning` (CIO 1400.0) is the order in progress: it goes to 1 when the line is activated (the machine is locked to work on that order) and to 0 on finishing, canceling or releasing. When a line is **canceled, released or aborted** the Link leaves the PLC clean (stop, preselection 0, `OrderRunning` 0, cuts and texts blank, panel to zero, `HmiActive` 1) **without touching the counter**; when a line finishes, the last order is left on the screen.

When the Link closes (closing the window, stopping the engine or the service): with the **machine stopped**, the order leaves the screen (`OrderRunning` 0, `HmiActive` 0, preselection, cuts and texts blank) and the line goes back to the queue keeping what was already cut; with the **machine running** the order is not touched (the machine finishes the job): only `HmiActive` drops to 0, `OrderRunning` stays at 1 and the line stays claimed to be resumed on start (a warning is left in the log). If the Link dies without saying goodbye it cannot clean anything: on start it resumes the line without resending it if the PLC still has that order, and cleans an order in progress that has no claimed line.

5. Zebra labels

For every confirmed cut the Link puts one row in the `sw_link_labels` queue. A print thread sends it to the machine printer over TCP/IP (port 9100) in ZPL, marks the row as printed and emits `label.printed`.

- Templates: `.zpl` files in the `labels` folder (one is provided, `ORDER_default.zpl`, 100 x 50 mm at 203 dpi). Data is inserted with `{{FIELD}}` marks in upper case.
- Available marks: every field of `sw_delivery` and `sw_delivery_items` by name (the header wins), plus `{{MEASURE}}`, `{{LAST_MEASURE}}`, `{{QTY_CUTTED}}`, `{{CUTNR}}`, `{{REELID}}`, `{{BARCODE}}`, `{{USUARIO}}` (`LINK`), `{{FECHA}}` and `{{ESTACION}}`.
- Before sending it asks the Zebra `~HS`: if it has no paper or is paused, it records the reason and retries every 30 s up to 5 times; then the label is left in error (visible in Status and in the web report) and can be reprinted by setting its status to `N`.
- A service restart never duplicates a label (primary key per cut) nor leaves one unprinted (pending ones are resumed).
- If the template is missing, a simple built-in one is used.

> **Note.** `ReelID` is optional and does not depend on stock: it is the identifier of the reel being cut. If the order carries it, it is printed on the label, recorded in the movement (if stock control is on) and returned in `line.done`. With stock control on and the reel in `sw_stock`, it is the reel that gets deducted (section 7.2).

6. Connecting a customer

6.1 Choosing the channel

The customer system never needs to touch the SoftWinding order tables. It sends its orders through one of these five channels; the Link validates them with the same rules, translates them into SoftWinding lines and returns the result through the same channel.

Table 11. Communication channels with the customer.

Channel	When	Registration	Detail
DB mailbox	The customer system already writes (or can write) to MySQL/MariaDB.	Yes: `DB` channel.	Section 6.3.
Files	The customer system exports a CSV, XML or Excel (.xlsx) file to a folder.	Yes: an `Import` with its `client` in the mapping file.	Section 6.4.
REST / JSON	Modern systems (ERP, MES, scripts). The easiest to integrate.	Yes: `REST` channel, with token.	Section 6.5 and `docs\openapi.json` .
TCP (AHP)	Jungheinrich systems or others that already use that framing protocol.	Yes: `TCP` channel, client IP.	Section 6.6.
MQTT / VDA 5050	Logistics robots and cloud platforms (AWS IoT Core): each machine appears as a vehicle.	Yes: `MQTT` channel and the `[Mqtt]` section of the ini.	Section 6.7.

The five channels share the same message contract (`order.create`, `order.cancel`, `line.accepted`, `line.done`, `line.canceled`, `line.error`, `line.waste`, `machine.status`, `label.printed`) and the same rules: registrations are idempotent by `(client, order, position)`: sending the same thing again duplicates nothing.

6.2 Quick guide for the developer

Steps to start sending orders to the LOIMEX machines.

1. Ask for registration. The LOIMEX administrator registers you as a client (**Clients** tab of the window): an identifier such as `ACME`, the channels you will use and the machines you may send to. You receive the machine identifiers (`CORTE1`, …) and, for REST, a token.
2. Choose the channel from the table above. If you can write SQL, the mailbox is the most direct; if your system already generates files, use the file channel; REST for web systems.
3. Send an order (examples below). A line is a product, the meters of each cut and how many cuts; the Link offers it to the machine named in `machineId` (or to the first free one if empty or `ALL`).
4. Read the answer through the same channel: accepted or rejected with its code (4001–4013) and then the `line.done` (cut done, with the measured meters) or `line.error` notices.
5. Try the integration first against a simulated machine (chapter 10); the Link behaves the same as with a real one.

Table 12. Fields of an order line (the same in every channel; in REST the names are the same in JSON).

Line field	Required	Meaning and limits
`orderId`	yes	Order number: 1 to 30 printable characters (no quotes, backslash or backtick).
`item`	yes (optional in files)	Position within the order, 1 to 99999. The same `orderId` and `item` identify the line.
`machineId`	no	Link machine; empty or `ALL` = any free machine that accepts `ALL`.
`productId`	yes	Product, up to 50 characters.
`lengthM`	yes	Meters of each cut, greater than 0 and up to 99999.999.
`cuts`	yes (default 1 in files and mailbox)	Number of cuts of that length, 1 to 99999.
`reelId`	no	Reel being cut (up to 20). Optional and independent of stock; if present it goes to the label and the movement. With stock control on and the reel in `sw_stock`, it is the one deducted (section 7.2).
`priority`	no	0 by default; higher number first.
`customer, description, notes`	no	Free texts: customer (50), description (100) and notes (500). They appear on the label and on the machine screen.
`batchId`	no	Batch (30). If missing, `<client>_<yyyymmdd>`.
`cableDiameterMm`	no	Cable diameter, 0 to 1000 mm (REST and MQTT / VDA 5050 only). The Link sends it to the machine when preparing the line and leaves it in manual mode (section 4.6). Out of range: 400.
`free1 … free10`	no	Ten free fields of 100 characters, available on the label.

> **Important.** Do not write to `sw_delivery_*` or `sw_movements`, and do not modify the SoftWinding master data (products, reels, stock, users): to receive a reel use the `S` action (section 6.3). Do not delete rows of `sw_link_*` either. The Link performs these operations for you, with checks that direct access skips: allowed client and machine, idempotency, machine ownership and a per-order answer.

6.3 Database mailbox

The customer writes its messages to the `sw_link_inbox` table (one row per order line or per request) and only reads the result in that same row and in `sw_link_events`. It does not need to know what the SoftWinding order tables look like or use `LOCK TABLES`. The Link checks the mailbox every second (`[Inbox] PollMs`), so the answer usually arrives within one or two seconds.

Table 13. Columns of sw_link_inbox.

Column	Written by	Meaning
`ClientID`	client	Client identifier, as registered (enabled and with the `DB` channel).
`Action`	client	`C` create or change a line (default), `X` cancel (with `Item` 0, the whole order), `K` confirm a cut, `S` receive or adjust a reel (stock control, below).
`Status`	both	`N` new (default), `I` draft, `W` processing, `D` done, `E` rejected.
`OrderID, Item, BatchID, Customer`	client	Order data (previous table).
`MachineID, ProductID, ProviderID, ReelID, LengthM, Cuts, Priority`	client	Line data. `ProviderID` is only used by the `S` action.
`Description, Notes, Free1 … Free10`	client	Free texts.
`Created`	client (optional)	Time of the request; if missing, the Link sets it.
`Code, Result`	Link	`Code` 0 = accepted; otherwise the error code (4001…4013). `Result` is the text: `accepted`, `already received`, `canceled`, the rejection reason…
`Assigned`	Link	Machine the line was offered to.
`Instance, Processed`	Link	Which Link processed it and when.

```
-- A multi-line order: ONE single statement (accepted or rejected as a whole)
INSERT INTO sw_link_inbox (ClientID, OrderID, Item, MachineID, ProductID, ReelID, LengthM, Cuts,
Description)
VALUES ('ACME', 'SO-100235', 10, 'CORTE1', 'CABLE6',  'R-77881', 47.2, 5, 'Cable 6 mm'),
       ('ACME', 'SO-100235', 20, 'CORTE1', 'CABLE10', '',        12.0, 2, 'Cable 10 mm');

-- The Link answer, in the same rows
SELECT Item, Status, Code, Result, Assigned FROM sw_link_inbox
 WHERE ClientID = 'ACME' AND OrderID = 'SO-100235' ORDER BY Item;

-- Cancel the whole order
INSERT INTO sw_link_inbox (ClientID, Action, OrderID, Item) VALUES ('ACME', 'X', 'SO-100235', 0);

-- The notices (cut done, error...) and the cursor
SELECT EventID, Type, MachineID, DeliveryID AS OrderID, Item, Code, Payload FROM sw_link_events
 WHERE ClientID = 'ACME' AND EventID > 1234 ORDER BY EventID;
UPDATE sw_link_clients SET LastEventID = 1260 WHERE ClientID = 'ACME';
```

More examples (draft, confirming a cut) in `docs\examples\inbox_examples.sql`.

6.3.1 Mailbox rules

- It is processed per order. The Link groups the new rows with the same `(ClientID, OrderID, Action)` and validates them together: if one line is not valid, none of that group is written and all stay in `E`. For a multi-line order to be all-or-nothing, write it in a single `INSERT` statement, or as a draft: rows with `Status = 'I'` (the Link ignores them) and, at the end, `UPDATE … SET Status = 'N'`.
- It is idempotent. Sending the same line again answers `already received` (`D`) and duplicates nothing. Sending it with different data modifies it if it is still pending (`N`/`I`); if a machine already has it, it answers 4003.
- Cancel (`Action = 'X'`): a pending line is deleted (`line.canceled` notice); one a machine already has is only canceled if it has not measured yet; a finished one answers 4011.

- **Cancel what is on the screen** without knowing the order number: a row `Action = 'X'` with an empty `OrderID` and the `MachineID` of the machine. It removes the order on the screen and cancels it even if it has already measured or cut, and leaves the PLC clean. It is rejected with code 4014 if the machine is running, and with 4002 if it is not online. It needs the engine running.
- Confirm (`Action = 'K'`) only applies to machines with external confirmation (`ConfirmMode = X`) and to a line waiting for confirmation.
- If the client does not exist, is disabled or has no `DB` channel, all its rows end in `E` with code 4012.
- If a Link crashes with rows in `W`, after 60 s another one recovers them (they are idempotent). With several Links on the same database, each order is processed by only one.
- Rows in `D` and `E` are deleted by themselves after 30 days (`EventRetentionDays`). The tables are latin1; accents are stored correctly even if the client connection is UTF-8.

6.3.2 Receiving reels (action S)

With stock control on (section 7.2), the customer receives a reel, or corrects its meters, with a row of action `S`. It is the same operation as a warehouse import in SoftWinding: it creates the product, the reel and its stock row if they are missing, and sets the stock of the reel to the meters given.

Table 14. sw_link_inbox columns used by the S action.

Column	Use in action S
`OrderID`	Delivery note or receipt reference: it groups the rows and has no other effect.
`ProductID`	Product of the reel (required). If it does not exist in `sw_products` with that provider, it is created.
`ProviderID`	Provider (optional, up to 50). It is part of the SoftWinding stock key.
`ReelID`	Reel identifier (required, up to 20). If it already exists in `sw_reels` with another product, it is rejected with 4005.
`LengthM`	Meters the reel has now: absolute value, from 0 to 99999.999. Sending the same message again changes nothing.
`Description, Notes`	Product description (if created) and reel notes.

```
-- Receipt of two reels (the second one already has 12.5 m less: it stays at 87.5)
INSERT INTO sw_link_inbox (ClientID, Action, OrderID, Item, ProductID, ProviderID, ReelID,
LengthM, Description)
VALUES ('ACME', 'S', 'ALB-2026-77', 1, 'CABLE6', 'PV1', 'R-90001', 100,  'Cable 6 mm'),
       ('ACME', 'S', 'ALB-2026-77', 2, 'CABLE6', 'PV1', 'R-90002', 87.5, 'Cable 6 mm');
```

- The Link writes the reel to `sw_reels` (type `N6`, status `A`) and its row to `sw_stock`, in the `[Stock] Warehouse` warehouse and unlocated (`Line`, `Colum`, `Level` and `Position` set to `-1`; placing it on a rack is done from SoftWinding). If the meters change, it writes an `I` (up) or `O` (down) movement with the difference and `UserID = LINK`.
- Each row is handled separately: a wrong one does not stop the others (unlike orders, which are accepted or rejected whole).
- With `[Stock] Control=0` every `S` action is rejected with error 4013. By file, an empty meters cell is rejected so that a blank cannot empty a reel.
- By file it is declared with `<Column field="action" value="S"/>` plus `providerId`, `reelId` and `lengthM`: example in `docs\examples\LinkFiles.xml` (`almacen` `Import`) and `docs\examples\stock_in.csv`.

6.3.3 Reading the status: the sw_link_v_* views

To see the status the customer does not need to look at the SoftWinding tables: it reads these four views, created by `sql\create_sw_link_views.sql` (the **Install or update the Link schema…** button runs it after the

schema). They run with the permissions of whoever created them (`SQL SECURITY DEFINER`), so the customer user only needs `SELECT` on them. **Always filter by your** `ClientID`: there is no per-client isolation at database level (one database user per trusted client).

Table 15. Query views.

View	What it holds
`sw_link_v_orders`	The status of each line, like REST `GET /orders/{id}`: `ClientID, OrderID, Item, BatchID, ProductID, Status` (`pending`, `claimed`, `done`, `error`, `canceled`), `MachineID, ReelID, LengthM, Cuts, CutsDone, MeasuredM, StartTime, EndTime, ErrorCode`. A line canceled while pending is deleted and no longer appears: the cancellation is seen in the mailbox answer (`Status D`, `Result canceled`) and in the `line.canceled` notice.
`sw_link_v_cuts`	One record per cut: `ClientID, OrderID, Item, CutNr, MachineID, ReelID, MeasuredM, Created`.
`sw_link_v_waste`	The wastes: `ClientID, OrderID, Item, Seq, CutNr, MachineID, ReelID, MeasuredM, PresetM, ShortM, Seconds, Created`.
`sw_link_v_machines`	The state of each machine: `MachineID, Description, State, StateText, Online, Cutting, CounterM, CutNr, OrderID, Item, OrderClientID, Updated`. `State` is `offline` (the Link is not talking to the PLC), `idle` (free), `ready` (line sent, waiting for the operator), `running` (measuring), `waiting` (cut done, waiting for confirmation), `error` or `disabled`. `Online` = `Y` if the Link updated it in the last 30 s and it is not `offline` or `disabled`; `Cutting` = `Y` if `State = running`; `CounterM` is the counter in meters (refreshed every second while measuring). `OrderID` and `Item` are the line the machine has now and `OrderClientID` the client of that line (the machine is shared). It does not expose host, port or profile.

```
-- Status of my lines
SELECT Item, Status, CutsDone, Cuts, MeasuredM FROM sw_link_v_orders
 WHERE ClientID = 'ACME' AND OrderID = 'SO-100235' ORDER BY Item;

-- Is the machine working? Is it cutting? What does the counter say?
SELECT State, Online, Cutting, CounterM, OrderID, Item, Updated FROM sw_link_v_machines
 WHERE MachineID = 'CORTE1';
```

6.3.4 Customer database user

The script `sql\create_sw_link_client_user.sql` creates a user with minimum privileges: it can write to `sw_link_inbox` (and change its `Status` to release drafts), read `sw_link_events` and the four `sw_link_v_*` views (previous section), and move its cursor (`sw_link_clients.LastEventID`). It cannot see the internal tables `sw_link_machines` or `sw_link_orders`. It has no privilege on the SoftWinding sw_ tables. Create one per customer and restrict the host it may connect from.

> **Note.** Direct access to the SoftWinding tables. It is still possible, as the SoftWinding manual describes (section 6.10): a line written to `sw_delivery_items` with the `StationID` of a Link machine is also picked up by the Link. But it is not supported for new customers: there is no validation, client, per-order answer or targeted notices, and it depends on the internal structure of SoftWinding. Use it only for SoftWinding's own tools.

6.4 Files: CSV, XML and Excel

The file channel turns a file that the customer system leaves in a folder into orders, and writes the results to another file. Everything is defined in one mapping file (`LinkFiles.xml`) in the same style as the SoftWinding *Export*: a list of `<Column>` elements that says which column of the file each field comes from. No programming is needed.

6.4.1 How it works

1. The Link watches the `folder` of each `<Import>` every 5 s (`[Files] PollSec`). It takes the files matching `mask` that have not been modified for at least 2 s (so as not to read one still being written).
2. It moves the file to `work\` (so two Links do not take it at once), reads it according to its format and applies the mapping.
3. If any row has a format error (a badly written number, a missing order or product, a column that does not exist), the whole file is rejected and nothing is sent.
4. Otherwise it writes the rows to the mailbox as a draft and releases them all at once: the Link never sees half a file. From there it is exactly the same process as the DB mailbox.
5. It waits for the Link answer (up to `[Files] WaitSec`, 120 s), writes the results file (`result`) and moves the original to `done\` if everything was accepted or to `error\` otherwise.

Dropping the same file again is safe: orders are idempotent and the rows answer `already received`.

6.4.2 Enabling it

1. Copy `docs\examples\LinkFiles.xml` to `<Link folder>\data\LinkFiles.xml` (or wherever `[Files] MappingFile` says) and adapt it.
2. Register the client in the window (**Clients**, `DB` channel). The `client` of the `<Import>` must match its identifier.
3. Put in the `.ini`: `[Files] Enabled=1`. Check the mapping with `SoftWindingLinkSvc.exe --checkfiles`, which lists the `Import` and `Export` elements or says what is wrong. Restart the service: afterwards the mapping is reread by itself whenever the file changes.
4. Drop a file into the folder and look at the `done\` or `error\` folder and the results file.

6.4.3 The Import element

Table 16. Attributes of <Import>.

Attribute	Value
`name, client, folder`	Required. Unique name, registered client and input folder (full path).
`mask`	File name mask; several separated by `;`. Default `*.*`.
`format`	Required: `csv`, `xml` or `xlsx`.
`delimiter, quote`	CSV only. `delimiter`: `auto` (default: guessed from the first line among `; ,` tab and `|`), one character or `tab`. `quote`: `"` by default.
`encoding`	CSV only: `latin1` (default, like the SoftWinding Export) or `utf8`. A file with a UTF-8 BOM is read as UTF-8.
`sheet`	XLSX only: sheet name or number (default 1).
`record`	XML only, required: XPath of the element that is a record, for example `/Pedidos/Pedido/Linea`.
`header_row, first_row`	CSV and XLSX. Header row (default 1; `0` = no header) and first data row (default the next one). Empty rows are ignored.
`decimal_separator`	`.` (default) or `,`: how numbers written as text come. It does not affect numeric Excel cells.
`done, error`	Folders for processed files. Default `done\` and `error\` inside `folder`.
`result, result_format`	Results file (full path) and its format (`csv`, `xml` or `xlsx`; default that of the original). It accepts `{file}`, `{import}`, `{client}`, `{date}`, `{time}`, `{datetime}`. Without `result`, none is written.

Table 17. Attributes of <Column> in an <Import>.

Column attribute	Meaning
`field`	Link field: `orderId`, `item`, `batchId`, `customer`, `machineId`, `productId`, `reelId`, `lengthM`, `cuts`, `priority`, `description`, `notes`, `free1` … `free10` and `action` (`C` create, `X` cancel, `K` confirm, `S` receive a reel; default `C`). Also `providerId`, only for `S`.
`source`	Where the value comes from. CSV and XLSX: the header name, the column number (1, 2…) or the letter (A, B…). XML: an XPath relative to the record (`@attribute`, `Element`, `../@number`).
`value`	A fixed value for all rows (instead of `source`).
`default`	Value if the source is empty.
`map`	Value translation: `1=CORTE1;2=CORTE2`. Anything not in the list is left unchanged.
`scale`	Factor for numeric fields: `0.001` turns millimeters into meters.
`prefix, suffix`	Text put before or after (text fields).

Row rules. `orderId` is required. For `C`, also `productId` and `lengthM` greater than 0; `cuts` is 1 if missing. For `S` (receiving a reel), `productId`, `reelId` and `lengthM` (0 or more, but the cell cannot be empty). If there is no `item` column (or it is empty), the position is numbered by itself, 1, 2, 3…, within each order in file order. A cancel (`X`) without a position cancels the whole order. Completely empty rows are ignored.

6.4.4 Examples

CSV from a Spanish ERP (`;` separator, decimal comma, millimeters), `docs\examples\orders_es.csv`:

```
Pedido;Pos;Maquina;Articulo;Largo_mm;Cantidad;Bobina;Texto
SO-100234;10;1;CABLE6;47200;5;R-77881;Cable 6 mm
SO-100235;10;2;CABLE6;8000,5;1;;"Texto con ; dentro"
```

```
<Import name="erp_csv" client="ACME" folder="C:\LinkData\in\acme_csv" mask="*.csv" format="csv"
        delimiter="auto" encoding="latin1" decimal_separator=","
        result="C:\LinkData\out\acme\{file}_resultado.csv">
  <Column field="orderId"   source="Pedido"/>
  <Column field="item"      source="Pos"/>
  <Column field="machineId" source="Maquina" map="1=CORTE1;2=CORTE2"/>
  <Column field="productId" source="Articulo"/>
  <Column field="lengthM"   source="Largo_mm" scale="0.001"/>
  <Column field="cuts"      source="Cantidad"/>
  <Column field="customer"  value="ACME Cables"/>
</Import>
```

XML with one element per line (`docs\examples\orders.xml`):

```
<Pedidos>
  <Pedido numero="SO-100240">
    <Linea art="CABLE6"  metros="47.2" cortes="5"/>
    <Linea art="CABLE10" metros="12"   cortes="2"/>
  </Pedido>
</Pedidos>
```

```
<Import name="wms_xml" client="ACME" folder="C:\LinkData\in\acme_xml" mask="*.xml" format="xml"
        record="/Pedidos/Pedido/Linea">
  <Column field="orderId"   source="../@numero"/>
  <Column field="productId" source="@art"/>
  <Column field="lengthM"   source="@metros"/>
  <Column field="cuts"      source="@cortes"/>
  <Column field="machineId" value="CORTE1"/>
</Import>
```

Excel (.xlsx): sheet `Pedidos` with the header `Order, Pos, Machine, Product, Length, Qty` (`docs\examples\orders_en.xlsx`). Numeric cells are read as numbers, whatever the Excel decimal separator. A formula is read by its last calculated value; an Excel date is a serial number, so if you need it as a date make it text.

```
<Import name="excel_orders" client="ACME" folder="C:\LinkData\in\acme_xlsx" mask="*.xlsx"
format="xlsx"
        sheet="Pedidos" result="C:\LinkData\out\acme\{file}_result.xlsx">
  <Column field="orderId"   source="Order"/>
  <Column field="item"      source="Pos"/>
  <Column field="machineId" source="Machine" default="ALL"/>
  <Column field="productId" source="Product"/>
  <Column field="lengthM"   source="Length"/>
  <Column field="cuts"      source="Qty"/>
</Import>
```

6.4.5 The results file

One row per row of the original, with the columns `row` (row of the original), `orderId`, `item`, `productId`, `machineId` (the assigned machine), `status`, `code` and `text`. `status` is `OK` (accepted), `ERROR` (rejected: `code` and `text` say why) or `PENDING` (the Link did not answer within `WaitSec`; the row stays in the mailbox and will be processed). In a file rejected for format, each row carries the reason and the others `not sent: the file has rows with errors`.

6.4.6 Machine results in a file (Export)

An `<Export>` reads the notices (`sw_link_events`) and writes them to a CSV, XML or Excel file, with the same `<Column>` style as the SoftWinding Export. Every round with new notices generates a file (or appends rows to a CSV with `append="1"`) and then advances its cursor: if the Link crashes half-way, the file is repeated and a notice is never lost.

Table 18. Attributes of <Export>.

Attribute	Value
`name, file, format`	Required. `file` is a full path and accepts `{export}`, `{client}`, `{date}`, `{time}`, `{datetime}`. Two files in the same second are numbered `_1`, `_2`.
`client`	Only the notices of that client; empty = all.
`events`	Notice types separated by commas: `line.done` (default), `line.error`, `line.canceled`, `label.printed`.
`append`	CSV only: `1` appends to the same file (the header is written once).
`delimiter, decimal_separator, encoding`	CSV: `;`, `.` and `latin1` by default. `decimal_separator` also applies to numbers written as text in XLSX and XML.
`sheet, numeric_as_text`	XLSX: sheet name (`Sheet1`); `numeric_as_text="1"` writes numbers as text.
`root, record`	XML: names of the root element (`LinkExport`) and of each record (`Line`).
`from_start`	`1` starts with the first existing notice; by default it starts with the new ones, so as not to dump the history.

The `<Column>` elements of an `<Export>` carry `header` (the column title), `field` or `value` (a fixed value), and optionally `default`, `map`, `format` (a mask such as `0.000`), `scale`, `prefix` and `suffix`. Available fields: `eventId`, `created`, `type`, `machineId`, `clientId`, `batchId`, `orderId`, `item`, `reelId`, `code`, `productId`, `lengthM` (ordered meters), `cutsOrdered`, `cutsDone`, `measuredM` (measured meters), `endTime`, `text` (error text), `description`, `customer` and `payload.<key>` for any other notice data.

```
<Export name="hechos_csv" client="ACME" events="line.done,line.error"
        file="C:\LinkData\out\acme\cortes.csv" format="csv" append="1"
        delimiter=";" decimal_separator="," encoding="latin1">
  <Column header="Pedido"  field="orderId"/>
  <Column header="Pos"     field="item"/>
  <Column header="Metros"  field="measuredM" format="0.000"/>
  <Column header="Cortes"  field="cutsDone"/>
  <Column header="Sistema" value="LOIMEX"/>
</Export>
```

> **Tip.** Before putting the channel into production, validate the mapping with `--checkfiles` and a test file: the separator, the encoding and the sheet name are what usually differ between systems.

6.5 REST / JSON

1. In **Settings**, enable REST/JSON (port 8090 by default), save and restart the service.
2. In **Clients**, **New**: identifier, `REST` channel, allowed IPs and allowed machines. Copy the token (shown only once).
3. The client sends `Authorization: Bearer <token>` with every request.

Table 19. REST API.

Method and path	Function
`POST /api/v1/orders`	Creates an order with its lines. Idempotent by `orderId`.
`GET /api/v1/orders/{orderId}`	Status of the order and of each line.
`DELETE /api/v1/orders/{orderId}?item=`	Cancels an order or a line (only if not started).
`POST /api/v1/orders/{orderId}/lines/{item}/confirm`	Confirms the cut (machines with confirmation `X`).
`GET /api/v1/events?after={id}&limit=`	Events with a greater `EventID`. Without `after` it uses the client cursor.
`POST /api/v1/events/ack`	`{"lastEventId": n}` moves the cursor.
`GET /api/v1/machines`	Machine status.
`DELETE /api/v1/machines/{machineId}/order`	Cancels what the machine has on its screen, without an order number (not if it is running).
`POST /api/v1/orders/{orderId}/lines/{item}/cut-screen`	`{"on":true|false}`: shows or hides the cut screen of the machine.
`GET /api/v1/machines/{machineId}/cable`	Current cable diameter and mode (laser or manual).
`PUT | POST /api/v1/machines/{machineId}/cable`	`{"diameterMm":10.5}` sends the diameter (manual mode); `{"laser":true}` goes back to the laser gauge.
`GET /api/v1/machines/{machineId}/process`	Process time (mm:ss) and working speed (m/min).
`GET /api/v1/health`	Health check, no authentication. Returns `status`, `version`, `schema` and `licensed` (`false` in DEMO mode).

```
curl -H "Authorization: Bearer <token>" -H "Content-Type: application/json" -d '{
  "orderId": "SO-100234", "customer": "ACME Cables Ltd",
  "lines": [ {"item": 10, "machineId": "CORTE1", "productId": "CABLE6",
              "reelId": "R-77881", "lengthM": 47.2, "cuts": 5} ] }' \
  http://server:8090/api/v1/orders

curl -H "Authorization: Bearer <token>" "http://server:8090/api/v1/events?after=0"
```

Responses: `201` created, `200` identical repetition, `400` validation (with `code` 4001–4012), `401`, `403`, `404`, `409` line already running, `429` too many requests or blocked IP and `503` machine offline. Errors carry `{"code":4001,"error":"unknown machine","orderId":"…","item":10}`. The complete definition (OpenAPI 3) is in `docs\openapi.json`.

6.5.1 Machine functions over REST

The five machine routes do not depend on the client having a line on the machine (except `cut-screen`). REST leaves a machine order in the `sw_link_commands` table and waits up to 6 s for its result; it is executed by the machine thread, so **the engine must be running** and connected to the PLC. If the machine does not answer in 6 s: 504 and the order is canceled so that it does not run later.

Table 20. Machine functions of the REST API.

Route	Body and response	Errors
`DELETE …/machines/{id}/order`	200 `{"machineId":"M1","canceled":true,"detail":"canceled SO-1/10"}`; with no line in the Link, `"detail":"cleared"` and the PLC is left clean. The client receives `line.canceled` with `measuredM` and `cuts`.	404 unknown or not allowed machine; 409 no heartbeat for 30 s or **running** (code 4014); 502 the PLC refuses; 504.
`POST …/lines/{item}/cut-screen`	Optional body `{"on":true|false}` (no body = `true`). 202 `{"status":"requested","orderId":"…","item":10,"cutScreen":true}`. The manual order stays until the next automatic change.	400 invalid body; 404 order or line; 409 the line is not active on a machine.
`GET …/machines/{id}/cable`	200 `{"machineId":"M1","laser":true,"manual":false,"diameterMm":12.5,"raw":125}`. `laser` and `manual` are missing if the profile has no `CableManual`; `raw` is the PLC value as is.	404; 409 no heartbeat; 501 the profile has no `CableDiam`; 502; 504.
`PUT | POST …/machines/{id}/cable`	`{"diameterMm":10.5}` (0 to 1000) writes the diameter and goes to manual; `{"laser":true}` goes back to the laser. The response is that of the query, read back from the PLC.	400 body or diameter outside 0 to 1000; 404; 409; 501 no `CableSet`; 502; 504.
`GET …/machines/{id}/process`	200 `{"machineId":"M1","processTime":"08:03","processSeconds":483,"speedMPerMin":12.5,"speedRaw":125,"currentOrderId":"PR-1","currentItem":10}`. Fields the profile lacks (`ProcTime`, `Speed`) are missing, and so are the line ones if the machine has none.	404; 409 no heartbeat; 501 the profile has neither `ProcTime` nor `Speed`; 502; 504.

In addition, each line of `POST /orders` accepts `cableDiameterMm` (0 to 1000): the Link sends it to the machine when preparing the line. The diameter and the time do not travel over AHP, the DB mailbox or files.

6.5.2 .NET and other HTTP clients

- The API is **HTTP/1.1 without TLS** (`http://` only). If the client requires HTTPS, put a reverse proxy with a certificate in front (IIS, nginx, Caddy).
- One request per connection: the server closes after answering and clients reconnect by themselves. JSON body in UTF-8 with `Content-Type: application/json` (the charset is optional). The `Bearer` scheme is case-insensitive.
- It accepts `Transfer-Encoding: chunked` bodies and `Expect: 100-continue`, which is what .NET `HttpClient.PostAsJsonAsync` sends. Body limit: 1 MB (also for chunked).
- Limits: 600 requests per minute and client; 8 consecutive authentication failures block the IP for 5 minutes (429).
- Codes: `200`, `201`, `202` (screen order accepted), `400`, `401`, `403`, `404`, `409`, `429`, `501` (the profile lacks the function), `502` (the PLC refuses it), `503` and `504` (the machine does not answer).

```
using var http = new HttpClient { BaseAddress = new Uri("http://servidor:8090/api/v1/") };
http.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", token);

var r = await http.PostAsJsonAsync("orders", new {
    orderId = "SO-100234", customer = "ACME Cables Ltd",
    lines = new[] { new { item = 10, machineId = "CORTE1", productId = "CABLE6", lengthM = 47.2,
cuts = 5 } } });
var estado = await http.GetFromJsonAsync<JsonElement>("orders/SO-100234");
var cable  = await http.GetFromJsonAsync<JsonElement>("machines/CORTE1/cable");
```

6.6 TCP: AHP protocol

AHP is the framing protocol used by Jungheinrich warehouse systems: `<STX>` + 5-digit sequence + 2-digit type + fixed-width fields + `<ETX>`. Every telegram is acknowledged with the same sequence.

- Two channels: the client connects to the Link receiving port (3001) and sends; the Link connects to the client IP and port (`TcpHost`:`TcpPort`) to send its replies.
- The client is identified by its IP: there can be only one TCP client per IP address. A connection from an unregistered IP is closed and logged.
- At-least-once, in order delivery: the Link does not send the next event until the previous one is acknowledged; with no acknowledgement in 1.5 s it retries up to 3 times with a new connection and then every 8 s, skipping none. It validates that the acknowledgement repeats the sequence.
- The acknowledgement format is chosen per client (`AckFormat`): `S` sequence only, `C` sequence and type.
- Telegram `00` is the keep-alive and is mirrored on the outgoing channel.

Table 21. Telegrams per dialect.

Dialect	Incoming (client → Link)	Outgoing (Link → client)
`L` Link (generic orders)	`10` order line, `11` cancel (with a blank order and a machine at position 36, it cancels what that machine has on its screen; a rejection arrives as `78` with the code)	`60` accepted, `70` done, `71` canceled, `78` error, `80` machine status
`T` Jungheinrich warehouse	`32` cut request, `05` cancel, `03` move platform (only logged)	`92` cut done, `65` canceled, `58` error (1001–3000)

The field lengths of every `L` telegram are in the specification (`docs\SPEC_Link_v1.md`, section 9.3). In the `T` dialect, a `32` request becomes a line (`DeliveryID` = TaskID, `ProductID` = cut type, `Qty` = QPick/1000…); existing batches and lines are never deleted.

6.7 MQTT / VDA 5050

To integrate the machines into a logistics system (robots, fleets, cloud platforms such as AWS IoT Core) the Link speaks MQTT with the VDA 5050 standard (version 2.0): every Link machine presents itself as a vehicle and the customer system is the master control. Between the two there is an MQTT server (the broker), for example AWS IoT Core. It uses the same order rules as the other channels: validation, idempotency, cancellation and the same error codes.

> **How the standard is applied.** VDA 5050 describes mobile vehicles. For a stationary cutting machine the Link applies the standard as follows: the transport (MQTT), the topics, the message structure (`order`, `instantActions`, `state`, `connection`, `factsheet`) and the action life cycle are those of VDA 5050; the cut command is a manufacturer-specific action (`cut`), which the Link declares in the `factsheet` as the standard allows; the movement fields (`agvPosition`, battery…) are filled with neutral values. The

customer's master control must send this action.

6.7.1 Getting started

1. Run `sql\create_sw_link.sql` (schema 6; the `sw_link_vda` table exists since schema 4) and `sql\create_sw_link_user.sql` for the grants. With schema 3 the Link keeps working, but without the MQTT channel.
2. In **Clients**, register a client with the `MQTT` channel (by default it is named `MQTT`; it is the client that owns the orders arriving through this channel). If the client has `AllowedMachines`, it can only send to those machines.
3. In **Settings**, MQTT / VDA 5050 group: server, port, TLS and the certificate files, and the manufacturer (part of the topic). The other keys are in the ini (next table).
4. With TLS, copy `libssl-3-x64.dll` and `libcrypto-3-x64.dll` (OpenSSL 3) next to `SoftWindingLinkSvc.exe`. They come in the distribution folder.
5. Restart the service. The log says `MQTT / VDA 5050 channel started` and, for each machine, `MQTT <machine> connected`.

Table 22. Keys of the [Mqtt] ini section.

Key of [Mqtt]	Default and meaning
`Enabled`	0. Enables the channel.
`Host, Port`	empty and 8883 (1883 without TLS). The MQTT server; on AWS: `xxxx-ats.iot.<region>.amazonaws.com`, port 8883.
`Tls, Verify`	1 and 1. TLS 1.2 or higher; `Verify=1` checks the server certificate and name.
`CAFile`	Trusted certificates (PEM). On AWS: `AmazonRootCA1.pem`.
`CertFile, KeyFile, KeyPassword`	X.509 client certificate and its private key (PEM), with optional password. AWS IoT Core requires a client certificate.
`User, Password`	For servers with user and password (the password is stored DPAPI-encrypted when saved from the window).
`ClientIdPrefix`	empty. The MQTT ClientID of each machine is `<prefix><MachineID>`. On AWS it must match the thing name.
`InterfaceName, MajorVersion, Manufacturer`	`uagv`, `v2` and `LOIMEX`: segments of the topic `uagv/v2/LOIMEX/<machine>/...`.
`ClientID`	`MQTT`. Link client (`sw_link_clients`) owning the orders of this channel.
`KeepAlive, StateSeconds`	30 and 30. MQTT keep-alive (AWS: 30 to 1200 s) and how many seconds between `state` messages even if nothing changes.

6.7.2 Topics

Each machine opens its own MQTT connection (like a vehicle) and uses the topics `<interface>/<version>/<manufacturer>/<machine>/...`; the VDA 5050 `serialNumber` is the Link machine identifier (`MachineID`, only letters, digits and `_ . : -`).

Table 23. Topics of the MQTT channel.

Topic (uagv/v2/LOIMEX/CORTE1/…)	Direction	QoS	Content
`order`	client → Link	0 / 1	An order: nodes with `cut` actions.
`instantActions`	client → Link	0 / 1	`cancelOrder`, `stateRequest`, `factsheetRequest`.
`state`	Link → client	0	Machine and action state. On every change and at least every `StateSeconds`.
`connection`	Link → client	1, retain	`ONLINE` on connect, `OFFLINE` when the Link stops and `CONNECTIONBROKEN` (the server will message) if the Link dies without saying goodbye.
`factsheet`	Link → client	0, retain	Description of the machine and of the `cut` action with its parameters.

6.7.3 Sending an order: the cut action

The order is an `order` message with one node per line; each node carries a `cut` action (other actions are ignored). `orderId` is the order number (up to 30 characters). An order with the same `orderId` and a higher `orderUpdateId` updates the lines that have not started; a lower one is rejected with the `orderUpdateError` error; sending the same message again duplicates nothing.

Table 24. Parameters of the cut action (actionParameters: list of {key, value}).

cut parameter	Required	Meaning
`productId`	yes	Product to cut.
`lengthM`	yes	Meters of each cut (number or text, with dot or comma).
`cuts`	no (1)	Number of cuts of the line.
`item`	no	Line position (default 10, 20, 30… in node order).
`reelId`	no	Identifier of the reel the material is cut from.
`description, customer, batchId, notes`	no	Line text (goes to the label), end customer, batch (default `MQTT_<date>`) and notes.
`priority, free1 … free10`	no	Priority and free fields.
`cableDiameterMm`	no	Cable diameter in mm (0 to 1000): the machine goes to manual with that diameter when the line is prepared.

```
{ "headerId": 1, "timestamp": "2026-10-21T10:00:00.000Z", "version": "2.0.0",
  "manufacturer": "LOIMEX", "serialNumber": "CORTE1",
  "orderId": "SO-100234", "orderUpdateId": 0,
  "nodes": [
    { "nodeId": "N10", "sequenceId": 0, "released": true, "actions": [
        { "actionType": "cut", "actionId": "SO-100234-10", "blockingType": "HARD",
          "actionParameters": [ {"key": "productId", "value": "CABLE6"},
                                {"key": "lengthM", "value": 47.2},
                                {"key": "cuts", "value": 5},
                                {"key": "reelId", "value": "R-77881"} ] } ] },
    { "nodeId": "N20", "sequenceId": 2, "released": true, "actions": [
        { "actionType": "cut", "actionId": "SO-100234-20", "blockingType": "HARD",
          "actionParameters": [ {"key": "productId", "value": "CABLE6"},
                                {"key": "lengthM", "value": 3.5}, {"key": "cuts", "value": 2} ] } ] }
],
  "edges": [ { "edgeId": "E0", "sequenceId": 1, "released": true,
               "startNodeId": "N10", "endNodeId": "N20", "actions": [] } ] }
```

6.7.4 The state: state

The Link publishes the `state` of each machine whenever something changes (order received, cut done, error, machine state change) and at least every `StateSeconds`. It carries the current `orderId`, the last finished node (`lastNodeId`), the pending nodes (`nodeStates`) and the state of every action:

Table 25. States of a cut action.

actionStatus	When
`WAITING`	The line is accepted and waits for its turn or a free machine.
`INITIALIZING`	The machine has claimed it and is getting ready (the PLC receives the order).
`RUNNING`	The machine is measuring.
`FINISHED`	Line finished. `resultDescription` and the `cutResult` entry of `information` give the measured meters and the cuts.
`FAILED`	Rejected (code 4001–4013 in `resultDescription`), failed on the machine, or canceled (`canceled`).

`errors` collects the problems: `lineError` (a rejected or failed line, with its code), `orderError` (invalid message or no `cut` action), `orderUpdateError`, `noOrderToCancel`, `emergencyStop` (`FATAL` level, with `safetyState.eStop`), `machineError` and `plcOffline` (no communication with the PLC). `information` carries the machine status (`machineStatus`: counter and cut number) and one `cutResult` per finished line. `driving` is true while the machine is measuring.

```
{ "headerId": 14, "timestamp": "2026-10-21T10:00:41.120Z", "version": "2.0.0",
  "manufacturer": "LOIMEX", "serialNumber": "CORTE1",
  "orderId": "SO-100234", "orderUpdateId": 0, "lastNodeId": "N10", "lastNodeSequenceId": 0,
  "nodeStates": [ { "nodeId": "N20", "sequenceId": 2, "released": true } ], "edgeStates": [],
  "driving": true, "paused": false, "newBaseRequest": false, "operatingMode": "AUTOMATIC",
  "actionStates": [ { "actionId": "SO-100234-10", "actionType": "cut", "actionStatus": "FINISHED",
                      "resultDescription": "measured 236.000 m, 5 cut(s)" },
                    { "actionId": "SO-100234-20", "actionType": "cut", "actionStatus": "RUNNING" } ],
  "batteryState": { "batteryCharge": 100.0, "charging": false },
  "errors": [],
  "information": [ { "infoType": "machineStatus", "infoDescription": "measuring 1.250 of 3.500 m",
"infoLevel": "INFO",
                     "infoReferences": [ {"referenceKey": "counterM", "referenceValue": "1.250"} ] } ],
  "safetyState": { "eStop": "NONE", "fieldViolation": false } }
```

With the machine online the `state` also carries live data: `velocity.vx` (m/s, the working speed in m/min divided by 60) and two `information` entries: `processTime` (`infoDescription` = `mm:ss`; references `orderId`, `item`, `seconds` and `speedMPerMin`) and `cableDiameter` (references `diameterMm` and `mode` = `laser`, `manual` or `unknown`). They are not published with the machine offline or with a schema older than 6.

6.7.5 Instant actions

Table 26. Supported instant actions; any other ends in FAILED.

actionType	Effect
`cancelOrder`	Removes what the machine has on its screen (even if it has already measured) and cancels the other pending lines of the current order; it works even when there is no current order. With the machine running the action ends in `FAILED` (`machineRunning` error). If there is nothing to cancel: `noOrderToCancel` error. Result: `N line(s) canceled`.
`setCableDiameter`	Parameter `diameterMm` (0 to 1000): sends the cable diameter and puts the machine in manual mode.
`cableLaser`	Goes back to the laser diameter gauge.
`stateRequest`	Publishes the `state` immediately.
`factsheetRequest`	Publishes the `factsheet` again.

6.7.6 Connecting to AWS IoT Core

1. In AWS IoT Core create one thing per machine with the same name as the `MachineID` (`CORTE1`), with its device certificate. Download the certificate, the private key and the Amazon root CA (`AmazonRootCA1.pem`).
2. Attach to the certificate a policy such as the one below: each machine can only connect with its own name and use its own topics.
3. Configure the Link: `Host` = the AWS IoT endpoint (IoT settings → Settings), `Port` = 8883, `Tls=1`, `Verify=1`, `CAFile`, `CertFile` and `KeyFile`. With a thing-name policy leave `ClientIdPrefix` empty.
4. If all machines share one certificate, `ClientIdPrefix` and the policy must agree with that (AWS accepts only one connection per ClientID).

```
{ "Version": "2012-10-17", "Statement": [
  { "Effect": "Allow", "Action": "iot:Connect",
    "Resource": "arn:aws:iot:REGION:ACCOUNT:client/${iot:Connection.Thing.ThingName}" },
  { "Effect": "Allow", "Action": ["iot:Publish", "iot:RetainPublish"],
    "Resource": "arn:aws:iot:REGION:ACCOUNT:topic/uagv/v2/LOIMEX/${iot:Connection.Thing.ThingName}/*"
},
  { "Effect": "Allow", "Action": "iot:Subscribe",
    "Resource":
"arn:aws:iot:REGION:ACCOUNT:topicfilter/uagv/v2/LOIMEX/${iot:Connection.Thing.ThingName}/*" },
  { "Effect": "Allow", "Action": "iot:Receive",
    "Resource": "arn:aws:iot:REGION:ACCOUNT:topic/uagv/v2/LOIMEX/${iot:Connection.Thing.ThingName}/*" }
] }
```

The customer system (master control) subscribes to `uagv/v2/LOIMEX/+/state`, `connection` and `factsheet`, and publishes to `.../<machine>/order` and `instantActions`. If the server closes the connection right after TLS, the Link log says so: it is almost always the client certificate (not accepted or without a policy) or a ClientID that is not allowed.

> **Tip.** Before going into production, check in the AWS console that the policy, the thing name and the certificates match the configuration of the Link. Connection errors (certificate not accepted, ClientID not allowed) are written to the Link log.

6.8 Events and cursor

Everything the Link has to tell the customer goes to the `sw_link_events` queue with an increasing `EventID`. Each client has its cursor (`LastEventID`): over REST it is moved with `POST /events/ack`; over TCP it moves by

itself on acknowledgement. Machine status (`machine.status`) is emitted only on changes, at most every 5 s per machine. Events are kept for 30 days (`EventRetentionDays`).

6.9 Functions per channel

What each function is called in each channel and whether it is available (OK = available, – = not available). The complete table, with notes, is in `docs\Tabla_Comandos_por_Protocolo_en.md`.

Table 27. Functions per channel.

Function	DB mailbox	REST	TCP dialect L	TCP dialect T	MQTT / VDA	Files
Send order / line	INSERT `Action=C`	POST /orders	telegram 10	telegram 32	topic `order`	`action=C`
Cancel by order number	`Action=X`	DELETE /orders/{id}	telegram 11	telegram 05	cancelOrder	`action=X`
Cancel what is on the screen	`X`, empty order + `MachineID`	DELETE /machines/{id}/order	11, blank order + machine	–	cancelOrder	`X`, empty order + `machineId`
Confirm a cut (`X` confirmation)	`Action=K`	POST …/confirm	–	–	–	`action=K`
Cut screen	–	POST …/cut-screen	–	–	–	–
Cable diameter	–	GET / PUT /machines/{id}/cable	–	–	setCableDiameter, cableLaser	–
Reel receipt (stock control)	`Action=S`	–	–	–	–	`action=S`
Order status	view `sw_link_v_orders`	GET /orders/{id}	notices 60 / 70 / 71 / 78	92 / 65 / 58	topic `state`	result file
Machine status	view `sw_link_v_machines`	GET /machines	telegram 80	–	topic `state`	–
Process time and speed	–	GET …/process	–	–	`state.information` and `velocity`	–
Events (done, error, canceled, waste, label)	`sw_link_events` + `LastEventID`	GET /events, POST /events/ack	sent by the Link	sent by the Link	topic `state`	–

7. How the Link works

7.1 The path of an order through the database

All channels end up on the same path. The customer writes its order only to the Link tables (the `sw_link_inbox` mailbox, a file that the Link turns into mailbox rows, or a REST/TCP call that the Link serves); it does not touch the SoftWinding tables. It is the Link that translates the order into the SoftWinding order tables, sends the line to the PLC and, when finished, writes the result: the label, the notice to the customer and, if stock control has been turned on, the movement and the reel deduction.

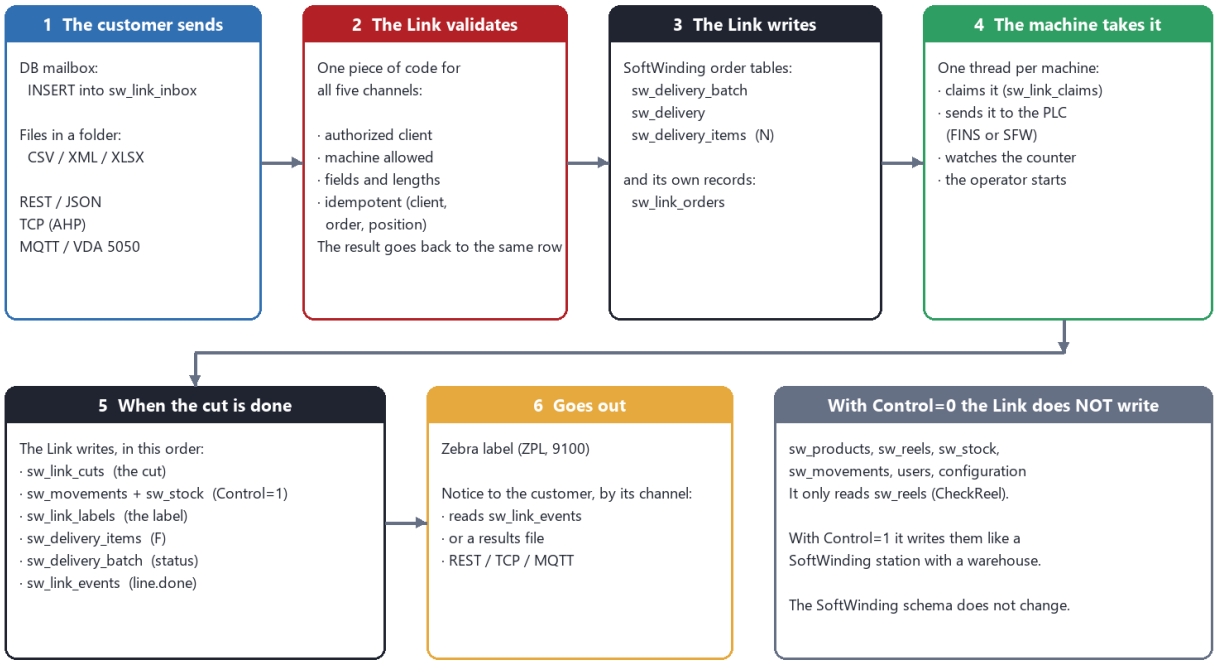


Figure 18. The path of an order, from the customer input to the label and the notice.

> **Note.** By default the SoftWinding master data is not touched: products, reels, stock, users and configuration. The Link does not change the SoftWinding schema (`SQLVer` 172) nor raise its version; it only adds its own `sw_link_*` tables. The only thing it writes outside them are the order lines, which classic SoftWinding stations see and can use. With stock control on (section 7.2) it also writes products, reels, stock and movements, as a SoftWinding station with the warehouse enabled would.

Table 28. What the Link reads and writes.

Table	Written by	What the Link does
`sw_link_inbox`	The customer (and the file channel)	Reads it, validates it, writes the answer in the same row (`Status`, `Code`, `Result`, `Assigned`) and deletes it after 30 days.
`sw_link_clients`, `_machines`, `_profiles`, `_printers`	The administrator, from the window	Reads them. In `sw_link_machines` it also writes the live state (state, heartbeat, counter, owner).
`sw_delivery_batch`, `sw_delivery`, `sw_delivery_items`	The Link, like a SoftWinding station	Creates the lines (`Status = N`), claims them (`C`), updates `Qty_Cutted` and `Qty_Measured` and closes them (`F`). A classic SoftWinding sees them and can work with them.
`sw_link_orders`	The Link	Relates the customer reference to the SoftWinding keys (idempotency and owner of each line).
`sw_link_claims`	The Link	Marks which machine has each line; allows recovery after a crash.
`sw_movements`	The Link, inserts only, and only with `Control=1`	One movement of type `O` with `UserID = LINK` for every closed cut, and `I` or `O` when receiving or adjusting a reel (`S` action). It never modifies or deletes a movement.
`sw_link_cuts`	The Link, always	One row per confirmed cut, with or without stock control. The window and web report production and the recovery after a crash come from it.
`sw_link_labels`	The Link	One row per cut; sends it to the Zebra and notes whether it was printed.
`sw_link_events`	The Link	One notice for each thing the customer must know (`line.done`, `line.error`…). The customer reads them with its cursor.
`sw_link_commands`, `sw_link_filestate`, `sw_link_log`	The Link	Confirm or cancel commands to a machine, cursors of the file `Export` elements and the log.
`sw_reels`, `sw_products`, `sw_stock`	SoftWinding; the Link only with `Control=1`	By default read only, and only `sw_reels` and only with reel checking on. With `Control=1` the Link creates products, reels and stock rows, and deducts `Qty` and `Date_Movement` of `sw_stock` (section 7.2).

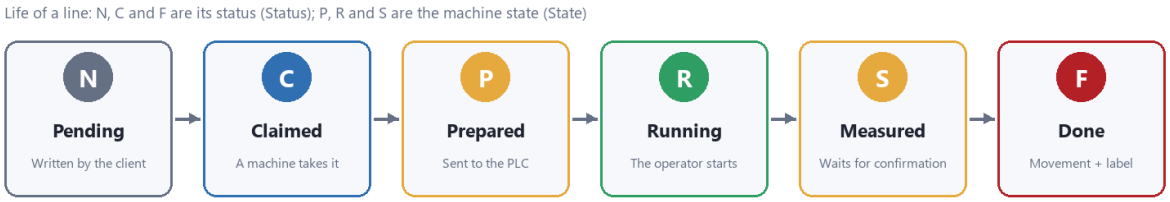


Figure 19. Life of a line.

7.2 Stock control and products (optional)

By default the Link is an order exchanger: it sends the lines to the machine and returns the result, and it writes to no SoftWinding product, reel, stock or movement table. If SoftWinding must keep the cable warehouse as its stations do, turn it on in `SoftWindingLink.ini` or in the **SoftWinding stock and products** group of the Settings tab:

```
[Stock]
Control=1          ; movements, sw_stock deduction and reel entries (0 = only exchange orders)
CreateProducts=1   ; create in sw_products the products that are missing (default: same as
Control)
Warehouse=300001   ; warehouse of the reels the Link creates (default -1)
```

Table 29. The three stock control modes.

Setting	What the Link does
`Control=0`, `CreateProducts=0` (default)	It only exchanges orders. It does not touch `sw_products`, `sw_reels`, `sw_stock` or `sw_movements`. It records every cut in `sw_link_cuts`, so the production of the window and the web report works the same. The `S` action is rejected with 4013. It does not need the permissions of `create_sw_link_user_stock.sql`.
`Control=0`, `CreateProducts=1`	When a line is accepted, if the product does not exist in `sw_products` it is created (`ProductID`, `Reference` = product, `Description` = line description). There are no movements or stock.
`Control=1`	Like a SoftWinding station with the warehouse enabled (`[Stock] Habilitado`): it creates missing products, writes one movement per cut, deducts the reel in `sw_stock` and accepts reel entries (`S` action).

7.2.1 What it does with Control=1

- When accepting an order: it creates in `sw_products` the product of every new line if it does not exist with any provider.
- When confirming a cut: it writes the movement (`sw_movements`, type `O`, `UserID = LINK`, with `ReelID`, `CutNr`, meters, order, position and location of the stock row) and then subtracts the meters from `sw_stock`: only `Qty` and `Date_Movement` change, as SoftWinding does.
- Which row it deducts: the one of the `ReelID` reel of the line (with its product and, if given, its provider); if the line has no reel, the one of its location (`WareHouse`, `Line`, `Colum`, `Level`, `Position`) if it has one. If there is no row it writes the movement but deducts nothing: the Link does not guess which reel was cut.
- Negative stock: it is allowed, as in SoftWinding, and a warning is left in the Log. The cut is not stopped (section 1.3).
- Order and idempotency: first the movement (unique per order-position-cut) and, only if it is new, the deduction. A retry after a crash does not deduct twice. If the Link crashes exactly between the two, that deduction is missing: fix it with an `S` entry that sets the real meters.
- Reel entries: `S` action of the mailbox or of files (section 6.3).

7.2.2 Database permissions

The script `sql\create_sw_link_user_stock.sql` gives the Link user `INSERT` on `sw_products`, `sw_reels` and `sw_stock`, and `UPDATE` only of `sw_stock.Qty`, `sw_stock.Date_Movement`, `sw_reels.ProductID` and `sw_reels.ProviderID`. Without it, with `Control=1` the order is cut anyway but the Log warns that the stock could not be deducted or the product created.

7.2.3 Typical imports and their equivalent in the Link

Table 30. Typical imports and where they are in the Link.

SoftWinding import	What it does	In the Link
Orders from an ERP Excel file	Creates the batch, the order and a line per row with `ReelID ALL`, and creates the product if it does not exist. It does not touch stock.	`Control=0`, `CreateProducts=1`. Order by Excel file with an `Import` (section 6.4).
Warehouse from an XML file	Creates the product (with provider), the reel and its stock row in the default warehouse unlocated, and adjusts the meters with an `I` or `O` movement.	`Control=1` and `S` action (mailbox or file).
Planning and notification	One order per cable; one notification per cut with the reel.	Orders by mailbox, file, REST, TCP or MQTT; `line.done` notices and file export (section 6.4).

The specific file formats of each customer (an ERP Excel, a warehouse XML) are described in the `LinkFiles.xml` mapping file (section 6.4).

7.3 Claiming a line

Every machine looks for the next pending line (`Status` `N` or `I`, with its `StationID` or `ALL`) ordered by batch priority, line priority and position. It claims it in two steps: an `INSERT` into `sw_link_claims` (unique key, arbitrates between machines) and a conditional `UPDATE` of the line (arbitrates between the Link and a classic SoftWinding). If it loses either, it moves to the next candidate. With 20 clients claiming the same line at once, exactly one wins.

7.4 Sending to the PLC

With the machine stopped and with no emergency or open guard, the Link repeats the order preparation sequence of SoftWinding: stop, preselection to 0, order running, counter reset, preselection = meters, cut number and maximum cuts, order texts (order, position, customer, address, warehouse, location, product, description, dates and reel) and a wake-up of the screen. It reads the preselection and the counter back to verify the PLC accepted them, and then the machine goes to "prepared".

7.5 Confirming and closing

When the cut is confirmed the Link writes in this order (no transactions, MyISAM): 1) the cut in `sw_link_cuts` (always, idempotent by order-position-cut); 2) with stock control on, the warehouse movement (`sw_movements`, type `O`, `UserID = LINK`) and the `sw_stock` deduction; 3) the queued label; 4) the line progress (`Qty_Cutted`, `Qty_Measured`) and, at the end, `Status = F`; 5) the batch status, using the SoftWinding rule (`F` if no line is left in `I`, `N` or `C`); 6) the `line.done` event and releasing the claim. If it crashes between two steps, recovery repeats them without duplicating.

7.6 Machine orders

The functions that are not an order (confirming a cut, canceling what is on the screen, the cut screen, the cable diameter, the time and the speed) reach the machine thread through the `sw_link_commands` table: whoever asks for them (REST, the mailbox, the MQTT channel or the window itself) writes a row with the machine and the command and waits up to 6 s for its result. The machine thread executes them **with or without a line in progress**; if it does not answer in that time, the order ends in error (`timeout`) and is not executed later. That is why these functions **need the engine running** (the service or **Start the engine here**).

Table 31. Machine orders.

Command	What it does
`confirm`	Confirms the cut (machines with external confirmation).
`cancel`	Cancels one specific line.
`clear`	Cancels what is on the screen and leaves the PLC clean. Refused with the machine running (4014).
`cutscreen`	Shows or hides the cut screen.
`cableget, cableset, cablelaser`	Reads the cable diameter, sends it (manual) or goes back to the laser.
`processget`	Reads the process time and the speed.

7.7 Recovery

Table 32. Recovery after a failure.

Situation at start-up or when the PLC returns	What it does
The PLC keeps the order	Resumes watching without preparing it again (the counter of a half-done cut is not lost).
The PLC is at zero and the line has no cuts, or the PLC lost the order	Resends the order and logs it.
There are cuts in `sw_link_cuts` that are not in the line	Repeats the closing with those data.
Another station changed the line	Releases the line (error 4006) and stops writing.
No PLC for 30 minutes	Releases the line (back to pending), keeping the cuts already done.

7.8 Machine safety

- No order is sent with the emergency stop active or a guard open.
- The Link never triggers the hydraulic cut.
- `Start` is only sent with `AutoStart` on that machine, line prepared and guards correct.
- If the PLC does not answer 3 reads in a row, the machine goes "offline" and nothing is sent until it recovers.
- Every order to the PLC is logged.

8. The "Link live" web report

SoftWindingWeb includes the **Link live** report (group **Stations**, minimum level **supervisor**) to follow the Link from any browser, without installing anything. It reloads by itself every 10 seconds while the tab is visible. The **Hours** parameter (24 by default) sets the period of the production figures, which come from `sw_link_cuts` and therefore work with or without stock control.

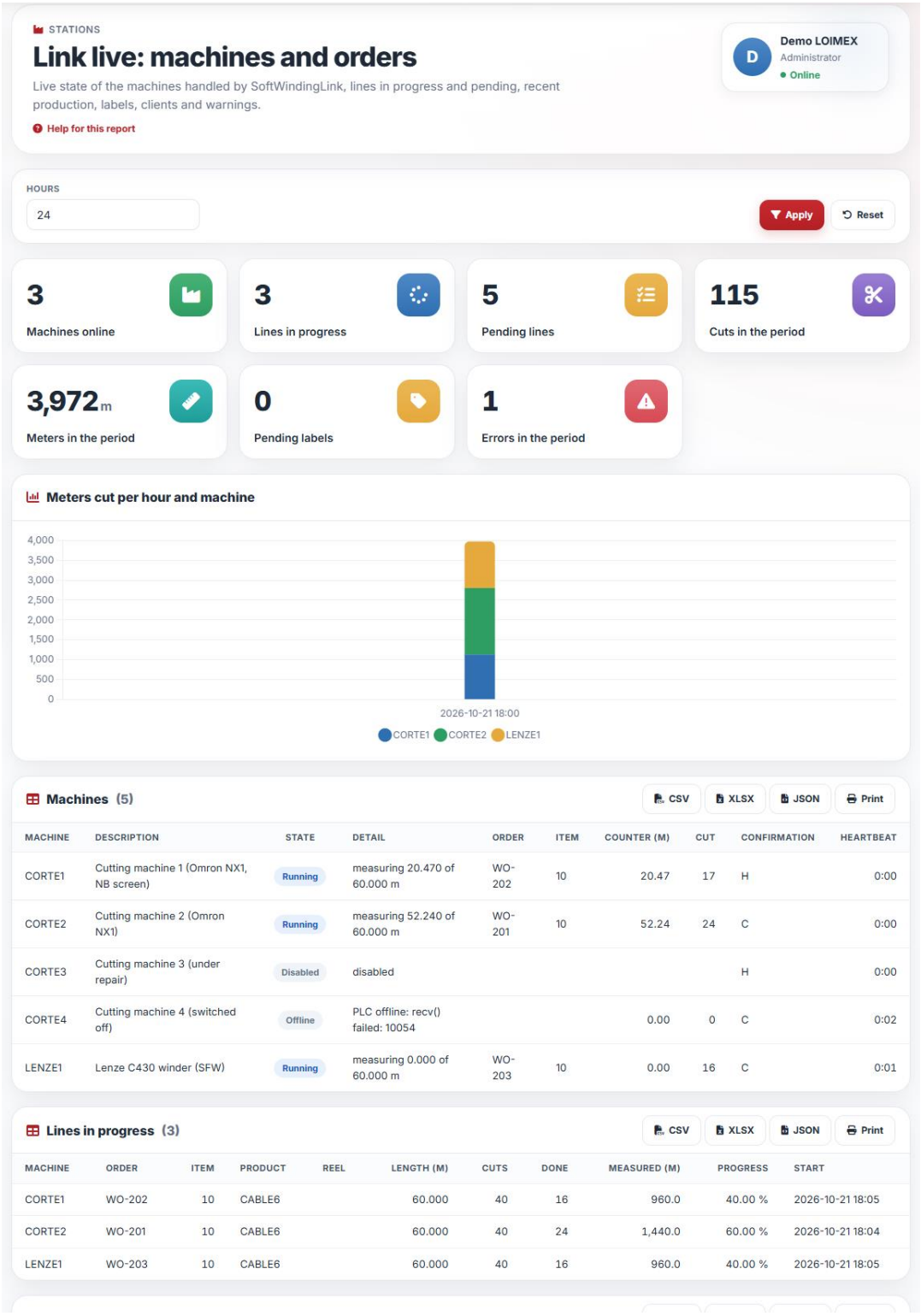


Figure 20. "Link live" report (upper part).

Table 33. Report content.

Block	Content
Indicators	Connected machines, lines in progress, pending lines, cuts and meters of the period, pending labels and errors of the period.
Chart	Meters cut per hour and machine (stacked bars).
Machines	Live state, detail, order and position, counter, cut, confirmation and heartbeat age.
Lines in progress / Pending per machine	Progress of each line; pending lines, cuts and meters per station.
Production per machine	Cuts and meters of the period with total.
Clients	Channel, dialect, activity and undelivered events (if it keeps growing, the client is not consuming).
Latest events / Warnings and errors	The latest events of the queue and the warnings and errors from the log.

Like every SoftWindingWeb report, each table can be exported to CSV, XLSX or JSON and printed. The report is available in six languages (Spanish, English, German, French, Italian and Portuguese). To enable it, the `www\reports\link_live` folder must be present in SoftWindingWeb, which reads the same database.

9. Operation and troubleshooting

9.1 What to check daily

- **Status**: service running, connected machines = enabled ones, failed labels = 0, undelivered events not growing, warnings and errors in the log.
- **Machines**: none "offline" or in "error" without reason.
- **Log**: filter by level `E` and `W`.

9.2 Common problems

Table 34. Troubleshooting.

Symptom	Probable cause	What to do
The machine is "offline"	Wrong IP or port, UDP 9600 / TCP 63000 blocked, FINS nodes wrong, PLC powered off.	**Test the PLC connection**; check firewall and nodes. The Link retries every 3 s.
The machine does not take lines	Disabled; line `StationID` different; `AcceptAll` unticked with `ALL` lines; `Qty` or `Qty_Cut` ≤ 0; emergency or open guard; another program controls it.	Look at Status and Log. A line with invalid data arriving by direct database is left alone.
"Machine taken by another computer"	Another instance (service, window or simulator) controls it; its heartbeat is recent.	Stop that instance. If it no longer exists, **Release claimed lines** or wait 30 s.
The line stays in `C` without progressing	The service stopped; the PLC lost the order.	On start, the Link recovers it. If not, **Release claimed lines**.
Error 4005	The reel of the line belongs to another product (with checking on).	Correct `ReelID` or `ProductID` in the order.
Still in DEMO mode with the code received	The machine ID is different (the disk changed or it is another server); the company name is not exactly the one given to LOIMEX (spaces, capitals, accents); the code is for another product; the service was not restarted.	Compare the ID in **Settings > License** with the one sent, type the name exactly and restart the service. `SoftWindingLinkSvc.exe --license` says whether the code is valid.
Error 4013	A reel entry (`S` action) with `[Stock] Control=0`.	Turn stock control on and grant the permissions (section 7.2) or do not send entries.
No movements or stock deduction	`Control=0` (default); the line has no `ReelID` or location with a row in `sw_stock`; the Link user lacks the permissions of `create_sw_link_user_stock.sql` (the Log says "command denied").	Check the Log. Turn `Control=1` on, run the permissions script and restart the service.
Negative reel stock	More was cut than the stock said (the Link does not stop for lack of stock) or the reel was never received.	Send an `S` entry with the real meters of the reel (section 6.3).
Error 4009	Cut measured, no confirmation within 30 minutes.	Confirm on the screen / send the external confirmation. The line does not close by itself.
Label in error	Zebra out of paper, paused, off or wrong IP.	**Printer status (~HS)**; load paper; set the label to `N` to reprint.
A mailbox row stays in `N`	The service is stopped; `[Inbox] Enabled=0`; the `sw_link_inbox` table does not exist (old schema); the Link user has no privileges on it.	Look at Status and Log (source `DB`). **Install or update the Link schema** and run `create_sw_link_user.sql` again.

Symptom	Probable cause	What to do
A mailbox row ends in `E`	`Code` and `Result` say it: 4001 unknown machine, 4002 disabled, 4004 invalid field, 4012 client or machine not allowed. If one line fails, all lines of the same order fail.	Correct and send the order again (it is idempotent).
A file is not processed	`[Files] Enabled=0`; the mapping is not valid (`--checkfiles`); the mask does not match; the file is still being written; the folder does not exist or the service cannot write to it.	Look at the Log (source `FILE`) and the `work\` folder. A file left in `work\` by a crash returns to the folder by itself after 10 minutes.
A file goes to `error\`	A row has a format error (number, missing column) or the Link rejected an order.	Open the results file: each row says why. Correct it and drop the file again.
REST 401 / 429	Wrong token or IP blocked after 8 failures (5 min).	**New token**; wait 5 minutes.
TCP client receives no events	Client IP/port wrong; does not acknowledge; two clients with the same IP; wrong `AckFormat`.	Look at the Log (source `AHP`) and the undelivered events; try the other `AckFormat`.
The machine goes to the "PLC refused the line" error (FINS code 1103 or 1104)	An address of the profile is outside the memory the PLC offers over FINS (for example DM 6000 onwards with `OMRON_NX` on a PLC that does not have it).	Change the machine profile to `OMRON_CP` or correct the addresses in **PLC profiles**. The line goes back to pending and is retried.
When a line is canceled another one appears on the machine	The machine takes the pending `ALL` lines of the database (**Also takes "ALL" lines** box) and, once free, claims the next one.	Untick the box and send each line with its `machineId` (chapter 4).
"Cancel order on screen" does nothing or gives an error	The engine is not running (no machine heartbeat), the machine is running (4014) or there is no line and the PLC was already clean.	Start the engine (service or **Start the engine here**), stop the machine and repeat.
In screen mode (`H`) no order appears on the machine	There are no pending lines in the first batch for that machine, or the profile writes the text block in an area the screen does not read.	Check the pending lines in **Orders** and the machine profile (`OMRON_CP` or `OMRON_NX`).

9.3 Error codes

Table 35. Error codes.

Code	Meaning
`4001`	Unknown machine
`4002`	Machine disabled or offline
`4003`	Duplicate or non-modifiable order (the line is already running or done)
`4004`	Invalid field (length, cuts, product, position)
`4005`	The reel belongs to another product
`4006`	Line lost (another station changed the line)
`4007`	PLC communication error
`4008`	Emergency or open guard
`4009`	No confirmation in the expected time
`4010`	Invalid character in an identifier
`4011`	The line cannot be canceled (running or done)
`4012`	Machine not allowed for this client

Code	Meaning
`4013`	Stock control is off (`S` action)
`4014`	The machine is running (cancel what is on the screen)
`4015`	The machine does not support this function (the profile lacks the parameters)
`4090`	Internal error
`1001–3000`	Codes of dialect `T`

9.4 Where the logs are

- **In the database:** the `sw_link_log` table (`LogID`, `Created`, `Level` D/I/W/E, `Source` ENGINE/DB/FILE/ADMIN, `MachineID`, `ClientID`, `Direction`, `Message`), the one shown by the **Log** tab. It is written by the engine and the machines (`ENGINE`), the DB mailbox (`DB`), the file channel (`FILE`) and the actions of the window (`ADMIN`). It purges itself after `LogRetentionDays` days.
- **In a file:** `link_YYYYMMDD.log` in the log folder (`[Logs] LogDir`): everything the service writes, one line per message (start-up, REST, AHP, MQTT, connection errors…).
- The REST, AHP, MQTT / VDA and start-up logs are **only in the file**, not in the database. The business notices for the client (`line.done`, `line.error`…) are not logs: they are in `sw_link_events`.

9.5 Maintenance

- Events and log are purged by themselves after 30 days (`EventRetentionDays`, `LogRetentionDays`).
- The `sw_link_*` tables live inside the SoftWinding database: the usual backup includes them.
- To update the Link: stop the service, replace the executables, press **Install or update the Link schema…** and start the service.

10. Simulators

To integrate and test without a real machine there are three programs that stand in for the two ends of the Link: the machine and the customer system. They use the same protocols as the real equipment. Both windows speak six languages (`--lang=es|en|de|fr|it|pt`, or `[Window] Lang` of their ini; when changed from the language list it applies when the window is opened again).

Table 36. Simulators.

Program	What it does
`SoftWindingLinkSim.exe`	The simulated machine in a console: FINS over UDP 9600 and SFW over TCP 63000, with an automatic operator, NB screen, emergency stop, guard and a simulated Zebra printer. All the options are in section 10.1.
`SoftWindingLink_MachineSimulator.exe`	The same machine with a window (section 10.1).
`SoftWindingLink_ServerSimulator.exe`	The customer system (ERP/WMS) with a window (section 10.2).

10.1 The simulated machine

It is an Omron NX1 PLC (FINS/UDP) and an SFW PLC (TCP) in the same program, with the counter DM 2000 (with 2 decimals, 57 m = 5700), the preselection DM 2005, the cuts (DM 2010 and DM 2260), the reset, the run state, the panel of the received order (texts and figures), the measure of the last cut (DM 2002 and DM 2026) and the link flag `HmiActive` (DM 2698, DM 2699). The window has the counter and the run, stop, counter reset, "reach the measure" and cut buttons, and a label with the link state: **SOFTWINDING DISCONNECTED** (DM 2698 = 0), **CONNECTED** "no order data" (1) or **TRANSFERRING** "orders on the screen" (2). For the screen mode it has the Previous (10), Next (20), Prepare (90) and Confirm (100) buttons.

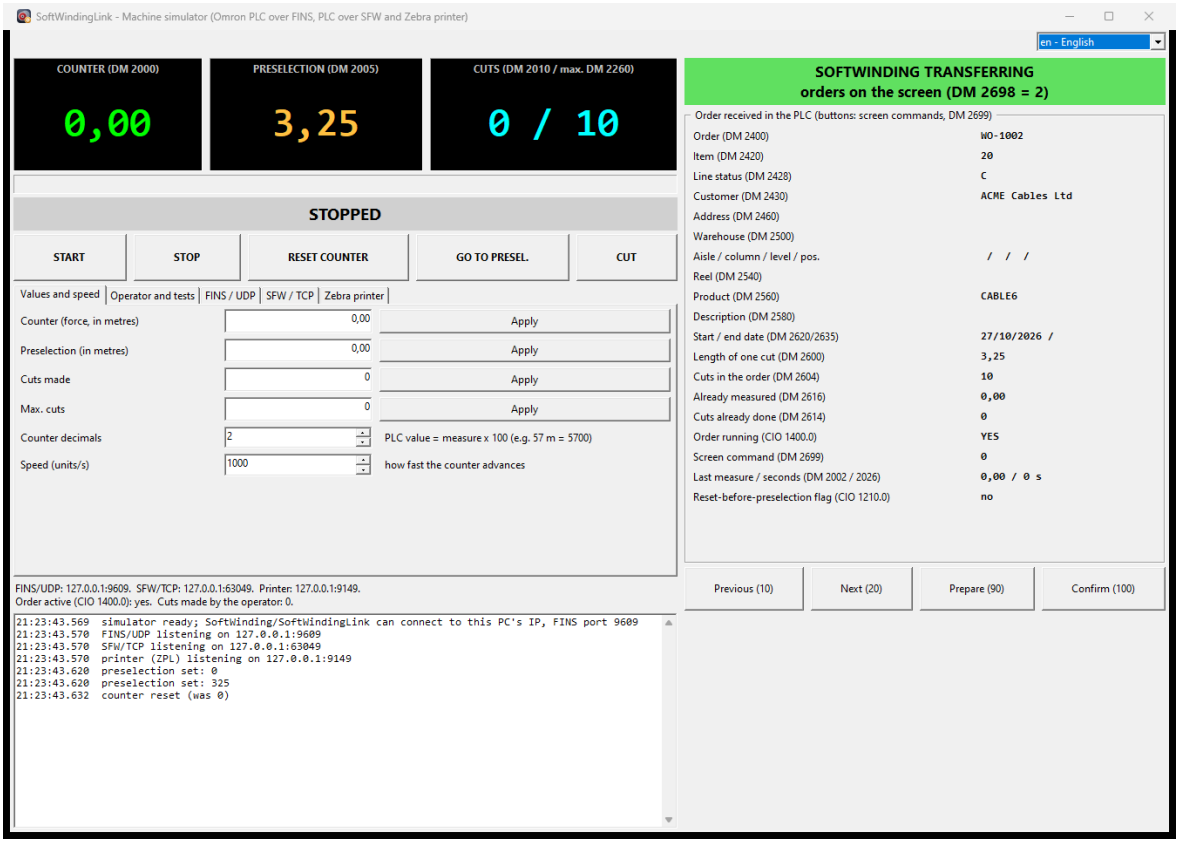


Figure 21. The machine simulator with an order received from the Link. The configuration area is a tab control.

Table 37. Tabs of the machine simulator window.

Tab	What is configured
Values and speed	Counter, preselection, cuts done and maximum, decimals and speed (units per second).
Operator and tests	Emergency stop, guard, automatic operator, "the PLC counts the cut", "the screen presses next" and the **test cases**: overshoot (`--overshoot`), waste at N units (`--waste`) and how many times (`--wastes`), in counter units.
FINS / UDP	**Active** box (starts or stops the server at once), listening IP, port, **Apply IP and port**, **State** (listening on IP:port / disabled / error with the reason, for example port in use) and **Activity** (requests and who sent the last one).
SFW / TCP	The same; the activity counts requests and connected clients. When disabled, open connections are closed.
Zebra printer	The same, with the labels received, the file where it stores the ZPL and the simulated state (ready, out of paper, paused). It answers `^HH`, `~JC` (the length goes from 1200 to 1218 on each calibration), `~WC`, `~JG`, `~PH` and `^JUS`, so the calibration can be tested (section 3.7).

Table 38. Machine simulator options (console and window).

Option	Effect
`--listen=IP --fins-port=9600 --sfw-port=63000`	Listening address and ports.
`--printer-port=9100 --printer-file=path`	Simulated Zebra printer and file where it stores what it receives (in the window it only starts active with `--printer-port`).
`--no-fins --no-sfw --tab=N`	Window: starts with FINS or SFW off; initial tab (0 to 4).
`--speed=2000 --decimals=2`	Speed (units per second) and counter decimals.
`--auto --start-delay=300 --cut-delay=500`	Automatic operator (starts, waits, cuts) and its delays in ms.
`--hmi-next`	After cutting, the screen presses "next" (D2699 = 100).
`--hmi-work`	With an order on the screen not yet activated, presses "work with the order" (D2699 = 90): tests the screen mode.
`--plc-counts-cuts`	The reset with the counter at the preselection adds a cut (`DM 2010`).
`--plc-auto-cuts`	The PLC cuts by itself, counts the cuts up to `CutsMax` and stores the measure (`DM 2002`) and the seconds (`DM 2026`): cut mode `R`.
`--overshoot=N`	Units it goes past the preselection before stopping.
`--waste=N --wastes=K`	The cut is interrupted at N units and the operator resets (waste; example flag `CIO 1210.0`); K times (1 by default).
`--text-base=6000`	The order block (texts and panel) starts at DM 6000: simulates the `OMRON_NX` profile (2400 by default, `OMRON_CP`).
`--emergency`	Starts with the emergency stop active.
`--cmdfile=path`	Console: reads commands from a file and deletes it (`emerg on|off`, `guard open|close`, `printer ok|nopaper|pause`, `hmi 10|20|90|100`).
`--run=SEC --quiet --noconsole`	Console: ends by itself, without traces, without waiting for Enter.
`--lang=es|en|de|fr|it|pt --ini=path`	Window: language and ini file (`SoftWindingLink_MachineSimulator.ini`, next to the exe).

10.2 The simulated customer system

`SoftWindingLink_ServerSimulator.exe` acts as an ERP or WMS: it sends orders through the DB mailbox, REST, TCP (AHP) or MQTT / VDA 5050 and shows the answers and the status of the lines. The **blue header** shows the client, the channel and whether the database is connected; the **Help** button opens a quick guide

and the **help bar** below explains what is under the mouse. The connection is configured in tabs (*Client and channel*, *DB (mailbox)*, *REST / JSON*, *TCP (AHP)* and *MQTT / VDA 5050*); the channel chosen for sending carries a dot (●). **Prepare client in the DB** registers the client and its token; **Test connection** tests the database and only the chosen channel, and **Test all channels** the four; the traffic shows what is sent and received.

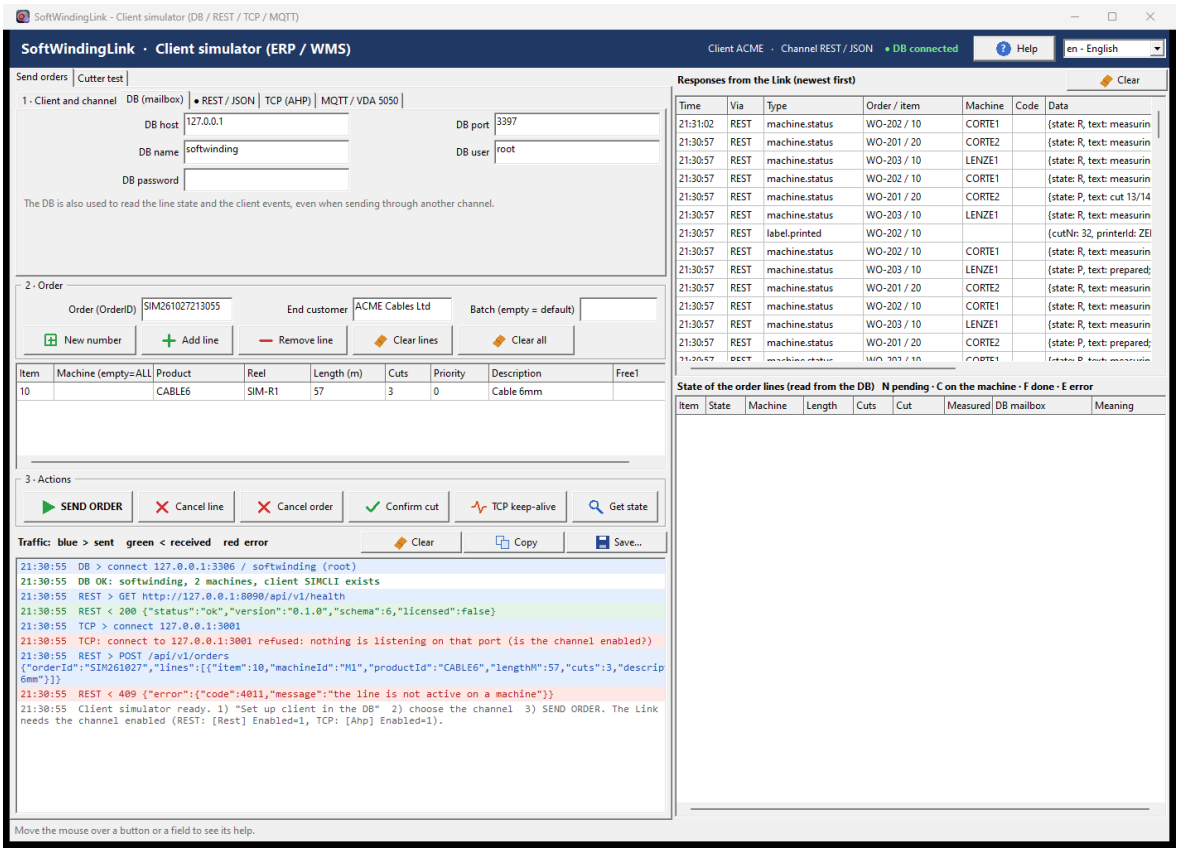


Figure 22. The customer simulator, "Send orders" tab. The traffic is colored: blue sent, green received, red error.

- **Send orders:** the order lines (new number, add, remove; *Clear lines* and *Clear all*), **Send**, **View status** and **Cancel**. The Link answers are green (`line.done`), red (`line.error`, HTTP 4xx/5xx), orange (`line.canceled`) and yellow (`line.waste`); the line status table fills in by itself with the meaning (pending, on the machine, done, error). The traffic can be cleared, copied to the clipboard or saved to a file. None of this deletes data from the database: it only empties the lists of the window.
- **Cutter test:** choose the machine (the list comes from `sw_link_machines` with its state), the product, the length and the cuts and press **Start cut test**: it sends a one-line `CUT…` order and ticks six steps with ✓ (order accepted, line claimed, PLC prepared, measuring, measure reached and line finished). It shows the machine state (red warning if the Link gives no heartbeat in 30 s), the live counter and the cuts done. During the test: *Confirm cut*, *Show / Hide cut screen*, *Cancel what is on the screen*, *Cancel test* and *New test*. There are also the **Cable diameter** box (*Send diameter*, *Use laser gauge*, *Query diameter*) and the **Time and speed** box (*Query time and speed*), all over REST. With a real machine it **writes to its PLC**.

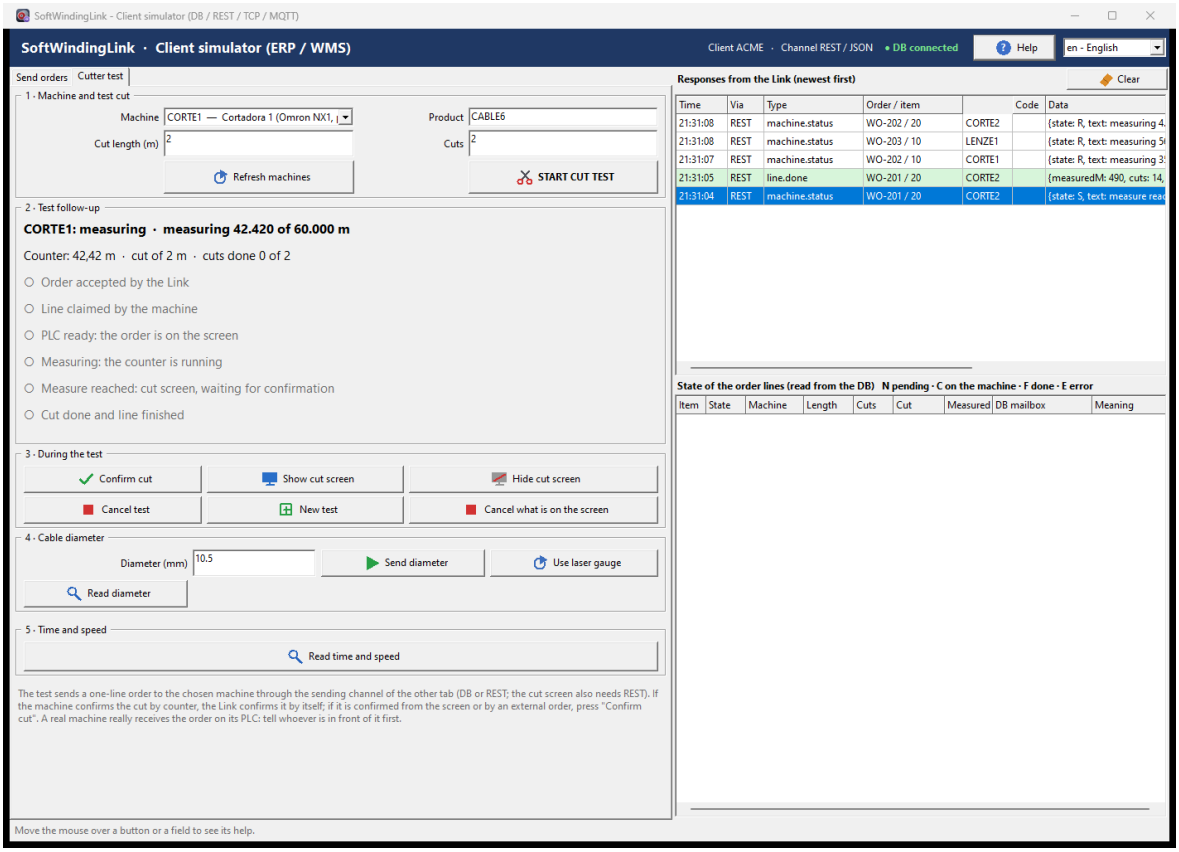


Figure 23. The customer simulator, "Cutter test" tab.

Table 39. Customer simulator options.

Option	Effect
`--lang=es|en|de|fr|it|pt --ini=path`	Language and ini file (`SoftWindingLink_ServerSimulator.ini`, next to the exe).
`--client=ID --main=0|1 --tab=N --machine=ID`	Initial client; main tab (0 send orders, 1 cutter test); connection tab (0 to 4); machine of the cutter test.
`--selftest=db|rest|tcp|cutter|clear`	Unattended self-test; the result goes to `--result=` (`selftest.txt` by default).

Warning. The ini of the customer simulator stores the database connection (by default `root` without a password on port 3306). Check that it points to a test database before sending anything: **with a real machine enabled in that database, the orders reach the machine**.

For an end-to-end test, start the machine simulator, define a Link machine with its address (section 4) and send orders with the client simulator. Always use a test database. To move to a real machine, define it as explained in chapter 4 and follow the commissioning steps (section 4.5).

Annex A. Configuration (SoftWindingLink.ini)

Table 40. Ini file keys.

Section	Key	Default and meaning
`[Database]`	`Host, Port, Name, User`	MariaDB connection.
`[Database]`	`Pass / PassEnc`	Clear password (first time only) / DPAPI-encrypted.
`[Database]`	`Library, PoolMax`	`libmariadb.dll`; maximum connections (16).
`[Service]`	`Tick`	300: ms between cycles of each machine.
`[Service]`	`HeartbeatSec, OwnerLeaseSec`	5 and 30: machine heartbeat and time after which another instance may take it.
`[Service]`	`ConfirmTimeoutMin, ClaimTimeoutMin`	30 and 30: warning without confirmation and release of a line without PLC.
`[Service]`	`MachineReloadSec`	10: how often the machines table is reread.
`[Fins]`	`BitMode`	`rmw` (read-modify-write the word) or `bit` (bit command).
`[Ahp]`	`Enabled, ListenIP, ListenPort`	0, 0.0.0.0, 3001.
`[Rest]`	`Enabled, ListenIP, Port`	0, 0.0.0.0, 8090.
`[Labels]`	`LabelDir, RetryCount, RetrySec`	`labels`, 5, 30.
`[License]`	`Licensed, UnLock`	empty: DEMO mode. Company name and unlock code (section 2, step 4).
`[Stock]`	`Control, CreateProducts, Warehouse`	`Control` 0; `CreateProducts` same as Control; `Warehouse` -1. Stock control (movements, `sw_stock` deduction, reel entries), creation of missing products and warehouse of new reels (section 7.2).
`[Inbox]`	`Enabled, PollMs`	1, 1000: DB mailbox enabled and how often in ms it is checked (minimum 200).
`[Files]`	`Enabled, MappingFile`	0 and `<Link folder>\data\LinkFiles.xml`: file channel and mapping file.
`[Files]`	`PollSec, WaitSec, StableSec`	5, 120, 2: how often in seconds it checks the folders, how long it waits for the Link answer and how long a file must be unmodified before it is read.
`[Mqtt]`	`Enabled, Host, Port, Tls, Verify`	0, empty, 8883, 1, 1: MQTT / VDA 5050 channel (section 6.7).
`[Mqtt]`	`CAFile, CertFile, KeyFile, KeyPassword`	PEM certificates for TLS (CA, client and private key).
`[Mqtt]`	`User, Password, ClientIdPrefix`	MQTT server user and password and prefix of the ClientID of each machine.
`[Mqtt]`	`InterfaceName, MajorVersion, Manufacturer, ClientID`	`uagv`, `v2`, `LOIMEX`, `MQTT`.
`[Mqtt]`	`KeepAlive, StateSeconds`	30 and 30.
`[Logs]`	`LogDir, EventRetentionDays, LogRetentionDays`	`logs`, 30, 30.
`[Window]`	`Lang`	`es`: language of the administration window (`es`, `en`, `de`, `fr`, `it`, `pt`); also `--lang=`.

Annex B. Command line

Table 41. Commands.

Command	Effect
`SoftWindingLinkSvc.exe --install / --uninstall`	Installs or uninstalls the service (as administrator).
`SoftWindingLinkSvc.exe --console [--run=SECONDS] [--noconsole] [--ini=path]`	Runs the engine in a console (Enter to stop; `--run` to stop by itself; `--noconsole` not to wait for Enter).
`SoftWindingLinkSvc.exe --newclient=ID [--channels=REST,TCP,MQTT] [--dialect=L|T] [--host=IP --port=N]`	Registers a client (or renews its token) and writes the token once. Equivalent to the **New** button of Clients.
`SoftWindingLinkSvc.exe --license [--ini=path]`	Shows the machine ID and whether the license is valid or in DEMO mode.
`SoftWindingLinkSvc.exe --checkfiles [--ini=path]`	Reads the file channel mapping file and lists its `Import` and `Export` elements, or says what is wrong (exit code 1).
`SoftWindingLink.exe`	The administration window (options in section 3.1).

Annex C. OMRON_CP and OMRON_NX profiles

Profiles installed with the Link. `Inv.` = inverted logic (bit at 0 = active). `OMRON_CP` is the map of the CP series PLCs; `OMRON_NX` is the same except for the block of order texts and panel, which starts at DM 6000 (next table). Everything can be corrected in the **PLC profiles** tab.

Table 42. Common parameters of OMRON_CP and OMRON_NX.

Parameter	Address	Notes
`Counter`	`DM 2000 (D)`	measurement counter
`Preselection`	`DM 2005 (D)`	preselection, in units of 10^-n m
`CutNr`	`DM 2010 (W / D)`	cut number (read: word; write: double word)
`CutsMax`	`DM 2260 (W)`	total cuts of the line
`LastCutLen`	`DM 2002 (D)`	mode `R`: `Counter_Last_Mts`, length of the last cut, stored by the PLC before adding 1 to `CutNr`
`LastCutSecs`	`DM 2026 (D)`	mode `R`: `Counter_Last_timer`, seconds the last cut took (optional; only noted in the Log)
`ShortReset`	`(no row)`	PLC bit that flags a reset before the preselection (waste). No default row: it is added when the PLC has it (section 4.2)
`CutScreen`	`DM 1508.03 (B)`	cut screen: 1 = measuring is done (optional)
`CableDiam / CableSet`	`DM 2326 (W) / DM 2322 (W)`	current cable diameter / diameter the PC sends in manual mode; `Size` = decimals (optional)
`CableManual`	`DM 2342.02 (B)`	`opt_rel_mm_user`: 1 = manual diameter, 0 = laser gauge (optional)
`ProcTime / Speed`	`DM 2030 (T 3) / DM 5491 (W)`	process time `mm:ss` / speed in m/min, `Size` = decimals (optional)
`Start`	`CIO 1201.7`	start from the PC
`ResetCounter`	`CIO 1201.15`	counter reset
`HydraulicCut`	`CIO 1200.11`	hydraulic cut: the Link does NOT drive it
`Run`	`CIO 1205.3 (Inv.)`	bit at 0 = running
`SlowSpeed / Hydraulic / Guard`	`CIO 1205.4 / .9 / .10`	slow speed / hydraulic active / guard closed
`Emergency`	`CIO 1205.15 (Inv.)`	bit at 1 = no emergency
`HmiActive`	`DM 2698`	SoftWinding order: 0 disconnected, 1 connected with no order data, 2 orders on the screen (section 4.6)
`HmiCommand`	`DM 2699`	screen commands: 10 previous, 20 next, 90 work with the order, 100 finish
`OrderRunning`	`CIO 1400.0`	order in progress
`Wake`	`CIO 205.10`	wakes the screen

Table 43. Block of order texts and panel: the only difference between the two profiles.

Parameter	OMRON_CP	OMRON_NX	Size
`TxtOrder`	`DM 2400`	`DM 6000`	text, 10 words
`TxtItem`	`DM 2420`	`DM 6020`	text, 6
`TxtStatus`	`DM 2428`	`DM 6028`	text, 2
`TxtCustomer`	`DM 2430`	`DM 6030`	text, 30

Parameter	OMRON_CP	OMRON_NX	Size
TxtAddress	DM 2460	DM 6060	text, 30
TxtWarehouse	DM 2500	DM 6100	text, 20
TxtAisle / TxtColumn / TxtLevel / TxtPosition	DM 2524 / 2528 / 2532 / 2536	DM 6124 / 6128 / 6132 / 6136	text, 4 each
TxtReel	DM 2540	DM 6140	text, 20 (reel identifier)
TxtProduct	DM 2560	DM 6160	text, 20
TxtDescription	DM 2580	DM 6180	text, 20
ToCutLen / ToCutCount	DM 2600 (D) / DM 2604	DM 6200 (D) / DM 6204	length of one cut and number of cuts
DoneCount / DoneLen	DM 2614 / DM 2616 (D)	DM 6214 / DM 6216 (D)	cuts done and measured
TxtDateStart / TxtDateEnd	DM 2620 / DM 2635	DM 6220 / DM 6235	text, 10 each

Annex D. sw_link_* tables

Table 44. Link tables. The fields of each one are in sql\create_sw_link.sql.

Table	Purpose
sw_link_meta	Link schema version (currently 6).
sw_link_machines	One row per machine: configuration and live state (LiveInfo: time, speed and cable).
sw_link_profiles	Memory address profiles.
sw_link_printers	Zebra printers.
sw_link_clients	TCP/REST clients and their event cursor.
sw_link_inbox	Order mailbox of the DB and file channels: one row per line or request, with its answer.
sw_link_orders	Orders received (all channels): customer reference and SoftWinding keys, idempotency and per-line cable diameter (CableMm).
sw_link_claims	Lines claimed by a machine.
sw_link_events	Event queue to the clients.
sw_link_commands	External commands to a machine (confirm, cancel, cancel screen, cut screen, cable, time and speed).
sw_link_labels	Label queue, one per cut.
sw_link_cuts	One confirmed cut per row, always: production of the window and web report, and recovery after a crash.
sw_link_filestate	Event cursor of each Export of the file channel.
sw_link_vda	MQTT / VDA 5050 channel: one row per cut action received, with its actionId, node and result.
sw_link_waste	Wastes: operator reset before the preselection (invalid measure), one row per waste with the measure, the preselection and the seconds.
sw_link_log	Log of telegrams, calls and actions.
sw_link_v_orders, sw_link_v_cuts, sw_link_v_waste, sw_link_v_machines	Query views for database clients (sql\create_sw_link_views.sql, section 6.3): status of the lines, cuts, wastes and machine state.