Database Mailbox Integration Guide

Send orders to the cutting machines by writing rows in a database table, and read the results with SQL

Interface	Database Mailbox
Document version	1.1
Date	October 2026

Contents

1. Introduction 4
1.1 What the mailbox is 4
1.2 How it works 4
1.3 What you need from the Link administrator 5
1.4 Conventions in this guide 5
2. Quick start 6
3. The table sw_link_inbox 8
3.1 Columns 8
3.2 Status of a row 9
3.3 How the Link processes the rows 9
4. Sending orders 10
4.1 One line 10
4.2 An order with several lines: one statement 10
4.3 A draft: lines written one by one 10
4.4 Which machine and in what order 11
4.5 Repeating and changing a line 11
4.6 Characters 12
5. Reading answers and results 13
5.1 The answer of your row 13
5.2 The state of the lines: sw_link_v_orders 13
5.3 Events: sw_link_events 13
5.4 The cursor 14
5.5 Other views 15
6. Cancel, confirm and stock entries 16
6.1 Cancel a line or an order (Action X) 16
6.2 Confirm a cut (Action K) 17
6.3 Stock entry (Action S) 17
7. Errors 19
8. Security and permissions 21
9. Debugging and troubleshooting 22
9.1 A method: follow the order step by step 22
9.2 Where the Link leaves its traces 22
9.3 My line does not start 23
9.4 Common problems with this interface 24
9.5 Tracing the traffic 24

9.6 What to send to LOIMEX support 25
10. Testing without a machine 26
10.1 Set up the test machine 26
10.2 Useful simulator options 27
11. Going to production 28
Appendix A. Example client (Python) 29
Appendix B. SQL cookbook 31

1. Introduction

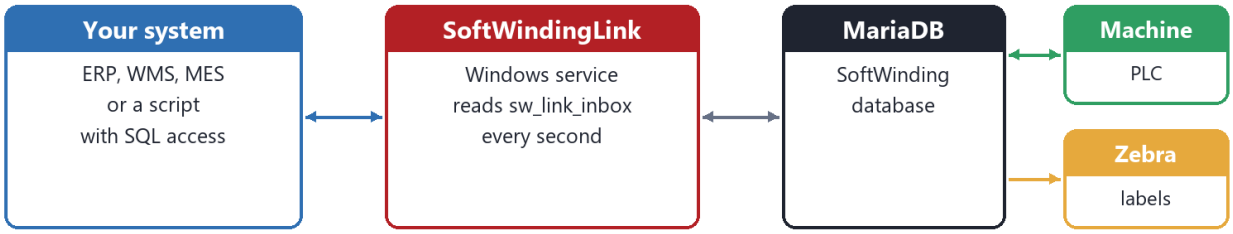
1.1 What the mailbox is

The **database mailbox** is the simplest way to connect a system that already talks SQL. Your system does not call a service and does not open a protocol: it **inserts rows** in one table, `sw_link_inbox`, and **reads the answers** with `SELECT`. SoftWindingLink reads the table every second, validates what it finds, creates the orders for the machines and writes its answer in the same row.

- **One place to write:** `sw_link_inbox` (new orders, cancellations, confirmations, stock entries).
- **Places to read:** the answer in your own row of `sw_link_inbox`; the state of each line in the view `sw_link_v_orders`; each cut in `sw_link_v_cuts`; wastes in `sw_link_v_waste`; machine state in `sw_link_v_machines`; and the history of what happened in `sw_link_events`.
- **No SoftWinding table is touched.** Your database user has no rights on any `sw_*` table of SoftWinding nor on the internal tables of the Link; it only gets the mailbox, the events and four read-only views.

Anything that can run an `INSERT` can use it: an ERP with a database link, a PLC gateway, a stored procedure, a Python script, a scheduled SQL job. Chapter 2 does a complete cycle with only SQL statements.

1.2 How it works



1. Your system writes rows into sw_link_inbox with an INSERT (Status N, or I for a draft).
2. Every second the Link claims the new rows, validates them together and creates the orders.
3. The Link writes its answer in the same rows: Status D (done) or E (rejected), with Code and Result.
4. A machine takes each line, the operator cuts, the Link records every cut and prints the labels.
5. You read the progress in sw_link_v_orders and the news in sw_link_events.

Figure 1. The path of an order through the database mailbox.

> **Important. It is asynchronous.** The `INSERT` only puts a request in the mailbox. The answer ("accepted" or "rejected") appears in the row within about a second, and the cut happens later, when the operator works on the line. Never wait for the cut inside your transaction: write, then read.

1.3 What you need from the Link administrator

Table 1. Data to ask the administrator for.

You need	Example	Where the administrator finds it
Database server, port and name	`link-server`, `3306`, `softwinding`	The MariaDB server that hosts the SoftWinding database.
A database user and password	`acme` / `********`	Created once with `sql\create_sw_link_client_user.sql` (one user per customer system).
Your ClientID	`ACME`	Administration window, **Clients** tab: the client must exist, be enabled and have the **DB** channel. You write this value in the `ClientID` column.
Machine identifiers	`M1`, `M2`	Administration window, **Machines** tab. You can also read them in the view `sw_link_v_machines`.
Behaviour of each machine	confirm by counter, from the screen or external	Machines tab. It decides when a cut counts as done and whether you must confirm it (section 6.2).

The mailbox is **on by default** (`[Inbox] Enabled=1`, polling every `PollMs` = 1000 ms). The Link engine must be running: if it is not, your rows stay in `N` and nothing happens.

1.4 Conventions in this guide

- Statements and results are shown as they were run in the **MariaDB command-line client** against a test system with two simulated machines (`M1`, `M2`) and a client `SHOP`. Your identifiers will differ.
- All the examples are in `docs\examples\inbox`, written in Python (`inbox_client.py`), Node.js (`inbox_client.js`) and C# (`InboxClient.cs`), and also in `docs\examples\inbox_examples.sql`.
- Lengths are in **metres** with up to 3 decimals; one line is *N cuts of L metres of product P*.

2. Quick start

Open a MariaDB client with the user the administrator gave you and run the six steps below. If you have no real machine, set up the simulator first (chapter 9).

```
mariadb -h link-server -P 3306 -u acme -p softwinding
```

2.1 Step 1. Send an order

One `INSERT` is one line. `ClientID` is yours; `Action` and `Status` take their defaults (`C` create, `N` new).

```
INSERT INTO sw_link_inbox (ClientID, OrderID, Item, MachineID, ProductID, ReelID, LengthM,
    Cuts, Customer, Description)
VALUES ('SHOP', 'PO-5001', 10, 'M1', 'H07V-K-1.5-BK', 'R-77881', 12.5, 3,
    'ACME Cables Ltd', 'H07V-K 1.5 mm2 black');

Query OK, 1 row affected
```

2.2 Step 2. Read the answer

A moment later the Link has processed the row. Right after the `INSERT` you still see `Status = N` and empty `Result`:

```
SELECT InboxID, Action, Status, OrderID, Item, MachineID, ProductID, LengthM, Cuts, Code,
    Result, Assigned
FROM sw_link_inbox
WHERE OrderID='PO-5001'
ORDER BY InboxID;

+---------+--------+---------+------+------+--------+----------+
| InboxID | Status | OrderID | Item | Code | Result | Assigned |
+---------+--------+---------+------+------+--------+----------+
| 1       | N      | PO-5001 | 10   | 0    |        |          |
+---------+--------+---------+------+------+--------+----------+
1 row in set
```

About a second later:

```
+---------+--------+---------+------+------+----------+----------+
| InboxID | Status | OrderID | Item | Code | Result   | Assigned |
+---------+--------+---------+------+------+----------+----------+
| 1       | D      | PO-5001 | 10   | 0    | accepted | M1       |
+---------+--------+---------+------+------+----------+----------+
1 row in set
```

`Status D` means the Link accepted it, `Result` says `accepted` and `Assigned` is the machine offered for it.

2.3 Step 3. Follow the line

```
SELECT OrderID, Item, Status, MachineID, CutsDone, Cuts, MeasuredM, StartTime, EndTime
  FROM sw_link_v_orders WHERE ClientID = 'SHOP' ORDER BY OrderID, Item;
```

```
+---------+------+--------+-----------+------+----------+-----------+---------------------+
| OrderID | Item | Status | MachineID | Cuts | CutsDone | MeasuredM | EndTime             |
+---------+------+--------+-----------+------+----------+-----------+---------------------+
| PO-5001 | 10   | done   | M1        | 3    | 3        | 37.500    | 2026-10-28 12:59:41 |
| PO-5002 | 10   | done   | M2        | 2    | 2        | 16.800    | 2026-10-28 12:59:32 |
| PO-5002 | 20   | done   | M2        | 2    | 2        | 12.000    | 2026-10-28 12:59:43 |
```

```
+---------+------+--------+-----------+------+----------+-----------+--------------------+
3 rows in set
```

`status` goes `pending` -> `claimed` (a machine has it) -> `done`. `MeasuredM` is what the machine really measured: 3 cuts of 12.5 m are 37.5 m.

2.4 Step 4. Read every cut

```
+---------+------+-------+-----------+---------+-----------+---------------------+
| OrderID | Item | CutNr | MachineID | ReelID  | MeasuredM | Created             |
+---------+------+-------+-----------+---------+-----------+---------------------+
| PO-5001 | 10   | 1     | M1        | R-77881 | 12.500    | 2026-10-28 12:59:27 |
| PO-5001 | 10   | 2     | M1        | R-77881 | 12.500    | 2026-10-28 12:59:34 |
| PO-5001 | 10   | 3     | M1        | R-77881 | 12.500    | 2026-10-28 12:59:41 |
| PO-5002 | 10   | 1     | M2        |         | 8.400     | 2026-10-28 12:59:26 |
+---------+------+-------+-----------+---------+-----------+---------------------+
4 rows in set
```

2.5 Step 5. Read the events

```
+---------+-----------+-----------+------------+------+---------+
| EventID | Type      | MachineID | DeliveryID | Item | ReelID  |
+---------+-----------+-----------+------------+------+---------+
| 67      | line.done | M2        | PO-5002    | 10   |         |
| 72      | line.done | M1        | PO-5001    | 10   | R-77881 |
| 75      | line.done | M2        | PO-5002    | 20   |         |
+---------+-----------+-----------+------------+------+---------+
3 rows in set
```

The `Payload` column is JSON text. For `line.done`, for example:

```
{"measuredM":37.500,"cuts":3,"endTime":"2026-10-28T12:59:41"}
```

`line.done` is the event your system waits for: it carries the metres measured and the end time.

2.6 Step 6. Mark the events as read

```
UPDATE sw_link_clients
SET LastEventID = 20
WHERE ClientID = 'SHOP';

Query OK, 1 row affected
```

You can keep the cursor in your own system instead; section 5.4 explains both ways.

3. The table sw_link_inbox

3.1 Columns

Table 2. Columns of sw_link_inbox.

Column	Written by	Meaning
`InboxID`	database	Automatic row number. Use it to find your answer.
`ClientID`	you	Your client identifier (up to 30). Required.
`Action`	you	`C` create or change a line (default), `X` cancel, `K` confirm a cut, `S` stock entry (section 6).
`Status`	you, then the Link	`N` new (default), `I` draft. The Link changes it to `W`, `D` or `E` (next table).
`OrderID`	you	Your order number (up to 30). Required.
`Item`	you	Line number in the order, 1 to 99999. Default 1. For `X`, 0 means the whole order.
`BatchID`	you	Optional work package (up to 30). Default `<ClientID>_<yyyymmdd>`.
`Customer`	you	Customer name (up to 50).
`MachineID`	you	Machine that must cut the line. Empty or `ALL`: any machine that accepts general lines.
`ProductID`	you	Product code (up to 50). Required for `C` and `S`.
`ProviderID`	you	Supplier, only for the stock entry (`S`).
`ReelID`	you	Reel (up to 20): label and traceability. Required for `S`.
`LengthM`	you	Length of **one** cut, in metres (> 0, up to 99999.999). For `S`: the stock of the reel.
`Cuts`	you	Number of cuts of that length. Default 1.
`Priority`	you	Lower goes first. Default 0.
`Description`, `Notes`	you	Text for the label and the machine screen (100) / free note (500).
`Free1` ... `Free10`	you	Free fields (100) for the label template.
`Created`	you or the Link	When the row was written. If you leave it empty the Link fills it when it reads the row.
`Code`	the Link	0 if it went well, otherwise the error code (section 7).
`Result`	the Link	The answer in words: `accepted`, `already received`, `canceled`, `requested`... or the reason of a rejection.
`Assigned`	the Link	The machine the line was offered to.
`Instance`, `Processed`	the Link	Which Link instance handled the row and when. For diagnosis.

3.2 Status of a row

Table 3. Row status.

Status	Meaning	Set by
`I`	**Draft.** The Link ignores it. Use it for an order written row by row; release it with `UPDATE ... SET Status = 'N'` (section 4.3).	you
`N`	**New.** Waiting for the Link (default).	you
`W`	**Working.** An instance of the Link has claimed the row. If it stays like this for more than 60 seconds (the instance died) another instance puts it back to `N`.	the Link
`D`	**Done.** Processed correctly. `Result` is `accepted`, `already received`, `canceled` or `requested`.	the Link
`E`	**Rejected.** `Code` is the error and `Result` the reason (section 7). Nothing was written for this row.	the Link

3.3 How the Link processes the rows

1. Every second it looks for rows in `N` and **groups them by (`ClientID`, `OrderID`, `Action`)**.
2. It **claims** each group in one atomic step (`N` -> `W`). If several Link instances run, only one gets it.
3. It checks that your **client exists, is enabled and has the DB channel**. If not, every row of the group is rejected with 4012.
4. For `C` it builds one order from all the rows of the group and validates **all lines together**: if one is invalid, **none** is written and the others are rejected with the order code and the text `not written: another line of the order was rejected`.
5. It writes the answer in each row (`D` or `E`, `Code`, `Result`, `Assigned`) and a line in the Link log.

> **Warning.** **Rows of one order written in separate statements are processed separately.** The Link does not wait for "the rest of the order". If you insert line 10, then line 20 two seconds later, line 10 may already be on the machine, even finished, when line 20 arrives. For an order that must appear complete, write all its lines in **one** `INSERT`, or write them as drafts (`I`) and release them together.

Rows in `D` and `E` are purged after `EventRetentionDays` (30 by default): read your answers before.

4. Sending orders

4.1 One line

The minimum is `ClientID`, `OrderID`, `ProductID`, `LengthM` and, for a given machine, `MachineID`. With a reel and texts for the label:

```
INSERT INTO sw_link_inbox (ClientID, OrderID, Item, MachineID, ProductID, ReelID, LengthM,
    Cuts, Customer, Description)
VALUES ('SHOP', 'PO-5001', 10, 'M1', 'H07V-K-1.5-BK', 'R-77881', 12.5, 3,
    'ACME Cables Ltd', 'H07V-K 1.5 mm2 black');

Query OK, 1 row affected
```

4.2 An order with several lines: one statement

Put all the lines in **one** `INSERT`. They are processed as one order, all or nothing:

```
INSERT INTO sw_link_inbox (ClientID, OrderID, Item, MachineID, ProductID, LengthM, Cuts,
    Description)
VALUES ('SHOP', 'PO-5002', 10, 'M2', 'H07V-K-1.5-BU', 8.4, 2, 'H07V-K 1.5 mm2 blue'),
       ('SHOP', 'PO-5002', 20, 'M2', 'H07V-K-2.5-GY', 6.0, 2, 'H07V-K 2.5 mm2 grey');

Query OK, 2 rows affected
```

```
+---------+--------+---------+------+------+----------+----------+
| InboxID | Status | OrderID | Item | Code | Result   | Assigned |
+---------+--------+---------+------+------+----------+----------+
| 2       | D      | PO-5002 | 10   | 0    | accepted | M2       |
| 3       | D      | PO-5002 | 20   | 0    | accepted | M2       |
+---------+--------+---------+------+------+----------+----------+
2 rows in set
```

4.3 A draft: lines written one by one

When your system cannot write all the lines in one statement, write each one with `Status = 'I'`. The Link ignores drafts. When the last line is in, release them all with one `UPDATE`:

```
INSERT INTO sw_link_inbox (ClientID, OrderID, Item, Status, MachineID, ProductID, LengthM,
    Cuts)
VALUES ('SHOP', 'PO-5003', 10, 'I', 'M1', 'P-A', 3, 2);

Query OK, 1 row affected
```

```
INSERT INTO sw_link_inbox (ClientID, OrderID, Item, Status, MachineID, ProductID, LengthM,
    Cuts)
VALUES ('SHOP', 'PO-5003', 20, 'I', 'M1', 'P-B', 4, 2);

Query OK, 1 row affected
```

```
+---------+--------+---------+------+------+--------+
| InboxID | Status | OrderID | Item | Code | Result |
+---------+--------+---------+------+------+--------+
| 4       | I      | PO-5003 | 10   | 0    |        |
| 5       | I      | PO-5003 | 20   | 0    |        |
```

```
+---------+--------+---------+------+------+--------+
2 rows in set
```

```
UPDATE sw_link_inbox
SET Status = 'N'
WHERE ClientID='SHOP' AND OrderID = 'PO-5003' AND Status = 'I';

Query OK, 2 rows affected
```

```
+---------+--------+---------+------+------+----------+----------+
| InboxID | Status | OrderID | Item | Code | Result   | Assigned |
+---------+--------+---------+------+------+----------+----------+
| 4       | D      | PO-5003 | 10   | 0    | accepted | M1       |
| 5       | D      | PO-5003 | 20   | 0    | accepted | M1       |
+---------+--------+---------+------+------+----------+----------+
2 rows in set
```

4.4 Which machine and in what order

- With `MachineID`, the line goes to that machine only (it must exist, be enabled and be allowed to your client).
- Empty or `ALL`: the line goes to any free machine that accepts general lines. The answer shows the offered machine in `Assigned`.
- Each machine takes **one line at a time**, in this order: batch priority, then `Priority` (lowest first), then `Item`, then order number.
- The Link prepares the machine; **the operator starts it**. A line can stay "prepared" as long as nobody starts the machine.

4.5 Repeating and changing a line

The key of a line is (`ClientID`, `OrderID`, `Item`). Sending the same line again is safe:

Table 4. Writing a line again.

You write	Row result	Effect
A new line	`D, accepted`	The line is created.
The same line, same content	`D, already received`	Nothing changes.
The same line with other content, **not yet taken by a machine**	`D, accepted`	The line is updated.
The same line with other content, **already taken or finished**	`E`, 4003	Rejected: the line stays as it was.

```
+---------+--------+---------+------+------+------------------+----------+
| InboxID | Status | OrderID | Item | Code | Result           | Assigned |
+---------+--------+---------+------+------+------------------+----------+
| 1       | D      | PO-5001 | 10   | 0    | accepted         | M1       |
| 13      | D      | PO-5001 | 10   | 0    | already received | M1       |
+---------+--------+---------+------+------+------------------+----------+
2 rows in set
```

```
+---------+--------+---------+------+------+--------------------------------+----------+
| InboxID | Status | OrderID | Item | Code | Result                         | Assigned |
+---------+--------+---------+------+------+--------------------------------+----------+
| 1       | D      | PO-5001 | 10   | 0    | accepted                       | M1       |
| 13      | D      | PO-5001 | 10   | 0    | already received               | M1       |
```

```
| 14      | E      | PO-5001 | 10   | 4003 | duplicate or unchangeable order |          |
+---------+--------+---------+------+------+---------------------------------+----------+
3 rows in set
```

4.6 Characters

The database is Latin-1. Western European accents are kept; characters outside Latin-1 in free texts (`Customer`, `Description`, `Notes`, `Free1`...) become `?`. In the **identifiers** (`OrderID`, `ProductID`, `ReelID`, `MachineID`, `BatchID`) only printable ASCII without spaces and without `' " \ ``` is accepted; anything else is rejected with 4010. Use either a Latin-1 or a UTF-8 connection: MariaDB converts.

5. Reading answers and results

5.1 The answer of your row

Poll your rows until they are `D` or `E` (a second or two). The key is `InboxID` (from `LAST_INSERT_ID()` or from the first column of your own `SELECT`):

```
SELECT InboxID, Status, Code, Result, Assigned FROM sw_link_inbox
 WHERE ClientID = 'SHOP' AND OrderID = 'PO-5002' ORDER BY InboxID;
```

5.2 The state of the lines: sw_link_v_orders

One row per line. This is the view to show progress or to find out that an order is finished:

Table 5. Columns of sw_link_v_orders.

Column	Meaning
`ClientID`, `OrderID`, `Item`, `BatchID`, `ProductID`	Identification of the line.
`Status`	`pending` (stored, waiting), `claimed` (a machine has it), `done`, `error`. A line canceled while it was pending is **deleted**: it does not appear (the answer of the cancel row says `canceled`).
`MachineID`, `ReelID`	Machine that has the line and reel of the line.
`LengthM`, `Cuts`	What you ordered: length of one cut and number of cuts.
`CutsDone`, `MeasuredM`	Cuts made and total metres measured so far.
`StartTime`, `EndTime`	When the machine took the line and when it finished.
`ErrorCode`	Error code when `Status` is `error`.

5.3 Events: sw_link_events

The events are the history of what the Link did for you. `ClientID` is your client for the events of your orders. Read them in order, after a cursor:

```
SELECT EventID, Type, MachineID, DeliveryID AS OrderID, Item, ReelID, Code, Payload
  FROM sw_link_events
 WHERE ClientID = 'SHOP' AND EventID > :last ORDER BY EventID LIMIT 200;
```

Table 6. Event types.

Type	When	Payload (JSON)
`line.done`	A line finished.	`{"measuredM":37.500,"cuts":3,"endTime":"..."}`
`line.error`	The Link could not cut the line. `Code` has the error.	`{"text":"..."}`
`line.canceled`	A line was canceled.	`{}` (with `measuredM` and `cuts` if it had been measured)
`label.printed`	A label was printed (one per cut).	`{"cutNr":2,"printerId":"ZEB1"}`
`line.waste`	The operator reset the counter before the preselection: a waste.	`{"valid":false,"waste":1,"cutNr":2,"measuredM":18.000,"presetM":20.000,...}`

> **Note.** `machine.status` events are not attributed to a client (`ClientID` is empty). To know the state of a machine read the view `sw_link_v_machines` instead of the events. Events are kept `EventRetentionDays` (30). The `label.printed` of the last cut can arrive after `line.done`.

5.4 The cursor

Remember the last `EventID` you processed so that you never read the same event twice. There are two ways:

- **In your system** (recommended): store the highest `EventID` you handled and query `EventID > that`.
- **In the Link**: your user may read and update the column `LastEventID` of your row in `sw_link_clients`. Query `EventID > (SELECT LastEventID FROM sw_link_clients WHERE ClientID = 'SHOP')` and, after handling, `UPDATE sw_link_clients SET LastEventID = <last> WHERE ClientID = 'SHOP'`. It is the only column you may change there.

```
SELECT ClientID, LastEventID
FROM sw_link_clients
WHERE ClientID='SHOP';

+----------+-------------+
| ClientID | LastEventID |
+----------+-------------+
| SHOP     | 0           |
+----------+-------------+
1 row in set
```

```
+----------+-------------+
| ClientID | LastEventID |
+----------+-------------+
| SHOP     | 20          |
+----------+-------------+
1 row in set
```

Delivery is **at least once**: make your handler idempotent.

5.5 Other views

Table 7. Read-only views.

View	What it shows
`sw_link_v_cuts`	One row per cut made: `OrderID`, `Item`, `CutNr`, `MachineID`, `ReelID`, `MeasuredM`, `Created`.
`sw_link_v_waste`	One row per waste: `OrderID`, `Item`, `Seq`, `CutNr`, `MeasuredM`, `PresetM`, `ShortM`, `Seconds`, `Created`.
`sw_link_v_machines`	State of the machines: `State` (`offline`, `idle`, `ready`, `running`, `waiting`, `error`, `disabled`), `StateText`, `Online` (`Y` if the Link updated it in the last 30 s), `Cutting`, `CounterM`, `CutNr`, the current `OrderID`/`Item` and its `OrderClientID`.

```
+-----------+-------+-----------+--------+---------+----------+-------+---------+
| MachineID | State | StateText | Online | Cutting | CounterM | CutNr | OrderID |
+-----------+-------+-----------+--------+---------+----------+-------+---------+
| M1        | idle  | ready     | Y      | N       | 0.000    | 2     | NULL    |
| M2        | idle  | line done | Y      | N       | 6.000    | 1     | NULL    |
+-----------+-------+-----------+--------+---------+----------+-------+---------+
2 rows in set
```

6. Cancel, confirm and stock entries

6.1 Cancel a line or an order (Action X)

Write a row with `Action = 'X'`. `Item = 0` cancels the whole order; otherwise only that line. The effect depends on the state of the line:

Table 8. Cancel, by line state.

Line state	Result
Pending	Deleted at once. Row `D`, `canceled`. Event `line.canceled`. The line disappears from `sw_link_v_orders`.
Claimed, nothing measured yet	The Link asks the machine to release it and deletes it. Event `line.canceled` shortly after.
Claimed, measurement started	Not canceled: event `line.error` with code 4011 (`measurement already started`).
Done	Row `E`, code 4011.

```
INSERT INTO sw_link_inbox (ClientID, Action, OrderID, Item)
VALUES ('SHOP', 'X', 'PO-6001', 20);

Query OK, 1 row affected
```

```
+---------+--------+--------+---------+------+------+----------+
| InboxID | Action | Status | OrderID | Item | Code | Result   |
+---------+--------+--------+---------+------+------+----------+
| 16      | C      | D      | PO-6001 | 10   | 0    | accepted |
| 17      | C      | D      | PO-6001 | 20   | 0    | accepted |
| 18      | X      | D      | PO-6001 | 20   | 0    | canceled |
+---------+--------+--------+---------+------+------+----------+
3 rows in set
```

```
INSERT INTO sw_link_inbox (ClientID, Action, OrderID, Item)
VALUES ('SHOP', 'X', 'PO-6001', 0);

Query OK, 1 row affected
```

```
+---------+--------+--------+---------+------+------+----------+
| InboxID | Action | Status | OrderID | Item | Code | Result   |
+---------+--------+--------+---------+------+------+----------+
| 18      | X      | D      | PO-6001 | 20   | 0    | canceled |
| 19      | X      | D      | PO-6001 | 0    | 0    | canceled |
+---------+--------+--------+---------+------+------+----------+
2 rows in set
```

6.1.1 Clear what the machine shows (X with an empty OrderID)

With `OrderID` empty and `MachineID` filled, the row removes **what that machine has on its screen**, even if the line was already measured. It is refused with 4014 if the machine is running.

```
INSERT INTO sw_link_inbox (ClientID, Action, OrderID, MachineID)
VALUES ('SHOP', 'X', '', 'M1');

Query OK, 1 row affected
```

```
+---------+--------+--------+-----------+------+--------------------+
| InboxID | Action | Status | MachineID | Code | Result             |
+---------+--------+--------+-----------+------+--------------------+
| 20      | X      | D      | M1        | 0    | canceled PO-6000/10 |
+---------+--------+--------+-----------+------+--------------------+
1 row in set
```

6.2 Confirm a cut (Action K)

Machines set to **external confirmation** stop when the length is reached and wait for you (`sw_link_v_machines.State = waiting`). Confirm with `Action = 'K'`:

```
+-----------+---------+-----------------------------------------------------+---------+------+
| MachineID | State   | StateText                                           | OrderID | Item |
+-----------+---------+-----------------------------------------------------+---------+------+
| M2        | waiting | measure reached (6.000 m); waiting for confirmation | PO-7000 | 10   |
+-----------+---------+-----------------------------------------------------+---------+------+
1 row in set
```

```
INSERT INTO sw_link_inbox (ClientID, Action, OrderID, Item)
VALUES ('SHOP', 'K', 'PO-7000', 10);

Query OK, 1 row affected
```

```
+---------+--------+--------+---------+------+------+-----------+
| InboxID | Action | Status | OrderID | Item | Code | Result    |
+---------+--------+--------+---------+------+------+-----------+
| 22      | K      | D      | PO-7000 | 10   | 0    | requested |
+---------+--------+--------+---------+------+------+-----------+
1 row in set
```

`requested` means the confirmation was passed to the machine; the line finishes a moment later (`sw_link_v_orders`: `done`).

```
+---------+------+--------+----------+-----------+
| OrderID | Item | Status | CutsDone | MeasuredM |
+---------+------+--------+----------+-----------+
| PO-7000 | 10   | done   | 1        | 6.000     |
+---------+------+--------+----------+-----------+
1 row in set
```

6.3 Stock entry (Action S)

If the administrator turned on **stock control** (`[Stock] Control=1`), your system can report the stock of reels. Each row sets the **absolute** stock in metres of a reel (so repeating it is harmless). `ProductID`, `ReelID` and `LengthM` are required; if the product or the reel do not exist the Link creates them.

```
INSERT INTO sw_link_inbox (ClientID, Action, OrderID, ProductID, ReelID, LengthM,
    Description)
VALUES ('SHOP', 'S', 'IN-0001', 'H07V-K-1.5-BK', 'R-77881', 850, 'Goods receipt'),
       ('SHOP', 'S', 'IN-0001', 'H07V-K-1.5-BU', 'R-77882', 420.5, 'Goods receipt');

Query OK, 2 rows affected
```

```
+---------+--------+---------------+---------+---------+------+-----------------------------------+
| InboxID | Status | ProductID     | ReelID  | LengthM | Code | Result                            |
+---------+--------+---------------+---------+---------+------+-----------------------------------+
| 23      | D      | H07V-K-1.5-BK | R-77881 | 850.000 | 0    | stock set to 850.000 m (was 0.000 m) |
```

```
| 24      | D      | H07V-K-1.5-BU | R-77882 | 420.500 | 0    | stock set to 420.500 m (was 0.000 m) |
+---------+--------+---------------+---------+---------+------+--------------------------------------+
2 rows in set
```

```
+---------+--------+---------------+---------+---------+------+----------------------------+
| InboxID | Status | ProductID     | ReelID  | LengthM | Code | Result                     |
+---------+--------+---------------+---------+---------+------+----------------------------+
| 25      | D      | H07V-K-1.5-BK | R-77881 | 850.000 | 0    | stock unchanged (850.000 m) |
+---------+--------+---------------+---------+---------+------+----------------------------+
1 row in set
```

Without stock control the row is rejected with 4013. With control on, every cut also writes a stock movement and subtracts the metres from the reel.

7. Errors

A rejected row is `Status = E`, with `Code` and `Result`. Nothing is written for that row, and for the others of the same order if it was a `C` group.

Table 9. Error codes in the mailbox.

Code	Result (typical)	Cause and what to do
4001	unknown machine	`MachineID` does not exist. Check `sw_link_v_machines`.
4002	machine disabled or offline	The machine is disabled. Try later or use another machine.
4003	duplicate or unchangeable order	The line was already taken or finished and the content differs.
4004	invalid field	`LengthM` not > 0 or > 99999.999, `Cuts` < 1, `Item` < 1, empty `ProductID` or `OrderID`.
4005	the reel belongs to another product	Only if the administrator turned on the reel check.
4010	invalid character in an identifier	An identifier has a space or a forbidden character.
4011	the line cannot be canceled / confirmed	It is done, or the request does not apply.
4012	client unknown, disabled or without the DB channel	Wrong `ClientID`, not registered, disabled, or without the DB channel. Ask the administrator.
4012	machine not allowed for this client	The client is not allowed to use that machine.
4013	stock control is off	Stock entry (`S`) while `[Stock] Control=0`.
4014	the machine is running	Clear-screen refused: wait until it stops.
4001..4012	not written: another line of the order was rejected	A line of the same group failed: the others are not written. Fix the failing line and write the whole order again.

Examples:

```
INSERT INTO sw_link_inbox (ClientID, OrderID, Item, MachineID, ProductID, LengthM, Cuts)
VALUES ('SHOP', 'PO-E1', 10, 'M9', 'P1', 5, 1);

Query OK, 1 row affected
```

```
+---------+--------+---------+------+------+-----------------+----------+
| InboxID | Status | OrderID | Item | Code | Result          | Assigned |
+---------+--------+---------+------+------+-----------------+----------+
| 6       | E      | PO-E1   | 10   | 4001 | unknown machine |          |
+---------+--------+---------+------+------+-----------------+----------+
1 row in set
```

```
+---------+--------+---------+------+------+-----------------------------------------------------+-----
-----+
| InboxID | Status | OrderID | Item | Code | Result                                              |
Assigned |
+---------+--------+---------+------+------+-----------------------------------------------------+-----
-----+
| 9       | E      | PO-E4   | 10   | 4001 | not written: another line of the order was rejected |
|
| 10      | E      | PO-E4   | 20   | 4001 | unknown machine                                     |
|
+---------+--------+---------+------+------+-----------------------------------------------------+-----
-----+
2 rows in set
```

```
+---------+--------+---------+------+------+-------------------------------------------------+------
----+
| InboxID | Status | OrderID | Item | Code | Result                                          |
Assigned |
+---------+--------+---------+------+------+-------------------------------------------------+------
----+
| 11      | E      | PO-E5   | 10   | 4012 | client unknown, disabled or without the DB channel |
|
+---------+--------+---------+------+------+-------------------------------------------------+------
----+
1 row in set
```

> **Tip.** Unlike the REST interface, the mailbox **does** write the reason of each rejection in the Link log, as a warning: `inbox: order SHOP/PO-E1 rejected (code 4001)`.

8. Security and permissions

The database user created for your customer system has exactly the rights it needs, and nothing else:

```
+-----------------------------------------------------------------------------------------+
| Grants for shop@%                                                                       |
+-----------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO `shop`@`%` IDENTIFIED BY PASSWORD '*...'                          |
| GRANT SELECT ON `softwinding`.`sw_link_events` TO `shop`@`%`                            |
| GRANT SELECT, UPDATE (`LastEventID`) ON `softwinding`.`sw_link_clients` TO `shop`@`%`   |
| GRANT SELECT ON `softwinding`.`sw_link_v_orders` TO `shop`@`%`                          |
| GRANT SELECT ON `softwinding`.`sw_link_v_waste` TO `shop`@`%`                           |
| GRANT SELECT, INSERT, UPDATE (`Status`) ON `softwinding`.`sw_link_inbox` TO `shop`@`%`  |
| GRANT SELECT ON `softwinding`.`sw_link_v_cuts` TO `shop`@`%`                            |
| GRANT SELECT ON `softwinding`.`sw_link_v_machines` TO `shop`@`%`                        |
+-----------------------------------------------------------------------------------------+
8 rows in set
```

Table 10. What the database user may do.

You can	You cannot
Insert rows in `sw_link_inbox`, read them, and change only their `Status` (to release a draft).	Change any other column of a row, or delete rows.
Read `sw_link_events` and the four views `sw_link_v_*`.	Read or write any SoftWinding table (`sw_delivery_items`, `sw_stock`, ...) or the internal tables of the Link (`sw_link_orders`, `sw_link_machines`, ...).
Read your row of `sw_link_clients` and update its `LastEventID`.	Change anything else in `sw_link_clients` (token, allowed machines...).

```
SELECT *
FROM sw_delivery_items LIMIT 1;

ERROR 1142: SELECT command denied to user 'shop'@'127.0.0.1' for table
`softwinding`.`sw_delivery_items`
```

```
UPDATE sw_link_inbox
SET LengthM = 99
WHERE OrderID = 'PO-5001';

ERROR 1143: UPDATE command denied to user 'shop'@'127.0.0.1' for column 'LengthM' in table
'sw_link_inbox'
```

> **Warning.** **There is no isolation between customers at database level.** The `ClientID` is a value you write: a database user could write another customer's `ClientID` and the Link would accept the row (in our test, a `SHOP` user wrote a row as `ERPFILES` and it was accepted). Create **one database user per customer system**, give the user only to systems you trust, restrict it by host (`CREATE USER 'acme'@'10.0.0.%'`), use a strong password, and use MariaDB TLS on untrusted networks.

```
+---------+--------+---------+------+------+----------+----------+
| InboxID | Status | OrderID | Item | Code | Result   | Assigned |
+---------+--------+---------+------+------+----------+----------+
| 26      | D      | PO-I1   | 10   | 0    | accepted | M2       |
+---------+--------+---------+------+------+----------+----------+
1 row in set
```

9. Debugging and troubleshooting

9.1 A method: follow the order step by step

When "nothing happens", do not guess: follow the order along its path, from your software to the machine, and find the first step where it stops.

Table 11. The five steps of a diagnosis.

Step	Question	Where to look
1	**Did the Link read my row?** Query your row in `sw_link_inbox`: `Status` `D` (done) or `E` (rejected, with `Code` and `Result`). `N` means it is waiting, `W` that the Link is working on it, `I` that it is a draft.	Your own row in `sw_link_inbox` and the line `inbox:` in the Link log.
2	**Is the order stored?**	Administration window, **Orders** tab (filter by order number), or the view `sw_link_v_orders` in the database.
3	**Is a machine working on it?**	Administration window, **Machines** tab: state and current order. In the database: the view `sw_link_v_machines`.
4	**What did the Link do?**	Administration window, **Log** and **Events** tabs; the daily log file `logs\link_YYYYMMDD.log`.
5	**What did the machine answer?**	The log lines tagged with the machine, for example `[M1] ...`, and the machine state text.

9.2 Where the Link leaves its traces

Table 12. Diagnostic tools.

Place	What it shows
Administration window	**Status**: engine, database and licence. **Machines**: state, counter and current line of each machine, with a connection test. **Orders**: every line with its status, searchable. **Clients**: your client, its channels, allowed IPs and machines. **Events**: the events your system receives. **Log**: the Link log with a level filter and a button that opens the log folder.
Log file `logs\link_YYYYMMDD.log`	One line per fact: starts, connections, every request of the channel, every order accepted, every line sent to the PLC, every cut confirmed, warnings and errors. One file per day, kept 30 days.
Table `sw_link_log`	The engine part of the log, queryable with SQL: `SELECT * FROM sw_link_log ORDER BY LogID DESC LIMIT 50`.
Events (`sw_link_events`, API, files, messages)	The business-level history: `line.done`, `line.error`, `line.canceled`, `label.printed`, `machine.status`, `line.waste`.
Views `sw_link_v_orders`, `_cuts`, `_waste`, `_machines`	The current state of lines, cuts made, wastes and machines, as plain SQL.
`SoftWindingLinkSvc.exe --console`	Runs the engine in a window instead of as a service, to watch it live (stop with Enter). Combine with `--ini=<path>` to use a test configuration.

Typical engine lines for an order that goes well (excerpt of the log of the test system):

```
12:46:16 INF REST order SO-100234 from ACME: 2 line(s) accepted
12:46:17 INF [M1] claimed ACME_20261028|SO-100234|H07V-K-1.5-BK|10
12:46:17 INF [M1] line ACME_20261028|SO-100234|H07V-K-1.5-BK|10 sent to the PLC:
```

```
             H07V-K-1.5-BK x 3 cuts of 12.500 m (preselection 1250)
12:46:25 INF [M1] cut 1/3 confirmed: 12.500 m
12:46:38 INF [M1] line ACME_20261028|SO-100234|H07V-K-1.5-BK|10 DONE: 3 cuts, 37.500 m
```

9.3 My line does not start

The line is stored (`pending`) but no machine takes it, or the machine takes it and nothing is cut. Check, in this order:

Table 13. Why a line does not start.

What you see	Probable cause	What to do
Machine state `O` (offline) or `D` (disabled), or a heartbeat older than 30 s	The Link cannot reach the PLC, the machine is switched off in the Link, or **the Link service is not running**.	Check the service, the machine network and the **Machines** tab (use the connection test).
The line stays `pending`; the machine is free (`I`)	The line is for another machine, or has no `machineId` and the machine accepts only addressed lines, or an earlier line is still ahead of it (batch priority, `priority`, `item`).	Compare `machineId` with the machine; look at the **Orders** tab ordering.
The machine is `P` (`prepared ...`) and nothing moves	The line was sent to the PLC and the machine waits for the **operator to start it**. The Link does not start machines.	Ask the operator to start. This is normal.
The machine shows the order list on its screen and waits	The machine works in **screen mode**: the operator chooses the order on the machine panel.	The operator picks and starts the order on the screen.
The machine is `S` (`waiting for confirmation`)	The machine is set to external confirmation and waits for your system.	Confirm the cut (REST `confirm`, mailbox action `K`...).
The machine is `E`	Emergency stop, open guard or a PLC error.	The operator clears the cause; the Link continues.
The line is `error`	The PLC refused it or the data were invalid (see `code` and the `line.error` event).	Read the code in chapter 7 (REST guide) / the error table of this guide and the Link log.
The first line works but the next one never starts	The machine waits for the operator after each line (reset, next order) or a previous line is still `claimed`.	Look at the machine state and current order.

9.4 Common problems with this interface

Table 14. Common problems.

Symptom	Cause	Solution
My row stays `N` for a long time	The Link service is not running, `[Inbox] Enabled` is 0, or the Link cannot reach the database.	Check the **Status** tab and the service. Look at `Processed` and `Instance` of other rows to see who is working.
My row stays `W` for more than a minute	The Link instance that claimed it stopped.	Another instance returns it to `N` after 60 s; if none is running, start the service.
`E 4012 client unknown, disabled or without the DB channel`	The `ClientID` you wrote is not registered, is disabled, has no DB channel, or has a typo.	Compare it with the **Clients** tab (the `ClientID` is case-insensitive, but must exist).
`E with not written: another line of the order was rejected`	Another row of the same order had an error.	Look at the other rows of the order: the one with the real reason has the order code and the reason.
I wrote the lines but only some appeared	The lines were written in separate statements and some are drafts (`I`), or the lines of one order were rejected together.	Write one order in one `INSERT`, or release drafts with `UPDATE ... Status = 'N'`.
Accents come out as `?` or garbage	Character set of your connection, or characters outside Latin-1.	Use a UTF-8 or Latin-1 connection; the tables are Latin-1.
`ERROR 1142 SELECT command denied`	You read a table your user is not allowed to read.	Use the views `sw_link_v_*`, not the base tables.
`ERROR 1143 UPDATE command denied for column`	You tried to change a column other than `Status`.	To correct a row, insert a new one (same key): see section 4.5.
I cannot find my canceled line	A line canceled while pending is deleted.	Read the answer row (`D`, `canceled`) and the event `line.canceled`.
I read events but see no `line.done`	The order is not finished, or you filter by another `ClientID`.	Look at `sw_link_v_orders` for the state; filter events by your `ClientID`.
The same event comes twice	You did not move the cursor, or delivery is at least once.	Make the handler idempotent and save the cursor after handling.

9.5 Tracing the traffic

Because the mailbox is plain SQL, you can trace everything with SQL:

```
-- the last rows of your client, newest first
SELECT InboxID, Action, Status, OrderID, Item, Code, Result, Assigned, Created, Processed
  FROM sw_link_inbox WHERE ClientID = 'SHOP' ORDER BY InboxID DESC LIMIT 20;

-- everything that was rejected in the last day
SELECT * FROM sw_link_inbox
 WHERE ClientID = 'SHOP' AND Status = 'E' AND Created > NOW() - INTERVAL 1 DAY;

-- what is on the machines now
SELECT * FROM sw_link_v_machines;
```

On your side, the MariaDB **general log** (`SET GLOBAL general_log = 1`) shows every statement your application sends, which helps to prove what was written and when. The Link log also contains one line per processed group, for example:

```
inbox: order SHOP/PO-7000 accepted (1 lines)
inbox: order SHOP/PO-E1 rejected (code 4001)
inbox: client NOBODY rejected (unknown, disabled or without the DB channel)
inbox: cancel SHOP/PO-6001 item 0 -> 0
```

```
inbox: confirm SHOP/PO-7000 item 10 -> 0
inbox: stock IN-0001 reel R-77881 -> 0
```

9.6 What to send to LOIMEX support

- Link version (`GET /health` or the window title) and Windows version.
- The order number, the line (`item`) and the machine.
- The exact request and the **complete answer** (headers and body), or the row / file / telegram / message you sent.
- The log file of the day (`logs\link_YYYYMMDD.log`) and, if possible, an export of the **Events** tab.
- What you expected and what happened, with the time (local time of the server).

10. Testing without a machine

You do not need a real machine to develop. LOIMEX provides **simulators** that behave like a cutting machine and like a customer system. With them you can run every example in this guide at your desk, repeat a failing case as many times as you like and see events appear in the Link window.

Table 15. Simulators (installed in the Link folder with the "Simulators and diagnostic tools" component of the installer).

Program	What it is	Typical use
`SoftWindingLinkSim.exe`	Machine simulator (console). Speaks the PLC protocols of the Omron NX/CP (FINS over UDP, port 9600) and of PLCs without FINS (SFW over TCP, port 63000) and simulates a Zebra printer on TCP 9100. With `--auto` an automatic "operator" starts, measures and resets the counter by itself.	Unattended tests and scripts.
`SoftWindingLink_MachineSimulator.exe`	The same simulator with a **window**: counter, preselection and cut count, with buttons to start, stop, reset the counter, reach the length, cut and edit values. The right panel shows the order the Link writes to the machine.	You play the operator by hand.
`SoftWindingLink_ServerSimulator.exe`	Customer-system simulator with a window: sends orders by database, REST, TCP (AHP) and MQTT and shows the answers.	Compare your integration with a known good client.

10.1 Set up the test machine

1. **Start the machine simulator** in a command prompt: `SoftWindingLinkSim.exe --auto --speed=400 --start-delay=1500 --cut-delay=2000 --printer-port=9100`. It listens on `127.0.0.1`. `--speed` is the counter speed in counter units per second (hundredths of a metre: 400 is 4 m/s; default 2000), `--start-delay` the time before the automatic operator starts, `--cut-delay` the time the operator takes to cut and reset. Add `--noconsole` to run it in the background (otherwise it waits for commands typed in its console) and `--run=SECONDS` to make it stop by itself.
2. **Add the machine to the Link.** In the administration window, **Machines** tab, create `M1`: PLC type *Omron NX1 (FINS)*, address `127.0.0.1`, port `9600`, profile `OMRON_CP`, confirm mode *counter*, cut mode *cut by cut*, enabled. Use the connection test: it must answer. (The database equivalent is the `INSERT INTO sw_link_machines ...` shown in the Link manual, section 4.)
3. **Create your client** in the **Clients** tab (or `SoftWindingLinkSvc.exe --newclient=ACME --channels=DB`) and keep the access data it shows.
4. **Check the channel** is enabled in the **Settings** tab and that the engine is running (Status tab).
5. **Run the examples.** The line goes `pending`, `claimed` and `done` in a few seconds; the simulated machine counts every cut and the simulated Zebra receives the labels.

> **Warning.** **Never point a test client at a Link that drives real machines**, unless the machine identifiers you use are reserved for tests: on a real machine an operator could start your test line. Use a separate test Link, or machines that exist only for testing.

10.2 Useful simulator options

Table 16. Machine simulator options.

Option	Effect
`--auto`	Automatic operator: starts, measures up to the preselection, confirms the cut and resets.
`--speed=N`, `--start-delay=MS`, `--cut-delay=MS`	Counter speed and operator delays. Slow values let you watch a line in progress.
`--fins-port=N`, `--sfw-port=N`	Ports. Use different ports to run several simulated machines on one PC (`M1` on 9600, `M2` on 9601).
`--printer-port=9100`, `--printer-file=labels.txt`	Simulated Zebra: the received ZPL is appended to the file.
`--waste=N`, `--wastes=K`	The cut is interrupted at N counter units and the operator resets the counter: a waste (a reset before the preselection). K = how many times (1 by default).
`--noconsole`, `--run=SECONDS`	Without them the simulator waits for typed commands in its console. Use `--noconsole` to run it in the background, and `--run` to make it stop after that many seconds.
`--listen=IP`	Address to listen on (default `127.0.0.1`).

To try the mailbox with a ready-made client, open `SoftWindingLink_ServerSimulator.exe`: its database tab writes rows into `sw_link_inbox` and shows the answers. The command `SoftWindingLink_ServerSimulator.exe --selftest=db` runs a complete cycle without any interaction.

11. Going to production

Table 17. Production checklist.

Topic	Recommendation
One INSERT per order	Write all the lines of an order in one statement (or as drafts released together).
Do not wait in your transaction	Commit the `INSERT`, then read the answer in a later step.
Check every answer	Poll `Status` until `D` or `E`, and handle `E` (log `Code` and `Result`; do not resend without fixing).
Idempotency	You can resend a row with the same key: you get `already received`. Use it to recover after a failure.
Cursor	Save the last `EventID` after handling, in your system or in `LastEventID`.
Polling	One query per second or less for answers and events. Avoid scanning big ranges: filter by `ClientID` and keys.
Security	One database user per customer system, restricted by host, strong password, TLS on untrusted networks.
Clock	All dates are the local time of the database server.
Monitoring	Check `sw_link_v_machines.Online` and the age of your oldest row in `N`.
Retention	Answers and events are purged after 30 days (configurable): copy what you need.

Appendix A. Example client (Python)

A complete client in one file (`pymysql`). The same cycle in Node.js and C# is in `docs\examples\inbox`.

```
import os
import sys
import time


import pymysql


FIELDS = ('item', 'machineId', 'productId', 'reelId', 'lengthM', 'cuts', 'priority', 'description', 'notes') + \
    tuple('free%d' % i for i in range(1, 11))
COLUMNS = {'item': 'Item', 'machineId': 'MachineID', 'productId': 'ProductID', 'reelId': 'ReelID', 'lengthM':
'LengthM',
           'cuts': 'Cuts', 'priority': 'Priority', 'description': 'Description', 'notes': 'Notes'}
COLUMNS.update({'free%d' % i: 'Free%d' % i for i in range(1, 11)})



class InboxClient:
    def __init__(self, host, port, user, password, client_id, database='softwinding'):
        self.client_id = client_id
        self.db = pymysql.connect(host=host, port=port, user=user, password=password, database=database,
                                  charset='latin1', autocommit=True)       # the tables are latin1


    def _q(self, sql, args=()):
        with self.db.cursor(pymysql.cursors.DictCursor) as c:
            c.execute(sql, args)
            return c.fetchall()


    # ---- send ----
    def send_order(self, order_id, lines, customer='', draft=False):
        """One INSERT statement with all the lines: the Link reads them together (all or nothing).
        Returns the InboxID of every row. With draft=True the rows wait (Status I) until release()."""
        cols = ['ClientID', 'OrderID', 'Customer', 'Status'] + [COLUMNS[k] for k in FIELDS if any(k in l for l in
lines)]
        keys = [k for k in FIELDS if any(k in l for l in lines)]
        row_sql = '(' + ', '.join(['%s'] * len(cols)) + ')'
        args = []
        for l in lines:
            args += [self.client_id, order_id, customer, 'I' if draft else 'N'] + [l.get(k, '' if k not in ('lengthM',
'cuts', 'item', 'priority') else 0) for k in keys]
        with self.db.cursor() as c:
            c.execute('INSERT INTO sw_link_inbox (%s) VALUES %s' % (', '.join(cols), ', '.join([row_sql] *
len(lines))), args)
            first = c.lastrowid
        return list(range(first, first + len(lines)))


    def release(self, order_id):
        """Release a draft written row by row (Status I -> N)."""
        with self.db.cursor() as c:
            c.execute("UPDATE sw_link_inbox SET Status = 'N' WHERE ClientID = %s AND OrderID = %s AND Status = 'I'",
(self.client_id, order_id))


    def cancel(self, order_id, item=0):
        """item=0 cancels the whole order. Returns the InboxID of the request."""
        return self._action('X', order_id, item)


    def confirm(self, order_id, item):
        return self._action('K', order_id, item)


    def clear_machine_screen(self, machine_id):
        with self.db.cursor() as c:
            c.execute("INSERT INTO sw_link_inbox (ClientID, Action, OrderID, MachineID) VALUES (%s, 'X', '', %s)",
(self.client_id, machine_id))
            return c.lastrowid


    def stock_entry(self, product_id, reel_id, length_m, ref='stock'):
        """Absolute stock of a reel (needs [Stock] Control=1 in the Link)."""
        with self.db.cursor() as c:
            c.execute("INSERT INTO sw_link_inbox (ClientID, Action, OrderID, ProductID, ReelID, LengthM) VALUES (%s,
'S', %s, %s, %s, %s)",
                      (self.client_id, ref, product_id, reel_id, length_m))
            return c.lastrowid
```

```
    def _action(self, action, order_id, item):
        with self.db.cursor() as c:
            c.execute("INSERT INTO sw_link_inbox (ClientID, Action, OrderID, Item) VALUES (%s, %s, %s, %s)",
(self.client_id, action, order_id, item))
            return c.lastrowid


    # ---- answers and state ----
    def answers(self, ids, timeout=30):
        """Waits until the Link has processed every row (Status D done or E rejected) and returns them."""
        end = time.time() + timeout
        while True:
            rows = self._q('SELECT InboxID, Action, Status, OrderID, Item, Code, Result, Assigned FROM sw_link_inbox
WHERE InboxID IN (%s) ORDER BY InboxID'
                           % ', '.join(['%s'] * len(ids)), ids)
            if all(r['Status'] in ('D', 'E') for r in rows) or time.time() > end:
                return rows
            time.sleep(0.5)


    def order_state(self, order_id):
        return self._q('SELECT Item, Status, MachineID, ReelID, LengthM, Cuts, CutsDone, MeasuredM, StartTime,
EndTime, ErrorCode '
                       'FROM sw_link_v_orders WHERE ClientID = %s AND OrderID = %s ORDER BY Item', (self.client_id,
order_id))


    def wait_until_finished(self, order_id, timeout=120, poll=2.0):
        end = time.time() + timeout
        while time.time() < end:
            state = self.order_state(order_id)
            if state and all(l['Status'] in ('done', 'error') for l in state):
                return state
            time.sleep(poll)
        raise TimeoutError('order %s did not finish in %s s' % (order_id, timeout))


    # ---- events with the cursor stored in the Link (sw_link_clients.LastEventID) ----
    def new_events(self, limit=200):
        return self._q("SELECT EventID, Type, MachineID, DeliveryID AS OrderID, Item, ReelID, Code, Payload FROM
sw_link_events "
                       "WHERE ClientID = %s AND EventID > (SELECT LastEventID FROM sw_link_clients WHERE ClientID =
%s) "
                       "ORDER BY EventID LIMIT %s", (self.client_id, self.client_id, limit))


    def ack(self, last_event_id):
        with self.db.cursor() as c:
            c.execute('UPDATE sw_link_clients SET LastEventID = %s WHERE ClientID = %s', (last_event_id,
self.client_id))
```

Use it like this:

```
link = InboxClient("link-server", 3306, "acme", password, "ACME")
ids = link.send_order("SO-100234", [
    {"item": 10, "machineId": "M1", "productId": "H07V-K-1.5-BK", "lengthM": 12.5, "cuts": 3}],
    customer="ACME Cables Ltd")
for answer in link.answers(ids):
    print(answer["Status"], answer["Code"], answer["Result"])
state = link.wait_until_finished("SO-100234")
for e in link.new_events():
    handle(e); link.ack(e["EventID"])
```

Appendix B. SQL cookbook

```
-- Lines of my client that are not finished
SELECT OrderID, Item, Status, MachineID, CutsDone, Cuts FROM sw_link_v_orders
 WHERE ClientID = 'ACME' AND Status IN ('pending', 'claimed') ORDER BY OrderID, Item;

-- Metres cut per day and machine
SELECT DATE(Created) AS day, MachineID, COUNT(*) AS cuts, SUM(MeasuredM) AS metres
  FROM sw_link_v_cuts WHERE ClientID = 'ACME' GROUP BY DATE(Created), MachineID;

-- Wastes of today
SELECT OrderID, Item, CutNr, MeasuredM, PresetM, ShortM FROM sw_link_v_waste
 WHERE ClientID = 'ACME' AND Created >= CURDATE();

-- Rows still waiting for an answer for more than one minute
SELECT InboxID, OrderID, Status, Created FROM sw_link_inbox
 WHERE ClientID = 'ACME' AND Status IN ('N', 'W') AND Created < NOW() - INTERVAL 1 MINUTE;

-- Is the Link alive and the machines online?
SELECT MachineID, State, Online, Updated FROM sw_link_v_machines;
```