Sublogin Documentation

Complete PDF Manual

Guide for Magento administrators, technical project owners, and implementation teams.

LAST UPDATED
2026-10-13

SOURCE
Open complete documentation online

EXTENSION
Magento 2 Extension Sublogin

Table of Contents

Sublogin Documentation · sublogin

Overview
Sublogin
Getting Started
Installation
Configuration
Create Your First Sublogin
Features
Address Management
Email and Account Recovery
Public Sublogin Signup
Extend the Frontend Form
Login Context and Impersonation
Multi-Account Management
Payment and Shipping Restrictions
Shared Cart and Wishlist
Order and Product Reports
Addons
Sublogin Web API
Budget Integration Boundary
Budget Management
Sublogin Custom Catalog
Hyvä Compatibility Packages
Import and Export
One-Time Budget
Order Approval
Roles, Permissions and Groups
Sample Data
Transform Customers and Sublogins
Role Web API
Mageplaza One Step Checkout Compatibility
Use Cases
Department Purchasing
Controlled Employee Ordering

Branch and Franchise Ordering

Reference

Configuration Fields

Database Reference

Troubleshooting

Common Issues

Permission Problems

FAQ

Sublogin

SoftwareSilo Sublogin lets a company use one Magento customer account without sharing one login. The main account can create separate sign-ins for buyers, branches or departments, decide which addresses and checkout methods each person may use, and keep orders tied to the person who placed them.

The base package covers account management, login context, address assignment, payment and shipping restrictions, shared carts and wishlists, and order ownership. Roles, budgets, approval workflows and catalogs are separate add-ons. Install only the parts your purchasing process needs.

Start with one real purchasing scenario

Before creating many accounts, choose one buyer and answer these questions:

1. Should this person see only their own orders or every order from the company account?
2. May they use every company address, selected company addresses, or an address of their own?
3. Are payment or shipping methods restricted for this person?
4. Should carts and wishlists be private or shared across the company account?
5. Do they need a role, a spending limit, an approval step or a restricted catalog?

Create that buyer first and place a complete trial order. This catches address, email and checkout conflicts before you import a larger account structure.

Main account and sublogin responsibilities

Account	Typical responsibility
Main customer account	Owns the company relationship, creates and manages sublogins, assigns addresses, and can sign in as a sublogin when impersonation is enabled
Sublogin	Signs in with a separate email and password, shops within assigned restrictions, and creates orders recorded with that sublogin identity
Magento administrator	Configures store-wide behavior, manages records in Admin, and supports account recovery and troubleshooting

Sublogin emails must be unique within Magento's configured customer-account scope. A sublogin cannot reuse the parent customer's email address.

What the base package includes

- customer and Admin screens for creating and maintaining sublogins;
- separate credentials with Magento password-policy checks;
- active, expiry, email and newsletter controls;
- optional main-account impersonation with a visible login-context banner;
- own-order or company-order visibility;
- parent-address assignment and optional sublogin-owned addresses;
- per-sublogin payment and shipping method restrictions;
- optional shared cart and wishlist;
- order and product reports for administrators and, when enabled, customer accounts;
- an optional public signup request for people joining an existing company account.

The public signup form is disabled by default. Its parent-email match helps attach a request to an account, but it does not verify the applicant's identity — every request starts inactive until the parent customer approves it through the emailed confirmation link or an administrator activates it.

Optional packages

Need	Package and guide
Reusable roles, storefront permissions and group hierarchy	`mageb2b/sublogin-role`: Roles and Permissions
Daily, monthly, yearly or per-order spending limits	`mageb2b/sublogin-budget`: Budget Management
A temporary allowance for the next order	`mageb2b/sublogin-budget-one-time`: One-Time Budget
Main-account or group approval before an order proceeds	`mageb2b/sublogin-orderapproval`: Order Approval
A product assortment for one or more sublogins	`mageb2b/sublogin-custom-catalog`: Sublogin Custom Catalog
REST and SOAP management of sublogins	`mageb2b/sublogin-api`: Web API
REST and SOAP management of roles, groups and permissions	`mageb2b/sublogin-role-api`: Role Web API
Magento CSV export and command-line import	`mageb2b/sublogin-importexport`: Import and Export

Need	Package and guide
Customer-to-sublogin or sublogin-to-customer conversion	`mageb2b/sublogin-transform`: Transform Accounts
Hyvä storefront templates	Hyvä Compatibility Packages
Mageplaza One Step Checkout address selection	`mageb2b/sublogin-mageplaza-osc`: Mageplaza OSC Compatibility
Disposable development fixtures	Sample Data

Reports are part of the current base package. There is no separate Reports package to install.

Recommended setup path

1. Install the base package.
2. Review the Configuration Guide, especially order visibility, impersonation and address handling.
3. Create one sublogin with a reserved example address in staging.
4. Sign in as that sublogin and test product access, cart, address selection, payment, shipping and order history.
5. Add Roles, Budget, Order Approval or Custom Catalog only after the base journey works.
6. Repeat the complete order test after every add-on or checkout integration.

Important boundaries

- Sublogin does not create a separate Magento customer for each employee. The records remain children of the main customer account.
- The base package does not provide product and checkout roles. Those controls require the Role add-on.
- Order Approval supports a per-sublogin subtotal threshold and an optional group hierarchy. It is not a general-purpose, amount-tier workflow engine.
- The Budget add-on has a PHP service contract but no `/V1/sublogin-budget` REST endpoint.
- Shared cart and wishlist are supported. Collaborative quote editing is not part of the verified base feature set.

Find the right guide

- Address Management

- Login Context and Impersonation
- Payment and Shipping Restrictions
- Shared Cart and Wishlist
- Order and Product Reports
- Public Sublogin Signup
- Frequently Asked Questions
- Email and Account Recovery
- Common Issues
- Permission Problems

Requirements

- PHP 8.1 or newer within the range supported by your Magento release
- Magento 2.4
- the Magento Integration module
- a separate license and Composer entitlement for each optional package you install

Installation

Install and test the base package before adding roles, budgets or approval workflows. Sublogin adds account, quote, wishlist and order data, so take a database backup and run the first installation in staging.

Before you begin

You need:

- PHP 8.1 or newer within your Magento release's supported range;
- Magento 2.4;
- the Magento Integration module;
- Composer 2;
- repository access for `mageb2b/sublogin`;
- a valid SoftwareSilo license;
- maintenance access to run Magento setup and deployment commands.

Check the package version Composer can resolve before changing the store:

```
composer show mageb2b/sublogin --all
```

Install the base package

From the Magento project root:

```
composer require mageb2b/sublogin
php bin/magento module:enable MageB2B_ExtensionManager
MageB2B_Sublogin
php bin/magento setup:upgrade
```

For a production-mode store, rebuild generated code and static assets using your normal deployment process:

```
php bin/magento setup:di:compile
php bin/magento setup:static-content:deploy
```

Refresh only the caches your deployment has invalidated. A blanket cache flush is not required for every installation.

Confirm the installation

Run:

```
php bin/magento module:status MageB2B_Sublogin
composer show mageb2b/sublogin
```

Then sign in to Magento Admin and open **Stores > Configuration > MageB2B > Sublogin**. Confirm that the module information is visible and **Enabled** is set for the intended website or store view.

The base package adds the main Sublogin tables and associates the current sublogin ID with quotes, orders, the sales order grid, wishlists and integration tokens. `setup:upgrade` must complete before anyone signs in as a sublogin.

Install optional packages only when needed

Each add-on is a separate Composer package and license entitlement. Install the base package first.

Workflow	Composer package	Magento module
Roles, permissions and groups	`mageb2b/sublogin-role`	`MageB2B_SubloginRole`
Spending limits and Pay on Budget	`mageb2b/sublogin-budget`	`MageB2B_SubloginBudget`
Allowance for the next order	`mageb2b/sublogin-budget-one-time`	`MageB2B_SubloginBudgetOneTime`
Order approval	`mageb2b/sublogin-orderapproval`	`MageB2B_SubloginOrderApproval`
Sublogin-specific catalogs	`mageb2b/sublogin-custom-catalog`	`MageB2B_SubloginCustomCatalog`
REST and SOAP endpoints	`mageb2b/sublogin-api`	`MageB2B_SubloginApi`
CSV import and export	`mageb2b/sublogin-importexport`	`MageB2B_SubloginImportExport`
Customer and sublogin conversion	`mageb2b/sublogin-transform`	`MageB2B_SubloginTransform`
Disposable development records	`mageb2b/sublogin-sample-data`	`MageB2B_SubloginSampleData`

Reports are included in `mageb2b/sublogin`. Do not install the retired `mageb2b/sublogin-report` package.

Install an add-on with the same sequence used for the base package. For example:

```
composer require mageb2b/sublogin-role
php bin/magento module:enable MageB2B_ExtensionManager
```

```
MageB2B_SubloginRole
php bin/magento setup:upgrade
```

After installation, open the add-on's guide and configure it before exposing its screens to customers.

Hyvä packages

Hyvä compatibility is split by feature. Install the bridge that matches each enabled add-on:

Storefront feature	Package
Base Sublogin screens	`mageb2b/sublogin-hyva`
Roles and groups	`mageb2b/sublogin-role-hyva`
Budgets	`mageb2b/sublogin-budget-hyva`
Sublogin Custom Catalog	`mageb2b/sublogin-custom-catalog-hyva`
Order Approval	`mageb2b/sublogin-orderapproval-hyva`

These packages require the Hyvä default theme and the corresponding parent Sublogin packages. Let Composer resolve the complete dependency set rather than copying one bridge package between stores.

See Hyvä Compatibility Packages for the exact package relationships.

First verification

Use a clearly fictional staging customer, for example `northstar-buyer@example.com`.

1. Create one active sublogin from the main customer account.
2. Sign out and sign in with the new credentials.
3. Add a product to the cart and complete checkout with an allowed address, payment method and shipping method.
4. Confirm the order appears under the correct sublogin and follows the configured order-visibility rule.
5. If impersonation is enabled, return to the main account, sign in as the sublogin and confirm the context banner is visible.
6. Test account setup or password reset without placing a password in an email or log.

Do not import a full company structure until this path works.

Updating

Review the package changelog and take a backup before updating:

```
composer update mageb2b/sublogin --with-dependencies
php bin/magento setup:upgrade
```

When add-ons are installed, include them in the update plan and verify that Composer selects compatible versions. Recompile and deploy static content when required by your Magento mode and release process.

Retest login, address selection, checkout, order visibility and every installed add-on after the update.

Removing the module

Disabling code is different from deleting business data. Start with:

```
php bin/magento module:disable MageB2B_Sublogin
```

Do not run `module:uninstall` or manually delete Sublogin tables on a live store without a reviewed data-retention plan. Sublogin IDs are connected to customer, quote, wishlist and sales records. Preserve the database backup and decide how historic orders should be presented before removing data.

Next steps

- Configure Sublogin
- Create the First Sublogin
- Common Installation and Login Issues

Configuration

Open **Stores > Configuration > MageB2B > Sublogin**. Choose the website or store view before changing values. Most settings are scope-aware, so an unexpected storefront result often comes from a website or store-view override rather than the default configuration.

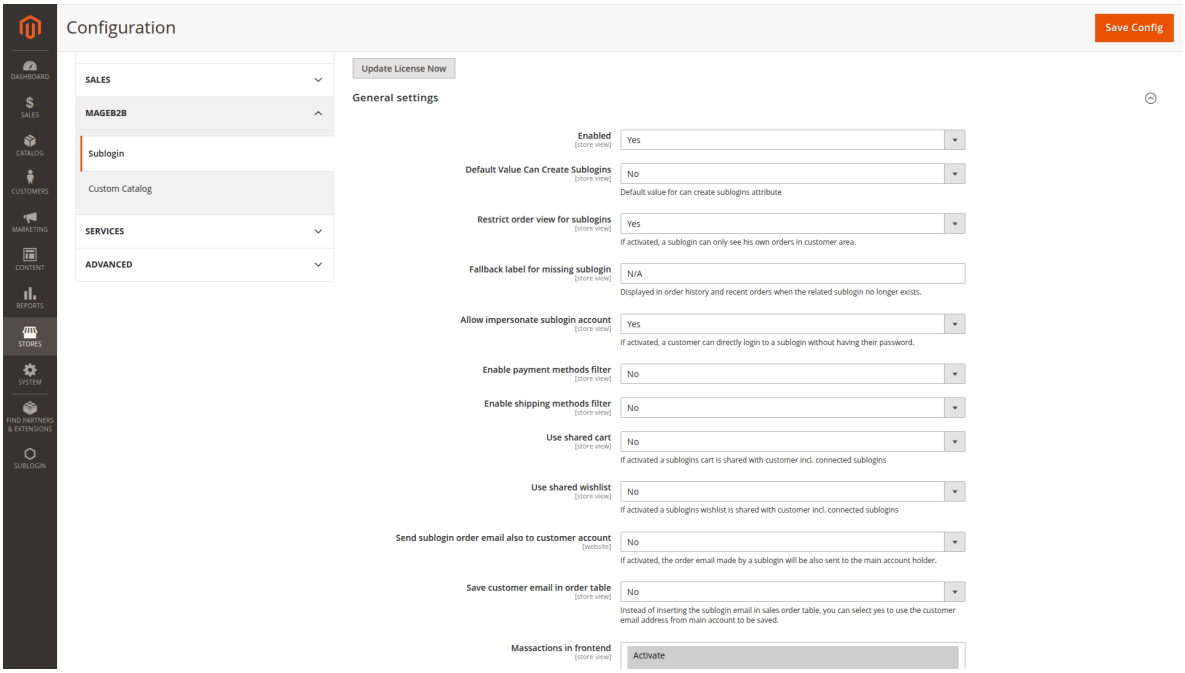
Configure the base account journey first. Leave Roles, Budget and Order Approval at their defaults until a sublogin can sign in and complete a normal checkout.

Safe starting configuration

For a first staging test:

- enable Sublogin;
- keep **Restrict order view for sublogins** enabled;
- keep payment and shipping filters disabled;
- keep cart and wishlist sharing disabled;
- keep email confirmation and public signup disabled;
- allow parent-account impersonation for support testing;
- use all parent customer addresses or assign a small approved set;
- keep plain-password emails disabled.

This gives one buyer a clear, isolated journey without introducing approval, budget or method-filter failures at the same time.



General settings

Setting	Default	What it changes
Enabled	Yes	Turns the base Sublogin functionality on for the selected scope.
Default Value Can Create Sublogins	No	Sets the initial value of the per-sublogin permission to manage other sublogins. Grant it only to trusted team leads.
Restrict order view for sublogins	Yes	Shows a sublogin only their own orders. Disable it only when every company user may see company-wide order history.
Fallback label for missing sublogin	`N/A`	Replaces the sublogin name in order views if the original record no longer exists. Use a phrase staff will understand, such as `Former company user`.
Allow impersonate sublogin account	Yes	Lets the main account enter a sublogin session without knowing that person's password. The storefront shows a context banner and a way back to the main session.
Enable payment methods filter	No	Applies the payment-method list stored on each sublogin. Enable only after every active sublogin has been reviewed.
Enable shipping methods filter	No	Applies the shipping-method list stored on each sublogin. Saved values use full carrier and method codes.
Use shared cart	No	Uses one cart for the main account and attached sublogins. One user's changes become visible to the others.
Use shared wishlist	No	Uses one wishlist for the account family.
Send sublogin order email also to customer account	No	Copies the main customer on order confirmation for a sublogin order.
Save customer email in order table	No	Stores the parent customer's email on the order instead of the sublogin email. The sublogin ID still records who placed it. Check exports and integrations before changing this.
Massactions in frontend	Activate, Deactivate	Selects bulk actions the main account may use. Other options include newsletter, email preference and the ability to manage sublogins.

Setting	Default	What it changes
Require email confirmation of new created sublogin	No	Keeps access blocked until the sublogin follows the confirmation flow. Use this when the email owner must prove access before signing in.
Sublogin can delete own account	No	Adds self-deletion in the storefront. Review retention and support implications before enabling it.
Reactivate account when disabled	Yes	Reactivates eligible sublogins when the parent customer account is enabled again. Disable it if reinstatement requires an individual review.

Shared cart and wishlist

Shared resources are convenient when a purchasing team builds one order together, but ownership becomes less clear. Test simultaneous sessions and decide who may remove items or change quantities. Sharing does not add collaborative quote editing.

Payment and shipping filters

The global switches activate restrictions already assigned to each sublogin. Roll them out in this order:

1. Assign allowed methods to every active sublogin.
2. Test an address that produces the expected shipping rates.
3. Enable one filter in staging.
4. Confirm checkout still offers at least one valid option.
5. Enable the second filter only after the first works.

An empty result can block checkout. See Payment and Shipping Restrictions.

Public signup

Setting	Default	What it changes
Enable public sublogin signup	No	Makes a request form available to guests who name an existing parent customer email.

The form validates the parent email, the applicant's email and password, but a matching parent email is not proof that the applicant belongs to that company. A new request always starts inactive: the parent customer receives an approval email with a confirmation link, and an administrator can also activate the record in Admin.

The storefront route is `/sublogin/signup/index` under the active store base URL.

Admin settings

Filter Sublogin Addresses on Customer Edit is enabled by default. It hides sublogin-owned addresses from the normal customer-address grid, which keeps the parent customer's address list focused on parent records. Disable it only when administrators need to manage both address types from that grid.

Address settings

Choose **Address management for sublogins** first:

Mode	Result
Disallow using customer addresses	The sublogin cannot select parent customer addresses. Use this only when sublogin-owned addresses are enabled and maintained.
Allow to use all customer addresses	Every parent customer address is available to the sublogin. This is the default.
Use custom address management	The main account assigns selected parent addresses to each sublogin.

Then review the related settings:

Setting	Default	What it changes
Use sublogin address as a customer	No	Lets the parent customer use addresses created for sublogins.
Bind sublogin to customer default billing address	No	Forces checkout to the parent account's default billing address.
Enable add new address option for sublogins in checkout	No	Lets a sublogin create an address during checkout. Enable only if buyers are allowed to expand the company's address book.
Show customer address under My Account for sublogin	Yes	Displays parent customer addresses in the sublogin account area, subject to restrictions.
Automatically add new customer address	No	In custom management mode, assigns a newly created parent address to restricted sublogin address books. Review whether broad automatic assignment is acceptable.
Set customer default address as sublogin default	No	Uses the parent defaults only when the sublogin has no default address.

Setting	Default	What it changes
Allow customer to delete an address used by sublogins	No	Allows parent users to remove an address still assigned to sublogins. Keeping this off avoids broken assignments.
Show main address under My Sublogins	No	Displays a headline address on the sublogin management page.
Not available address message	`N/A`	Text shown when that headline address is missing.

Test billing and shipping separately. A setup can allow a delivery address while still binding billing to the parent default.

See Address Management for workflow examples.

Report settings

Reports are included in the base package.

Setting	Default	Guidance
Enable reports for customer	Yes	Shows order and product reports to the main account.
Enable reports for sublogin	No	Shows reports in a sublogin session. Enable only if the report scope matches that user's visibility policy.

The relevant Admin ACL permissions must also be granted to administrator roles.

Email settings

Setting	Default	Guidance
Email Sender	Sales	Magento sender identity used by Sublogin messages.
Use Sublogin Store ID	No	Uses the sublogin record's store for templates and labels instead of the store active during the order. Useful only for deliberate cross-store account management.
New, confirmation, reset and expiry templates	Module defaults	Create store-specific overrides through Magento's email template system rather than editing vendor files.

Setting	Default	Guidance
Send plain password in account emails	No	Keep disabled. Use setup and reset links so passwords are not copied into mailboxes.
BCC Receiver	Empty	Separate multiple recipients with semicolons. Consider privacy and retention before copying account messages.
Add order details to selected templates	Empty	Adds order context only to the selected Sublogin messages.
Email templates for store owner	Empty	Sends store-owner alerts for selected events.
Customer optional email templates	Module event list	Decides which messages may use the optional parent-customer email field.
Sublogin optional email templates	Module event list	Decides which messages may use the optional sublogin email field.

Send a setup message, a password-reset message and an order message to a staging mailbox. Confirm the store name, sender, recipient and links for every store view you operate.

See Email and Account Recovery.

Address templates and content

The address template fields control text, one-line and HTML rendering for Sublogin address data. Keep Magento template variables intact and test the result anywhere it is used. The HTML field accepts a limited set of safe formatting tags.

Static block for sublogin grid at frontend accepts a CMS block identifier. Use it for short company instructions above the grid, such as whom to contact before creating a buyer. Leave it empty when no instruction is needed.

Debug logging

Debug logging is disabled by default and available only at default scope. Enable it for a specific investigation, reproduce the issue, collect the relevant entries, and disable it again. Do not leave verbose logs enabled as routine monitoring.

Go-live check

Before enabling Sublogin for customers:

1. Verify one main account with two sublogins.

2. Confirm email uniqueness and account recovery.
3. Test own-order and company-order visibility.
4. Test every address mode you intend to offer.
5. Confirm at least one payment and shipping combination for each restricted buyer type.
6. Verify impersonation starts and ends clearly.
7. Test shared cart and wishlist with two simultaneous sessions if enabled.
8. Review the public signup setting.
9. Run the same checkout checks with every installed add-on.

Continue with Create Your First Sublogin.

Create Your First Sublogin

Create the first sublogin in staging and use it to complete a real buyer journey. One successful record is more useful than importing a full organization before address, checkout and email rules have been tested.

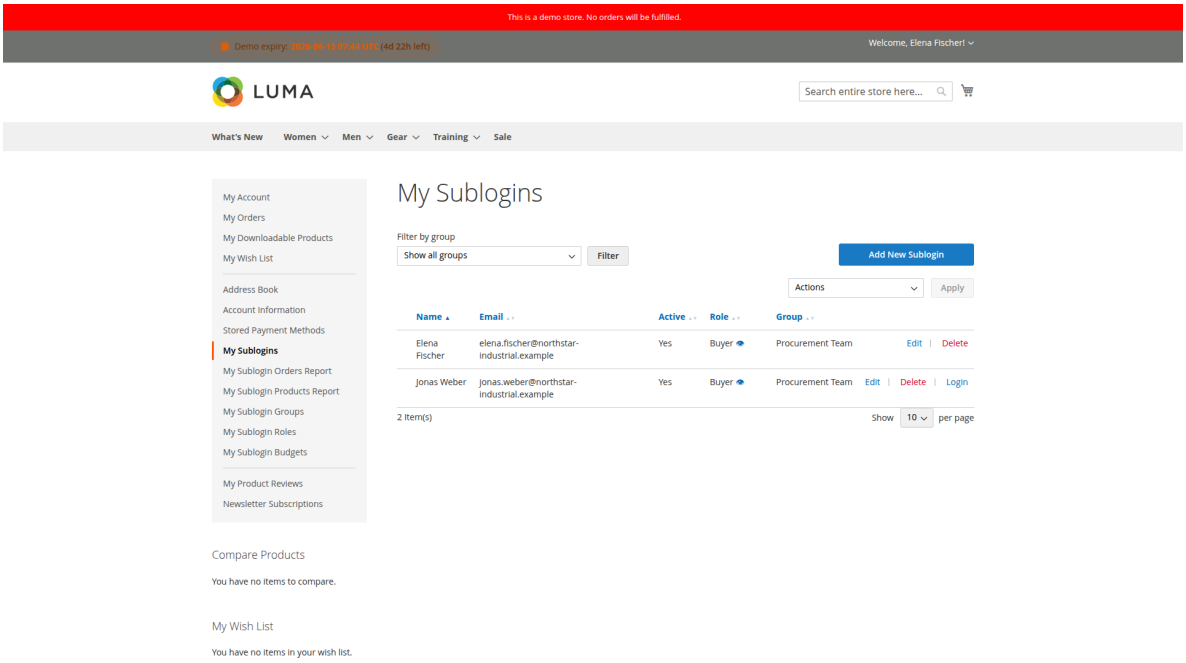
Use a fictional buyer with a normal-looking demo address, for example:

- Name: Elena Fischer
- Company role: Procurement Manager
- Email: `elena.fischer@northstar-industrial.example`

Do not use a real employee or customer record in documentation or training captures.

Create it from the customer account

1. Sign in as the main customer account.
2. Open **My Sublogins**.
3. Select **Add New Sublogin**.
4. Enter the buyer's name and unique email address.
5. Set an active status or expiry date according to your access policy.
6. Decide whether the person receives Sublogin emails and newsletters.
7. Decide whether they may manage other sublogins. Keep this off for ordinary buyers.
8. Assign addresses and, when the global filters are enabled, allowed payment and shipping methods.
9. Save the record.



The exact fields depend on installed add-ons. Role, group, budget, catalog and order-

approval controls appear only when their packages are active.

Set up the password safely

Use the configured account setup or confirmation message so the new user creates their own password. The password must satisfy Magento's configured minimum length and character-class rules.

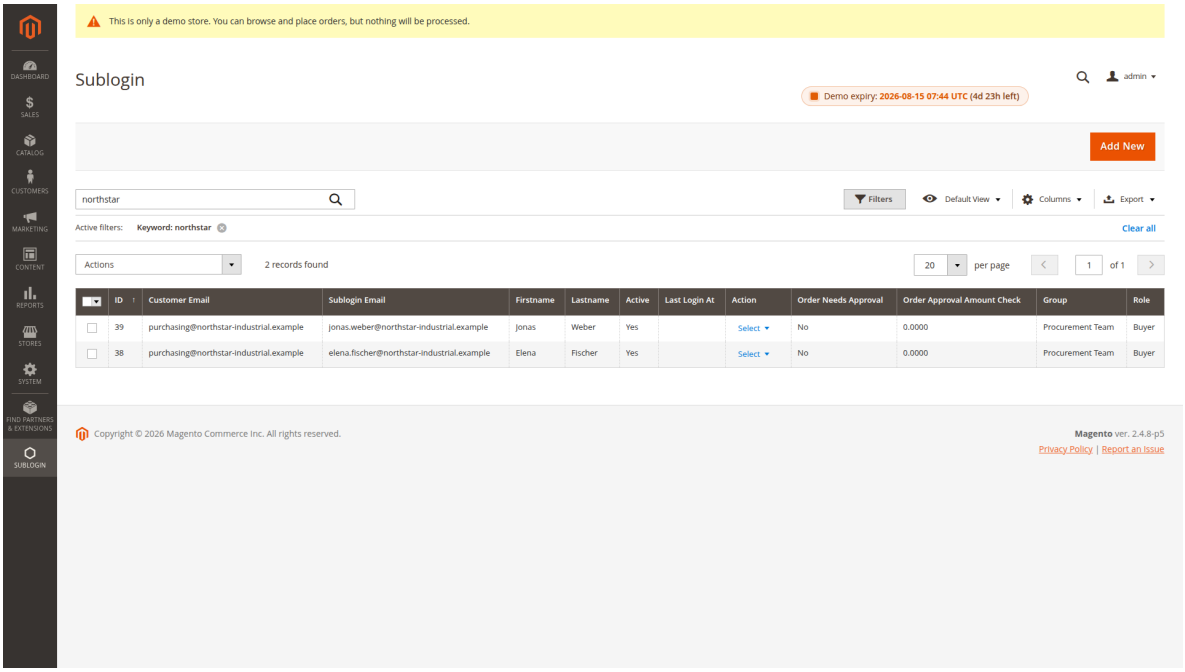
Keep **Send plain password in sublogin account emails** disabled. If no message arrives:

1. confirm that the email address is correct and unique;
2. check whether the sublogin is configured to receive emails;
3. inspect the Magento mail transport or staging mailbox;
4. send a fresh setup or reset message instead of sharing a password manually.

If email confirmation is required, the account cannot be used until its confirmation flow is complete.

Create or review it in Magento Admin

Administrators can manage records from **Sublogin > All Sublogins**. This is useful for support, bulk review and add-on fields that a customer may not control.



An administrator should confirm:

- the correct parent customer and website scope;
- active and expiry state;
- the sublogin email is not used by a customer or another sublogin in the same account scope;

- assigned addresses and checkout methods;
- any Role, Budget, Custom Catalog or Order Approval values contributed by add-ons.

Do not treat the Admin grid as a substitute for a storefront test. A record can look complete in Admin and still have no valid checkout combination.

Test direct sign-in

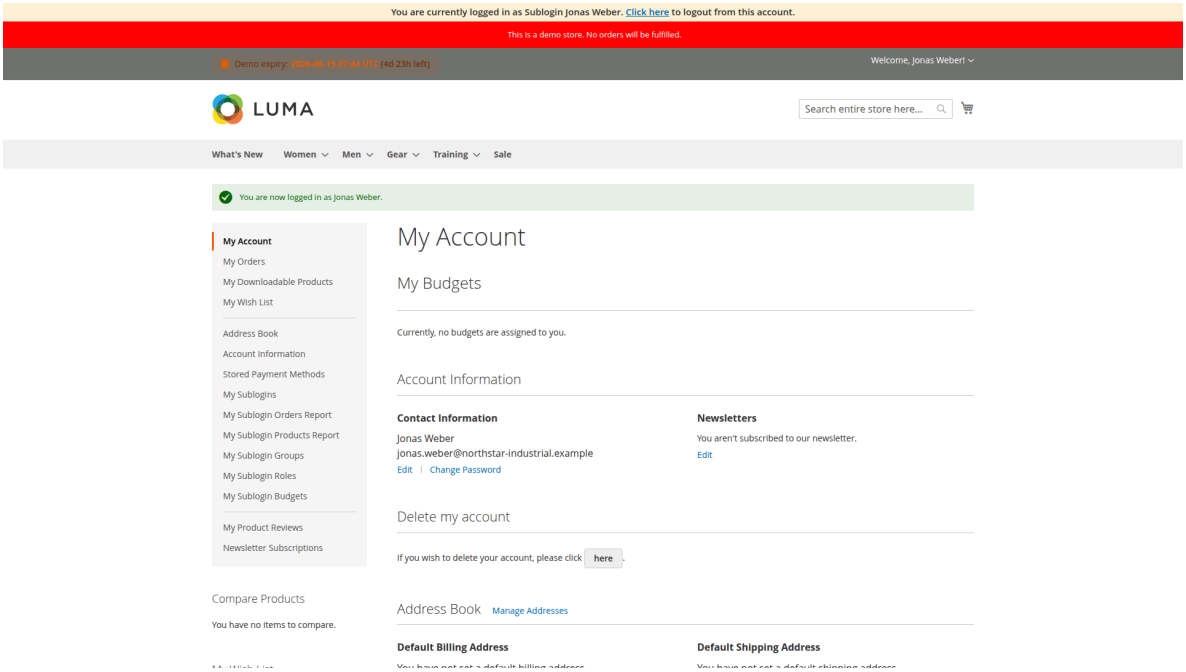
1. Sign out of the main account.
2. Open the normal Magento customer login page.
3. Sign in with the sublogin email and password.
4. Confirm the account page shows the sublogin identity.
5. Confirm order history follows **Restrict order view for sublogins**.
6. Add a product to the cart and proceed through address, shipping and payment selection.
7. Place a staging order and confirm its Sublogin value in Admin.

Sublogins use Magento's customer login entry point. They are not separate Magento customer entities, even though they have separate credentials.

Test main-account impersonation

When **Allow impersonate sublogin account** is enabled:

1. Sign in as the main customer.
2. Open **My Sublogins**.
3. Select **Login** for the new buyer.
4. Confirm the storefront clearly states that the session is acting as that sublogin.
5. Perform the support check you need.
6. Use the banner link to return to the main account.



Impersonation is intended for account owners and support workflows. It should never require learning or sending the sublogin's password.

Assign addresses deliberately

The available choices depend on **Address management for sublogins**:

- **Allow all customer addresses** makes every parent address available.
- **Custom address management** requires the main account to assign selected addresses.
- **Disallow customer addresses** requires a usable sublogin-owned address when the workflow permits one.

Test shipping and billing separately. The store may bind billing to the parent default even when the buyer can choose a different delivery address.

Add optional controls in this order

Once the base checkout works, add controls one at a time:

1. **Role and group**, if the buyer needs storefront restrictions or a place in an approval hierarchy.
2. **Custom Catalog**, if the buyer should see a limited assortment.
3. **Budget**, if spending must be checked at checkout.
4. **Order Approval**, if an order must wait for the main account or a parent group.
5. **One-Time Budget**, only for a temporary next-order allowance.

Place another order after each change. This makes it clear which rule caused a missing product, checkout method or approval state.

Common first-record failures

The email is rejected

The address already belongs to the parent customer, another Magento customer or another sublogin in the same account-sharing scope. Use a unique address or review **Stores > Configuration > Customers > Customer Configuration > Account Sharing Options**.

Sign-in fails after creation

Check active status, expiry date and required email confirmation. If the parent account was disabled and restored, also review **Reactivate account when disabled**.

No address is available at checkout

Review the address-management mode, selected parent addresses, permission to add a new address and any default-billing restriction.

No payment or shipping method is available

Temporarily disable the relevant Sublogin filter in staging. If checkout returns, assign at least one method that is also valid for the current cart, address and Magento shipping or payment configuration.

The buyer sees the wrong orders

Review **Restrict order view for sublogins**, then confirm the order carries the expected sublogin identity. Integrations that create orders outside the normal Sublogin session may need explicit ownership handling.

Next steps

- Address Management
- Roles and Permissions
- Budget Management
- Order Approval
- Common Issues

Address Management

Sublogin can keep purchasing addresses under the main customer account while controlling which buyer may use each one. Choose one ownership model, then test billing and shipping separately.

Choose the parent-address mode

Open **Stores > Configuration > MageB2B > Sublogin > Address settings**.

Mode	Use it when
Disallow using customer addresses	Each sublogin must use an address created for that sublogin. Make sure this workflow is enabled before selecting the mode.
Allow to use all customer addresses	Every buyer may ship to every address on the parent account. This is the default.
Use custom address management	The main account must assign an approved subset of parent addresses to each sublogin.

Custom management works well for branches and warehouses. A buyer assigned to Düsseldorf can use that address without seeing every delivery location in the company account.

Decide who may create and reuse addresses

- **Enable add new address option for sublogins in checkout** lets a buyer add an address during checkout.
- **Use sublogin address as a customer** lets the parent customer select addresses created for sublogins.
- **Automatically add new customer address** assigns newly created parent addresses to restricted sublogin address books in custom mode.
- **Allow customer to delete an address used by sublogins** controls whether the parent can remove a still-assigned address.

Keep address creation and deletion restricted when delivery locations require internal approval.

Billing can follow a stricter rule

Bind sublogin to customer default billing address forces billing to the parent customer's default even when the buyer may choose another shipping address. This is useful when branches receive deliveries but all invoices go to the head office.

If **Set customer default address as sublogin default** is enabled, the parent defaults

are used only when the sublogin has no defaults of its own.

Assign addresses to a sublogin

1. Select **Use custom address management**.
2. Open the sublogin from the main customer's **My Sublogins** page or Magento Admin.
3. Select the approved parent addresses.
4. Save the sublogin.
5. Sign in as that buyer and confirm only the expected addresses are available.
6. Test one order with different shipping and billing choices if your policy allows it.

Troubleshoot an empty checkout address list

Check these in order:

1. Does the parent account have a valid address for the current website?
2. In custom mode, is that address assigned to the sublogin?
3. In disallow mode, does the sublogin have its own usable address?
4. Is adding a new checkout address enabled when no stored address exists?
5. Is billing bound to a missing parent default?

Do not loosen every address setting at once. Change one condition in staging and repeat the checkout step.

Related: Configuration and Payment and Shipping Restrictions.

Email and Account Recovery

Sublogin sends account setup, confirmation, password reset, expiry and selected workflow messages through Magento's email system. Configure the sender and templates per store view, then test every link in a staging mailbox.

Protect credentials

Leave **Send plain password in sublogin account emails** disabled. New users should receive a setup or confirmation link and choose their own password. Passwords are checked against Magento's configured length and character-class policy and stored as hashes.

Password reset, setup confirmation and expiry refresh use separate tokens. Do not copy tokens into tickets, screenshots or logs.

Configure recipients

Under **Stores > Configuration > MageB2B > Sublogin > Email settings**:

- choose the Magento sender identity;
- select the template for each account event;
- decide whether the main customer receives a copy of sublogin order confirmation;
- use store-owner alerts only for events staff must act on;
- add BCC recipients only after reviewing privacy and retention;
- select which messages may use the optional parent or sublogin contact email.

The extension does not route invoice or shipment messages by a generic “Finance role”. Standard Magento sales email behavior and installed workflow modules still apply.

Choose the store context

By default, the store active for the order determines email templates and labels. **Use Sublogin Store ID** instead uses the store recorded on the sublogin. Enable it only when that is the intended behavior for accounts used across store views.

Test the complete sequence

1. Create an inactive staging sublogin at a reserved `.example` address that your local mail capture accepts.
2. Send the setup or confirmation message.
3. Open the link and set a compliant password.
4. Request a password reset and confirm the old link cannot be reused after completion.
5. Place a staging order and verify the main-account copy setting.
6. Check sender name, store label and URLs in every store view.

If a message is missing, verify the sublogin's email preference, selected template, Magento mail transport and queue or cron state before resending it.

Related: Configuration and Order Approval.

Public Sublogin Signup

The base package can show a guest form for someone who wants to join an existing customer account. The feature is disabled by default and should normally create an inactive request for the parent customer or an administrator to review.

Enable the request form

Open **Stores > Configuration > MageB2B > Sublogin > Public signup**.

1. Set **Enable public sublogin signup** to **Yes** for the intended website.
2. Save configuration.
3. Open `/sublogin/signup/index` under the store's base URL.

The applicant enters the existing parent customer's email, their own details, a unique email, password and password confirmation.

Approval flow

A new request always starts inactive. The parent customer receives the **public signup approval** email containing a confirmation link; following that link activates the sublogin. An administrator can also activate the record in Admin.

Understand the security boundary

The form checks that the named parent email exists. Knowing that email does not prove the applicant belongs to the company — the approval step is the control that establishes membership.

Before approving, verify the person through your normal company process, such as an approved contact or internal directory. The confirmation link is a credential: do not forward the approval email outside the account owner's hands.

What a public request does not grant

A new request does not receive permission to create sublogins and is not subscribed to the newsletter by default. Role, group, budget, catalog and approval values still need an explicit assignment where those add-ons are installed.

Test the flow

1. Use a fictional parent customer in staging.
2. Submit a request with a normal-looking demo email.
3. Confirm duplicate emails and mismatched passwords are rejected.
4. Confirm the new record is inactive.
5. Review and activate it through the intended account-management route.

6. Test sign-in and checkout.

Related: Email and Account Recovery and Create Your First Sublogin.

Extend the Frontend Form

Use a Magento plugin on `MageB2B\Sublogin\Block\Sublogin\Edit::getFormFields()` when a custom module needs to change the fields rendered on the customer-facing Sublogin create or edit form.

This hook controls presentation only. A new field still needs its own validation, persistence, loading, authorization and API or import behavior where applicable.

Inspect the current field array first

The returned fields depend on store configuration and installed add-ons. Base fields can include name, unique email, password handling, active and expiry state, email preference, delegated sublogin management, parent addresses and payment or shipping restrictions.

Role, Budget, Order Approval and Custom Catalog can add their own definitions. Inspect the result from your installed version before choosing a position or renderer.

Register a frontend plugin

In your custom module's `etc/frontend/di.xml`:

```
<?xml version="1.0"?>
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="urn:magento:framework:ObjectManag
er/etc/config.xsd">
    <type name="MageB2B\Sublogin\Block\Sublogin\Edit">
        <plugin name="company_sublogin_add_form_field"
type="Company\SubloginFields\Plugin\AddFormField" />
    </type>
</config>
```

Then append a field without replacing definitions contributed by other modules:

```
<?php

declare(strict_types=1);

namespace Company\SubloginFields\Plugin;

use MageB2B\Sublogin\Block\Sublogin\Edit;

final class AddFormField
{
```

```
    public function afterGetFormFields(Edit $subject, array
$fields): array
    {
        $fields[] = [
            'name' => 'purchasing_reference',
            'label' => __('Purchasing Reference'),
            'type' => 'text',
            'default' => '',
            'visible' => true,
            'position' => 1400,
        ];

        return $fields;
    }
}
```

Common definition keys include `name`, `label`, `type`, `default`, `options`, `visible` and `position`. Copy renderer conventions from the current array instead of assuming every Magento form type is supported.

Register the field for saving

Rendering the field is not enough. Since Sublogin 2.3.9, storefront create and edit requests pass through a POST field allow-list in `MageB2B\Sublogin\Controller\Sublogin\PostDataProcessor`: any posted key that is not registered is silently dropped before the sublogin is saved. A custom field therefore renders but never persists unless you register it.

Add the field name to the `frontendAllowedFields` argument in your module's `etc/frontend/di.xml` (or global `di.xml`):

```
<type
name="MageB2B\Sublogin\Controller\Sublogin\PostDataProcessor">
    <arguments>
        <argument name="frontendAllowedFields" xsi:type="array">
            <item name="purchasing_reference"
xsi:type="string">purchasing_reference</item>
        </argument>
    </arguments>
</type>
```

This is the mechanism the first-party add-ons use: `mageb2b/sublogin-role` registers `role_id` and `group_id`, `mageb2b/sublogin-orderapproval` registers `order_needs_approval` and `order_approval_amount_check`. Registering a field only allows it through the filter — validation and persistence are still your code's job.

Complete the data contract

Before releasing the field, decide:

1. where the value is stored;
2. which main customer, sublogin or administrator may read and change it;
3. how create and edit requests validate it;
4. whether Admin, REST, SOAP, import and export need the value;
5. how it behaves when a customer is transformed into a sublogin or back;
6. whether it contains personal or sensitive data.

Do not write directly into vendor module tables without a reviewed schema and service boundary.

Verify

Test create and edit as the main customer and as a delegated sublogin. Submit invalid input, reload the saved record, and check every installed storefront theme. If Hyvä is used, confirm the relevant compatibility package and custom template render the new field.

Login Context and Impersonation

A sublogin signs in through Magento's normal customer login page. The session keeps the parent customer identity and the current sublogin identity together so checkout, orders and add-ons can apply the correct restrictions.

Direct sublogin sign-in

The email must be unique within Magento's customer account-sharing scope. The account must also be active, unexpired and, when enabled, confirmed by email.

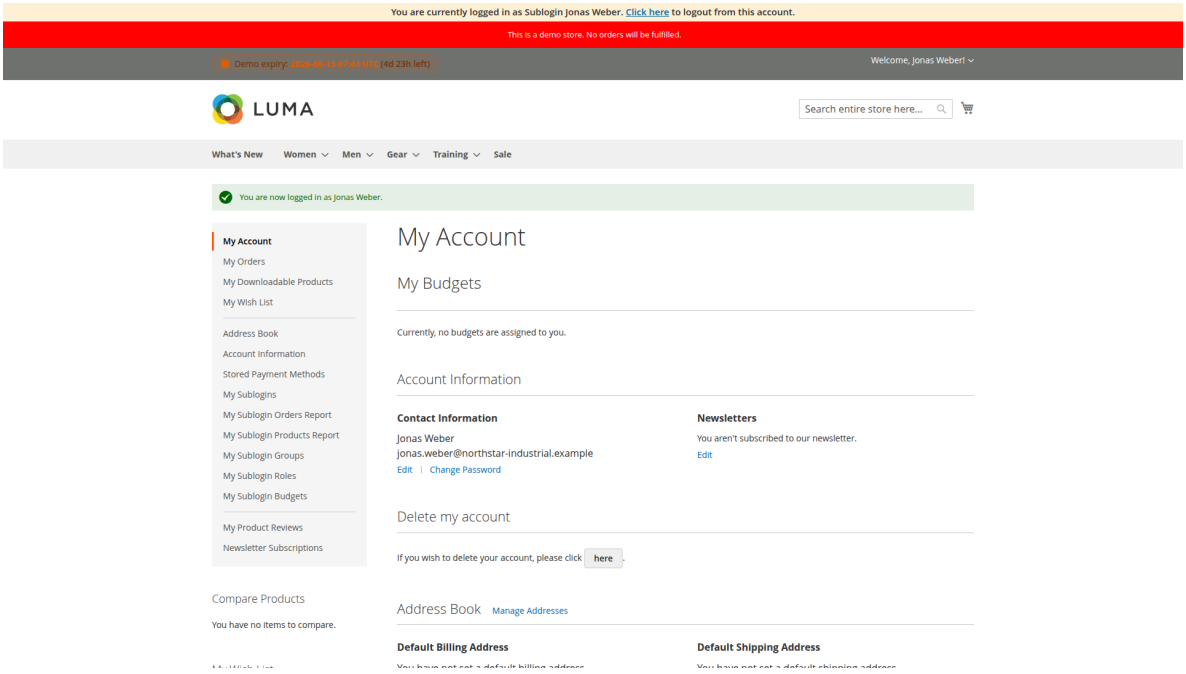
When a sublogin is active, the session can affect:

- address availability;
- payment and shipping methods;
- cart and wishlist ownership;
- order ownership and history;
- permissions contributed by Role and other add-ons;
- budgets, catalogs and approval state.

Stale or invalid sublogin context is cleared rather than carried into an unrelated customer session.

Main-account impersonation

With **Allow impersonate sublogin account** enabled, the main customer can select **Login** beside a sublogin. The storefront displays the active sublogin context and a link back to the main account.



Use impersonation to reproduce a buyer's storefront view or help build a cart. It does not reveal or replace the buyer's password.

Before leaving the session, return through the context banner and confirm the main account identity is restored. Shared cart or wishlist changes made during impersonation remain shared when those features are enabled.

Admin impersonation

Magento administrator permissions are controlled by Admin ACL, and managing a Sublogin record is not the same as acting as that buyer in the storefront. Administrators can, however, start an impersonation session from Admin: the **Login as Sublogin** button on the Admin sublogin edit page opens the storefront logged in as that sublogin.

The button is available only on an existing record and requires the `MageB2B_Sublogin::save` ACL resource. It uses Magento's Login as Customer infrastructure (`Magento_LoginAsCustomer*`): the admin request generates a one-time secret and redirects to the storefront `loginascustomer/login/index` route with the sublogin ID attached, on the store view that matches the sublogin's scope and Magento's account-sharing configuration. The resulting session carries the sublogin context, so role, catalog, budget and address restrictions apply exactly as they do for the buyer.

The optional Sublogin API manages records under its own Web API ACL resources. It does not turn a customer storefront session into a general integration credential.

Diagnose the wrong context

1. Sign out completely and start a fresh browser session.
2. Confirm the email belongs to the intended customer website scope.
3. Check active, expiry and confirmation state.
4. Inspect the visible impersonation banner.
5. Confirm the order or quote carries the expected sublogin ID.

Related: Multi-Account Management and Permission Problems.

Multi-Account Management

The main customer account can maintain separate buyers without creating a separate Magento customer for each employee. Each sublogin gets its own credentials and identity while staying attached to the company account.

What the base package controls

For each sublogin, the available base fields can include:

- first and last name;
- unique email and password setup;
- store context;
- active and expiry state;
- email and newsletter preferences;
- permission to manage other sublogins;
- selected parent addresses;
- allowed payment and shipping methods.

Installed add-ons contribute their own fields, such as role, group, budget, approval threshold or catalog assignment.

Delegate account management carefully

The main account can allow a sublogin to create and manage other sublogins. The global **Default Value Can Create Sublogins** is off, which is a sensible default for ordinary buyers.

When delegated management is needed:

1. grant it to a small number of named team leads;
2. decide which frontend mass actions they may use;
3. test whether add-on fields are visible and editable as intended;
4. keep account activation and deletion aligned with your offboarding process.

Per-customer settings in Admin

The Admin customer edit form contains a **Sublogin** tab with a **Sublogin Configuration** fieldset. These account-level values apply to the parent customer rather than a single sublogin:

Field	What it changes
Max Number of Sublogins	Caps how many sublogins the account may have. `0` means unlimited. The limit is enforced server-side when a sublogin is created.

Field	What it changes
Sublogin Optional Email	Additional customer email address usable by the configured optional-email templates.
Manage Sublogins	Whether the customer may create and manage sublogins at all.

Installed add-ons extend the same fieldset: Role adds **Manage Roles & Permissions**, Budget adds **Manage Budgets**.

These customer attributes are admin-managed. Customer-side API writes that try to set them through `custom_attributes` are rejected — see Protected Customer Attributes.

Manage records in Admin

Sublogin > All Sublogins lists sublogins across customers. Its mass actions are **Delete**, **Activate**, **Deactivate** and **Unset expire date**; the storefront's newsletter and management mass actions are configured separately under **Massactions in frontend**.

The Admin edit page adds tools a storefront user does not have: a **Reset Password** button that sends a Magento password-reset message, a **Login as Sublogin** button for storefront impersonation (see Login Context and Impersonation), and a **Sublogin Shopping Cart** section listing the sublogin's active cart contents with mass delete for stale entries.

When an administrator creates an order in Admin, the order form includes a **Sublogin** select listing the chosen customer's sublogins, so back-office orders can be attributed to the right buyer. The Admin order view shows the stored sublogin in the customer information area, linked to its edit page.

Base accounts are flat

The base package attaches sublogins directly to the main customer. Hierarchical groups and reusable roles require `mageb2b/sublogin-role`.

A common structure with that add-on is:

```
Main customer account
└── Operations
    ├── Procurement
    └── Warehouse
```

The hierarchy can also support approval routing when Order Approval is installed and group approval is enabled. It does not create arbitrary amount-based approval tiers.

Offboard a buyer

Deactivate access before deleting a record. Historic orders retain the sublogin association and use the configured fallback label if the record is later removed. Check reports, exports and integrations before permanent deletion.

For a person who should become an independent Magento customer, use the reviewed Transform Accounts workflow in staging rather than manually moving identifiers.

Related: Create Your First Sublogin and Roles and Permissions.

Payment and Shipping Restrictions

The base Sublogin package can store allowed payment and shipping methods for each buyer. Two global switches decide whether Magento enforces those lists.

These are per-sublogin restrictions, not Role rules and not spending limits.

Configure without blocking checkout

1. Leave both global filters disabled.
2. Edit every active sublogin and assign the methods that fit that buyer.
3. Test the buyer's usual cart and delivery address.
4. Enable **Payment methods filter** in staging and confirm one valid method remains.
5. Enable **Shipping methods filter** and repeat the test.

A configured method can still disappear because Magento considers it unavailable for the cart, country, currency, customer group or address. Sublogin only narrows Magento's valid result.

Payment examples

A purchasing account might be allowed to use purchase order and bank transfer but not a card. The exact method codes come from installed Magento payment modules.

Budget's **Pay on Budget** method is separate. It appears only when the Budget add-on is installed, enabled and valid for the current sublogin and cart.

Shipping examples

A warehouse buyer might use a freight method while an office buyer uses parcel delivery. Store the complete carrier and method combination, such as the value Magento exposes for that exact rate.

Address restrictions do not prove that a shipping rate is available. Always test the assigned address with the intended products and quantities.

No method appears

Check in this order:

1. Does Magento offer the method when the Sublogin filter is disabled?
2. Is the full method code assigned to this sublogin?
3. Is the global filter enabled at the correct website or store-view scope?
4. Does the address satisfy the carrier or payment module's own rules?
5. Does another module, such as Budget or a checkout customization, remove the method later?

Related: Address Management, Budget Management, and Common Issues.

Shared Cart and Wishlist

Sublogin can share one cart and one wishlist across the main customer and its attached sublogins. Both features are disabled by default.

Shared cart

Enable **Use shared cart** when a team intentionally builds one order together. Products, quantities and removals become part of the same working cart, so one user's change affects everyone in that account family.

Before enabling it:

- decide who is responsible for final checkout;
- test two simultaneous sessions;
- confirm customer-specific prices and catalog restrictions produce an acceptable shared result;
- explain to buyers that another user may change the cart.

A shared cart does not merge separate Magento customer accounts.

Shared wishlist

Enable **Use shared wishlist** when the company treats the wishlist as a common replenishment list. Keep it disabled when buyers maintain personal shortlists.

Resources that use separate controls

- Parent addresses use the address-management settings.
- Order history uses **Restrict order view for sublogins**.
- Payment and shipping methods use per-sublogin filters.
- Products can be restricted by Role or Sublogin Custom Catalog.
- Quotes are not a verified shared resource in the base package. B2B Quote workflows have their own extension and documentation.

Rollout check

Sign in as two sublogins in separate browser sessions. Add, change and remove items, then verify which user is recorded when the shared cart becomes an order. Repeat after enabling any catalog, pricing, budget or approval add-on.

Related: Multi-Account Management and Configuration.

Order and Product Reports

Order and product reports are features of the current `mageb2b/sublogin` base package. There is no separate Reports add-on to install. The base package replaces and conflicts with the retired `mageb2b/sublogin-report` package name.

Choose who may open reports

Open **Stores > Configuration > MageB2B > Sublogin > Report settings**.

Setting	Default	Effect
Enable reports for customer	Yes	Makes the report screens available to the main customer account.
Enable reports for sublogin	No	Makes report screens available during a sublogin session.

Keep sublogin access disabled when company-wide purchasing data should remain visible only to the main account. If you enable it, sign in as a restricted buyer and confirm the data scope matches your policy.

Available views

The base module provides order and product reporting in Magento Admin and customer-account areas. Admin roles also need the matching Sublogin report ACL resources.

Use the reports to answer practical questions such as which sublogin placed an order or which products were ordered through the company account. Reports do not replace an accounting export or a general analytics platform.

Export and column layout

Both storefront report grids can download their current rows as CSV — `sublogin/report/exportordercsv` for orders and `sublogin/report/exportproductcsv` for products. The export respects the same report ACL as the screen, and matching CSV export is available in the Admin report grids.

Each user can hide or show grid columns through the column-visibility control above the grid. The choice is saved per user — on the sublogin record for a sublogin session and on the customer attribute for the main account — so a restricted buyer's layout does not affect the main account's.

Upgrade from the former package

If an older installation still lists `MageB2B_SubloginReport`:

1. take a database and code backup;
2. update `mageb2b/sublogin` to a version that includes reports;
3. let Composer remove the retired package according to the resolved dependency set;
4. run `php bin/magento setup:upgrade`;
5. review report configuration and administrator ACL;
6. test the customer and Admin report routes.

Do not force both package names into the same Composer installation.

Related: Configuration and Multi-Account Management.

Sublogin Web API

Install `mageb2b/sublogin-api` when an ERP, identity workflow or administration service must create and maintain sublogin records through Magento's REST or SOAP layer. The package exposes sublogin record CRUD. Role, group and permission records have their own API package: `mageb2b/sublogin-role-api` (see Role API). Budgets and approvals do not expose API routes.

Install and authorize

```
composer require mageb2b/sublogin-api
php bin/magento module:enable MageB2B_SubloginApi
php bin/magento setup:upgrade
```

Create a dedicated Magento integration and grant only the Sublogin resources it needs:

Operation	Route	ACL resource
Read one	`GET /V1/sublogin/:id`	`MageB2B_Sublogin::manage`
Search	`GET /V1/sublogin/search`	`MageB2B_Sublogin::manage`
Create	`POST /V1/sublogin`	`MageB2B_Sublogin::save`
Update	`PUT /V1/sublogin/:id`	`MageB2B_Sublogin::save`
Delete	`DELETE /V1/sublogin/:id`	`MageB2B_Sublogin::delete`

Use HTTPS and store the integration token outside source code and logs.

Identity and scope fields

Field	Meaning
`id`	Sublogin record ID. The update and delete routes use this value.
`entity_id`	Parent Magento customer entity ID.
`website_id`	Magento website used for email uniqueness and account scope.
`store_id`	Store context used by the sublogin and, depending on configuration, email templates.
`email`	Sublogin login address. It must not collide with a customer or sublogin in the same account-sharing scope.

Field	Meaning
`firstname`, `lastname`, `prefix`	Person details.
`active`, `expire_date`	Access state and optional expiry date.
`send_backendmails`, `is_subscribed`, `create_sublogins`	Email, newsletter and delegated-management flags.
`customer_address_ids`	Parent customer address IDs assigned to the sublogin.
`addresses`	Sublogin-owned address objects.
`default_billing`, `default_shipping`	Default sublogin address IDs.

The data interface also contains internal reset-token and last-login fields. Do not send, copy or log reset tokens in integrations.

Fields added by other packages

The base package and add-ons expose additional data through Magento extension attributes rather than the primary Sublogin payload:

Attribute	On	Added by
`role_id`, `group_id`	`SubloginInterface`	`mageb2b/sublogin-role`
`order_needs_approval`, `order_approval_amount_check`	`SubloginInterface`	`mageb2b/sublogin-orderapproval`
`sales_report_config`	`SubloginInterface`	base package (stores the user's report column visibility)
`sublogin_id`	`OrderInterface`, `CartInterface`, `AddressInterface`	base package
`sublogin_approval_token`, `sublogin_approver_ids`	`OrderInterface`	`mageb2b/sublogin-orderapproval`
`can_create_sublogins`, `max_number_sublogins`, `sublogin_optional_email`	`CustomerInterface`	base package
`can_manage_role`	`CustomerInterface`	`mageb2b/sublogin-role`

Read these through the object's `extension_attributes` in REST responses. The customer-level attributes are admin-managed: customer-side writes that submit them through `custom_attributes` are rejected — see Protected Customer Attributes.

Sublogin API tokens

API authentication with sublogin credentials issues an `oauth_token` bound to that sublogin identity (`sublogin_id` plus the parent `customer_id`). The token keeps the sublogin's permissions during REST requests — it is not equivalent to the parent customer's token. Tokens are invalidated when the sublogin is deactivated, expires or is deleted, so a removed buyer cannot keep calling the API.

Address shape

An address supports name, company, street, city, country, region, postcode, telephone and default flags. In this API contract, `street` is one string, not Magento's usual array of street lines.

Create a sublogin

Replace the IDs and secrets with values from your staging system:

```
curl --request POST "https://store.example/rest/V1/sublogin" \
  --header "Authorization: Bearer <integration-token>" \
  --header "Content-Type: application/json" \
  --data '{
    "sublogin": {
      "entity_id": 123,
      "website_id": 1,
      "store_id": 1,
      "email": "elena.fischer@northstar-industrial.example",
      "password": "<new-password>",
      "firstname": "Elena",
      "lastname": "Fischer",
      "active": true,
      "customer_address_ids": [456],
      "addresses": [
        {
          "firstname": "Elena",
          "lastname": "Fischer",
          "company": "Northstar Industrial Supplies GmbH",
          "street": "Werkstraße 18",
          "city": "Düsseldorf",
          "country_id": "DE",
          "postcode": "40210",
          "telephone": "+49 211 5550100",
          "default_shipping": true,
          "default_billing": false
        }
```

```
        ]
      }
    }'
```

The password is validated against Magento's customer password policy and stored as a hash. Prefer a secure account-setup flow when an external system should not know the user's final password.

Search with Magento search criteria

```
curl --get "https://store.example/rest/V1/sublogin/search" \
  --header "Authorization: Bearer <integration-token>" \
  --data-urlencode
"searchCriteria[filterGroups][0][filters][0][field]=entity_id" \
  --data-urlencode
"searchCriteria[filterGroups][0][filters][0][value]=123" \
  --data-urlencode "searchCriteria[pageSize]=50" \
  --data-urlencode "searchCriteria[currentPage]=1"
```

Paginate every synchronization. Do not assume one response contains the full company account.

Update and delete safely

Use `PUT /V1/sublogin/:id` with a complete, validated `sublogin` payload. Read the current record first when your integration does not own every field.

Before `DELETE`, decide how historic orders should display after the person is removed. Deactivation is safer when access must end but the identity should remain available for audit and reporting.

Error checklist

- **401 or 403:** verify the integration token and the route's exact ACL resource.
- **Email already exists:** confirm `website_id` and Magento account-sharing scope, then search both customers and sublogins.
- **Password rejected:** use the Magento password policy configured for the store.
- **Address rejected:** send `street` as a string and provide the required country, postcode and telephone fields.
- **Record found in the wrong website:** always supply and filter by the intended website scope.

Related: Budget Integration Boundary and Import and Export.

Budget Integration Boundary

`mageb2b/sublogin-budget` does not register REST or SOAP routes. Endpoints such as `/V1/sublogin-budget` are not part of the current package and should not be used in an integration plan.

What is available

The Budget package exposes PHP interfaces for Magento modules running inside the same application. Custom code can use those service contracts through dependency injection to read or work with budget records while keeping the owning module's validation in place.

This is an internal Magento extension point, not a remote API contract.

If an external system must manage budgets

Build a small integration module that:

1. defines explicit Magento Web API routes;
2. uses dedicated Admin ACL resources;
3. validates customer, sublogin, budget type, period, amount and website scope;
4. calls the Budget service layer instead of writing its tables directly;
5. returns stable data-transfer objects rather than exposing models;
6. records operational errors without logging credentials or sensitive account data.

Do not expose generic model save operations or direct database access over HTTP.

Sublogin record API

The separate `mageb2b/sublogin-api` package does provide supported CRUD endpoints for sublogin records. It does not add budget fields or budget routes.

See Sublogin Web API and Budget Management.

Budget Management

`mageb2b/sublogin-budget` checks a sublogin's current cart against configured spending limits. Limits can belong to the sublogin, the main account or both, and can cover a fixed period, a recurring period or one order.

Install and test Budget after the base Sublogin checkout works.

Install

```
composer require mageb2b/sublogin-budget
php bin/magento module:enable MageB2B_SubloginBudget
php bin/magento setup:upgrade
```

Choose the budget types

Open **Stores > Configuration > MageB2B > Sublogin > Budget settings**.

Type	Meaning
Year	Fixed start and end within one selected year.
Yearly	Recurring yearly limit.
Month	Fixed selected month.
Monthly	Recurring monthly limit.
Day	Fixed selected day.
Daily	Recurring daily limit.
Per Order	Maximum for one order.

The default allowed-type list includes the six period types but not **Per Order**. Add it under **Allowed Budget Types** when your buyers need an order cap.

Use a fixed type for a project or a specific accounting period. Use a recurring type for an ongoing purchasing policy.

Decide whose budget applies

Budget Logic provides five choices:

Choice	Result
Disable	Budget validation is not applied.

Choice	Result
Apply main account budget limit when no sublogin budget found.	Use the buyer's budget when present, otherwise fall back to the parent account. This is the default.
Apply main account budget limit and skip sublogin budget.	Ignore sublogin budgets.
Use sublogin budget only.	Ignore main-account budgets.
Combine sublogin and main account budgets.	Evaluate applicable values from both owners.

Combined mode does not turn every record into one unrestricted total. Applicable amounts are combined by budget type and period; the effective per-order value uses the maximum applicable per-order allowance. Test the exact record combination you intend to use.

Assign budgets per website

Every budget record carries a required **Website** selection in the Admin budget form, and budgets apply only in that website's storefronts. A buyer who purchases in two websites can therefore have independent limits and independent usage per website.

Existing budgets created before website scoping was introduced are migrated to their owners' websites during the upgrade. On multi-website installations, verify a few migrated records and the Admin website selection before assuming a shared limit.

Choose what counts as spending

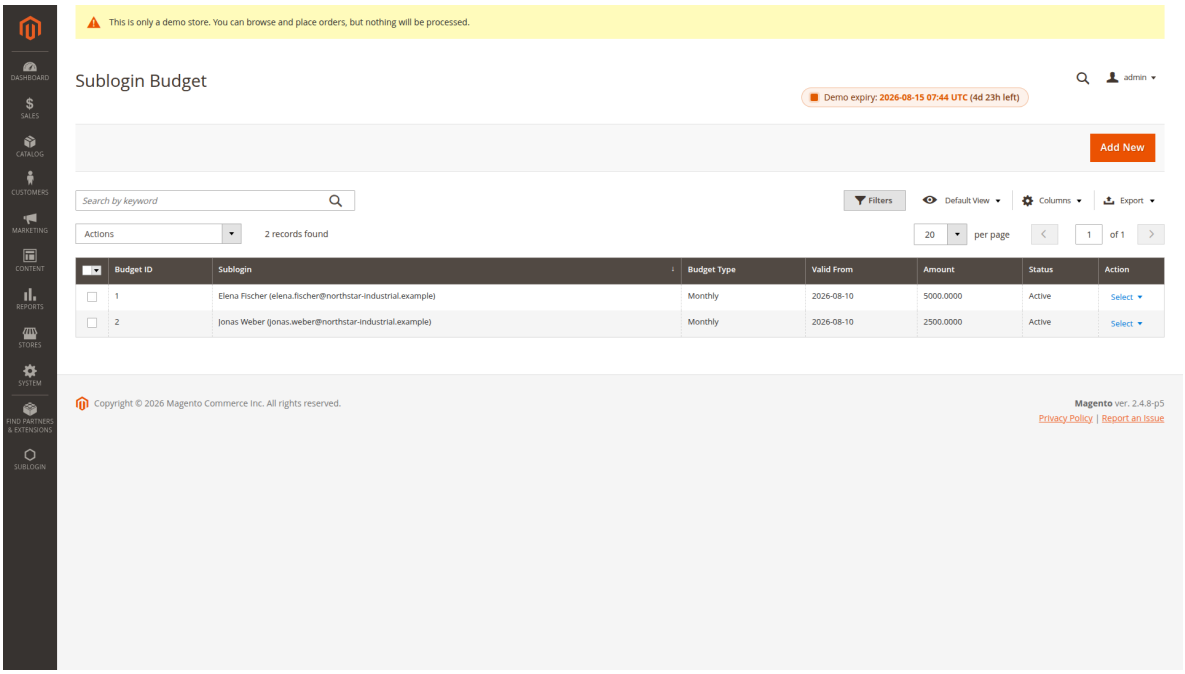
Setting	Default	Guidance
Budget Calculation Basis	Grand total	Choose subtotal when shipping and tax should not consume budget. Choose grand total for the final payable amount.
Order Status for budget consideration	Processing, Complete, Closed	Only orders in selected statuses contribute to used spend. Align this list with your cancellation and payment flow.
Use only Pay on Budget orders for budget usage	No	When enabled, orders paid another way do not consume the tracked budget.

Changing statuses or calculation basis changes future validation and can also change how past orders contribute to the current period. Review the effect before changing a live

website.

Create and test a budget

1. Choose one staging sublogin.
2. Create one active Monthly budget for the current month.
3. Set a round amount that makes the test easy to understand.
4. Add a cart below the remaining budget and proceed to checkout.
5. Place the order and move it into a status counted by Budget.
6. Start a second cart that exceeds the remaining amount.
7. Confirm the configured period, spent amount and exceeded amount appear in the message.



Add more periods only after this single record behaves as expected.

Checkout enforcement

Budget validates the quote during checkout. An over-budget cart is normally blocked.

When Order Approval is installed, **Allow budget restricted order to checkout** can send an over-budget order into the approval state instead. In that path, the sublogin's normal approval flag and threshold are ignored for the budget-restricted order.

Enable order approval on empty budget applies only to that integration and only when the sublogin is configured to need approval. Test an account with no applicable budget so the fallback is deliberate.

Multishipping checkout follows the same rule: an over-budget cart disables the multishipping flow unless **Allow budget restricted order to checkout** routes the order

into approval instead. If buyers report a missing "Check Out with Multiple Addresses" option, check the current cart against the applicable budgets first.

Budget summaries and messages

Budget summaries are disabled by default for both the sublogin and main customer. Enable the appropriate storefront summary when users should see the limit, used amount and remaining amount.

The exceeded messages support placeholders shown in Admin. Keep the wording factual and tell the buyer whom to contact. Do not expose internal IDs or approval tokens.

Pay on Budget

Pay on Budget is an offline Magento payment method and is disabled by default under **Stores > Configuration > Sales > Payment Methods**.

It is available only when:

- the shopper is an active sublogin;
- an applicable budget exists;
- the current cart passes budget validation;
- the method is enabled for the website and country.

Enforce Pay on Budget as only payment method when budgets are available hides other payment methods for an eligible buyer. Enable it only after confirming Pay on Budget works for every affected checkout country and address.

Expired fixed budgets

The cron job `mageb2b_subloginbudget_disable_expired_budgets` runs daily at 04:00 and deactivates expired fixed Day, Month and Year records. Recurring Daily, Monthly and Yearly records remain active because their period rolls forward.

Make sure Magento cron runs reliably. If a fixed record stays active after its end date, check the cron schedule and job history before editing the database.

One-time allowance

The separate `mageb2b/sublogin-budget-one-time` package lets the latest active one-time record override normal budgets for the next successful order. See One-Time Budget.

Integration boundary

Budget exposes PHP interfaces for code inside Magento. It does not register a REST or SOAP budget endpoint. See Budget Integration Boundary.

Hyvä storefront

Install `mageb2b/sublogin-budget-hyva` when the store uses Hyvä. See Hyvä Compatibility Packages.

Troubleshooting

- **Used amount is zero:** confirm the order reached a counted status and review Pay-on-Budget-only mode.
- **The wrong total is checked:** review subtotal versus grand total.
- **No budget appears:** check owner, type, active flag, period and Budget Logic.
- **Checkout is blocked:** compare the current cart with every applicable period and per-order limit.
- **Pay on Budget is missing:** check payment-method enablement, country, active sublogin and budget errors.

Related: Order Approval and Payment and Shipping Restrictions.

Sublogin Custom Catalog

`mageb2b/sublogin-custom-catalog` links products or reusable catalogs to a sublogin. The configured mode decides what that assignment means: restricting the storefront to the assigned set, or only re-showing assigned products that another rule hides. The package can work on its own or combine its result with the separate SoftwareSilo Custom Catalog extension when that module is present.

The separate Custom Catalog extension is optional. Sublogin Custom Catalog owns its own catalog, assignment and direct-product records.

Install and enable

```
composer require mageb2b/sublogin-custom-catalog
php bin/magento module:enable MageB2B_SubloginCustomCatalog
php bin/magento setup:upgrade
```

Open **Stores > Configuration > MageB2B > Sublogin > Sublogin Custom Catalog**.

Setting	Default	Effect
Activate	Yes	Enables product filtering and the Sublogin catalog context.
Mode	Override hidden if assigned	Store-scoped visibility mode, see below.
Allow sublogin "My Products" access	Yes	Controls access when the Role add-on is not active.
Enforce ACL check	Yes	Checks `MageB2B_SubloginCustomCatalog::view` for a sublogin.

Visibility modes

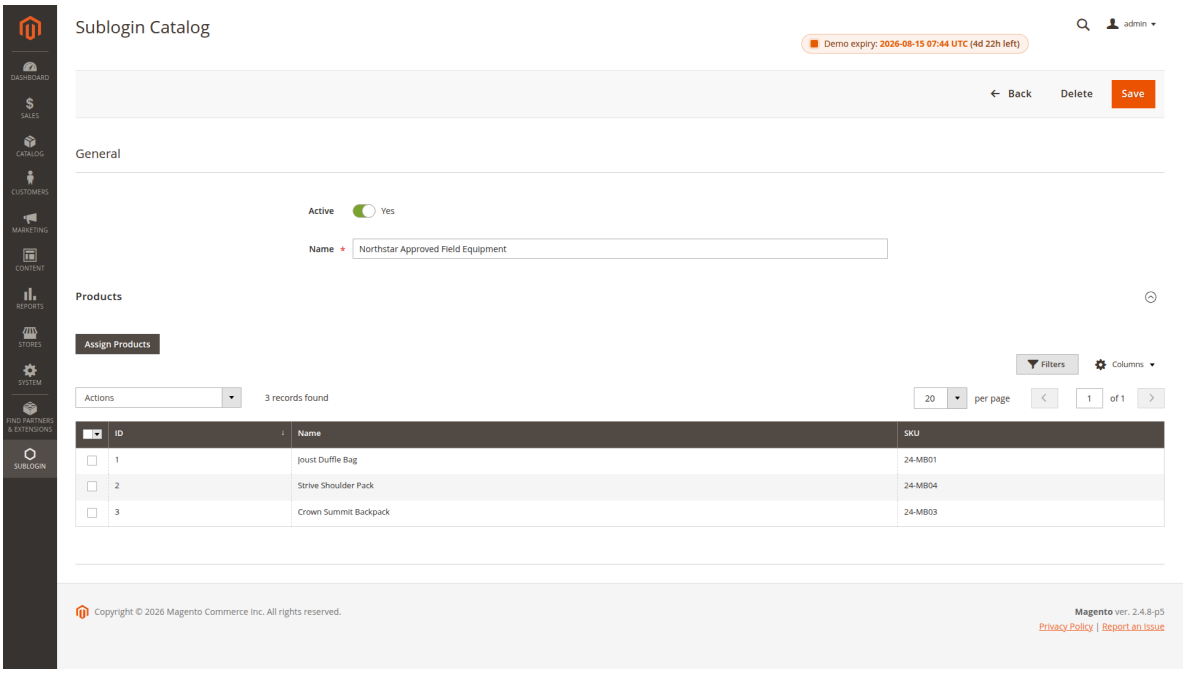
Mode	Storefront result
Override hidden if assigned	The catalog keeps its normal visibility; an assigned product is shown even when another rule — typically the separate Custom Catalog extension — would hide it. Assignment can un-hide products but does not remove unassigned ones.
Strict whitelist (assigned only)	Only assigned products are visible and purchasable. Category and search collections, direct product access and add-to-cart validation are all limited to the assigned set.

Choose the mode per store view. Strict whitelist gives a closed buying catalog without requiring the separate Custom Catalog extension; the override mode is mainly useful when that extension already hides products for the customer and selected buyers should see exceptions.

When Role is active, grant or deny **View My Products** through the assigned role. The permission is allowed by default, but your role definition remains the source of truth.

Build a catalog

1. Create a Sublogin catalog in Magento Admin.
2. Add products to the catalog.
3. Assign the catalog to one or more sublogins.
4. Add direct product assignments only for exceptions that do not belong in a reusable catalog.
5. Sign in as each affected buyer and test category, search, product detail and cart access.



The main customer can manage assignments in the supported storefront flow. A sublogin cannot manage its own allowed catalog.

What is filtered

In **Strict whitelist** mode the allowed product set is applied to storefront product collections, direct product access and cart validation. Test search and recently viewed or recommendation components supplied by other modules because they may build product collections differently. In **Override hidden if assigned** mode the assigned set can only add visibility, so the storefront does not shrink for the sublogin.

If an allowed product later becomes disabled, out of stock or unavailable on the store website, Magento can still hide it. Catalog assignment does not override core product availability.

Combine with SoftwareSilo Custom Catalog

When `MageB2B_CustomCatalog` is installed, the same **Stores > Configuration > MageB2B > Sublogin > Sublogin Custom Catalog** group gains two more fields: **Enable Custom Catalog Integration** and **Sublogin Merge Operator**. They appear only while the Custom Catalog module is enabled — the values are stored under `customcatalog/general/...` for the separate extension, but they are edited from the Sublogin section, not from **MageB2B > Custom Catalog**.

Operator	Effective products
Intersection	Only products allowed by both systems. This is the default and the strictest option.
Union	Products allowed by either system.
Sublogin Only	Ignore the separate Custom Catalog result for this calculation.
Custom Catalog Only	Ignore Sublogin-specific assignments and use the separate Custom Catalog result.

Use **Intersection** when company rules and individual buyer rules must both pass. Use **Union** only when either assignment is sufficient; it can broaden visibility.

A practical example

Northstar Industrial Supplies has a company catalog containing approved workshop equipment. Its maintenance buyer receives a smaller Sublogin catalog containing only consumables.

- **Intersection** shows approved consumables that occur in both sets.
- **Union** shows the complete company set plus the buyer's direct additions.
- **Sublogin Only** uses only the maintenance buyer's assignments.
- **Custom Catalog Only** applies only the company catalog.

Test before rollout

Use three products:

1. one assigned by both systems;
2. one assigned only to the sublogin;
3. one assigned only through the separate Custom Catalog.

For every merge operator you intend to use, check category, search, direct URL and add-to-cart behavior. Also test a cart created before an assignment changed.

Hyvä storefront

Install `mageb2b/sublogin-custom-catalog-hyva` when the storefront uses Hyvä. It requires the base Sublogin Hyvä bridge as well as this package. See Hyvä Compatibility Packages.

Troubleshooting

- **My Products is missing:** check Activate, Allow Sublogin View and Role ACL.
- **Every product disappeared:** in Strict whitelist mode an empty assignment empties the storefront; in Intersection mode an empty result from either system produces an empty effective set.
- **Assignments seem to have no effect:** check the store-scoped Mode — in Override hidden if assigned the sublogin still sees the full normal catalog.
- **A direct URL still opens:** confirm filtering is active for the current store view and clear only the relevant Magento cache after configuration changes.
- **A previously added cart item fails:** the cart is revalidated against current visibility; review recent assignment changes.

Related: Roles and Permissions and Permission Problems.

Hyvä Compatibility Packages

Sublogin uses separate Hyvä bridge packages for the base storefront and four feature add-ons. Install one bridge for every Sublogin package whose storefront screens you use.

Package map

Storefront area	Hyvä package	Requires
My Sublogins, create/edit, account navigation, reports and order labels	`mageb2b/sublogin-hyva`	the Sublogin base extension
Roles, groups and permission-aware catalog/cart UI	`mageb2b/sublogin-role-hyva`	the Sublogin Role add-on
Budget pages and summaries	`mageb2b/sublogin-budget-hyva`	the Sublogin Budget add-on
Sublogin catalog pages	`mageb2b/sublogin-custom-catalog-hyva`	the Sublogin Custom Catalog add-on and the base Hyvä bridge
Approval notices and order actions	`mageb2b/sublogin-orderapproval-hyva`	the Order Approval add-on and the base Hyvä bridge

All five bridges require the Hyvä default theme. Let Composer resolve the exact compatible versions rather than pinning package versions yourself.

Install the base bridge

```
composer require mageb2b/sublogin-hyva
php bin/magento module:enable MageB2B_SubloginHyva
php bin/magento setup:upgrade
```

Install feature bridges in the same way. For example, a store using Role and Order Approval needs both corresponding Hyvä packages in addition to the base bridge.

Verify the storefront

Test with the same buyer in Luma-compatible staging and Hyvä staging where possible:

1. open **My Sublogins** and create or edit a record;
2. test roles and groups if installed;
3. check budget summaries and checkout enforcement;
4. verify catalog product visibility and cart access;
5. place an order that requires approval, then approve, decline and edit it as permitted;
6. test mobile navigation and validation messages.

A successful `setup:upgrade` only proves the modules are registered. It does not prove that every customized Hyvä checkout or account template still exposes the required Sublogin data.

This KB pass intentionally uses no separate Hyvä screenshots. The package and workflow guidance remains source-backed.

Import and Export

`mageb2b/sublogin-importexport` adds Magento's `sublogin` import and export entity, an Admin grid export action and a command-line CSV importer. Use it for reviewed bulk changes, not as an unmonitored identity feed.

Install

```
composer require mageb2b/sublogin-importexport
php bin/magento module:enable MageB2B_SubloginImportExport
php bin/magento setup:upgrade
```

Start from an export

Open **Sublogin > All Sublogins**, select a small set of representative records and use the grid's export action. The export includes up to three sublogin-owned address groups, using columns such as `address_1_*` through `address_3_*`.

Passwords are not exported. Treat the file as sensitive because it still contains names, email addresses, company relationships and address data.

Use a copy of this export as the template for your first staging import. Do not build a production CSV from the package's fixture data.

Required import data

For create and update behavior, the importer requires:

- `entity_id`, the parent Magento customer ID;
- `email`;
- `firstname` and `lastname`;
- `active`;
- `store_id`;
- `password` when automatic password generation is disabled.

Include `website_id` whenever accounts span multiple websites. Role and group IDs are supported when Role is installed. `order_needs_approval` and `order_approval_amount_check` are supported when Order Approval is installed.

Address columns support three numbered address groups. Verify country, region and default billing or shipping fields with a small import before adding hundreds of records.

Password behavior

Stores > Configuration > MageB2B > Sublogin > Import/Export settings > Generate Missing Import Passwords is enabled by default. A row without `password`

receives a generated password.

Generated credentials are not a substitute for an account-setup process. Decide how the buyer will securely establish access before importing active records.

Run the CLI importer

The command is:

```
php bin/magento sublogin:import-sublogin
var/import/sublogins.csv \
  --behavior=add_update \
  --delete_file_after_import=0
```

Supported behaviors are `append`, `add_update`, `replace` and `delete`.

The separator options are:

```
--field_separator=,
--field_multiple_value_separator=,
--fields_enclosure=0
```

Important file-deletion default

`--delete_file_after_import` defaults to `1`. After a successful import, the command deletes the source CSV. Pass `--delete_file_after_import=0` while validating a file or when your workflow archives the source elsewhere.

Keep import files outside public web directories and remove them according to your data-retention policy.

Understand the behaviors

- **append / add_update** saves valid rows and updates matching records according to Magento's import behavior.
- **replace** can replace matching entity data, including address relationships. Test it with a backup.
- **delete** removes matching sublogin records. Email is required.

When the same email can match more than one scoped record, a delete row must also provide `store_id`, `website_id` or `entity_id`. This prevents a broad email-only delete across scopes.

Verify the result

1. Read the command output and stop on validation errors.

2. Open **Sublogin > All Sublogins** and compare the imported parent, store and website.
3. Check active state, role, group, addresses and method restrictions.
4. Use one imported buyer to sign in and complete checkout.
5. Confirm password setup or reset works without exposing the generated password.

For recurring external synchronization, consider the authenticated Sublogin Web API instead of repeatedly moving CSV files.

One-Time Budget

`mageb2b/sublogin-budget-one-time` adds a temporary allowance intended for the sublogin's next successful order. It requires the Budget add-on, not Order Approval.

Install

```
composer require mageb2b/sublogin-budget-one-time
php bin/magento module:enable MageB2B_SubloginBudgetOneTime
php bin/magento setup:upgrade
```

How the allowance is evaluated

When more than one active one-time budget exists, the latest active record is used. Its amount becomes the effective per-order allowance and overrides the normal periodic budget calculation for that checkout.

After successful order placement, active one-time budgets for that sublogin are deactivated. The same cleanup applies to multishipping order placement.

The optional popup setting controls the explanatory message only. It does not turn enforcement on or off.

Mark a budget as one-time

A budget becomes a one-time allowance through the **One-Time Budget** (Yes/No) field rather than a separate record type:

- In Admin, the field appears on the sublogin budget form. Its notice states that the budget is used once, deactivates after an order and overrides the sublogin's other budget types while active.
- In the storefront, the budget create/edit form offers the same Yes/No select, and the budget list marks one-time records.

The flag is also available as the `is_one_time` extension attribute on the Sublogin Budget API contract.

Grant an allowance

1. Confirm the buyer's normal Budget configuration and calculation basis.
2. Create one active one-time budget with the approved amount.
3. Tell the buyer the allowance is for the next successful order.
4. Complete a staging checkout below the amount.
5. Confirm the order succeeds and the one-time record becomes inactive.
6. Start another cart and confirm normal budget rules apply again.

Avoid creating several active one-time records for the same person. Although the latest record wins, overlapping approvals are difficult to explain and audit.

Failed and abandoned checkouts

The allowance is consumed after successful order placement, not merely by opening checkout. If a payment integration creates an order before later failing externally, review the resulting Magento order and allowance state as part of reconciliation.

Related: Budget Management and Order Approval.

Order Approval

`mageb2b/sublogin-orderapproval` lets a sublogin place an order that waits for a decision before normal processing continues. Approval can go directly to the main customer or, with the Role add-on, through parent groups first.

The package supports one subtotal threshold per sublogin. It is not a general workflow builder with separate manager, director and finance thresholds.

Install

```
composer require mageb2b/sublogin-orderapproval
php bin/magento module:enable MageB2B_SubloginOrderApproval
php bin/magento setup:upgrade
```

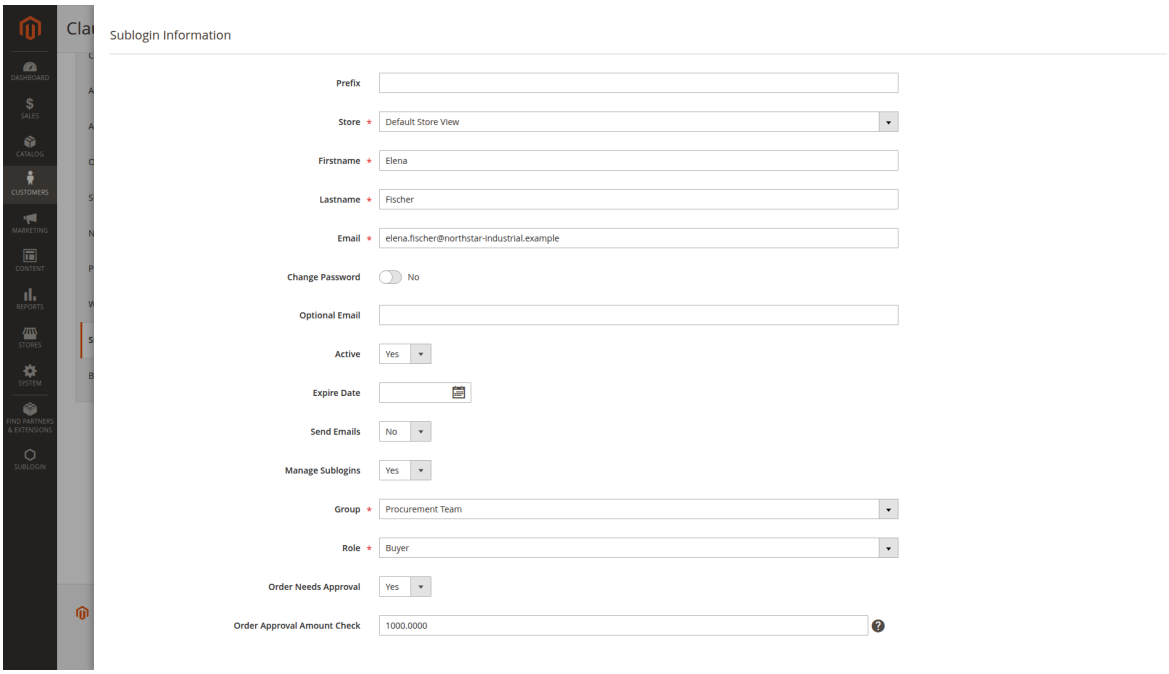
Role is optional. Install `mageb2b/sublogin-role` only when you need group-based routing or reusable approval permissions.

Decide when a buyer needs approval

Edit the sublogin and configure:

- **Order needs approval:** enables approval for this buyer;
- **Order approval amount check:** subtotal threshold.

A threshold of `0` sends every applicable order for approval. A positive threshold sends the order for approval when its subtotal exceeds that amount.



Test tax, discounts and shipping separately. The trigger uses subtotal, not the Budget

add-on's configurable calculation basis.

Direct approval without Role

When Role is not installed:

1. the sublogin completes checkout;
2. the order enters the configured approval state and status;
3. the normal Magento order confirmation is held back;
4. the main customer reviews the order;
5. approve moves it to the configured approved state, while decline moves it to the configured decline state;
6. final approval releases the normal order confirmation to the configured recipient.

The default final confirmation recipient is the sublogin.

Group approval with Role

When Role is active and **Final Approve/Decline order when all group approved an order** is enabled:

1. the order enters the pre-approval state;
2. eligible users in a parent group use the contributed approve or decline permission;
3. approval advances through the configured group hierarchy;
4. after group approval is complete, the main customer makes the final decision.

Can decline preapproval orders decides whether a group approver may decline before the order reaches final approval.

The hierarchy follows group parent relationships. There are no amount-specific group levels in this package.

Configure states and statuses

Open **Stores > Configuration > MageB2B > Sublogin > Order approval settings**.

Stage	Default state/status	Use
Pre-approval	`pre_approval`	Intermediate group decision when Role hierarchy is used.
Approval	`approval`	Waiting for the main customer's final decision.
Approved	`approved`	Final approval completed.
Declined	`not_approved`	Order was rejected.

The selected status must belong to the selected state in Magento. Test invoicing,

shipment, payment capture and external exports with these custom states before launch.

Cart notice and deletion

Order Approval Notice appears in cart and checkout. Leaving it blank hides the notice but does not disable approval.

Delete not approved orders allows the main customer to delete an order that was not approved. Keep it disabled when your audit or accounting process requires declined orders to remain visible.

On massaction check approval state restricts the Admin mass action to orders currently in the approval state. It is enabled by default.

Edit an order awaiting approval

Enable **Can Edit Order In Approval State** when the main customer may revise a pending order. The edit flow rebuilds order details from a quote and can change items, addresses and payment data while keeping the order identity.

This is a high-impact action. Test tax, discounts, inventory, payment integrations and exports before enabling it.

Auto approve edited order moves an edited approval order directly to approved after the main account finishes editing. It is disabled by default. Keep it off when the changed order needs a separate final review.

Comments

Enable comments for selects which actions request a comment, such as edit, approve or decline. Separate settings control whether decline and edit comments are visible in the storefront.

Use comments for a useful purchasing reason, not sensitive personal data. Buyers and approvers may see the text according to configuration.

Email flow

The module provides templates for:

- approval required;
- next group approval level;
- approved and declined decisions;
- main-account order alert;
- edited order;
- deleted order.

Recipient settings can include the main customer and sublogin. The normal Magento

order confirmation is delayed until final approval, then sent to the configured final recipient list.

Approval links use internal order tokens. Do not place token values in screenshots, tickets or logs.

Budget interaction

With Budget installed, an over-budget cart is normally blocked. **Allow budget restricted order to checkout** can instead create it in the approval state. For this case, Budget's restriction drives approval and the sublogin's normal approval threshold is ignored.

Test these cases separately:

1. within budget and below approval threshold;
2. within budget and above approval threshold;
3. over budget with approval fallback enabled;
4. no applicable budget with the empty-budget option enabled or disabled.

A complete staging test

1. Create one sublogin whose threshold is easy to cross.
2. Place an order below the threshold and confirm it follows normal processing.
3. Place an order above the threshold.
4. Confirm the approval notice, state and email recipient.
5. Approve it from the main account and verify final confirmation.
6. Place another order and decline it with a comment.
7. If Role is installed, repeat with a child and parent group.
8. If editing is enabled, change an item and address before final approval.

Hyvä storefront

Install `mageb2b/sublogin-orderapproval-hyva` together with the base Sublogin Hyvä bridge. See Hyvä Compatibility Packages.

Troubleshooting

- **Order skipped approval:** check the module switch, sublogin flag and whether subtotal actually exceeded the threshold.
- **No group approver appears:** check Role activation, group parents and approve/decline permission.
- **Order never reaches the main customer:** inspect the group chain and pre-approval state.
- **Confirmation sent too early or to the wrong party:** review Order Approval recipient settings and test the final decision path.

- **Admin mass action changes nothing:** confirm the order is in the configured approval state when state checking is enabled.

Related: Roles, Permissions and Groups, Budget Management, and Permission Problems.

Roles, Permissions and Groups

`mageb2b/sublogin-role` adds reusable storefront roles and a group hierarchy. Use roles to define what a buyer may do. Use groups to represent reporting or approval relationships inside the company.

The base Sublogin package works without this add-on. Install Role only when buyers need different storefront permissions or an approval hierarchy.

Install and enable

```
composer require mageb2b/sublogin-role
php bin/magento module:enable MageB2B_SubloginRole
php bin/magento setup:upgrade
```

Open **Stores > Configuration > MageB2B > Sublogin > Role settings** and confirm **Is Active** is enabled for the intended scope.

Roles and groups solve different problems

Object	Purpose	Example
Role	Reusable set of allowed storefront actions	Buyer, Approver, Catalog Viewer
Group	Position in a parent-child company hierarchy	Operations > Procurement

A sublogin can be assigned to a role and a group. A group can have a parent group, which gives Order Approval a route through the organization when its group mode is enabled.

Built-in permission areas

The Role package contributes permissions for:

- product list, product details and product prices;
- adding products to cart, viewing cart and applying coupons;
- opening checkout, viewing totals and placing an order;
- viewing the wishlist;
- viewing orders and invoices;
- listing, saving and deleting Sublogin roles;
- listing, saving and deleting Sublogin groups.

The base package and installed add-ons contribute additional permissions to the same tree, such as Sublogin management, reports, Custom Catalog and order actions. The

number is therefore installation-dependent. It should not be described as a fixed “30+” feature.

Create a buyer role

1. Sign in as the main customer account.
2. Open **My Sublogin Roles**.
3. Create a role named for the job, such as `Procurement Buyer`.
4. Grant product-list, product-detail and price access as needed.
5. Grant cart and checkout permissions as a coherent set.
6. Save the role and assign it to one staging sublogin.
7. Sign in as that buyer and test navigation, direct URLs and actions.



For a buyer who may place orders, grant **View Checkout** as well as **Place Order**. A permission to perform the final action is not useful if the buyer cannot reach the checkout screen.

Create a group hierarchy

1. Open **My Sublogin Groups**.
2. Create the top-level business group.
3. Add child groups only where a real reporting or approval relationship exists.
4. Assign each sublogin to the appropriate group.
5. Review parent relationships before enabling hierarchical approval.

Keep the hierarchy shallow. A chart copied from HR often contains levels that add no value to purchasing and make approval routing difficult to explain.

Templates and default assignments

An administrator can mark a role or group **Is Template** on its Admin form. Templates are global: they have no customer owner, and every new Magento customer is provisioned with a copy of each template role and group when the customer is created. The **Copy as Template** button on an existing role drafts a template from that role's permissions; template names must be unique. Storefront save actions ignore a posted `is_template` value, so templates stay admin-managed.

Each customer account can also flag one role and one group as the default for new subaccounts:

- In Admin, use the **Default Role for New Subaccounts** or **Default Group for New Subaccounts** checkbox on the role or group form.
- In the storefront, the main account gets the same choice as a Yes/No select on the role and group edit forms, including the forms shipped by the Hyvä package.

The default is stored per customer account, or per template scope for template records, and applies to sublogins created afterwards. Choosing a new default replaces the previous one for that account. Explicit assignments and existing sublogins are not changed.

Permission behavior

Permissions are enforced in storefront actions and, for supported resources, product collections and views. Test both visible navigation and direct URLs. Hiding a link alone is not an authorization check.

Some permissions are marked as group permissions. They can participate in group-aware behavior, including order and invoice visibility or approval actions, depending on the installed modules.

Role and Order Approval

Order Approval does not require Role. Without Role, an order that needs approval goes to the main customer.

With both packages installed, **Final Approve/Decline order when all group approved an order** can enable a group path:

1. the order enters the configured pre-approval state;
2. eligible users in parent groups approve it in sequence;
3. the main account makes the final decision after group approval completes.

This hierarchy is based on groups, not on configurable amount tiers. The per-sublogin subtotal threshold only decides whether approval is required.

Safe role design

Start with the smallest useful roles:

- **Catalog Viewer:** products and prices, no cart or checkout;
- **Buyer:** products, prices, cart and checkout;
- **Approver:** order visibility plus approve or decline permissions when Order Approval is installed;
- **Account Coordinator:** selected role, group or sublogin management permissions.

These are examples, not roles created automatically by the package.

Avoid one-off roles for every person. Reusable roles make support and offboarding easier.

Bulk actions

The Admin menus **Sublogin > Sublogin Role** and **Sublogin > Sublogin Group** list roles and groups across all customers, with a **Template** column marking global records. Besides delete, their mass actions can copy a role's permissions or a group definition to every customer account — useful for rolling out a reviewed role to an existing customer base.

On the storefront, **My Sublogin Roles** offers a **Copy Selected** mass action that duplicates the chosen roles together with their permissions. **My Sublogins** gains **Assign Role** and **Assign Group** mass actions for reassigning selected sublogins. A sublogin's assigned role can be inspected through the storefront permission view, which returns that sublogin's effective permission tree.

Changes and troubleshooting

After changing a role:

1. save the role;
2. start a fresh sublogin session;
3. test the affected direct route as well as navigation;
4. check whether another add-on adds its own permission;
5. verify the role is assigned to the expected sublogin.

If every buyer loses access, confirm Role is active and a valid default or assigned role exists. If only one feature is missing, inspect that feature's contributed ACL permission.

Hyvä storefront

Install `mageb2b/sublogin-role-hyva` in addition to the base Sublogin Hyvä bridge. See Hyvä Compatibility Packages.

Related: Order Approval, Sublogin Custom Catalog, and Permission Problems.

Sample Data

`mageb2b/sublogin-sample-data` installs disposable records for development and demonstrations. It is not a production-data starter kit.

The current family ships one sample-data package. Separate Role, Budget, Order Approval or Reports sample-data packages described in older documentation are not part of the current package set.

What it creates

The fixtures cover a broad Sublogin scenario, including customers, sublogins, products, orders and address, payment and shipping relationships. They use predictable credentials and obvious fixture identities.

Install them only in an isolated development database that contains no customer or production copy.

Install in a disposable environment

```
composer require mageb2b/sublogin-sample-data
php bin/magento module:enable MageB2B_SubloginSampleData
php bin/magento setup:upgrade
```

Never include this package in a production deployment artifact.

Verify the fixtures

```
php bin/magento sampledata:verify:sublogin
```

Use the result to confirm the data patches ran. Then test the feature, but do not reuse fixture names, passwords, domains or product labels in public screenshots.

Reset the fixtures

```
php bin/magento sampledata:reset:sublogin
```

This command is interactive and destructive. It removes Sublogin sample records and resets the relevant data patch so `setup:upgrade` can install them again.

Before running it:

1. confirm the database is disposable;
2. take a snapshot if other development work matters;
3. read the confirmation target carefully;

4. run `php bin/magento setup:upgrade` only when you intend to recreate the fixtures.

For demonstrations intended for customers or screenshots, create a small coherent fictional company through the normal UI instead. Those records are easier to explain and do not expose predictable fixture credentials.

Transform Customers and Sublogins

`mageb2b/sublogin-transform` changes account ownership. It can turn a Magento customer into a sublogin of another customer, or turn a sublogin into an independent Magento customer.

This is a structural, destructive operation. Rehearse it in staging, take a database backup and verify orders and addresses before running it in production.

Install

```
composer require mageb2b/sublogin-transform
php bin/magento module:enable MageB2B_SubloginTransform
php bin/magento setup:upgrade
```

Only administrator roles with `MageB2B_SubloginTransform::transform_manage` may use the tool.

Customer to sublogin

Open the wizard through **Sublogin > Sublogin Transform** in Admin. The flow has four steps: select the source customer and the target parent customer, review a preview of the affected data, execute, then confirm the result page.

The preview also offers **Additional Sublogin Settings** for the new record — **Active**, **Expire Date** and **Send Emails** — so the resulting account state is chosen deliberately rather than inherited blindly.

This direction can:

- create a sublogin under the chosen parent customer;
- preserve the source account's password hash and newsletter state;
- move addresses into the new ownership model;
- reassign historic orders;
- reparent sublogins that belonged to the source customer;
- delete the original Magento customer record after the transformation.

The source customer ceases to be an independent account. Review website scope, duplicate email rules, address ownership and every nested sublogin shown in the preview.

Sublogin to customer

This direction creates an independent Magento customer, transfers sublogin-owned addresses, reassigns orders and deletes the original sublogin record. It is available from two places:

- the **Transfer to Customer** mass action on the Admin sublogin grids;

- the **Transform to Customer** button on the Admin sublogin edit page.

Both paths execute immediately after the browser confirmation — there is no preview step for this direction. The record is deleted at the end, so verify the sublogin's addresses and orders before triggering it.

Parent-customer addresses merely assigned to the sublogin are not the same as sublogin-owned addresses. Check the sublogin record and its orders in Admin so the new customer receives the intended data.

Safe procedure

1. Stop integrations that might update either account during the change.
2. Take a database backup.
3. Reproduce the accounts in staging.
4. For customer to sublogin, open the wizard and review its preview; for sublogin to customer, review the record in Admin because the action runs without a preview.
5. Record the source, target and affected orders for internal change control.
6. Execute once.
7. Test login or password reset for the resulting account.
8. Verify addresses, newsletter state, order history and website scope.
9. Resume integrations only after their external identifiers still resolve correctly.

Do not use transform to merge duplicates casually

The tool changes one account type into another; it is not a general customer deduplication engine. If two independent customers already contain overlapping orders, addresses or external IDs, define the desired surviving identity before transforming either record.

Related: Multi-Account Management and Sublogin Web API.

Role Web API

Install `mageb2b/sublogin-role-api` when an ERP, identity workflow or administration service must create and maintain sublogin roles, groups and permissions through Magento's REST or SOAP layer. The package requires the Sublogin Role add-on (`mageb2b/sublogin-role`).

Sublogin records themselves are managed by `mageb2b/sublogin-api` — see Sublogin Web API.

Routes

The package exposes 15 routes. Each entity supports create, read, update, delete and search:

Entity	Route pattern	Operations
Roles	`/V1/sublogin-role` and `/V1/sublogin-role/:id`	GET, POST, PUT, DELETE plus `/V1/sublogin-role/search`
Groups	`/V1/sublogin-group` and `/V1/sublogin-group/:id`	GET, POST, PUT, DELETE plus `/V1/sublogin-group/search`
Permissions	`/V1/sublogin-permission` and `/V1/sublogin-permission/:id`	GET, POST, PUT, DELETE plus `/V1/sublogin-permission/search`

Search endpoints accept Magento's standard `searchCriteria` filters, so integrations can query by parent customer, name or other stored fields.

Permissions

Read operations on all three entities require the `MageB2B_Sublogin::role` and `MageB2B_Sublogin::group` ACL resources. Save and delete operations use the dedicated `role_save`, `role_delete`, `group_save` and `group_delete` resources. Grant these resources only to trusted admin users and integrations — a role change affects every sublogin assigned to it.

Mageplaza One Step Checkout Compatibility

`mageb2b/sublogin-mageplaza-osc` is a small compatibility package for shops that run the Sublogin extension together with Mageplaza One Step Checkout. It requires the Sublogin base extension (`mageb2b/sublogin`).

Sublogin lets additional buyers pick from shared and sublogin-owned addresses during checkout. On the standard Magento checkout this is handled by Sublogin's own UI components. Mageplaza One Step Checkout replaces the checkout templates, so the address selection on the billing-address step needs dedicated markup and scripts — that is what this package adds.

What the package contains

File area	Purpose
`onestepcheckout_index_index.xml` layout	Registers the compatibility assets on the Mageplaza checkout page only.
Billing-address JS mixin	Connects Sublogin's address logic to the Mageplaza checkout UI.
Billing-address templates	Render the shared/sublogin-owned address selection inside the one-step layout.

Installation

```
composer require mageb2b/sublogin-mageplaza-osc
php bin/magento module:enable MageB2B_SubloginMageplazaOsc
php bin/magento setup:upgrade
```

After deployment, place a test order as a sublogin through the one-step checkout and verify that the billing-address selector lists the expected shared and sublogin-owned addresses.

Department Purchasing

Use Sublogin when several departments buy through one Magento customer account but need separate people, addresses or controls.

Example structure

```
Northstar Industrial Supplies GmbH
└── Operations
    ├── Procurement
    └── Warehouse
```

The base package creates the individual buyer accounts. The Role add-on creates the groups shown above and gives each job a reusable permission set.

A practical setup

Buyer	Base controls	Optional add-ons
Procurement buyer	Selected office and warehouse addresses, own-order view	Buyer role, monthly budget
Warehouse buyer	Warehouse delivery address, restricted shipping methods	Restricted product catalog
Operations approver	Company-order visibility	Approver role and parent group

These are examples, not records or roles installed by the package.

Build it in a safe order

1. Create the main customer and verified company addresses.
2. Create one buyer per real person, not one shared login per department.
3. Test base checkout and order ownership.
4. Add Role and create the smallest set of reusable roles.
5. Add groups only when the hierarchy is needed for administration or approval.
6. Add Budget, Custom Catalog or Order Approval one at a time.
7. Repeat checkout as every buyer type.

Avoid hidden conflicts

A product may be allowed by the role but removed by Custom Catalog. A payment method may be allowed for the sublogin but hidden by Budget enforcement. An order

may pass Budget but still cross the buyer's approval threshold.

When a journey fails, inspect the controls in that order rather than granting broad permissions.

Related: Roles, Permissions and Groups, Budget Management, and Order Approval.

Controlled Employee Ordering

Sublogin can let employees order approved supplies through the company account without sharing the main customer's password.

Choose the controls you actually need

Requirement	Module
Separate login and own-order history	Sublogin base
Approved delivery addresses	Sublogin base
Allowed payment and shipping methods	Sublogin base
Product, cart or checkout permissions	Role
Approved assortment	Sublogin Custom Catalog
Daily, monthly, yearly or per-order limit	Budget
Main-account decision above a subtotal	Order Approval

Do not install every add-on by default. A simple buyer may need only a separate login, one address and own-order visibility.

Example buyer policy

Elena Fischer may order workshop consumables to the Düsseldorf warehouse. She may see prices, use the cart and place orders. A monthly budget limits total spend, while orders above the company's chosen subtotal wait for approval.

To implement it:

1. create Elena as an active sublogin;
2. assign the warehouse address;
3. test ordinary checkout;
4. assign a buyer role;
5. assign the approved catalog;
6. create one monthly budget;
7. enable approval and set the subtotal threshold;
8. test below and above both limits.

Budget and approval answer different questions. Budget checks available spending. Approval decides whether another person must review the order.

Offboarding

Deactivate the sublogin when the employee leaves or changes responsibility. Keep the record long enough for historic order attribution and only delete it after checking reporting and retention needs.

Related: Create Your First Sublogin and Multi-Account Management.

Branch and Franchise Ordering

A company can represent branches or franchise locations as sublogins under one Magento customer account. This works when the parent organization should own the commercial account and each location needs a separate login and delivery context.

Decide whether one parent account is appropriate

Use this model when locations may share the parent customer's commercial identity. If every franchise must be a legally and financially independent customer, create separate Magento customer accounts instead.

Example layout

```
Northstar Service Network
├── Düsseldorf Branch
│   ├── Branch Manager
│   └── Buyer
└── Essen Branch
    ├── Branch Manager
    └── Buyer
```

Groups and nested relationships require the Role add-on. The base package itself attaches sublogins directly to the main customer.

Configure one location first

1. Add the location's approved shipping address to the parent account.
2. Create one branch buyer and assign only that address.
3. Restrict payment or shipping methods if the branch has a different fulfillment policy.
4. Test order visibility. Keep own-order view enabled unless cross-location visibility is required.
5. Add a location catalog or budget only after checkout succeeds.
6. If a branch manager approves orders, add Role and Order Approval and test the group path.

Reporting boundary

Sublogin reports can attribute orders and products to buyers within the shared customer account. They are not a statutory consolidation or franchise accounting system. Use the order's sublogin identity as an integration input if downstream systems need location reporting.

Related: Address Management, Order and Product Reports, and Transform Customers and

Sublogins.

Configuration Fields

This reference lists the Magento Admin settings exposed by the base package and its documented add-ons: `mageb2b/sublogin`, `mageb2b/sublogin-budget`, `mageb2b/sublogin-budget-one-time`, `mageb2b/sublogin-custom-catalog`, `mageb2b/sublogin-importexport`, `mageb2b/sublogin-orderapproval`, `mageb2b/sublogin-role`.

Configuration paths are included for controlled deployments and `config:set`. Use Magento Admin for normal changes so field validation and scope inheritance still apply. The extension status display is not listed because it reports installation state rather than a configurable value.

`mageb2b/sublogin`

General settings

Field	Configuration path	Input	Scope
Enabled	`sublogin/general/enabled`	Select	Default, Website, Store view
Default Value Can Create Sublogins	`sublogin/general/default_value_can_create_sublogins`	Select	Default, Website, Store view
Restrict order view for sublogins	`sublogin/general/restrict_order_view`	Select	Default, Website, Store view
Fallback label for missing sublogin	`sublogin/general/fallback_sublogin_title`	Text	Default, Website, Store view
Allow impersonate sublogin account	`sublogin/general/enable_mainaccount_login`	Select	Default, Website, Store view
Enable payment methods filter	`sublogin/general/enable_payment_methods_filter`	Select	Default, Website, Store view
Enable shipping methods filter	`sublogin/general/enable_shipping_methods_filter`	Select	Default, Website, Store view
Use shared cart	`sublogin/general/use_shared_cart`	Select	Default, Website, Store view

Field	Configuration path	Input	Scope
Use shared wishlist	`sublogin/general/use_shared_wishlist`	Select	Default, Website, Store view
Send sublogin order email also to customer account	`sublogin/general/cc_order_mainlogin`	Select	Default, Website
Save customer email in order table	`sublogin/general/save_customer_email_in_order`	Select	Default, Website, Store view
Massactions in frontend	`sublogin/general/massactions_frontend`	Multi-select	Default, Website, Store view
Require email confirmation of new created sublogin	`sublogin/general/require_confirmation`	Select	Default, Website, Store view
If sublogin is able to delete own account in frontend	`sublogin/general/delete_own_account`	Select	Default, Website
Reactivate account when disabled	`sublogin/general/reactivate_account_when_disabled`	Select	Default, Website

Public signup

Field	Configuration path	Input	Scope
Enable public sublogin signup	`sublogin/signup/enabled`	Select	Default, Website

Public signup requests always start inactive — the parent customer approves them through an emailed confirmation link, or an administrator activates the record. There is no immediate-activation switch.

Admin settings

Field	Configuration path	Input	Scope
Filter Sublogin Addresses on Customer Edit	`sublogin/admin/filter_sublogin_addresses_on_customer`	Select	Default, Website

Address settings

Field	Configuration path	Input	Scope
Address management for sublogins	`sublogin/address/management`	Select	Default, Website, Store view
Use sublogin address as a customer	`sublogin/address/use_sublogin_address`	Select	Default, Website, Store view
If set, a sublogin is bound to the default billing address of the customer	`sublogin/address/enable_billing_address_restriction`	Select	Default, Website
Enable add new address option for sublogins in checkout	`sublogin/address/add_new_address_option`	Select	Default, Website
Show customer address under my account for sublogin	`sublogin/address/show_customeraddress_under_myaccount`	Select	Default, Website, Store view
Automatically add new customer address	`sublogin/address/automatically_add_customer_address`	Select	Default, Website, Store view
Set customer default address as sublogin default address	`sublogin/address/set_customer_default_address_as_sublogin_default`	Select	Default, Website, Store view
Allow customer to delete an address used by his sublogins in frontend	`sublogin/address/allow_customer_delete_used_sublogin_address`	Select	Default, Website, Store view
Show main address under My Sublogins	`sublogin/address/show_main_address`	Select	Default, Website, Store view
Not available address message	`sublogin/address/not_available_address_message`	Text	Default, Website, Store view

Visibility dependencies:

- `sublogin/address/automatically_add_customer_address` is available when `sublogin/address/management` is `2`.
- `sublogin/address/set_customer_default_address_as_sublogin_default` is available when `sublogin/address/management` is `1,2`.
- `sublogin/address/not_available_address_message` is available when `sublogin/address/show_main_address` is `1`.

Report settings

Field	Configuration path	Input	Scope
Enable reports for customer	`sublogin/report/enable_for_customer`	Select	Default, Website, Store view
Enable reports for sublogin	`sublogin/report/enable_for_sublogin`	Select	Default, Website, Store view

Address Templates

Field	Configuration path	Input	Scope
Text	`sublogin/address_templates/text`	Text area	Default, Website, Store view
Text One Line	`sublogin/address_templates/oneline`	Text area	Default, Website, Store view
HTML	`sublogin/address_templates/html`	Text area	Default, Website, Store view

Email settings

Field	Configuration path	Input	Scope
Email Sender	`sublogin/email/identity`	Select	Default, Website, Store view
Use Sublogin Store ID	`sublogin/email/use_sublogin_store_id`	Select	Default, Website, Store view

Field	Configuration path	Input	Scope
Email template new sublogin	`sublogin/email/new`	Select	Default, Website, Store view
Email template public signup approval	`sublogin/email/signup_approval`	Select	Default, Website, Store view
Email template new sublogin confirmation	`sublogin/email/confirmation`	Select	Default, Website, Store view
Email template new password	`sublogin/email/reset_password`	Select	Default, Website, Store view
Email template expired account refreshing	`sublogin/email/expire_refresh`	Select	Default, Website, Store view
Email template Password Reset Confirmation	`sublogin/email/password_reset_confirmation`	Select	Default, Website, Store view
Send plain password in sublogin account emails	`sublogin/email/use_plain_password`	Select	Default, Website
BCC Receiver for emails	`sublogin/email/send_bcc`	Text	Default, Website, Store view
Add order details to selected email templates	`sublogin/email/add_orderdetails_to_emails`	Multi-select	Default, Website, Store view
Email templates for store owner	`sublogin/email/email_templates_for_store`	Multi-select	Default, Website, Store view
Customer optional email templates	`sublogin/email/customer_optional_email_templates`	Multi-select	Default, Website, Store view

Field	Configuration path	Input	Scope
Sublogin optional email templates	sublogin/email/sublogin_optional_email_templates	Multi-select	Default, Website, Store view

Content settings

Field	Configuration path	Input	Scope
Static block for sublogin grid at frontend	sublogin/content/frontend_sublogin_grid_before	Text	Default, Website, Store view

Debug

Field	Configuration path	Input	Scope
Enable log	sublogin/debug/enable_log	Select	Default

mageb2b/sublogin-budget

Budget settings

Field	Configuration path	Input	Scope
Allowed Budget Types	sublogin/budget/allowed_budgettypes	Multi-select	Default, Website, Store view
Order Status for budget consideration	sublogin/budget/order_status	Multi-select	Default, Website
Budget Logic	sublogin/budget/budget_logic	Select	Default, Website
Allow budget restricted order to checkout	sublogin/budget/allow_budget_restricted_order_to_checkout	Select	Default, Website
Enable order approval on empty budget	sublogin/budget/enable_order_approval_on_empty_budget	Select	Default, Website
Enable Budget Summary in Frontend for Sublogin	sublogin/budget/view_own_budget	Select	Default, Website

Field	Configuration path	Input	Scope
Enable Budget Summary in Frontend for Customer	`sublogin/budget/view_own_budget_customer`	Select	Default, Website
Budget Amount Exceeded Message	`sublogin/budget/budget_amount_exceeded_message`	Text area	Default, Website, Store view
Budget Order Limit Exceeded Message	`sublogin/budget/budget_order_limit_exceeded_message`	Text area	Default, Website, Store view
Budget Calculation Basis	`sublogin/budget/budget_calculation_basis`	Select	Default, Website
Progress Bar Color Thresholds	`sublogin/budget/progress_bar_colors`	Text	Default, Website
Use only Pay on Budget orders for budget usage	`sublogin/budget/use_pay_on_budget_only`	Select	Default, Website
Enforce Pay on Budget as only payment method when budgets are available	`sublogin/budget/enforce_budget_payment_method`	Select	Default, Website

Visibility dependencies:

- `sublogin/budget/allow_budget_restricted_order_to_checkout` is available when `sublogin/order_approval/enabled` is `1`.
- `sublogin/budget/enable_order_approval_on_empty_budget` is available when `sublogin/budget/allow_budget_restricted_order_to_checkout` is `1`.

Pay on Budget

Field	Configuration path	Input	Scope
Enable Pay on Budget	`payment/sublogin_budget/active`	Select	Default, Website
Title	`payment/sublogin_budget/title`	Text	Default, Website, Store view

Field	Configuration path	Input	Scope
Sort Order	`payment/sublogin_budget/sort_order`	Text	Default, Website, Store view
Payment from Applicable Countries	`payment/sublogin_budget/allowspecific`	Select	Default, Website
Payment from Specific Countries	`payment/sublogin_budget/specificcountry`	Multi-select	Default, Website

Visibility dependencies:

- `payment/sublogin_budget/specificcountry` is available when `payment/sublogin_budget/allowspecific` is `1`.

`mageb2b/sublogin-budget-one-time`

One-Time Budget

Field	Configuration path	Input	Scope
Enable One-Time Budget Popup	`sublogin/budget_one_time/enabled`	Select	Default, Website
One-Time Budget Popup Message	`sublogin/budget_one_time/popup_message`	Text area	Default, Website, Store view

`mageb2b/sublogin-custom-catalog`

Sublogin Custom Catalog

All fields render in the **Sublogin Custom Catalog** group of the `sublogin` section (**Stores > Configuration > MageB2B > Sublogin**). The last two store their values under `customcatalog/general/...` and appear only while `MageB2B_CustomCatalog` is enabled.

Field	Configuration path	Input	Scope
Activate	`sublogin/custom_catalog/active`	Select	Default, Website, Store view
Mode	`sublogin/custom_catalog/mode`	Select	Default, Website, Store view
Allow sublogin "My Products" access	`sublogin/custom_catalog/allow_sublogin_view`	Select	Default, Website, Store view
Enforce ACL check	`sublogin/custom_catalog/enforce_acl`	Select	Default, Website, Store view
Enable Custom Catalog Integration	`customcatalog/general/sublogin_integration_active`	Select	Default, Website, Store view
Sublogin Merge Operator	`customcatalog/general/sublogin_merge_operator`	Select	Default, Website, Store view

Mode values: `override_hidden_if_assigned` (default) or `strict_whitelist`.

Visibility dependencies:

- `customcatalog/general/sublogin_merge_operator` is available when `customcatalog/general/sublogin_integration_active` is `1`.

`mageb2b/sublogin-importexport`

Import/Export settings

Field	Configuration path	Input	Scope
Generate Missing Import Passwords	`sublogin/import_export/generate_missing_passwords`	Select	Default, Website, Store view

`mageb2b/sublogin-orderapproval`

Order approval settings

Field	Configuration path	Input	Scope
Enable Order Approval	`sublogin/order_approval/enabled`	Select	Default, Website, Store view
Can Edit Order In Approval State	`sublogin/order_approval/can_edit_in_approval_order`	Select	Default, Website, Store view
Order Approval Notice	`sublogin/order_approval/order_approval_notice`	Text area	Default, Website, Store view
Delete not approved orders	`sublogin/order_approval/delete_notapproved_order`	Select	Default, Website, Store view
On massaction check approval state	`sublogin/order_approval/massaction_check_on_approval_state`	Select	Default, Website, Store view
Order Pre-approval state	`sublogin/order_approval/preapproval_state`	Select	Default, Website, Store view
Order Pre-approval status	`sublogin/order_approval/preapproval_status`	Select	Default, Website, Store view
Order approval state	`sublogin/order_approval/approval_state`	Select	Default, Website, Store view
Order approval status	`sublogin/order_approval/approval_status`	Select	Default, Website, Store view
Order approved state	`sublogin/order_approval/approved_state`	Select	Default, Website, Store view
Order approved status	`sublogin/order_approval/approved_status`	Select	Default, Website, Store view
Order decline state	`sublogin/order_approval/not_approved_state`	Select	Default, Website, Store view
Order declinestate	`sublogin/order_approval/not_approved_status`	Select	Default, Website, Store view
Can decline preapproval orders	`sublogin/order_approval/can_decline_preapproval_order`	Select	Default, Website, Store view

Field	Configuration path	Input	Scope
Auto approve edited order	`sublogin/order_approval/auto_approve_edited_order`	Select	Default, Website, Store view
Show decline comment on frontend	`sublogin/order_approval/show_decline_comment_on_frontend`	Select	Default, Website, Store view
Enable comments for	`sublogin/order_approval/enable_comments_for`	Multi-select	Default, Website, Store view
Show order edit comment on frontend	`sublogin/order_approval/show_order_edit_comment_on_frontend`	Select	Default, Website, Store view

Email

Field	Configuration path	Input	Scope
Email template Order Require Approval	`sublogin/email/order_require_approval`	Select	Default, Website, Store view
Email template Order Approved	`sublogin/email/order_approved`	Select	Default, Website, Store view
Email template Order Declined	`sublogin/email/order_declined`	Select	Default, Website, Store view
Mainlogin Edited Order Template	`sublogin/email/mainlogin_edited_order_template`	Select	Default, Website, Store view
Deleted Order Email Template	`sublogin/email/deleted_order_template`	Select	Default, Website, Store view
Email template Main login order alert	`sublogin/email/mainlogin_orderalert`	Select	Default, Website, Store view
Receivers - Email template Order Approved	`sublogin/email/receivers_order_approved`	Multi-select	Default, Website

Field	Configuration path	Input	Scope
Receivers - Email template Order Declined	`sublogin/email/receivers_order_declined`	Multi-select	Default, Website
Receivers - Email template Main login order alert	`sublogin/email/receivers_mainlogin_orderalert`	Multi-select	Default, Website
Receivers - Final Order Confirmation	`sublogin/email/final_order_confirmation_receivers`	Multi-select	Default, Website

`mageb2b/sublogin-role`

Role settings

Field	Configuration path	Input	Scope
Is Active	`sublogin/role/is_active`	Select	Default, Website, Store view
Include sibling descendants	`sublogin/role/include_sibling_descendants`	Select	Default, Website, Store view
Final Approve/Decline order when all group approved an order	`sublogin/role/all_group_approve_decline_mode`	Select	Default, Website, Store view

Visibility dependencies:

- `sublogin/role/all_group_approve_decline_mode` is available when `sublogin/order_approval/enabled` is `1`.

Database Reference

Sublogin creates additional users beneath a Magento customer account. Orders and the commercial customer relationship remain with the main customer, while each sublogin can receive its own permissions, addresses, payment methods and optional add-on assignments.

Sublogin ownership and restrictions

Relationship	Why it matters
`customer_sublogin` belongs to one `customer_entity`	Keeps every additional user under the correct company customer account.
`customer_sublogin_address` links one `customer_sublogin` with one `customer_address_entity`	Limits the customer addresses available to that user.
`customer_sublogin_payment` belongs to one `customer_sublogin`	Restricts the payment methods the user may select.
`customer_sublogin_shipping` belongs to one `customer_sublogin`	Restricts the shipping methods the user may select.
Magento wishlist and API token records can belong to one `customer_sublogin`	Keeps user-specific shopping and API activity distinct within the main account.

Budget add-on

Relationship	Why it matters
`customer_sublogin_budget` can belong to one `customer_sublogin`	Applies a budget directly to an additional user.
`customer_sublogin_budget` can belong to one `customer_entity` and one `store_website`	Supports account-level budgets with website-specific periods and currency context.

Custom Catalog add-on

Relationship	Why it matters
A sublogin product assignment links one `customer_sublogin` with one `catalog_product_entity`	Provides a direct product allowlist for an individual user.
A sublogin catalog product assignment links one catalog with one `catalog_product_entity`	Builds a reusable catalog for several sublogins.
A sublogin catalog customer assignment links one catalog with one `customer_entity`	Keeps the catalog under the correct company account.
A sublogin catalog assignment links one catalog with one `customer_sublogin`	Reuses the same curated product set for selected additional users.
Catalog assignments belong to one `store`	Allows access to vary by Store View.

Role add-on

Relationship	Why it matters
`customer_sublogin_role` can belong to one `customer_entity`	Keeps reusable roles private to the company account that created them.
`customer_sublogin_role_permission` belongs to one `customer_sublogin_role`	Defines what users assigned to the role may do.
`customer_sublogin_group` can belong to one `customer_entity`	Organizes additional users within the same company account.
`customer_sublogin` can belong to one `customer_sublogin_role` and one `customer_sublogin_group`	Applies reusable permissions and organization to each user.

Passwords, tokens and internal version records are deliberately excluded. Use the customer account, Admin screens or service contracts so permissions and cache invalidation remain consistent.

Common Issues

Start with the buyer's exact website, store view and parent customer. Most Sublogin failures come from a scoped setting, inactive identity or a combination of address, method, budget and permission rules.

A sublogin cannot sign in

Check:

1. the base extension is enabled for the current store view;
2. the record is active and its expiry date is empty or in the future;
3. required email confirmation is complete;
4. the parent customer account is active;
5. the email is unique within Magento's account-sharing scope;
6. the password meets the current Magento policy.

If password reset reports an invalid token, search customers and sublogins for a duplicate scoped email before sending another reset message.

The main account cannot create sublogins

The main customer and delegated sublogins use separate controls.

- The main customer account must have its Sublogin creation capability enabled.
- A sublogin needs the per-record **Can Create Sublogins** value or the matching delegated action.
- **Default Value Can Create Sublogins** only sets the initial value for new records. It does not repair existing accounts.
- With Role installed, check the effective list, save and related management permissions.

Impersonation is missing

Confirm **Allow impersonate sublogin account** is enabled at the active scope. The acting user must also be allowed to manage or impersonate the target record.

After configuration changes, start a new customer session. A stale session can preserve the old navigation until sign-out.

No checkout address is available

Review Address Management in this order:

1. parent-address mode;
2. selected parent address in custom mode;

3. sublogin-owned address in disallow mode;
4. permission to add a new address;
5. parent default billing restriction.

Test billing and shipping independently.

Payment or shipping methods are missing

Disable the relevant Sublogin filter in staging. If the method still does not appear, Magento or another extension considers it invalid for the cart, address or website.

If it returns, check the exact method assigned to the sublogin. Shipping restrictions use the full carrier and method combination. Budget may also hide payment methods when Pay on Budget is enforced.

A product or cart item is unavailable

Check Role product permissions, Sublogin Custom Catalog activation and assignments, the optional merge operator, store website assignment, product status and stock.

An item placed in the cart before a catalog change can fail later validation. Remove it, start a new session and retest the direct product URL.

Budget blocks the wrong cart

Check:

- Budget Logic and owner;
- fixed or recurring period;
- subtotal versus grand-total basis;
- per-order limit;
- order statuses counted as used spend;
- Pay-on-Budget-only mode;
- active one-time allowance.

Do not add a second budget as a workaround until the effective record set is understood.

An order skips or stalls in approval

Check the sublogin approval flag and subtotal threshold first. Then review the configured state/status pair.

With Role hierarchy, inspect the buyer's group, every parent group and approver permissions. Without Role, the main customer is the direct approver.

Account emails do not arrive

Check the selected template, sender identity, sublogin email preference, store context and Magento mail transport. In a development environment, inspect the configured mail capture service.

Keep plain-password email disabled. Resend a setup or reset link after fixing delivery.

Import succeeded but records are wrong

Review `entity_id`, `website_id`, `store_id` and the selected import behavior. For `replace`, compare addresses and add-on fields against the source CSV.

Remember that the CLI importer deletes a successfully imported source file by default unless `--delete_file_after_import=0` is supplied.

Targeted logging

Enable **Stores > Configuration > MageB2B > Sublogin > Debug > Enable log** only while reproducing a problem. The module log is:

```
tail -100 var/log/sublogin.log
```

Related add-ons may also write to Magento's normal system or exception logs. Remove or redact personal data before sharing log excerpts, then disable debug logging after the investigation.

For custom storefront fields, see Extending Frontend Form Fields.

Permission Problems

Permission failures can come from the base Sublogin ACL, a Role assignment or an add-on permission. Identify the exact action before widening access.

A page is missing or returns access denied

1. Confirm the required package is installed and active.
2. Confirm the sublogin has the expected role.
3. Open that role and find the permission contributed by the owning module.
4. Save the role and start a fresh sublogin session.
5. Test both navigation and the direct route.

Examples:

- product and checkout actions belong to Role;
- **View My Products** belongs to Sublogin Custom Catalog;
- approve, decline and edit order actions belong to Order Approval;
- base account-management and report permissions come from Sublogin itself.

Checkout cannot be opened

Granting **Place Order** alone is not enough when **View Checkout** is denied. Review product details, add-to-cart, cart, checkout, totals and place-order permissions as one buyer journey.

Then check non-permission rules: address availability, method filters, Budget and approval configuration.

The buyer sees too much

Review:

- role assignment and allowed product permissions;
- **Restrict order view for sublogins**;
- parent-address mode and assignments;
- Custom Catalog merge operator;
- shared cart and wishlist settings;
- delegated role, group or sublogin management permissions.

Do not rely on a hidden menu item. Verify a direct product, order or management URL is rejected when access should be denied.

Group approval does not advance

Confirm:

1. Role and Order Approval are both active;
2. group approval mode is enabled;
3. the buyer's group has the intended parent chain;
4. the next user has the approve or decline permission;
5. the order is in the configured pre-approval state.

Group routing is hierarchical. Changing the buyer's subtotal threshold does not choose a different group level.

Role changes appear stale

Sign out and start a new session. If the problem remains, verify the assignment at the correct website and check whether another module contributes a separate permission for the same visible workflow.

Enable Sublogin debug logging only for a focused reproduction. See Common Issues.

FAQ

Is a sublogin a separate Magento customer?

No. A sublogin has separate credentials and identity but remains attached to the main Magento customer account.

Can a customer and sublogin use the same email?

No. Login and password-reset handling require a unique address within Magento's configured customer account-sharing scope.

Can a sublogin see only their own orders?

Yes. **Restrict order view for sublogins** is enabled by default. Disable it only when company-wide order visibility is intended.

Can the main customer sign in as a sublogin?

Yes, when impersonation is enabled. The storefront shows the active sublogin context and a link back to the main account. The main customer does not need the sublogin's password.

Are roles included in the base package?

No. Product, cart, checkout and reusable role permissions require `mageb2b/sublogin-role`. The base package still provides core account-management permissions.

Does Order Approval require Role?

No. Without Role, the main customer approves or declines the order. Role is needed for group-based pre-approval and reusable approver permissions.

Can approval use several amount tiers?

No. Each sublogin has an approval flag and one subtotal threshold. Optional group approval follows the group hierarchy, not separate amount bands.

What budget periods are available?

Fixed and recurring day, month and year limits are available, plus a per-order limit. Per Order is implemented but is not selected in the default allowed-type list.

Does Budget have a REST API?

No. It exposes PHP service contracts for Magento modules but does not register REST or SOAP routes. The separate Sublogin API manages sublogin records only.

Can a buyer receive a temporary next-order allowance?

Yes, with `mageb2b/sublogin-budget-one-time`. The latest active one-time allowance overrides normal budgets for the next successful order and is then deactivated.

Is a separate Custom Catalog extension required?

No. Sublogin Custom Catalog owns its own assignments. If SoftwareSilo Custom Catalog is also installed, the two product sets can be combined.

Are reports an add-on?

No. Order and product reports are features of the current Sublogin base package. Do not install the retired `mageb2b/sublogin-report` package.

Can guests request a sublogin?

Yes, when public signup is enabled. New requests are inactive by default. Matching a parent email is not identity verification, so review the applicant before activation.

Should plain passwords be sent by email?

No. Keep plain-password email disabled and use the account setup, confirmation or password-reset flow.

Are Hyvä templates included in the base package?

No. Install the base Hyvä bridge and the matching bridge for each Role, Budget, Custom Catalog or Order Approval storefront feature you use.