



Go-live procedure

Connecting Doli2Shop to an existing, already populated Dolibarr

Doli2Shop · Last updated : 2026-09-28

Generic document. It applies to any installation where Shopify is populated by migrating an existing store (PrestaShop, WooCommerce, Magento...) while Dolibarr already holds the corresponding sales history.

1. The risk this procedure removes

When your historical orders are pushed into Shopify, they receive **brand-new Shopify identifiers**, and their creation and modification timestamps are those of the **migration day** — Shopify lets you back-date the processed date, never those two.

Doli2Shop recognises an already-imported order by its **Shopify identifier**. An order that exists in Dolibarr from another origin carries no Shopify identifier, so the connector has no way of knowing it is the same sale. Without precautions, it is imported a second time, dated today, into your accounts.

Three paths can import an order, and they are independent of one another:

Path	What it does	Date filter
Order webhook	Treats any unknown order as a creation	none
<code>Doli2ShopOrdersCatchup</code> job	Asks Shopify for recently modified orders and imports those missing from Dolibarr	configurable rolling window
<code>Doli2ShopOrdersHistoricalImport</code> job	Deliberate historical import	its own date boundaries

2. Before the migration — settings

- **Stock synchronisation direction.** `DOLI2SHOP_SYNC_STOCKS_DIRECTION` must be set to `dolibarr_to_shopify` if Dolibarr is your stock master. The default is `both`: with that value, a Shopify inventory webhook also writes stock movements back into Dolibarr.
- **Unpaid orders.** By default, only orders whose financial status is *paid* are imported. If orders awaiting payment must come in (B2B payment on terms, bank transfer), enable `DOLI2SHOP_SYNC_NON_PAID_ORDERS`.
- **Product references.** Shopify variant SKUs must match Dolibarr references exactly: that matching is what prevents duplicate products and variants from being created.

- **Third-party emails.** An existing customer is matched on the email held on the **company record itself**, not on its contacts, and within the same entity. An email that only lives on a contact, or that differs from the Shopify one, will produce a duplicate company. Reconcile them before go-live — this is a data task, not a configuration one.

3. During the migration — what must stay switched off

Until the migration is finished **and verified**, all three import paths must be inactive:

1. the **order webhooks** — *Webhooks* screen in the module configuration;
2. the scheduled job `Doli2ShopOrdersCatchup` ;
3. the scheduled job `Doli2ShopOrdersHistoricalImport` .

One read-only query to check:

```
SELECT label, entity, status FROM llx_cronjob WHERE label LIKE 'Doli2Shop%';
```

Both jobs must show `status = 0` . On a Multicompany installation, take the opportunity to check that the `entity` column matches the entity where the module is configured: a job created in the wrong entity runs empty, without raising any error.

Product and stock synchronisation may keep running during this phase.

4. Go-live

1. Complete the migration and verify it in Shopify.
2. **Write down the exact timestamp** at which the migration ended — call it T0.
3. Lower `DOLI2SHOP_CATCHUP_LOOKBACK_HOURS` to a value that **does not reach back before T0**: 1 or 2 hours. The default is 168 (7 days), which would pull your migrated orders straight in.
4. Enable the order webhooks first, then the catch-up job.

5. After go-live — checks

- Open the **webhook events** screen: it lists what comes in, with its status.
- Check that **no order dated before T0** has been created in Dolibarr.
- Only restore `DOLI2SHOP_CATCHUP_LOOKBACK_HOURS` to its usual value (168) **once T0 is more than seven days old**. Before that, the window would reach your migrated orders.

6. What this procedure does not cover

The catch-up window bounds **the scheduled job**, not the webhooks.

After go-live, any **change made in Shopify to a migrated order** — an edit, a refund, an address correction, adding a tag — fires an `orders/updated` webhook. The connector does not know that order, so it creates it. You end up with a historical order re-imported under today's date.

Procedure alone cannot close this gap. It would be closed by a **go-live cut-off date** setting: any Shopify order created before the chosen date would be excluded from import, **whichever path it takes** — creation webhook, update webhook, catch-up job — while the deliberate historical import feature keeps its own boundaries.



This setting does not exist in the module yet. It is a development, available on quotation. Without it, the only safeguard is vigilance over migrated orders: do not edit them in Shopify after go-live, and check regularly that no order predating T0 has appeared in Dolibarr.

7. If a duplicate is created anyway

1. Identify it from the **webhook events** screen: the event carries the Shopify order identifier.
2. As long as the Dolibarr order created has been neither validated nor invoiced, deleting it has no accounting consequence.
3. If it has already been invoiced, correcting it follows your usual Dolibarr practice (credit note or cancellation, depending on your accounting policy) — nothing specific to the connector is required.
4. Remove the cause before cleaning up, otherwise the next webhook will recreate the same order.