Sublogin Documentation

PDF Manual

MODULE

sublogin

LAST UPDATED

2026-03-29

Table of Contents

Sublogin Documentation · sublogin

Overview

Sublogin Documentation · /sublogin

Welcome to the Sublogin extension documentation. This comprehensive B2B solution allows business customers to create and manage multiple sub-accounts with granular permissions and access control.

Quick Links

Getting Started

- Installation & Setup
- Configuration
- Create Your First Sublogin

Core Features

- Multi-Account Management
- Address Management
- Login Context System
- Payment & Shipping Restrictions
- Shared Resources
- Email System

Add-Ons

- Roles & Permissions
- Budget Management
- Order Approval
- Catalog Visibility
- API
- Import/Export
- Reports
- Hyvä Compatibility
- Mageplaza OSC Compatibility
- Sample Data

Use Cases

- B2B Company with Departments
- Franchise System
- Employee Ordering

Troubleshooting

- Common Issues
- Permission Problems

Overview

The Sublogin extension is a complete B2B solution that enables sophisticated user management within

your Magento store. Business customers can create multiple sub-accounts for employees, departments, or branches, each with their own credentials and permissions.

Core Concept

Main Account (Customer)

The primary business customer account that owns all sublogins. This account has full control over creating, editing, and managing sublogins.

Sublogins (Sub-Accounts)

Individual accounts created under the main customer account. Each sublogin has:

- Own email address and password
- Own profile information
- Assigned role and permissions (optional)
- Budget limits (optional)
- Address restrictions
- Order history

Hierarchical Organization

Sublogins can be organized into groups with parent-child relationships, creating a complete organizational structure:

```
Customer Account
├── General (Group)
│   ├── Branch A (Group)
│   │   ├── Department 1 (Group)
│   │   └── Department 2 (Group)
│   └── Branch B (Group)
└── Sublogins assigned to groups
```

Key Features

1. Unlimited Sub-Accounts

Create as many sublogins as needed for your organization. No limits on the number of sub-accounts.

2. Granular Permissions

Control exactly what each sublogin can do with role-based permissions (requires Roles add-on):

- View products and prices
- Add to cart and checkout
- Place orders
- View orders (own or all)
- Manage other sublogins
- And 30+ more permissions

3. Budget Control

Set spending limits per sublogin or department (requires Budget add-on):

- Daily, monthly, yearly budgets
- Per-order limits
- Automatic tracking and enforcement

4. Approval Workflows

Implement multi-level order approval processes (requires Order Approval add-on):

- Amount-based thresholds
- Role-based approvers
- Email notifications

5. Flexible Address Management

Three address modes:

- **Shared**: Sublogins use customer addresses
- **Restricted**: Sublogins can only use assigned addresses
- **Own**: Sublogins create their own addresses

6. Impersonation

Main account holders and admins can "login as" sublogins to:

- Test permissions
- Assist with orders
- Troubleshoot issues

Modular Architecture

The Sublogin system is modular with optional add-ons:

Core Module (Required)

- Basic sublogin management
- Address handling
- Email system
- Session management

Optional Add-Ons

- **Roles & Permissions**: Advanced ACL system with 30+ permissions
- **Budget Management**: Spending limits and tracking
- **Order Approval**: Multi-level approval workflows
- **Catalog Visibility**: Product restrictions per sublogin
- **API**: Web API (REST/SOAP) endpoints for integrations
- **Import/Export**: CSV export from admin grid + CLI import
- **Reports**: Orders/products reporting in customer account + admin
- **Hyvä Compatibility**: Layout/template compatibility for Hyvä storefronts

- **Mageplaza OSC Compatibility**: Ensures checkout compatibility with Mageplaza One Step Checkout
- **Sample Data**: Demo data for staging/dev environments

Common Scenarios

B2B Company

- Main account: Company
- Sublogins: Employees in Purchasing, Sales, Warehouse
- Each department has own budget and permissions

Franchise

- Main account: Franchise owner
- Sublogins: Individual store locations
- Each store has own addresses and order approval

Multi-Location Business

- Main account: Head office
- Sublogins: Branch offices
- Shared catalog, separate budgets per location

Educational Institution

- Main account: University
- Sublogins: Departments/Faculties
- Budget management with approval workflow

Database Structure

The extension uses several database tables:

- `customer_sublogin` - Main sublogin accounts
- `customer_sublogin_address` - Address assignments
- `customer_sublogin_payment` - Payment method restrictions
- `customer_sublogin_shipping` - Shipping method restrictions
- `customer_sublogin_role` - Roles (add-on)
- `customer_sublogin_role_permission` - Permissions (add-on)
- `customer_sublogin_group` - Hierarchical groups (add-on)
- `customer_sublogin_budget` - Budgets (add-on)

Integration Points

Customer ID Extension

Sublogins can have auto-generated customer IDs in format: `{customer_id}-{number}`

Sales Staff Extension

Assign sales representatives to sublogins for commission tracking

Price System Extensions

Customer-specific prices apply to all sublogins of that customer

Payment Profiles Extension

Combine with sublogin payment restrictions for advanced control

Installation

Sublogin Documentation · /sublogin/getting-started/installation

Requirements

- Magento 2.4.x or higher
- PHP 7.4 / 8.0 or higher
- MySQL 5.7 or higher

Installation via Composer

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/sublogin:*
php bin/magento module:enable MageB2B_Sublogin
php bin/magento setup:upgrade
php bin/magento setup:di:compile
php bin/magento setup:static-content:deploy
php bin/magento cache:flush
```

Installation via ZIP

1. Download the extension ZIP file
2. Extract to `app/code/MageB2B/Sublogin`
3. Run the following commands:

```
php bin/magento module:enable MageB2B_Sublogin
php bin/magento setup:upgrade
php bin/magento setup:di:compile
php bin/magento setup:static-content:deploy
php bin/magento cache:flush
```

Verify Installation

1. Log in to Magento Admin
2. Navigate to **Stores > Configuration > MageB2B > Sublogin**
3. You should see the configuration options
4. Navigate to **Customers > All Customers**
5. Edit a customer and you should see a "Sublogins" tab

Database Tables Created

The extension creates the following tables:

Core Tables

- `customer_sublogin` - Main sublogin accounts

- `customer_sublogin_address` - Address assignments
- `customer_sublogin_payment` - Payment method restrictions
- `customer_sublogin_shipping` - Shipping method restrictions

Extended Tables

- `quote` - Added `sublogin_id` column
- `sales_order` - Added `sublogin_id` column
- `sales_order_grid` - Added `sublogin_id` column
- `wishlist` - Added `sublogin_id` column

Initial Configuration

After installation, configure the extension:

1. Go to **Stores > Configuration > MageB2B > Sublogin**
2. Set **Enabled** to "Yes"
3. Configure basic settings:
    - Default value for "Can Create Sublogins"
    - Restrict order view for sublogins
    - Allow impersonate sublogin account
    - Use shared cart/wishlist
4. Click **Save Config**
5. Clear cache: `php bin/magento cache:flush`

Optional Add-Ons

Install optional add-ons for extended functionality:

Roles & Permissions

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/sublogin-role:*
php bin/magento module:enable MageB2B_SubloginRole
php bin/magento setup:upgrade
php bin/magento cache:flush
```

Budget Management

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/sublogin-budget:*
php bin/magento module:enable MageB2B_SubloginBudget
php bin/magento setup:upgrade
php bin/magento cache:flush
```

Order Approval

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/sublogin-orderapproval:*
php bin/magento module:enable MageB2B_SubloginOrderApproval
php bin/magento setup:upgrade
php bin/magento cache:flush
```

Catalog Visibility

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/sublogin-catalogvisibility:*
php bin/magento module:enable MageB2B_SubloginCatalogVisibility
php bin/magento setup:upgrade
php bin/magento cache:flush
```

API (Web API)

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/sublogin-api:*
php bin/magento module:enable MageB2B_SubloginApi
php bin/magento setup:upgrade
php bin/magento cache:flush
```

Import/Export

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/sublogin-importexport:*
php bin/magento module:enable MageB2B_SubloginImportExport
php bin/magento setup:upgrade
php bin/magento cache:flush
```

Reports

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/sublogin-report:*
php bin/magento module:enable MageB2B_SubloginReport
php bin/magento setup:upgrade
```

```
php bin/magento cache:flush
```

Hyvä Compatibility

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/sublogin-hyva:*
php bin/magento module:enable MageB2B_SubloginHyva
php bin/magento setup:upgrade
php bin/magento cache:flush
```

If you also use the Roles & Permissions add-on with Hyvä:

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/sublogin-role-hyva:*
php bin/magento module:enable MageB2B_SubloginRoleHyva
php bin/magento setup:upgrade
php bin/magento cache:flush
```

Mageplaza One Step Checkout Compatibility

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/sublogin-mageplaza-osc:*
php bin/magento module:enable MageB2B_SubloginMageplazaOsc
php bin/magento setup:upgrade
php bin/magento cache:flush
```

Sample Data

For staging/dev environments:

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/sublogin-sample-data:*
php bin/magento module:enable MageB2B_SubloginSampleData
php bin/magento setup:upgrade
php bin/magento cache:flush
```

If you installed the Reports add-on and want demo data:

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
```

```
composer require mageb2b/sublogin-report-sample-data:*
php bin/magento module:enable MageB2B_SubloginReportSampleData
php bin/magento setup:upgrade
php bin/magento cache:flush
```

Post-Installation Steps

1. Configure Customer Accounts

Enable "Can Create Sublogins" for B2B customers:

1. Navigate to **Customers > All Customers**
2. Edit a customer
3. Go to "Account Information" tab
4. Set "Can Create Sublogins" to "Yes"
5. Save customer

2. Configure Email Templates

1. Go to **Stores > Configuration > MageB2B > Sublogin > Email Settings**
2. Select email templates for:
    - New sublogin created
    - Email confirmation
    - Password reset
    - Expired account refresh
3. Configure email sender identity
4. Save configuration

3. Configure Address Management

1. Go to **Stores > Configuration > MageB2B > Sublogin > Address Settings**
2. Select address management mode:
    - Shared addresses
    - Restricted addresses
    - Own addresses
3. Configure additional address options
4. Save configuration

Uninstallation

To uninstall the extension:

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

This will:

- Remove all database tables
- Remove all sublogin data
- Remove module files

Warning: This action cannot be undone. All sublogin data will be permanently deleted.

Troubleshooting Installation

Module Not Showing in Admin

1. Verify module is enabled: `php bin/magento module:status`
2. Clear cache: `php bin/magento cache:flush`
3. Check file permissions
4. Review `var/log/system.log` for errors

Database Tables Not Created

1. Run setup:upgrade again: `php bin/magento setup:upgrade`
2. Check database user permissions
3. Review `var/log/system.log` for SQL errors

Compilation Errors

1. Remove generated files: `rm -rf generated/*`
2. Run di:compile again: `php bin/magento setup:di:compile`
3. Check for conflicting modules

Next Steps

- Configuration Guide
- Create Your First Sublogin
- Multi-Account Management

Configuration

Sublogin Documentation · /sublogin/getting-started/configuration

Complete reference for all Sublogin system configuration options.

Configuration Location

Navigate to **Stores > Configuration > MageB2B > Sublogin**

General Settings

Enabled

Path: `sublogin/general/enabled`
Type: Yes/No
Default: Yes
Scope: Store View

```
Yes: Sublogin system is active
No: Sublogin system is disabled (all features inactive)
```

When to use:

- Enable in production after testing
- Disable temporarily for maintenance
- Enable per store view for multi-store setups

Default Value Can Create Sublogins

Path: `sublogin/general/default_value_can_create_sublogins`
Type: Yes/No
Default: No
Scope: Store View

```
Yes: New customers can create sublogins by default
No: Admin must enable per customer
```

Impact: Sets default value for "Can Create Sublogins" attribute on new customers.

Restrict Order View for Sublogins

Path: `sublogin/general/restrict_order_view`
Type: Yes/No
Default: Yes

Scope: Store View

```
Yes: Sublogins see only their own orders
No: Sublogins see all customer orders
```

Use Cases:

- Yes: Privacy between departments/employees
- No: Shared order visibility for coordination

Fallback Label for Missing Sublogin

Path: `sublogin/general/fallback_sublogin_title`
Type: Text
Default: `N/A`
Scope: Store View

Displayed in order history and grids if the related sublogin no longer exists.

Allow Impersonate Sublogin Account

Path: `sublogin/general/enable_mainaccount_login`
Type: Yes/No
Default: Yes
Scope: Store View

```
Yes: Customer can "Login As" sublogin without password
No: Impersonation disabled
```

Security Note: Disable if you want to prevent impersonation for compliance reasons.

Enable Payment Methods Filter

Path: `sublogin/general/enable_payment_methods_filter`
Type: Yes/No
Default: No
Scope: Store View

```
Yes: Restrict payment methods per sublogin
No: All payment methods available
```

Requires: Payment method assignment per sublogin.

Enable Shipping Methods Filter

Path: `sublogin/general/enable_shipping_methods_filter`
Type: Yes/No
Default: No
Scope: Store View

```
Yes: Restrict shipping methods per sublogin
No: All shipping methods available
```

Requires: Shipping method assignment per sublogin.

Use Shared Cart

Path: `sublogin/general/use_shared_cart`
Type: Yes/No
Default: No
Scope: Store View

```
Yes: Customer and all sublogins share one cart
No: Each sublogin has separate cart
```

Impact:

- Yes: Items added by any sublogin appear in shared cart
- No: Each sublogin maintains independent cart

Use Shared Wishlist

Path: `sublogin/general/use_shared_wishlist`
Type: Yes/No
Default: No
Scope: Store View

```
Yes: Customer and all sublogins share one wishlist
No: Each sublogin has separate wishlist
```

Send Sublogin Order Email Also to Customer Account

Path: `sublogin/general/cc_order_mainlogin`

Type: Yes/No
Default: No
Scope: Website

```
Yes: Order emails sent to both sublogin AND customer
No: Order emails sent only to sublogin
```

Use Case: Customer wants to monitor all sublogin orders.

Save Customer Email in Order Table

Path: `sublogin/general/save_customer_email_in_order`
Type: Yes/No
Default: No
Scope: Store View

```
Yes: Customer email saved in sales_order table
No: Sublogin email saved in sales_order table
```

Impact: Affects order email field in database and reports.

Mass Actions in Frontend

Path: `sublogin/general/massactions_frontend`
Type: Multiselect
Default: Activate, Deactivate
Scope: Store View

```
Options:
- Activate
- Deactivate
- Delete
```

Allows: Customer to perform bulk actions on sublogins.

Require Email Confirmation of New Created Sublogin

Path: `sublogin/general/require_confirmation`
Type: Yes/No
Default: No
Scope: Store View

```
Yes: Sublogin must confirm email before activation
```

```
No: Sublogin active immediately
```

Workflow:

1. Sublogin created with active=0
2. Confirmation email sent
3. Sublogin clicks link
4. Account activated

If Sublogin is Able to Delete Own Account in Frontend

Path: `sublogin/general/delete_own_account`
Type: Yes/No
Default: No
Scope: Website

```
Yes: Sublogin can delete their own account
No: Only customer/admin can delete
```

Requires: `delete_own_account` permission (if using Roles add-on).

Reactivate Account When Disabled

Path: `sublogin/general/reactivate_account_when_disabled`
Type: Yes/No
Default: Yes
Scope: Website

```
Yes: Disabled sublogins can be reactivated
No: Once disabled, cannot be reactivated
```

Admin Settings

Filter Sublogin Addresses on Customer Edit

Path: `sublogin/admin/filter_sublogin_addresses_on_customer`
Type: Yes/No
Default: Yes
Scope: Website

```
Yes: Sublogin addresses hidden in customer address list
No: All addresses shown (customer + sublogin)
```

Use Case: Cleaner customer address view in admin.

Address Settings

Address Management for Sublogins

Path: `sublogin/address/management`
Type: Select
Default: 1 (Shared)
Scope: Store View

```
Options:
1 = Shared: Sublogins use customer addresses
2 = Restricted: Sublogins use only assigned addresses
3 = Own: Sublogins create own addresses
```

Details:

- **Shared**: All customer addresses available to all sublogins
- **Restricted**: Admin assigns specific addresses to each sublogin
- **Own**: Sublogins create and manage their own addresses

Use Sublogin Address as a Customer

Path: `sublogin/address/use_sublogin_address`
Type: Yes/No
Default: No
Scope: Store View

```
Yes: Customer can use sublogin-created addresses
No: Customer cannot see sublogin addresses
```

If Set, a Sublogin is Bound to the Default Billing Address of the Customer

Path: `sublogin/address/enable_billing_address_restriction`
Type: Yes/No
Default: No
Scope: Website

```
Yes: Sublogin must use customer's default billing address
```

```
No: Sublogin can choose any assigned address
```

Enable Add New Address Option for Sublogins in Checkout

Path: `sublogin/address/add_new_address_option`
Type: Yes/No
Default: No
Scope: Website

```
Yes: Sublogin can add new address during checkout
No: Sublogin must use existing addresses only
```

Show Customer Address Under My Account for Sublogin

Path: `sublogin/address/show_customeraddress_under_myaccount`
Type: Yes/No
Default: Yes
Scope: Store View

```
Yes: Customer addresses visible in sublogin account
No: Only sublogin's own addresses visible
```

Automatically Add New Customer Address

Path: `sublogin/address/automatically_add_customer_address`
Type: Yes/No
Default: No
Scope: Store View

```
Yes: New customer addresses auto-assigned to sublogins
No: Manual assignment required
```

Depends on: Address management mode = Restricted

Set Customer Default Address as Sublogin Default Address

Path: `sublogin/address/set_customer_default_address_as_sublogin_default`
Type: Yes/No
Default: No

Scope: Store View

```
Yes: Customer's default addresses become sublogin defaults
No: Sublogin has no default addresses initially
```

Applies: Only when sublogin has no default addresses set.

Allow Customer to Delete an Address Used by His Sublogins in Frontend

Path: `sublogin/address/allow_customer_delete_used_sublogin_address`
Type: Yes/No
Default: No
Scope: Store View

```
Yes: Customer can delete addresses even if used by sublogins
No: Cannot delete addresses assigned to sublogins
```

Show Main Address Under My Sublogins

Path: `sublogin/address/show_main_address`
Type: Yes/No
Default: No
Scope: Store View

```
Yes: Customer's main address shown in sublogin list
No: Main address hidden
```

Not Available Address Message

Path: `sublogin/address/not_available_address_message`
Type: Text
Default: `N/A`
Scope: Store View

Message displayed when no address is set.

Email Settings

Email Sender

Path: `sublogin/email/identity`
Type: Select
Default: Sales Representative
Scope: Store View

```
Options:
- General Contact
- Sales Representative
- Customer Support
- Custom Email 1
- Custom Email 2
```

Use Sublogin Store ID

Path: `sublogin/email/use_sublogin_store_id`
Type: Yes/No
Default: No
Scope: Store View

```
Yes: Use sublogin's store ID for email templates
No: Use current store ID
```

Email Template New Sublogin

Path: `sublogin/email/new`
Type: Select
Default: Sublogin New Account
Scope: Store View

Template sent when new sublogin is created.

Email Template New Sublogin Confirmation

Path: `sublogin/email/confirmation`
Type: Select
Default: Sublogin Email Confirmation
Scope: Store View

Template sent for email confirmation.

Email Template New Password

Path: `sublogin/email/reset_password`
Type: Select
Default: Sublogin Reset Password
Scope: Store View

Template sent when password is reset.

Email Template Expired Account Refreshing

Path: `sublogin/email/expire_refresh`
Type: Select
Default: Sublogin Account Expired
Scope: Store View

Template sent when expired account is refreshed.

Email Template Password Reset Confirmation

Path: `sublogin/email/password_reset_confirmation`
Type: Select
Default: Sublogin Password Reset Confirmation
Scope: Store View

BCC Receiver for Emails

Path: `sublogin/email/send_bcc`
Type: Text
Default: Empty
Scope: Store View

```
Format: email1@example.com;email2@example.com
```

Multiple emails separated by semicolon.

Add Order Details to Selected Email Templates

Path: `sublogin/email/add_orderdetails_to_emails`

Type: Multiselect
Default: None
Scope: Store View

```
Options:
- New Order
- Order Update
- Order Invoice
- Order Shipment
```

Email Templates for Store Owner

Path: `sublogin/email/email_templates_for_store`
Type: Multiselect
Default: None
Scope: Store View

Store owner receives copy of selected email types.

Customer Optional Email Templates

Path: `sublogin/email/customer_optional_email_templates`
Type: Multiselect
Default: `sublogin/email/new, sublogin/email/reset_password, sublogin/email/mainlogin_orderalert, sublogin/email/order_require_approval, sublogin/email/expire_refresh, sublogin/email/order_declined`
Scope: Store View

Customer's optional email receives copy.

Sublogin Optional Email Templates

Path: `sublogin/email/sublogin_optional_email_templates`
Type: Multiselect
Default: `sublogin/email/new, sublogin/email/reset_password, sublogin/email/mainlogin_orderalert, sublogin/email/order_require_approval, sublogin/email/expire_refresh, sublogin/email/order_declined`
Scope: Store View

Sublogin's optional email receives copy.

Content Settings

Static Block for Sublogin Grid at Frontend

Path: `sublogin/content/frontend_sublogin_grid_before`
Type: Text
Default: Empty
Scope: Store View

CMS block identifier to display before sublogin grid.

Debug Settings

Enable Log

Path: `sublogin/debug/enable_log`
Type: Yes/No
Default: No
Scope: Default

```
Yes: Log sublogin events to var/log/sublogin.log
No: No logging
```

Use for: Debugging issues in development.

Configuration Best Practices

Development Environment

```
Enabled: Yes
Require Confirmation: No (faster testing)
Restrict Order View: No (see all data)
Use Shared Cart: No (test isolation)
Enable Log: Yes (debugging)
```

Staging Environment

```
Enabled: Yes
Require Confirmation: Yes (test workflow)
Restrict Order View: Yes (test privacy)
Use Shared Cart: Per requirements
Enable Log: Yes (troubleshooting)
```

Production Environment

```
Enabled: Yes
Require Confirmation: Yes (security)
Restrict Order View: Yes (privacy)
Use Shared Cart: Per business requirements
Enable Log: No (performance)
Allow Impersonate: Per compliance requirements
```

Configuration by Use Case

B2B Company with Departments

```
Restrict Order View: Yes
Use Shared Cart: No
Use Shared Wishlist: No
Address Management: Restricted
Enable Payment Filter: Yes
Enable Shipping Filter: Yes
```

Franchise System

```
Restrict Order View: Yes
Use Shared Cart: No
Address Management: Own
Enable Payment Filter: Yes
Save Customer Email: No
```

Employee Ordering

```
Restrict Order View: Yes
Use Shared Cart: No
CC Order to Customer: Yes
Address Management: Shared
Require Confirmation: Yes
```

Next Steps

- Create Your First Sublogin
- Roles & Permissions
- Budget Management
- Troubleshooting

First Sublogin

Sublogin Documentation · /sublogin/getting-started/first-sublogin

This guide walks you through creating your first sublogin account.

Prerequisites

- Sublogin extension installed and enabled
- Customer account with "Can Create Sublogins" enabled
- Logged in as customer or admin

Method 1: Create from Customer Account (Frontend)

Step 1: Navigate to Sublogins

1. Log in to your customer account
2. Go to **My Account > My Sublogins**
3. Click **Add New Sublogin**

Step 2: Fill in Basic Information

Email Address

```
john.doe@company.com
```

Must be unique across all customers and sublogins.

First Name

```
John
```

Last Name

```
Doe
```

Password

```
SecurePassword123!
```

Or leave empty to auto-generate and send via email.

Step 3: Configure Options

Active

- Yes - Sublogin can log in immediately
- ☐ No - Sublogin is disabled

Send Backend Mails

- Yes - Sublogin receives email notifications
- ☐ No - No emails sent to this sublogin

Can Create Sublogins

- Yes - This sublogin can create other sublogins
- ☐ No - Cannot create sublogins

Expire Date (Optional)

```
2024-12-31
```

Sublogin will be automatically deactivated after this date.

Step 4: Assign Addresses (Optional)

Customer Addresses Select which customer addresses this sublogin can use:

- Main Office - 123 Business St, New York, NY
- Warehouse - 456 Storage Ave, Brooklyn, NY
- ☐ Home Office - 789 Remote Rd, Queens, NY

Or Create New Address Click "Add New Address" to create a sublogin-specific address.

Step 5: Save Sublogin

Click **Save Sublogin**

The sublogin will receive a welcome email (if "Send Backend Mails" is enabled) with:

- Login credentials
- Link to customer account
- Instructions for first login

Method 2: Create from Admin Panel

Step 1: Navigate to Customer

1. Log in to Magento Admin
2. Go to **Customers > All Customers**
3. Edit the customer account
4. Click on **Sublogins** tab

Step 2: Add New Sublogin

1. Click **Add Sublogin**
2. Fill in the same information as Method 1
3. Additional admin-only options:
    - **Role** (if SubloginRole add-on installed)
    - **Group** (if SubloginRole add-on installed)
    - **Budget** (if SubloginBudget add-on installed)
    - **Order Needs Approval** (if SubloginOrderApproval add-on installed)

Step 3: Configure Advanced Settings

Payment Methods (if enabled in config) Select which payment methods this sublogin can use:

- Bank Transfer
- Credit Card
- ☐ Cash on Delivery
- ☐ PayPal

Shipping Methods (if enabled in config) Select which shipping methods this sublogin can use:

- Standard Shipping
- Express Shipping
- ☐ Overnight Shipping

Step 4: Save

Click **Save Sublogin**

Testing the Sublogin

Test Login

1. Log out of your current account
2. Go to customer login page
3. Enter sublogin email and password
4. Click **Sign In**

You should be logged in as the sublogin.

Test Permissions

Try the following actions:

- Browse products
- Add products to cart
- View cart
- Proceed to checkout
- Place an order
- View order history

Depending on your configuration and permissions (if using Roles add-on), some actions may be restricted.

Test Impersonation

1. Log in as main customer account
2. Go to **My Account > My Sublogins**
3. Find the sublogin you created
4. Click **Login As**

You will be logged in as the sublogin without needing their password.

Common Scenarios

Scenario 1: Employee with Full Access

```
Email: employee@company.com
Active: Yes
Can Create Sublogins: No
Addresses: All customer addresses
Payment Methods: All methods
Shipping Methods: All methods
```

Use Case: Regular employee who can place orders

Scenario 2: Department Manager

```
Email: manager@company.com
Active: Yes
Can Create Sublogins: Yes
Role: Manager (if using Roles add-on)
Budget: €10,000/month (if using Budget add-on)
Addresses: All customer addresses
```

Use Case: Manager who can create sublogins for their team

Scenario 3: Sales Representative

```
Email: sales@company.com
Active: Yes
Can Create Sublogins: No
Role: Sales Rep (if using Roles add-on)
Catalog Visibility: Restricted products (if using Catalog Visibility add-
on)
Payment Methods: None (cannot checkout)
```

Use Case: Sales rep who can browse and add to cart but cannot place orders

Scenario 4: Temporary Contractor

```
Email: contractor@company.com
Active: Yes
Expire Date: 2024-06-30
Can Create Sublogins: No
Budget: €1,000 total (if using Budget add-on)
Addresses: Restricted to specific address
```

Use Case: Temporary worker with limited access and budget

Email Confirmation Flow

If "Require Email Confirmation" is enabled in configuration:

1. Sublogin is created with `active = 0`
2. Confirmation email is sent with unique token
3. Sublogin clicks confirmation link
4. Account is activated (`active = 1`)
5. Sublogin can now log in

Troubleshooting

Sublogin Cannot Log In

1. Check if sublogin is active
2. Verify email address is correct
3. Check if expire date has passed
4. Ensure password is correct
5. Check if email confirmation is required

Email Not Received

1. Check "Send Backend Mails" is enabled
2. Verify email address is correct
3. Check spam folder
4. Review email configuration in admin
5. Check `var/log/system.log` for email errors

Cannot Create Sublogin

1. Verify customer has "Can Create Sublogins" enabled
2. Check if email address is already in use
3. Review `var/log/system.log` for errors
4. Ensure all required fields are filled

Next Steps

- Configure Roles & Permissions
- Set Up Budgets
- Configure Order Approval
- Address Management

Address Management

Sublogin Documentation · /sublogin/features/address-management

Control address access and management for sub-accounts.

Overview

Configure which addresses sub-accounts can use and manage.

Address Sharing

Mode	Description
Shared	All sub-accounts access main account addresses
Individual	Each sub-account has own addresses
Hybrid	Shared + individual addresses

Configuration

Stores > Config > Sublogin > Address Settings

Setting	Description
Share Addresses	Enable address sharing
Allow Add	Sub-accounts can add addresses
Allow Edit	Sub-accounts can modify
Allow Delete	Sub-accounts can remove

Permissions

Per sub-account:

- View all addresses
- Use specific addresses only
- Add new addresses
- Manage own addresses only

Address Types

Type	Access
Company HQ	All users
Warehouse 1	Logistics team
Personal	Individual only

Related

- Multi-Account Management
- Configuration

Email System

Sublogin Documentation · /sublogin/features/email-system

Configure email notifications for sub-accounts.

Overview

Control who receives which email notifications.

Email Types

Email	Recipients
Order Confirmation	Orderer + Main account
Shipping Update	Orderer
Invoice	Main account + Finance role
Account Created	New sub-account

Configuration

Stores > Config > Sublogin > Email

Setting	Description
CC Main Account	Copy main on all emails
Role-Based	Send based on roles
Custom Templates	Sublogin-specific templates

Email Templates

Template	Purpose
`sublogin_welcome`	New sub-account created
`sublogin_order`	Order placed by sub
`sublogin_approval`	Order needs approval
`sublogin_budget`	Budget threshold alert

Notification Preferences

Sub-accounts can set:

- Order notifications
- Shipping updates
- Marketing emails
- Account alerts

Related

- Configuration
- Order Approval

Extending Frontend Form Fields

Sublogin Documentation · /sublogin/features/extending-frontend-form-fields

This guide explains how to add or adjust fields on the **Sublogin frontend create/edit form** (Customer Account area) by extending the field definition returned by `getFormFields()`.

When To Use This

Use this if you need:

- custom, customer-specific fields on the Sublogin form
- additional toggles/selectors that depend on your business logic

This approach is useful to avoid forking the Sublogin module.

Field Reference (What The Form Can Show)

The exact form fields can vary by configuration and installed add-ons. Common base fields are:

- **Customer** (Admin-only): selects the main customer account the Sublogin belongs to.
- **Store**: the store view the Sublogin is created for (can affect store context and emails).
- **Prefix**: only shown if prefixes are enabled in the Magento customer configuration.
- **Firstname / Lastname**
- **Email**: must be unique across the whole system (customers and sublogins).
- **Password**: set manually or auto-generate (UI-dependent).
- **Shared Customer Addresses**: shown when address assignment is restricted; lets you choose which main-account addresses the Sublogin can use.

Other common (optional) fields/sections:

- **Active/Enabled**
- **Send Backend Mails** (notifications)
- **Can Create Sublogins**
- **Expire Date**
- **Payment/Shipping Method Restrictions** (if enabled)

Add-ons may add additional fields (examples):

- Roles & Permissions: role/group assignment
- Budget Management: budget configuration
- Order Approval: approval flags/rules
- Catalog Visibility: restriction type and assignments

Extending `getFormFields()` Via Plugin

You can extend the frontend form fields by creating a Magento plugin for:

- `MageB2B\\Sublogin\\Block\\Sublogin\\Edit::getFormFields()`

1. Define The Plugin (di.xml)

Create `etc/frontend/di.xml` in your custom module:

```
<?xml version="1.0"?>
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="urn:magento:framework:ObjectManager/etc/conf
ig.xsd">
    <type name="MageB2B\Sublogin\Block\Sublogin\Edit">
        <plugin name="company_module_sublogin_form_fields"
type="Company\Module\Plugin\MageB2B\Sublogin\Block\Sublogin\Edit\AddCustomF
ield" />
    </type>
</config>
```

2. Implement `afterGetFormFields()`

Create the plugin class:

```
<?php

declare(strict_types=1);

namespace Company\Module\Plugin\MageB2B\Sublogin\Block\Sublogin\Edit;

use MageB2B\Sublogin\Block\Sublogin\Edit;

class AddCustomField
{
    public function afterGetFormFields(Edit $subject, array $result): array
    {
        $customFields = [
            [
                'name' => 'yes_no_dropdown',
                'label' => __('Yes No Dropdown'),
                'type' => 'dropdown',
                'default' => 1,
                'options' => [
                    ['value' => 1, 'label' => __('Yes')],
                    ['value' => 0, 'label' => __('No')],
                ],
                'visible' => true,
                'position' => 1400,
            ],
        ];

        return array_merge($result, $customFields);
    }
}
```

Field Definition Keys

The field array typically uses keys like:

- `name`: unique field identifier (used as input name)

- `label`: displayed label (use `__()` for translation)
- `type`: renderer/type used by the form (e.g. `text`, `password`, `dropdown`)
- `default`: default value
- `options`: for select/dropdown fields (array of `value`/`label`)
- `visible`: whether the field is shown
- `position`: sorting position (higher usually means later)

Important Notes

- This hook changes what is **rendered** on the form. How a new field is **validated, saved, and loaded** depends on your implementation and where you persist the value.
- If you are unsure which field definition keys/types are supported in your installation, inspect the existing `$result` structure returned by `getFormFields()` first, then mirror the same patterns for your custom field.

Login Context System

Sublogin Documentation · /sublogin/features/login-context-system

Understand how sub-account login sessions work.

Overview

The login context determines permissions and data access for each session.

Context Types

Context	Description
Main Account	Full access, admin rights
Sub-Account	Limited by role permissions
Impersonation	Admin acting as customer

Session Data

Each login stores:

- Account type (main/sub)
- Parent account ID
- Permission set
- Session token

Context Switching

Main account holders can:

1. View as sub-account
2. Test permissions
3. Switch back instantly

API Behavior

REST/SOAP (via API add-on) respects context:

- Returns appropriate data
- Enforces permissions
- Logs actions correctly

Security

- Sessions are isolated
- Permissions enforced server-side
- Audit logging enabled

Related

- Multi-Account Management
- Roles & Permissions

Multi-Account Management

Sublogin Documentation · /sublogin/features/multi-account-management

Manage multiple sub-accounts under a main customer account.

Important: Roles & Groups require an add-on

You can create and manage sublogins with the base Sublogin module.

If you want to **define roles**, assign **granular permissions**, and use **hierarchical groups/teams**, you must install the **Roles & Permissions Add-On**:

- Add-On: `mageb2b/sublogin-role` (`MageB2B_SubloginRole`)
- Docs: Roles & Permissions

Overview

Main account holders can create and manage sub-accounts for their organization.

Features

Feature	Description
Create Sub-Accounts	Add users under main account
Manage Permissions	Control what users can do (requires Roles & Permissions add-on for role-based permissions)
View Activity	Monitor sub-account actions
Deactivate Users	Suspend access

Account Hierarchy

```
Main Account (Company Admin)
├── Sub-Account 1 (Purchaser)
├── Sub-Account 2 (Approver)
├── Sub-Account 3 (Viewer)
└── Sub-Account 4 (Finance)
```

Creating Sub-Accounts

1. Login as main account
2. Go to "My Sub-Accounts"
3. Click "Create New"
4. Enter user details
5. Assign role/permissions (requires Roles & Permissions add-on)
6. Save

Sub-Account Settings

Setting	Description
Name	User's full name
Email	Login email
Role	Permission role (requires Roles & Permissions add-on)
Active	Enable/disable
Department / Group	Optional grouping (requires Roles & Permissions add-on for hierarchical groups)

Related

- Address Management
- Roles & Permissions

Payment & Shipping Restrictions

Sublogin Documentation · /sublogin/features/payment-shipping-restrictions

Control payment and shipping options for sub-accounts.

Overview

Restrict which payment methods and shipping options sub-accounts can use.

Payment Restrictions

By Role

Role	Allowed Methods
Admin	All methods
Purchaser	Purchase Order, Invoice
Limited	Credit Card only

Configuration

Per sub-account or role:

- Enable specific methods
- Disable specific methods
- Set spending limits

Shipping Restrictions

By Role

Role	Shipping Options
Admin	All carriers
Standard	Ground only
Urgent	Express available

Address Restrictions

- Ship to company addresses only
- No residential delivery
- Specific warehouses only

Configuration

Stores > Config > Sublogin > Restrictions

Setting	Description
Enforce Payment	Yes/No
Enforce Shipping	Yes/No
Default Profile	Fallback restrictions

Related

- Roles & Permissions
- Budget Management

Shared Resources

Sublogin Documentation · /sublogin/features/shared-resources

Share data and resources across sub-accounts.

Overview

Configure what data is shared between main account and sub-accounts.

Shareable Resources

Resource	Sharing Options
Addresses	Shared / Individual
Payment Methods	Inherited / Own
Order History	Visible / Hidden
Wishlists	Shared / Private
Quotes	Collaborative / Individual

Configuration

My Account > Sublogin Settings > Sharing

Address Sharing

- All addresses visible to subs
- Specific addresses only
- No sharing

Order Visibility

- Sub-accounts see own orders only
- Sub-accounts see all company orders
- Department-based visibility

Quote Collaboration

- Multiple users work on same quote
- Comments and history shared
- Approval workflow

Use Cases

Large Organization

- Shared company addresses
- Individual payment methods
- Department-based orders

Small Team

- Full sharing enabled
- Collaborative quotes
- Unified order history

Related

- Multi-Account Management
- Address Management

Budget API

Sublogin Documentation · /sublogin/addons/api-budget

The Sublogin Budget API allows management of order limits and periodic budgets for both main customers and sublogins.

Base URL

`https://yourstore.com/rest/V1/sublogin-budget`

Endpoints

1. Get Budget by ID

`GET /V1/sublogin-budget/:id` **ACL:** `MageB2B_SubloginBudget::budget_manage`

2. Search Budgets

`GET /V1/sublogin-budget/search` **ACL:** `MageB2B_SubloginBudget::budget_manage`

3. Create Budget

`POST /V1/sublogin-budget` **ACL:** `MageB2B_SubloginBudget::budget_save`

4. Update Budget

`PUT /V1/sublogin-budget/:id` **ACL:** `MageB2B_SubloginBudget::budget_save`

5. Delete Budget

`DELETE /V1/sublogin-budget/:id` **ACL:** `MageB2B_SubloginBudget::budget_delete`

Field Reference

Field	Type	Required	Description
`budget_id`	int	No (auto)	Primary key
`sublogin_id`	int	Conditional*	ID of the sublogin user (FK to `customer_sublogin.id`)
`customer_id`	int	Conditional*	Magento Customer Entity ID (FK to `customer_entity.entity_id`)
`budget_type_id`	int	Yes	Budget period type (see below)
`per_order`	float	No	Maximum amount per single order
`amount`	float	No	Total budget for the period
`valid_from_date`	string	No	Start date (YYYY-MM-DD)
`status`	int	No	1 = active, 0 = disabled

> **Important:** Set either `sublogin_id` OR `customer_id`, never both!

Budget Type IDs (`budget_type_id`)

ID	Label	Description
1	Year	One-time budget for a specific calendar year
2	Yearly	Recurring budget, resets every year
3	Month	One-time budget for a specific calendar month
4	Monthly	Recurring budget, resets every month
5	Day	One-time budget for a specific calendar day
6	Daily	Recurring budget, resets every day
7	Per Order	Limit per single order (no time period)

Extension Attributes

`is_one_time` (int)

If set to `1`, the budget is used once and automatically disabled after one order. All other budget types are ignored while a one-time budget is active.

Examples

Create Monthly Recurring Budget for Sublogin

```
curl -X POST "https://yourstore.com/rest/V1/sublogin-budget"
  -H "Authorization: Bearer <token>"
  -H "Content-Type: application/json"
  -d '{
    "subloginBudget": {
        "sublogin_id": 5,
        "budget_type_id": 4,
        "per_order": 500.00,
        "amount": 2000.00,
        "status": 1
    }
}'
```

Create One-Time Budget

```
curl -X POST "https://yourstore.com/rest/V1/sublogin-budget"
  -H "Authorization: Bearer <token>"
  -H "Content-Type: application/json"
  -d '{
```

```
    "subloginBudget": {
        "sublogin_id": 5,
        "budget_type_id": 7,
        "amount": 500.00,
        "status": 1,
        "extension_attributes": {
            "is_one_time": 1
        }
    }
}'
```

REST API

Sublogin Documentation · /sublogin/addons/api

The Sublogin API add-on (`MageB2B_SubloginApi`) exposes Sublogin CRUD endpoints via Magento Web API (REST/SOAP) for integration with ERP, CRM, or automation scripts.

Base URL

`https://yourstore.com/rest/V1/sublogin`

Endpoints

1. Get Sublogin by ID

`GET /V1/sublogin/:id` **ACL:** `MageB2B_Sublogin::manage`

2. Search Sublogins

`GET /V1/sublogin/search` **ACL:** `MageB2B_Sublogin::manage` Uses standard Magento `searchCriteria`.

3. Create Sublogin

`POST /V1/sublogin` **ACL:** `MageB2B_Sublogin::save`

4. Update Sublogin

`PUT /V1/sublogin/:id` **ACL:** `MageB2B_Sublogin::save`

5. Delete Sublogin

`DELETE /V1/sublogin/:id` **ACL:** `MageB2B_Sublogin::delete`

Field Reference: Sublogin

Field	Type	Required	Description
`id`	int	No (auto)	Primary key of the sublogin
`entity_id`	int	Yes	Magento Customer Entity ID (FK to `customer_entity.entity_id`)
`customer_id`	string	No	External customer number (e.g., ERP number)
`email`	string	Yes	Login email address (must be unique)
`optional_email`	string	No	Alternative email for notifications
`password`	string	Yes (POST)	Password (stored hashed)
`firstname`	string	Yes	First name
`lastname`	string	Yes	Last name

Field	Type	Required	Description
`prefix`	string	No	Prefix (e.g., "Mr.", "Ms.", "Dr.")
`expire_date`	string	No	Account expiration date (YYYY-MM-DD), null = no expiration
`active`	bool	No	true = active, false = disabled
`send_backendmails`	bool	No	Receives copies of backend emails (order confirmations, etc.)
`store_id`	int	No	Store View ID
`create_sublogins`	bool	No	Permission to create further sublogins
`is_subscribed`	bool	No	Newsletter subscription
`last_login_at`	string	No (auto)	Last login timestamp
`default_billing`	int	No	ID of the default billing address
`default_shipping`	int	No	ID of the default shipping address
`customer_address_ids`	string[]	No	IDs of allowed customer addresses (from parent customer)
`addresses`	Address[]	No	Custom sublogin addresses

Field Reference: Address

Field	Type	Required	Description
`id`	int	No (auto)	Primary key
`prefix`	string	No	Prefix
`firstname`	string	Yes	First name
`lastname`	string	Yes	Last name
`company`	string	No	Company name
`street`	string[]	Yes	Street and house number (Array for multi-line)
`city`	string	Yes	City
`country_id`	string	Yes	Country code (ISO 3166-1 alpha-2)
`postcode`	string	Yes	Postcode
`telephone`	string	Yes	Telephone number
`default_billing`	bool	No	Set as default billing address
`default_shipping`	bool	No	Set as default shipping address

ID Logic

- **id**: Primary key of the sublogin record.
- **entity_id**: Magento Customer Entity ID (parent customer).
- **customer_id**: External ID (e.g., "K-2024-00123" from ERP).

Address Concept

Sublogins can use two types of addresses:

1. **Customer Addresses**: Referenced by IDs from the parent customer.
2. **Custom Addresses**: Independent addresses specific to the sublogin.

Examples

Create Sublogin with Custom Address

```
curl -X POST "https://yourstore.com/rest/V1/sublogin" \
  -H "Authorization: Bearer <token>" \
  -H "Content-Type: application/json" \
  -d '{
    "sublogin": {
        "entity_id": 123,
        "email": "purchasing@company.com",
        "password": "Secure123!",
        "firstname": "John",
        "lastname": "Doe",
        "active": true,
        "addresses": [
            {
                "firstname": "John",
                "lastname": "Doe",
                "company": "Company Ltd - Purchasing",
                "street": ["Industrial St 10"],
                "city": "Hamburg",
                "country_id": "DE",
                "postcode": "20095",
                "telephone": "+49 40 987654",
                "default_billing": true,
                "default_shipping": true
            }
        ]
    }
}'
```

Search Sublogins for a Customer

```
curl -X GET
"https://yourstore.com/rest/V1/sublogin/search?searchCriteria[filterGroups]
[0][filters][0][field]=entity_id&searchCriteria[filterGroups][0][filters][0
][value]=123" \
  -H "Authorization: Bearer <token>"
```

Budget Management

Sublogin Documentation · /sublogin/addons/budget-management

The SubloginBudget add-on provides comprehensive budget management with flexible budget types, automatic tracking, and enforcement.

Overview

The Budget Management add-on extends the base Sublogin module with:

- **7 Budget Types**: Day, Daily, Month, Monthly, Year, Yearly, Per Order
- **Automatic Tracking**: Monitors spending from orders
- **Budget Enforcement**: Blocks checkout when budget exceeded
- **Pay on Budget**: Optional payment method
- **Cron Automation**: Auto-expires budgets

Installation

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/sublogin-budget:*
php bin/magento module:enable MageB2B_SubloginBudget
php bin/magento setup:upgrade
php bin/magento cache:flush
```

Budget Types

1. Day (One-Time Daily Budget)

Type ID: 5
Validity: Valid **only on the specified day**

```
Valid From: 2024-01-15
Amount: €500
```

Behavior:

- Active only on January 15, 2024
- Expires automatically after that day
- Used for special one-day events or promotions

Use Case: Special event purchasing (trade show, conference)

2. Daily (Recurring Daily Budget)

Type ID: 6
Validity: Valid **from start date, renews daily**

```
Valid From: 2024-01-01
Amount: €200
```

Behavior:

- Active every day starting January 1, 2024
- Resets at midnight each day
- No expiration date

Use Case: Daily employee spending limit

3. Month (One-Time Monthly Budget)

Type ID: 3
Validity: Valid **for the entire month from start date**

```
Valid From: 2024-01-15
Amount: €5000
```

Behavior:

- Active for entire January 2024 (even if starting mid-month)
- Expires at end of January
- Covers whole month regardless of start date

Use Case: Project budget for specific month

4. Monthly (Recurring Monthly Budget)

Type ID: 4
Validity: Valid **from start date, renews monthly**

```
Valid From: 2024-01-01
Amount: €10000
```

Behavior:

- Active every month starting January 2024
- Resets on 1st of each month
- No expiration date

Use Case: Department monthly budget

5. Year (One-Time Yearly Budget)

Type ID: 1
Validity: Valid **for the entire year from start date**

```
Valid From: 2024-01-01
Amount: €50000
```

Behavior:

- Active for entire year 2024
- Expires at end of 2024
- Covers whole year

Use Case: Annual project budget

6. Yearly (Recurring Yearly Budget)

Type ID: 2
Validity: Valid **from start date, renews yearly**

```
Valid From: 2024-01-01
Amount: €100000
```

Behavior:

- Active every year starting 2024
- Resets on January 1st each year
- No expiration date

Use Case: Annual department budget

7. Per Order

Type ID: 7
Validity: Applies to **each individual order**

```
Per Order Limit: €1000
```

Behavior:

- Checks each order individually
- No time period
- Always active

Use Case: Maximum order value limit

Creating Budgets

Method 1: Admin Panel

1. Navigate to **Customers > All Customers**
2. Edit customer
3. Go to **Sublogins** tab
4. Edit a sublogin
5. Scroll to **Budget** section
6. Click **Add Budget**

Budget Form:

```
Budget Type: Monthly
Amount: 5000.00
```

```
Per Order Limit: 1000.00
Valid From Date: 2024-01-01
Status: Active
```

7. Click **Save Budget**

Method 2: Customer Account (if enabled)

1. Log in as customer
2. Go to **My Sublogins**
3. Edit sublogin
4. Manage budgets
5. Add/edit budgets

Budget Calculation

Calculation Basis

Configure in **Stores > Configuration > MageB2B > Sublogin Budget**:

Option 1: Subtotal

```
Order Subtotal: €1000
Tax: €200
Shipping: €50
Grand Total: €1250

Budget Used: €1000 (subtotal only)
```

Option 2: Grand Total

```
Order Subtotal: €1000
Tax: €200
Shipping: €50
Grand Total: €1250

Budget Used: €1250 (full amount)
```

Order Status Filtering

Configure which order statuses count toward budget:

- Pending
- Processing
- Complete
- (Exclude: Canceled, Closed)

Multiple Budget Stacking

If multiple budget types apply, they **stack (sum)**:

```
Day Budget: €300
Daily Budget: €500
Total Available Today: €800
```

Example:

- Sublogin has Daily budget of €500
- Special Day budget of €300 for today
- Can spend up to €800 today

Budget Enforcement

Checkout Blocking

When budget would be exceeded:

1. Cart total is calculated
2. System checks available budget
3. If exceeded, checkout is blocked
4. Error message displayed:

```
We are sorry, the checkout is not possible.
Your budget limit for this Month of €5000.00
will be exceeded by €250.00.
```

Error Message Customization

Configure in **Stores > Configuration > MageB2B > Sublogin Budget**:

```
Budget Error Message:
We are sorry, the checkout is not possible.
Your budget limit for this %1 of %2 will be exceeded by %3.

Variables:
%1 = Budget period (Day, Month, Year)
%2 = Budget limit
%3 = Amount exceeded
```

Per Order Limit

Separate from period budgets:

```
Monthly Budget: €10000 (remaining: €8000)
Per Order Limit: €1000

Order Total: €1200
Result: BLOCKED (exceeds per order limit)

Order Total: €900
Result: ALLOWED (within per order limit and monthly budget)
```

Pay on Budget Payment Method

Overview

Optional payment method that deducts from budget.

Configuration

1. Go to **Stores > Configuration > Sales > Payment Methods**
2. Find **Pay on Budget**
3. Enable and configure:
    - Title: "Pay from Budget"
    - Sort Order: 10
    - Applicable Countries: All

Behavior

- Only shown if order is within budget
- Automatically hidden if budget insufficient
- Order is marked as "paid from budget"

Budget Calculation Options

Option 1: Count All Orders

```
All orders in configured statuses count toward budget
```

Option 2: Count Only "Pay on Budget" Orders

```
Only orders paid with "Pay on Budget" method count
Other payment methods don't affect budget
```

Configure in **Stores > Configuration > MageB2B > Sublogin Budget**:

```
Use only Pay on Budget orders for budget usage: Yes/No
```

Cron Jobs

Auto-Expire Budgets

Runs daily to disable expired budgets:

```
Cron Schedule: 0 2 * * * (2 AM daily)
```

What it does:

- Checks all active budgets
- Disables Day budgets after their date

- Disables Month budgets at end of month
- Disables Year budgets at end of year
- Sets status = 0 (inactive)

Manual Cron Execution

```
php bin/magento cron:run --group=default
```

Budget Reports

Frontend Budget Summary

Enable in **Stores > Configuration > MageB2B > Sublogin Budget**:

```
Enable Budget Summary in Frontend for Customer: Yes
```

Displays:

- Current budget limits
- Used budget
- Remaining budget
- Budget period
- Expiration date

Admin Budget Overview

Navigate to **Customers > All Customers > Edit Customer > Sublogins > Edit Sublogin > Budget**

Shows:

- All budgets for sublogin
- Budget type
- Amount and used amount
- Valid from date
- Status
- Actions (Edit, Delete)

Practical Examples

Example 1: Department Monthly Budget

```
Scenario: Marketing department with €10,000/month

Budget Type: Monthly
Amount: 10000.00
Per Order Limit: 2000.00
Valid From: 2024-01-01
Status: Active
```

```
Behavior:
- Resets on 1st of each month
- Maximum €2000 per order
- Tracks all orders in "Processing" and "Complete" status
```

Example 2: Employee Daily Limit

```
Scenario: Employee with €200/day spending limit

Budget Type: Daily
Amount: 200.00
Per Order Limit: 200.00
Valid From: 2024-01-01
Status: Active

Behavior:
- Resets at midnight each day
- Cannot exceed €200 per day
- Cannot exceed €200 per order
```

Example 3: Project Budget

```
Scenario: 6-month project with €30,000 budget

Budget Type: Month (one-time)
Amount: 30000.00
Valid From: 2024-01-01
Status: Active

Note: Create 6 separate Month budgets for Jan-Jun
Or use Year budget with €30,000 for entire year
```

Example 4: Special Event

```
Scenario: Trade show with €5000 budget for one day

Budget Type: Day
Amount: 5000.00
Valid From: 2024-03-15
Status: Active

Behavior:
- Active only on March 15, 2024
- Auto-expires after that day
- Combined with Daily budget if exists
```

Troubleshooting

Budget Not Enforcing

1. Check budget status is Active
2. Verify valid from date is in past
3. Check budget type is correct
4. Ensure order status is in configured list
5. Clear cache

Budget Calculation Wrong

1. Check calculation basis (Subtotal vs Grand Total)
2. Verify order statuses configuration
3. Check if "Pay on Budget only" is enabled
4. Review order history

Cron Not Running

1. Check cron is configured: `crontab -l`
2. Run manually: `php bin/magento cron:run`
3. Check `cron_schedule` table
4. Review `var/log/system.log`

Pay on Budget Not Showing

1. Verify payment method is enabled
2. Check budget is sufficient
3. Ensure cart total is within budget
4. Clear cache

Configuration Reference

General Settings

```
Stores > Configuration > MageB2B > Sublogin Budget

- Enable Budget System: Yes/No
- Budget Calculation Basis: Subtotal / Grand Total
- Order Statuses for Budget: Pending, Processing, Complete
- Use only Pay on Budget orders: Yes/No
- Enable Budget Summary in Frontend: Yes/No
```

Error Messages

```
- Budget Exceeded Message
- Per Order Limit Exceeded Message
- Budget Expired Message
```

Email Notifications

```
- Budget Warning Threshold: 80%
- Budget Warning Email Template
- Budget Exceeded Email Template
```

Next Steps

- Order Approval
- Roles & Permissions
- Catalog Visibility

Catalog Visibility

Sublogin Documentation · /sublogin/addons/catalog-visibility

The SubloginCatalogVisibility add-on provides product visibility restrictions per sublogin with whitelist and blacklist modes.

Overview

The Catalog Visibility add-on extends the base Sublogin module with:

- **Product Restrictions**: Control which products each sublogin can see
- **Whitelist Mode**: Show only assigned products
- **Blacklist Mode**: Hide assigned products
- **Collection Filtering**: Automatic filtering across all product lists
- **Search Integration**: Restricted products don't appear in search

Installation

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/sublogin-catalogvisibility:*
php bin/magento module:enable MageB2B_SubloginCatalogVisibility
php bin/magento setup:upgrade
php bin/magento cache:flush
```

Core Concepts

Restriction Type

Determines how product assignments are interpreted:

- **Whitelist (0)**: Show ONLY assigned products
- **Blacklist (1)**: Hide assigned products, show all others

Product Assignment

Products are assigned to sublogins via `customer_sublogin_catalog_visibility` table.

Collection Filtering

System automatically filters product collections based on restrictions.

Configuration

Set Restriction Type for Sublogin

Method 1: Admin Panel

1. Navigate to **Customers > All Customers**
2. Edit customer
3. Go to **Sublogins** tab
4. Edit sublogin
5. Find **Catalog Visibility** section
6. Set **Restriction Type**:
    - Whitelist: Show only assigned products
    - Blacklist: Hide assigned products
7. Save sublogin

Method 2: Customer Account

1. Log in as customer
2. Go to **My Sublogins**
3. Edit sublogin
4. Configure catalog visibility
5. Save

Restriction Modes

Whitelist Mode (Restrictive)

Behavior: Sublogin can ONLY see assigned products.

Configuration:

```
Restriction Type: Whitelist (0)
Assigned Products: Product A, Product B, Product C
```

Result:

- Product A: ✓ Visible
- Product B: ✓ Visible
- Product C: ✓ Visible
- Product D: ✗ Hidden
- Product E: ✗ Hidden

Use Cases:

- Sales rep with specific product portfolio
- Department-specific catalogs
- Regional product restrictions
- Customer-specific assortments

Blacklist Mode (Permissive)

Behavior: Sublogin can see ALL products EXCEPT assigned ones.

Configuration:

```
Restriction Type: Blacklist (1)
Assigned Products: Product X, Product Y
```

Result:

- Product A: ✓ Visible
- Product B: ✓ Visible
- Product C: ✓ Visible
- Product X: ✗ Hidden
- Product Y: ✗ Hidden

Use Cases:

- Hide discontinued products
- Exclude competitor products
- Remove restricted items
- Hide internal-use products

Assigning Products

Method 1: Admin Panel - Sublogin Edit

1. Edit sublogin
2. Go to **Catalog Visibility** section
3. Click **Assign Products**
4. Product grid appears with checkboxes
5. Search/filter products
6. Check products to assign
7. Click **Save**

Product Grid Columns:

- Checkbox
- Product ID
- Name
- SKU
- Price
- Status
- Visibility

Method 2: Admin Panel - Product Edit

1. Edit product
2. Go to **Sublogin Visibility** section
3. Select sublogins to assign
4. Save product

Method 3: Bulk Assignment

1. Go to **Catalog > Products**
2. Select multiple products
3. Actions dropdown: **Assign to Sublogins**
4. Select sublogins
5. Submit

Where Restrictions Apply

Category Pages

Products are filtered based on restrictions:

```
Category: Electronics
Total Products: 100

Sublogin with Whitelist (10 products assigned):
→ Shows 10 products

Sublogin with Blacklist (5 products assigned):
→ Shows 95 products
```

Search Results

Restricted products don't appear in search:

```
Search: "laptop"
Total Results: 20

Sublogin with Whitelist (3 laptops assigned):
→ Shows 3 results

Sublogin with Blacklist (2 laptops assigned):
→ Shows 18 results
```

Product Detail Pages

Direct access to restricted products:

```
Whitelist Mode:
- Assigned product: Page loads normally
- Non-assigned product: 404 Not Found

Blacklist Mode:
- Non-assigned product: Page loads normally
- Assigned product: 404 Not Found
```

Related Products

Related products are also filtered:

```
Product A has related products: B, C, D, E

Sublogin can only see: A, B, C
→ Related products shown: B, C
→ Related products hidden: D, E
```

Upsell/Cross-sell

Same filtering applies to upsell and cross-sell products.

Layered Navigation

Filters reflect only visible products:

```
Category: Clothing
Filter: Size

Sublogin sees 10 products (sizes: S, M, L)
→ Size filter shows: S, M, L
→ Size filter hides: XL, XXL (no visible products)
```

Practical Examples

Example 1: Sales Representative Portfolio

Scenario: Sales rep can only sell specific product lines.

```
Sublogin: Sales Rep A
Restriction Type: Whitelist
Assigned Products:
- Product Line: Office Furniture
- Categories: Desks, Chairs, Storage
- Total Products: 45

Result:
- Can see: 45 office furniture products
- Cannot see: All other products
- Search: Only returns office furniture
- Categories: Only office furniture categories visible
```

Use Case: Territory-based sales, product specialization

Example 2: Regional Restrictions

Scenario: EU sublogin cannot see US-only products.

```
Sublogin: EU Branch
Restriction Type: Blacklist
Assigned Products:
- Products with attribute "region" = "US Only"
- Total Products: 120

Result:
- Can see: All products except 120 US-only
- Cannot see: 120 US-only products
```

```
- Search: US-only products excluded
```

Use Case: Regional compliance, shipping restrictions

Example 3: Department-Specific Catalog

Scenario: IT department only sees IT products.

```
Sublogin: IT Department
Restriction Type: Whitelist
Assigned Products:
- Category: Computers & Electronics
- Category: Software
- Category: Accessories
- Total Products: 200

Result:
- Can see: 200 IT-related products
- Cannot see: Office supplies, furniture, etc.
- Budget: Applied only to visible products
```

Use Case: Department budgets, purchasing control

Example 4: Hide Discontinued Products

Scenario: Hide discontinued products from all sublogins.

```
Sublogin: All Employees
Restriction Type: Blacklist
Assigned Products:
- Products with status "Discontinued"
- Total Products: 35

Result:
- Can see: All active products
- Cannot see: 35 discontinued products
- Main customer: Can still see all products
```

Use Case: Product lifecycle management

Integration with Other Features

With Roles & Permissions

```
Sublogin: Sales Rep
Role: Sales Representative
Permissions:
- view_product_list
- view_product_details
- view_product_prices
```

```
- add_product_to_cart
Catalog Visibility:
- Whitelist: 50 assigned products

Result:
- Can view and add to cart: 50 assigned products
- Cannot checkout (no place_order permission)
```

With Budget Management

```
Sublogin: Department Manager
Budget: €5000/month
Catalog Visibility:
- Whitelist: Department-specific products

Result:
- Budget applies only to visible products
- Cannot exceed budget on visible products
- Cannot see or order restricted products
```

With Order Approval

```
Sublogin: Employee
Order Approval: Required
Catalog Visibility:
- Whitelist: Approved products only

Result:
- Can only order from approved product list
- Orders require manager approval
- Manager sees same product restrictions
```

Performance Considerations

Collection Filtering

System adds WHERE clause to product collections:

```
-- Whitelist Mode
WHERE entity_id IN (SELECT product_id FROM
customer_sublogin_catalog_visibility WHERE sublogin_id = ?)

-- Blacklist Mode
WHERE entity_id NOT IN (SELECT product_id FROM
customer_sublogin_catalog_visibility WHERE sublogin_id = ?)
```

Caching

- Product collections are cached per sublogin

- Cache is invalidated when assignments change
- Full page cache considers sublogin ID

Indexing

- No additional indexes required
- Uses existing product indexes
- Foreign key indexes on visibility table

Troubleshooting

Products Not Filtering

1. Check restriction type is set
2. Verify products are assigned
3. Clear cache: `php bin/magento cache:flush`
4. Reindex: `php bin/magento indexer:reindex`

Wrong Products Showing

1. Verify restriction type (Whitelist vs Blacklist)
2. Check product assignments
3. Review sublogin configuration
4. Check if logged in as correct sublogin

Product Detail Page 404

1. Check if product is assigned (Whitelist mode)
2. Check if product is blacklisted (Blacklist mode)
3. Verify product is enabled and visible
4. Check product website assignment

Search Not Filtering

1. Verify search index is up to date
2. Reindex catalog search: `php bin/magento indexer:reindex catalogsearch_fulltext`
3. Clear cache
4. Check search configuration

Configuration Reference

```
Stores > Configuration > MageB2B > Sublogin Catalog Visibility

- Enable Catalog Visibility: Yes/No
- Default Restriction Type: Whitelist/Blacklist
- Apply to Search: Yes/No
- Apply to Related Products: Yes/No
- Apply to Upsell/Cross-sell: Yes/No
- Show Empty Categories: Yes/No
- Redirect Restricted Products: 404/Home/Category
```

API Integration

Get Visible Products for Sublogin

```
GET /rest/V1/sublogin/:subloginId/catalog/products
```

Assign Products to Sublogin

```
POST /rest/V1/sublogin/:subloginId/catalog/assign
{
  "productIds": [1, 2, 3, 4, 5]
}
```

Remove Product Assignments

```
DELETE /rest/V1/sublogin/:subloginId/catalog/product/:productId
```

Import/Export

Export Product Assignments

```
php bin/magento sublogin:catalog:export --sublogin-id=5 --
output=assignments.csv
```

Import Product Assignments

```
php bin/magento sublogin:catalog:import --file=assignments.csv
```

CSV Format:

```
sublogin_id,product_id,restriction_type
5,101,0
5,102,0
5,103,0
```

Next Steps

- Roles & Permissions
- Budget Management
- Order Approval
- Multi-Account Management

Hyva

Sublogin Documentation · /sublogin/addons/hyva

If your storefront uses the Hyvä theme, install the Sublogin Hyvä compatibility add-on(s) to ensure the Sublogin pages and account navigation render correctly.

Sublogin Hyvä

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/sublogin-hyva:*
php bin/magento module:enable MageB2B_SubloginHyva
php bin/magento setup:upgrade
php bin/magento cache:flush
```

This add-on provides Hyvä-compatible layouts for common Sublogin customer pages (e.g. sublogin list/edit and account dashboard integration).

> Screenshot placeholder: Hyvä “My Sublogins” page

Sublogin Role Hyvä

If you use the Roles & Permissions add-on, also install the Hyvä compatibility module for it:

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/sublogin-role-hyva:*
php bin/magento module:enable MageB2B_SubloginRoleHyva
php bin/magento setup:upgrade
php bin/magento cache:flush
```

> Screenshot placeholder: Hyvä role/group management pages

Import Export

Sublogin Documentation · /sublogin/addons/import-export

The Import/Export add-on (`MageB2B_SubloginImportExport`) adds **bulk tooling** for Sublogins:

- Export selected sublogins from the **Admin grid** as CSV
- Import sublogins from a CSV via a **CLI command**

Installation

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/sublogin-importexport:*
php bin/magento module:enable MageB2B_SubloginImportExport
php bin/magento setup:upgrade
php bin/magento cache:flush
```

Admin Export (Grid Mass Action)

After enabling the module you get an **Export** mass action in:

- **Customers > Sublogins** (Sublogin grid)

Workflow:

1. Select one or more sublogins in the grid
2. Choose **Export**
3. Magento downloads a generated CSV

> Screenshot placeholder: Sublogin grid with “Export” mass action

Export Format Notes

- Passwords are **not** exported.
- The export includes **up to 3 addresses** per sublogin (`address_1_*` … `address_3_*`).

CLI Import

The module provides a CLI command:

```
php bin/magento sublogin:import-sublogin <path-to-csv> --
behavior=add_update
```

Supported behaviors:

- `append`

- `add_update`
- `replace`
- `delete`

Optional flags:

- `--field_separator=,`
- `--field_multiple_value_separator=,`
- `--fields_enclosure=0`
- `--delete_file_after_import=1` (defaults to `1`)

> Screenshot placeholder: CLI import run + resulting sublogins in admin grid

Sample CSV

A sample CSV file is shipped with the module:

- `vendor/mageb2b/sublogin-importexport/Files/Sample/sublogin.csv`

Tip: Start by exporting a few existing sublogins and use that CSV as your “real” template to avoid missing fields.

Mageplaza Osc

Sublogin Documentation · /sublogin/addons/mageplaza-osc

If you use **Mageplaza One Step Checkout**, this compatibility add-on ensures Sublogin address/billing logic works correctly within the Mageplaza checkout UI.

Installation

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/sublogin-mageplaza-osc:*
php bin/magento module:enable MageB2B_SubloginMageplazaOsc
php bin/magento setup:upgrade
php bin/magento cache:flush
```

What It Does

The module applies a RequireJS mixin to Mageplaza’s billing address component so Sublogin billing address restrictions (and related logic) are respected during checkout.

> Screenshot placeholder: Mageplaza OSC checkout with Sublogin billing address selection

Order Approval

Sublogin Documentation · /sublogin/addons/order-approval

The SubloginOrderApproval add-on provides multi-level order approval workflows with email notifications and role-based approvers.

Overview

The Order Approval add-on extends the base Sublogin module with:

- **Approval Workflows**: Orders require approval before processing
- **Amount Thresholds**: Approval based on order value
- **Role-Based Approvers**: Permissions control who can approve
- **Email Notifications**: Automatic approval request emails
- **Approval Tokens**: Secure approval links
- **Multi-Level Approval**: Hierarchical approval chains

Installation

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/sublogin-orderapproval:*
php bin/magento module:enable MageB2B_SubloginOrderApproval
php bin/magento setup:upgrade
php bin/magento cache:flush
```

Note: Requires SubloginRole add-on for role-based approvers.

Core Concepts

Order Needs Approval

Flag on sublogin that determines if their orders require approval.

Approval Amount Check

Threshold value - orders above this amount require approval.

Approval Token

Unique token generated for each order requiring approval.

Approver IDs

List of sublogin IDs who can approve the order.

Configuration

Enable Order Approval for Sublogin

Method 1: Admin Panel

1. Navigate to **Customers > All Customers**
2. Edit customer
3. Go to **Sublogins** tab
4. Edit sublogin
5. Find **Order Approval** section
6. Set **Order Needs Approval**: Yes
7. Set **Order Approval Amount Check**: 1000.00
8. Save sublogin

Method 2: Customer Account

1. Log in as customer
2. Go to **My Sublogins**
3. Edit sublogin
4. Configure approval settings
5. Save

Approval Settings

Order Needs Approval

```
Yes: All orders require approval
No: Orders process normally
```

Order Approval Amount Check

```
Amount: 1000.00

Orders <= €1000: Process normally
Orders > €1000: Require approval
```

Example:

```
Order Needs Approval: Yes
Amount Check: 1000.00

Order €500: Requires approval (flag is Yes)
Order €1500: Requires approval (flag is Yes AND exceeds amount)

Order Needs Approval: No
Amount Check: 1000.00

Order €500: No approval needed
Order €1500: No approval needed (flag is No)
```

Approval Workflow

Step 1: Sublogin Places Order

1. Sublogin adds products to cart
2. Proceeds to checkout
3. Places order
4. Order is created with status: **Pending Approval**

Step 2: Approval Token Generated

```
Order #000123
Approval Token: a1b2c3d4e5f6g7h8i9j0
Approver IDs: 5,12,18
```

Step 3: Email Notification Sent

Email sent to all approvers:

```
Subject: Order Approval Required - Order #000123

Dear [Approver Name],

A new order requires your approval:

Order Number: #000123
Placed By: John Doe (john@company.com)
Order Total: €1,250.00
Date: January 15, 2024

[Approve Order] [Decline Order]

Or use this link:
https://store.com/sublogin/approval/approve/token/a1b2c3d4e5f6g7h8i9j0
```

Step 4: Approver Reviews Order

Approver can:

- View order details
- Check budget availability
- Review products ordered
- Approve or decline

Step 5: Approval Decision

If Approved:

1. Order status changes to: **Processing**
2. Order is processed normally
3. Confirmation email sent to sublogin

4. Approver ID added to `sublogin_approver_ids`

If Declined:

1. Order status changes to: **Canceled**
2. Decline reason recorded
3. Notification email sent to sublogin
4. Order cannot be reactivated

Approval Permissions

Requires SubloginRole add-on for permission-based approval.

Approval Permissions

- `approve_order_all` - Approve any order
- `approve_order_same_level` - Approve orders from same group level
- `approve_order_lower_level` - Approve orders from lower group levels
- `decline_order_all` - Decline any order
- `decline_order_same_level` - Decline orders from same level
- `decline_order_lower_level` - Decline orders from lower levels

Example Role: Manager

```
Role: Department Manager
Permissions:
- approve_order_same_level
- approve_order_lower_level
- decline_order_same_level
- decline_order_lower_level

Can approve/decline:
- Orders from same department
- Orders from sub-departments
Cannot approve/decline:
- Orders from other departments
- Orders from higher levels
```

Example Role: Director

```
Role: Director
Permissions:
- approve_order_all
- decline_order_all

Can approve/decline:
- All orders regardless of level
```

Determining Approvers

Method 1: Role-Based (Automatic)

System automatically finds approvers based on permissions:

```
Sublogin A (Employee) places order
Group: Sales Department

System searches for sublogins with:
- approve_order_all permission (any group)
- approve_order_same_level permission (Sales Department)
- approve_order_lower_level permission (parent groups)

Found Approvers:
- Manager B (Sales Department, same level)
- Director C (Company, higher level with _all permission)
```

Method 2: Manual Assignment

Admin manually assigns approvers to sublogin:

```
Sublogin Settings:
Approvers: Manager B, Director C

All orders from this sublogin go to these approvers
```

Multi-Level Approval

Scenario: Hierarchical Approval Chain

```
Company Structure:
├── Director (Level 0)
│   └── Manager (Level 1)
│       └── Employee (Level 2)

Approval Rules:
- Orders €0-€500: No approval
- Orders €500-€2000: Manager approval
- Orders €2000+: Director approval
```

Implementation:

```
Employee Sublogin:
- Order Needs Approval: Yes
- Amount Check: 500.00

Manager Role:
- approve_order_lower_level (for Employee orders)
```

```
- Amount limit: €2000

Director Role:
- approve_order_all
- No amount limit
```

Workflow:

```
Employee orders €300:
→ No approval needed (< €500)

Employee orders €1000:
→ Manager approval required (€500-€2000)

Employee orders €3000:
→ Director approval required (> €2000)
→ Manager cannot approve (exceeds their limit)
```

Approval Methods

Method 1: Email Link

1. Approver receives email
2. Clicks "Approve Order" button
3. Redirected to approval page
4. Confirms approval
5. Order is approved

Method 2: Frontend Account

1. Approver logs in
2. Goes to **My Account > Orders Pending Approval**
3. Views order details
4. Clicks **Approve** or **Decline**
5. Adds optional comment
6. Confirms decision

Method 3: Admin Panel

1. Admin logs in
2. Goes to **Sales > Orders**
3. Filters by "Pending Approval" status
4. Opens order
5. Clicks **Approve Order** or **Decline Order**
6. Adds comment
7. Saves

Decline Reasons

When declining an order, approver can provide reason:

```
Decline Reasons:
- Exceeds budget
- Unauthorized purchase
- Wrong products
- Duplicate order
- Other (custom reason)
```

Example:

```
Order #000123 - DECLINED

Declined By: Manager B
Decline Reason: Exceeds budget
Comment: Monthly budget already exhausted.
         Please resubmit next month.
Date: January 15, 2024
```

Email Templates

Approval Request Email

```
Template: sublogin_order_approval_request

Variables:
- {{var order.increment_id}}
- {{var sublogin.name}}
- {{var sublogin.email}}
- {{var order.grand_total}}
- {{var approval_url}}
```

Approval Confirmation Email

```
Template: sublogin_order_approved

Variables:
- {{var order.increment_id}}
- {{var approver.name}}
- {{var approval_date}}
- {{var approver_comment}}
```

Decline Notification Email

```
Template: sublogin_order_declined

Variables:
- {{var order.increment_id}}
- {{var approver.name}}
- {{var decline_reason}}
```

```
- {{var decline_comment}}
```

Integration with Budget Add-On

When both Order Approval and Budget add-ons are installed:

Scenario 1: Budget Check Before Approval

```
Order Total: €1500
Budget Remaining: €1000

Result: Order blocked at checkout
Reason: Exceeds budget (no approval needed)
```

Scenario 2: Budget Check After Approval

```
Order Total: €1500
Budget Remaining: €2000
Requires Approval: Yes

Workflow:
1. Order placed (within budget)
2. Order status: Pending Approval
3. Approver approves
4. Budget is deducted
5. Order status: Processing
```

Scenario 3: Budget Exceeded During Approval

```
Order Total: €1500
Budget at Order Time: €2000
Budget at Approval Time: €500 (other orders placed)

Result: Approval fails
Reason: Budget no longer sufficient
```

Practical Examples

Example 1: Simple Manager Approval

```
Company: Small Business
Structure: Owner > Employees

Configuration:
- All employee orders require approval
- Owner is the only approver
- No amount threshold
```

```
Employee Sublogin:
- Order Needs Approval: Yes
- Amount Check: 0 (all orders)

Owner Role:
- approve_order_all
- decline_order_all
```

Example 2: Department-Based Approval

```
Company: Medium Business
Structure:
- Company
  - Sales Department
    - Sales Manager
    - Sales Reps
  - IT Department
    - IT Manager
    - IT Staff

Configuration:
- Sales Manager approves Sales Rep orders
- IT Manager approves IT Staff orders
- No cross-department approval

Sales Rep Sublogin:
- Order Needs Approval: Yes
- Group: Sales Department

Sales Manager Role:
- approve_order_same_level
- approve_order_lower_level
- Group: Sales Department
```

Example 3: Amount-Based Escalation

```
Company: Large Enterprise
Structure: Employee > Manager > Director

Configuration:
- €0-€1000: No approval
- €1000-€5000: Manager approval
- €5000+: Director approval

Employee Sublogin:
- Order Needs Approval: Yes
- Amount Check: 1000.00

Manager Role:
- approve_order_lower_level
```

```
- Max Amount: €5000

Director Role:
- approve_order_all
- No amount limit
```

Troubleshooting

Order Not Requiring Approval

1. Check "Order Needs Approval" is Yes
2. Verify order amount vs threshold
3. Check sublogin configuration
4. Clear cache

No Approvers Found

1. Verify SubloginRole add-on is installed
2. Check approval permissions are assigned
3. Verify group structure
4. Check role assignments

Approval Email Not Sent

1. Check email configuration
2. Verify approver email addresses
3. Check email template
4. Review `var/log/system.log`

Cannot Approve Order

1. Check approver has correct permissions
2. Verify approval token is valid
3. Check order status is "Pending Approval"
4. Ensure approver is logged in

Configuration Reference

```
Stores > Configuration > MageB2B > Sublogin Order Approval

- Enable Order Approval: Yes/No
- Default Approval Behavior: Require/Optional
- Email Sender: Sales/Support/Custom
- Approval Request Email Template
- Approval Confirmation Email Template
- Decline Notification Email Template
- Approval Token Expiration: 7 days
- Allow Multiple Approvers: Yes/No
- Require All Approvers: Yes/No (if multiple)
```

Next Steps

- Budget Management
- Roles & Permissions
- Multi-Account Management

Reports

Sublogin Documentation · /sublogin/addons/reports

The Reports add-on (`MageB2B_SubloginReport`) adds reporting screens for Sublogin activity in both **Admin** and **Customer Account**.

Installation

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/sublogin-report:*
php bin/magento module:enable MageB2B_SubloginReport
php bin/magento setup:upgrade
php bin/magento cache:flush
```

Admin: Reports Menu

After enabling the module, Admin gets a new menu branch under **Customers > Sublogins**:

- **Reports > Orders**
- **Reports > Products**

> Screenshot placeholder: Admin “Customers > Sublogins > Reports” menu

Customer Account: Report Links

The module adds two entries to the customer account navigation (when Sublogin is enabled):

- **My Sublogin Orders Report**
- **My Sublogin Products Report**

> Screenshot placeholder: Customer account navigation with report links

Configuration

Location: **Stores > Configuration > MageB2B > Sublogin > Report settings**

- `sublogin/report/enable_for_customer` – enable reports for the main customer account
- `sublogin/report/enable_for_sublogin` – enable reports for sublogin accounts

ACL (Admin Permissions)

To allow admin users to access the report pages, grant:

- `MageB2B_SubloginReport::report`
- `MageB2B_SubloginReport::report_orders`
- `MageB2B_SubloginReport::report_products`

Roles Permissions

Sublogin Documentation · /sublogin/addons/roles-permissions

The SubloginRole add-on provides a sophisticated permission system with roles, groups, and 30+ granular permissions.

Overview

The Roles & Permissions add-on extends the base Sublogin module with:

- **Roles**: Define custom roles with specific permissions
- **Groups**: Hierarchical organization structure
- **Permissions**: 30+ predefined permissions covering all aspects
- **Hierarchical Access**: All/Same Level/Lower Level permissions

Installation

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/sublogin-role:*
php bin/magento module:enable MageB2B_SubloginRole
php bin/magento setup:upgrade
php bin/magento cache:flush
```

Core Concepts

Roles

A role is a collection of permissions that can be assigned to sublogins.

Example Roles:

- **Purchaser**: Can view products, add to cart, and place orders
- **Manager**: Full access including order approval
- **Sales Rep**: Can view products but cannot checkout
- **Viewer**: Read-only access to orders and products

Groups

Groups create a hierarchical organization structure.

Example Structure:

```
Customer Account
├── General (Group, Level 0)
│   ├── Branch A (Group, Level 1)
│   │   ├── Department 1 (Group, Level 2)
│   │   └── Department 2 (Group, Level 2)
```

```
│   └── Branch B (Group, Level 1)
│       ├── Sales Team (Group, Level 2)
│       └── Support Team (Group, Level 2)
```

Permissions

Permissions control what actions sublogins can perform.

Permission Categories:

- Catalog
- Checkout
- Order
- Invoice
- Sublogin Management
- Role Management
- Wishlist

Available Permissions

Catalog Permissions

- `view_product_list` - View product listings
- `view_product_details` - View product detail pages
- `view_product_prices` - See product prices

Checkout Permissions

- `add_product_to_cart` - Add products to cart
- `view_cart` - View shopping cart
- `view_checkout` - Access checkout page
- `place_order` - Complete order placement

Order Permissions

- `view_order_all` - View all customer orders
- `view_order_same_level` - View orders from same group level
- `view_order_lower_level` - View orders from lower group levels

Invoice Permissions

- `view_invoice_all` - View all invoices
- `view_invoice_same_level` - View invoices from same level
- `view_invoice_lower_level` - View invoices from lower levels

Sublogin Management Permissions

- `list` - View sublogin list
- `save` - Create new sublogins
- `edit_all` - Edit all sublogins
- `edit_same_level` - Edit sublogins at same level
- `edit_lower_level` - Edit sublogins at lower levels
- `login_as_sublogin_all` - Impersonate any sublogin

- `login_as_sublogin_same_level` - Impersonate same level
- `login_as_sublogin_lower_level` - Impersonate lower levels
- `delete_all` - Delete any sublogin
- `delete_same_level` - Delete same level sublogins
- `delete_lower_level` - Delete lower level sublogins
- `delete_own_account` - Delete own account

Role Management Permissions

- `list` - View roles
- `save` - Create/edit roles
- `delete` - Delete roles
- `list_group` - View groups
- `save_group` - Create/edit groups
- `delete_group` - Delete groups

Wishlist Permissions

- `view_wishlist` - View wishlist

Order Approval Permissions (requires Order Approval add-on)

- `approve_order_all` - Approve any order
- `approve_order_same_level` - Approve same level orders
- `approve_order_lower_level` - Approve lower level orders
- `decline_order_all` - Decline any order
- `decline_order_same_level` - Decline same level orders
- `decline_order_lower_level` - Decline lower level orders

Budget Permissions (requires Budget add-on)

- `view_own_budget` - View own budget
- `manage_budget` - Manage budgets

Creating Roles

Step 1: Navigate to Roles

1. Log in to Magento Admin
2. Go to **Customers > Sublogin Roles**
3. Click **Add New Role**

Step 2: Basic Information

Role Name

```
Purchaser
```

Description

```
Can view products, add to cart, and place orders
```

Customer Select the customer account this role belongs to.

Step 3: Assign Permissions

Check the permissions this role should have:

For Purchaser Role:

- view_product_list
- view_product_details
- view_product_prices
- add_product_to_cart
- view_cart
- view_checkout
- place_order
- view_order_all
- ☐ edit_all (no sublogin management)
- ☐ approve_order_all (no approval rights)

Step 4: Save Role

Click **Save Role**

Creating Groups

Step 1: Navigate to Groups

1. Go to **Customers > Sublogin Groups**
2. Click **Add New Group**

Step 2: Group Information

Group Name

```
Branch A
```

Description

```
East Coast Branch
```

Parent Group Select parent group (or "General" for top-level)

Customer Select the customer account

Step 3: Save Group

Click **Save Group**

The group will be assigned a level automatically based on its parent.

Assigning Roles to Sublogins

Method 1: During Sublogin Creation

1. Create new sublogin
2. In "Role" dropdown, select the role
3. In "Group" dropdown, select the group
4. Save sublogin

Method 2: Edit Existing Sublogin

1. Edit sublogin
2. Change "Role" dropdown
3. Change "Group" dropdown
4. Save sublogin

Hierarchical Permissions

Hierarchical permissions use the group structure to determine access.

Example Scenario

```
Customer Account
├── General (Level 0)
│   ├── Branch A (Level 1)
│   │   ├── Sublogin A1 (Manager)
│   │   └── Sublogin A2 (Employee)
│   └── Branch B (Level 1)
│       └── Sublogin B1 (Employee)
```

Sublogin A1 (Manager) with `view_order_same_level`:

- Can view orders from: Sublogin A1, Sublogin A2 (same branch)
- Cannot view orders from: Sublogin B1 (different branch)

Sublogin A1 (Manager) with `view_order_lower_level`:

- Can view orders from: Sublogin A2 (lower in hierarchy)
- Cannot view orders from: Sublogin B1 (different branch)

Sublogin A1 (Manager) with `view_order_all`:

- Can view orders from: All sublogins (A1, A2, B1)

Permission Resolution

The system resolves permissions in this order:

1. **Custom Permission Provider** (if defined via DI)
2. **Context-Aware Check** (collection filtering, entity checks)
3. **Direct Permission Match** (sublogin has permission via role)
4. **Default Permission** (defined in `sublogin_acl.xml`)

Default Permissions

Each permission can have a default behavior:

- `allow` - Allowed if not explicitly denied
- `disallow` - Denied if not explicitly allowed

Custom Permissions

You can add custom permissions via `sublogin_acl.xml`:

```
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <acl>
        <resources>
            <resource id="Magento_Sublogin::all">
                <resource id="MyCompany_MyModule::all" title="Custom
Permissions" sortOrder="100">
                    <resource
                        id="MyCompany_MyModule::my_permission"
                        title="My Custom Permission"
                        sortOrder="10"
                        showInFrontend="true"
                        showInAdmin="false"
                        defaultPermission="allow"
                        priority="200"
                    />
                </resource>
            </resource>
        </resources>
    </acl>
</config>
```

Checking Custom Permissions

```
use MageB2B\SubloginRole\Model\AclService;

class MyClass {
    private $aclService;
    public function __construct(AclService $aclService) {
        $this->aclService = $aclService;
    }
    public function myFunction() {
        if
($this->aclService->isAllowed('MyCompany_MyModule::my_permission')) {
            // Permission granted
        }
    }
}
```

Common Role Examples

Role 1: Full Manager

```
Permissions:
- All catalog permissions
- All checkout permissions
- view_order_all
- view_invoice_all
- edit_all, delete_all
- approve_order_all, decline_order_all
- manage_budget
```

Role 2: Department Purchaser

```
Permissions:
- All catalog permissions
- All checkout permissions
- view_order_same_level
- view_invoice_same_level
- No sublogin management
- No approval rights
```

Role 3: Sales Representative

```
Permissions:
- view_product_list
- view_product_details
- view_product_prices
- add_product_to_cart
- view_cart
- No checkout permission
- view_order_same_level (read-only)
```

Role 4: Order Approver

```
Permissions:
- view_order_all
- view_invoice_all
- approve_order_lower_level
- decline_order_lower_level
- No purchasing permissions
```

Troubleshooting

Permission Not Working

1. Check if role is assigned to sublogin

2. Verify permission is checked in role
3. Clear cache
4. Check group hierarchy for hierarchical permissions
5. Review `var/log/system.log` for permission errors

Hierarchical Permission Issues

1. Verify group structure is correct
2. Check parent-child relationships
3. Ensure sublogins are assigned to correct groups
4. Use `_all` permission for testing

Custom Permission Not Showing

1. Verify `sublogin_acl.xml` syntax
2. Run `setup:upgrade`
3. Clear cache
4. Check `showInFrontend` and `showInAdmin` flags

Next Steps

- Budget Management
- Order Approval
- Multi-Account Management

Sample Data

Sublogin Documentation · /sublogin/addons/sample-data

Sample data modules help you evaluate and demo Sublogin features quickly on a staging/dev system.

Do not install sample data on production.

Sublogin Sample Data

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/sublogin-sample-data:*
php bin/magento module:enable MageB2B_SubloginSampleData
php bin/magento setup:upgrade
php bin/magento cache:flush
```

> Screenshot placeholder: Demo customer with multiple sublogins

What gets installed

This package ships fixtures and installs demo data such as:

- A few **B2B customer groups** and **demo customers** (company-like accounts)
- **Customer addresses** for realistic checkout scenarios
- Multiple **sublogins per customer** (e.g. buyer/manager patterns with different flags)
- Example **payment method restrictions** per sublogin
- Example **shipping method restrictions** per sublogin
- A small set of **products** and **orders** linked to the demo customers/sublogins

Useful CLI commands

Verify installation:

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

Reset (delete) the installed Sublogin sample data:

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

After a reset, reinstall via:

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

> Screenshot placeholder: Admin customer list filtered by `*-demo.com`

Feature-specific sample data (requires add-ons)

There are additional sample data packages that only make sense if the corresponding **Sublogin add-on** is installed.

Install the add-on first, then install its sample data.

Roles & Permissions sample data (requires Roles & Permissions add-on)

Prerequisite:

- Roles & Permissions add-on: `mageb2b/sublogin-role` (`MageB2B_SubloginRole`)

Sample data package (separate add-on):

- `mageb2b/sublogin-role-sample-data` (`MageB2B_SubloginRoleSampleData`)

Typical data created:

- Example roles (e.g. buyer / manager / admin patterns)
- Example permission assignments for common storefront actions

Budget sample data (requires Budget Management add-on)

Prerequisite:

- Budget Management add-on: `mageb2b/sublogin-budget` (`MageB2B_SubloginBudget`)

Sample data package (separate add-on):

- `mageb2b/sublogin-budget-sample-data` (`MageB2B_SubloginBudgetSampleData`)

Typical data created:

- Example budgets and limits per customer/sublogin (e.g. a few budgets per demo customer)
- Example spending counters (depending on module configuration)

Catalog Visibility sample data (requires Catalog Visibility add-on)

Prerequisite:

- Catalog Visibility add-on: `mageb2b/sublogin-catalogvisibility` (`MageB2B_SubloginCatalogVisibility`)

Sample data package (separate add-on):

- `mageb2b/sublogin-catalogvisibility-sample-data` (`MageB2B_SubloginCatalogVisibilitySampleData`)

Typical data created:

- Example visibility rules (allowed/blocked categories/products for different sublogin roles)

Order Approval sample data (requires Order Approval add-on)

Prerequisite:

- Order Approval add-on: `mageb2b/sublogin-orderapproval` (`MageB2B_SubloginOrderApproval`)

Sample data package (separate add-on):

- `mageb2b/sublogin-orderapproval-sample-data` (`MageB2B_SubloginOrderApprovalSampleData`)

Typical data created:

- Example approval workflows (thresholds, approvers, and “who can place orders” scenarios)

Sublogin Report Sample Data (optional)

If you use the Reports add-on, you can also install report sample data.

Important: this sample data package expects that you already have sublogins (e.g. from **Sublogin Sample Data** above). It updates report configuration fields on existing sublogins.

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/sublogin-report-sample-data:*
php bin/magento module:enable MageB2B_SubloginReportSampleData
php bin/magento setup:upgrade
php bin/magento cache:flush
```

> Screenshot placeholder: Reports with sample data populated

What gets installed

- Adds example report configurations to existing sublogins (enabled reports, schedule, export format, recipients).

Useful CLI commands

Verify installation:

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

Reset (allows reinstall):

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

Sublogin Budget

Sublogin Documentation · /sublogin/addons/sublogin-budget

This page is an alias for the Budget Management documentation.

- Full documentation: Budget Management

Installation

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/sublogin-budget:*
php bin/magento module:enable MageB2B_SubloginBudget
php bin/magento setup:upgrade
php bin/magento cache:flush
```

Sublogin Orderapproval

Sublogin Documentation · /sublogin/addons/sublogin-orderapproval

This page is an alias for the Order Approval documentation.

- Full documentation: Order Approval

Installation

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/sublogin-orderapproval:*
php bin/magento module:enable MageB2B_SubloginOrderApproval
php bin/magento setup:upgrade
php bin/magento cache:flush
```

Sublogin Role

Sublogin Documentation · /sublogin/addons/sublogin-role

This page is an alias for the Roles & Permissions documentation.

- Full documentation: Roles & Permissions

Installation

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/sublogin-role:*
php bin/magento module:enable MageB2B_SubloginRole
php bin/magento setup:upgrade
php bin/magento cache:flush
```

Common Issues

Sublogin Documentation · /sublogin/troubleshooting/common-issues

Practical troubleshooting checklist for the Sublogin module and its add-ons.

Sublogin Cannot Log In

Check Account Status

In Admin:

1. Go to **Customers > Sublogins**
2. Open the sublogin
3. Verify:
    - **Active** is enabled
    - **Expire date** is empty or in the future (if used)

If email confirmation is enabled:

- Verify the sublogin completed the confirmation flow (otherwise it stays inactive).

Customer and Sublogin Share the Same Email

Magento’s login and password reset flows expect email addresses to be unique. A customer and a sublogin must not share the same email address.

Symptoms:

- Login behaves unexpectedly
- Password reset fails with messages like “Invalid token”

Fix:

- Change one of the emails so that every customer and every sublogin has a unique email.

“Can Create Sublogins” Missing / Cannot Create Sublogins

Checks:

- The main customer must have the attribute **Can Create Sublogins** enabled.
- If you want it enabled for new customers by default, set:
    - **Stores > Configuration > MageB2B > Sublogin > General settings**
    - **Default Value Can Create Sublogins** = Yes

Impersonation (“Login As”) Not Available

Checks:

- **Stores > Configuration > MageB2B > Sublogin > General settings**
    - **Allow impersonate sublogin account** must be enabled
- If you use the Roles & Permissions add-on, ensure the effective permissions allow impersonation for the acting user.

Payment/Shipping Methods Missing at Checkout

If you restrict payment/shipping per sublogin:

- Enable filters:
  - `sublogin/general/enable_payment_methods_filter`
  - `sublogin/general/enable_shipping_methods_filter`
- Then make sure the sublogin actually has allowed methods assigned (otherwise checkout may show none).

Address Problems (No Addresses / Wrong Addresses in Checkout)

Check the address mode:

- **Stores > Configuration > MageB2B > Sublogin > Address settings**
  - Shared / Restricted / Own

Common pitfalls:

- Restricted mode but no addresses assigned to the sublogin
- “Add new address in checkout” disabled but restricted list is empty
- Default address expectations (customer defaults vs sublogin defaults) don’t match your configuration

Emails Not Sent (Welcome / Confirmation / Reset)

Checks:

- Email templates selected in:
  - **Stores > Configuration > MageB2B > Sublogin > Email settings**
- For welcome/notification behavior verify the sublogin flags (e.g. “Send backend mails”) where applicable.
- If you’re testing locally/staging, confirm SMTP/Mailpit configuration and clear config cache if you changed `.env`.

Debugging

Enable Sublogin Logging

Enable:

- **Stores > Configuration > MageB2B > Sublogin > Debug > Enable log**

Then inspect:

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

Developer Notes (Extending the Frontend Form)

If you need to add/modify fields on the sublogin frontend edit form, the module supports extending the

field definition via a plugin on:

- `MageB2B\\Sublogin\\Block\\Sublogin\\Edit::getFormFields()`

This is useful when you want custom customer-specific fields without forking the module.

Permission Problems

Sublogin Documentation · /sublogin/troubleshooting/permission-problems

Troubleshoot sub-account permission issues.

Common Issues

Can't Access Feature

Symptoms:

- Page shows "Access Denied"
- Feature not visible in menu

Solutions:

1. Check role permissions
2. Verify role assignment
3. Clear customer session
4. Check parent account status

Wrong Permissions Applied

Symptoms:

- User has more/less access than expected

Solutions:

1. Review role configuration
2. Check for multiple role assignments
3. Verify permission inheritance
4. Test with different user

Can't Create Orders

Symptoms:

- Checkout blocked
- Payment methods missing

Solutions:

1. Check order permissions
2. Verify budget remaining
3. Check payment restrictions
4. Confirm shipping address access

Debugging

Check Current Permissions

1. Login as sub-account
2. Go to My Account
3. View "My Permissions"

Admin View

1. Go to Customers > Sublogins
2. Find sub-account
3. Check "Effective Permissions"

Log Analysis

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

Look for:

- Permission checks
- Denied actions
- Role evaluations

Related

- Roles & Permissions
- Configuration

B2B Departments

Sublogin Documentation · /sublogin/use-cases/b2b-departments

Organize sub-accounts by company departments.

Scenario

Large organization with multiple departments needing different permissions.

Department Setup

Department	Role	Permissions
Purchasing	Purchaser	Create orders, manage quotes
Finance	Finance	View invoices, payment history
Operations	Viewer	View orders, track shipments
Management	Approver	Approve orders, full access

Implementation

Create Roles

1. Purchasing Role
    - Create orders: Yes
    - View prices: Yes
    - Approve orders: No
2. Finance Role
    - View invoices: Yes
    - Download statements: Yes
    - Place orders: No
3. Operations Role
    - View orders: Yes
    - Track shipments: Yes
    - Modify orders: No

Assign Users

Each employee gets:

- Sub-account under company
- Department role assigned
- Appropriate permissions

Budget by Department

Department	Monthly Budget
Purchasing	$50,000
Operations	$10,000
Marketing	$5,000

Related

- Multi-Account Management
- Roles & Permissions

Employee Ordering

Sublogin Documentation · /sublogin/use-cases/employee-ordering

Enable employees to order with company account.

Scenario

Company provides employees ability to order supplies or products using company account with controls.

Setup

Account Structure

```
Company Account (HR/Admin)
├── Employee 1
├── Employee 2
├── Employee 3
└── ...
```

Permissions

Permission	Setting
Order Limit	$100/order
Monthly Budget	$500/month
Products	Approved catalog only
Approval	Required above $50

Employee Features

- Browse approved products
- Place orders within limits
- Track own orders
- View remaining budget

Admin Features

- Create employee accounts
- Set individual limits
- Review pending approvals
- Generate reports

Approval Workflow

1. Employee places order
2. If over threshold → Pending

3. Manager receives notification
4. Manager approves/rejects
5. Order processes or cancels

Budget Controls

Control	Example
Per Order	Max $100
Daily	Max $200
Monthly	Max $500
Yearly	Max $5,000

Related

- Budget Management
- Order Approval

Franchise System

Sublogin Documentation · /sublogin/use-cases/franchise-system

Manage franchise locations with sub-accounts.

Scenario

Franchisor managing multiple franchise locations, each with their own ordering needs.

Structure

```
Franchisor (Main Account)
├── Franchise Location 1
│   ├── Manager
│   └── Staff
├── Franchise Location 2
│   ├── Manager
│   └── Staff
└── Franchise Location 3
    ├── Manager
    └── Staff
```

Configuration

Location-Based Access

Each franchise location:

- Own shipping addresses
- Specific product catalog
- Individual budgets
- Separate order history

Permissions

Role	Access
Franchisor	All locations, reports
Location Manager	Own location only
Staff	Order with approval

Pricing

- Franchise pricing tier
- Location-specific promotions
- Volume discounts per location

Reporting

Franchisor can view:

- Orders by location
- Spending by franchise
- Product preferences
- Compliance reports

Related

- Multi-Account Management
- Budget Management