Logix MVC - Accounting API Controllers Documentation

This documentation covers the main accounting API controllers within the Logix.MVC.LogixAPIs.Acc namespace. Each controller manages endpoints for various accounting entities such as accounts, cost centers, banks, cash on hand, financial years, journals, settlements, and more. The APIs are secured with permission checks, and follow a consistent, robust structure for data handling and error reporting.

Common Controller Structure

Most controllers:

- Inherit from `BaseAccApiController` for standardized API routing.
- Use dependency injection for services, helpers, and session/context objects.
- Check user permissions before processing logic.
- Wrap results with a `Result<T>` object for consistency.
- Handle exceptions and return friendly error messages.
- Provide CRUD and search endpoints for their specific entity.

BaseAccApiController

This is a base controller that applies the API version and routing conventions for all accounting controllers.

```
1  [Route($"api/{ApiConfig.ApiVersion}/Acc/[controller]")]
2  [ApiController]
3  public abstract class BaseAccApiController : ControllerBase { }
```

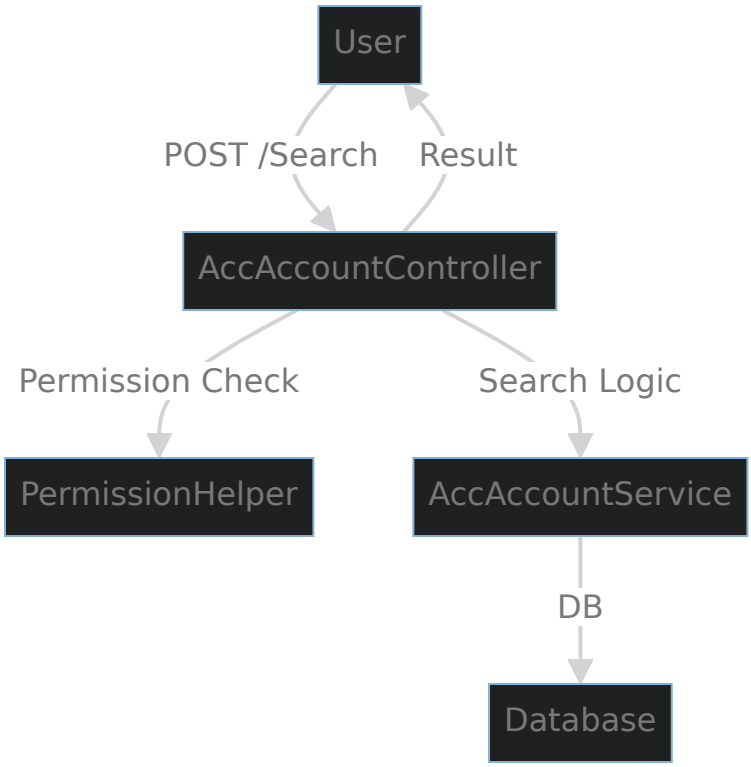
Account Controllers and Endpoints

AccAccountController

Handles operations related to accounting accounts (chart of accounts).

- **Search, Add, Edit, Delete accounts**
- **Get account by ID, Code, or for editing**
- **Excel import/export for bulk account management**
- **Parent account dropdowns and account trees**
- **Report-specific searches**

Main Flows



API: Search Accounts

Search Accounts

Export to Postman

Search for accounting accounts based on filter criteria.

POST `https://api.example.com/api/v1/Acc/AccAccount/Search`

Request body

JSON payload required for this request.

```
{
  "AccAccountCode": "1001",
  "AccGroupId": 2
}
```

Code examples

```
curl -X POST "https://api.example.com/api/v1/Acc/AccAccount/Search" \
  -H "Content-Type: application/json" \
  -d '{
  \"AccAccountCode\": \"1001\",
```

```
  \"AccGroupId\": 2
}'
```

Responses

Success

```
{ "data": [{ "AccAccountCode": "1001", "AccAccountName": "Cash" }] }
```

API: Add Account

Add Account

Export to Postman

Add a new account.

POST `https://api.example.com/api/v1/Acc/AccAccount/Add`

Request body

JSON payload required for this request.

```
{ "AccAccountCode": "2001", "AccAccountName": "Bank Account" }
```

Code examples

```
curl -X POST "https://api.example.com/api/v1/Acc/AccAccount/Add" \
  -H "Content-Type: application/json" \
  -d '{ \"AccAccountCode\": \"2001\", \"AccAccountName\": \"Bank Account\" }'
```

Responses

Account created

```
{ "data": { "AccAccountId": 45 } }
```

API: Delete Account

Delete Account

Export to Postman

Delete an account by its ID.

DELETE `https://api.example.com/api/v1/Acc/AccAccount/Delete`

Path parameters

Id string • path required

Account ID

Code examples

```
curl -X DELETE "https://api.example.com/api/v1/Acc/AccAccount/Delete"
```

Responses

Account deleted

```
{ "data": true }
```

AccAccountsCostcenter

Manages the linking between accounts and cost centers.

- **CRUD operations for account-cost center links**
- **Search, get by account, get by ID**

Example: Link Account to Cost Center

Add Account-CostCenter Link

Export to Postman

Link an account to a cost center.

POST `https://api.example.com/api/v1/Acc/AccAccountsCostcenter/Add`

Request body

JSON payload required for this request.

```
{ "AccAccountId": 25, "CostCenterId": 4 }
```

Code examples

```
curl -X POST "https://api.example.com/api/v1/Acc/AccAccountsCostcenter/Add" \
  -H "Content-Type: application/json" \
  -d '{ \"AccAccountId\": 25, \"CostCenterId\": 4 }'
```

Responses

Link added

```
{ "data": true }
```

AccAccountsLevelController

Controls the structure of the chart of accounts (levels, digits, colors).

- **Get all levels**
- **Update account level digit/format**

Example: Update Account Level Digit

Update Account Level Digit

Export to Postman

Update digit count and color for a specific account level.

POST `https://api.example.com/api/v1/Acc/AccAccountsLevel/Edit`

Request body

JSON payload required for this request.

```
{ "LevelId": 2, "NoOfDigit": 4, "Color": "#FF0000" }
```

Code examples

```
curl -X POST "https://api.example.com/api/v1/Acc/AccAccountsLevel/Edit" \
  -H "Content-Type: application/json" \
  -d '{ \"LevelId\": 2, \"NoOfDigit\": 4, \"Color\": \"#FF0000\" }'
```

Responses

Level updated

```
{ "succeeded": true }
```

AccAccountsTreeController

Builds and returns the hierarchical structure (tree) of accounts.

- **Get the full account tree**

Get Accounts Tree

Export to Postman

Returns the hierarchical tree of accounts.

GET `https://api.example.com/api/v1/Acc/AccAccountsTree/AccountsTree`

Code examples

```
curl -X GET "https://api.example.com/api/v1/Acc/AccAccountsTree/AccountsTree"
```

Responses

Accounts tree

```
{ "data": [ { "AccountId": 1, "AccountName": "Assets", "Children": [ ... ] } ] }
```

□ Bank and Cash Controllers

AccBankController

Manages bank records.

- **CRUD for banks**
- **Search, get by ID, editing**
- **Handles bank account mapping**

AccCashOnHandController

Manages cash on hand (petty cash registers).

- **CRUD for cash boxes**
- **Search, get by ID, editing**

ACCBankUsersApiController and ACCCashonhandUsersApiController

Manages permissions for users on banks/cash registers.

- **Get users with access**
- **Add, edit, remove user permissions on cash/bank**

□ Financial Periods & Years

ACCFinancialYearController

Handles financial year CRUD.

- **Add, edit, delete, get financial years**
- **Permission-based access**

ACCPeriodsController

Handles accounting periods (monthly/quarterly, etc).

- **Add, edit, delete, get periods**
- **Permission-based access**

□ Journals & Balances

AccJournalApiController

Handles generic journal entries for accounting.

- **CRUD for journals**
- **Search, review, get by ID**
- **Handles financial checks (e.g., period, cost center validation)**

AccJournalAutoApiController

Handles automated journal entry processes.

- **Search, get, add, edit, delete for auto journals**
- **Uses permission checks**

AccJournalReverseApiController

Handles reversed journal entries.

- **CRUD for reverse journals**
- **Get by reference/ID**

AccFirstTimeBalancesApiController

Manages first-time (opening) balances for all accounts.

- **CRUD for opening balances**
- **Advanced filtering and financial validation**

AccOpeningBalanceApiController

Similar to `AccFirstTimeBalancesApiController` , but for regular opening balances.

- **CRUD for opening balances**
- **Advanced filtering and financial validation**

□ Expenses & Income

AccExpensesController

Handles expense entries and reports.

- **CRUD for expenses**
- **Search, filter, report expenses**
- **Get expense book serials**
- **Detailed report endpoints**

AccIncomeController

Handles income entries and reports.

- **CRUD for incomes**
- **Search, filter, report income**
- **Get income book serials**
- **Detailed report endpoints**

AccTypesExpensesController

Manages expense types for petty cash.

- **CRUD for petty cash expense types**
- **Filter/search by name**

AccPettyCashController

Handles all petty cash box activities.

- **CRUD for petty cash**
- **Create journal from petty cash**
- **Detailed report endpoints**
- **Detailed information for each petty cash transaction**

□ Cost Centers

ACCCostCenterController & AccCostCenterTreeApiController

Handle cost center management and hierarchical structure.

- **CRUD for cost centers**
- **Get cost center tree**
- **Search cost centers**
- **Detailed validation and deletion checks**

□ Cheque Books

AccChequeBookApiController

Manages cheque books for banks.

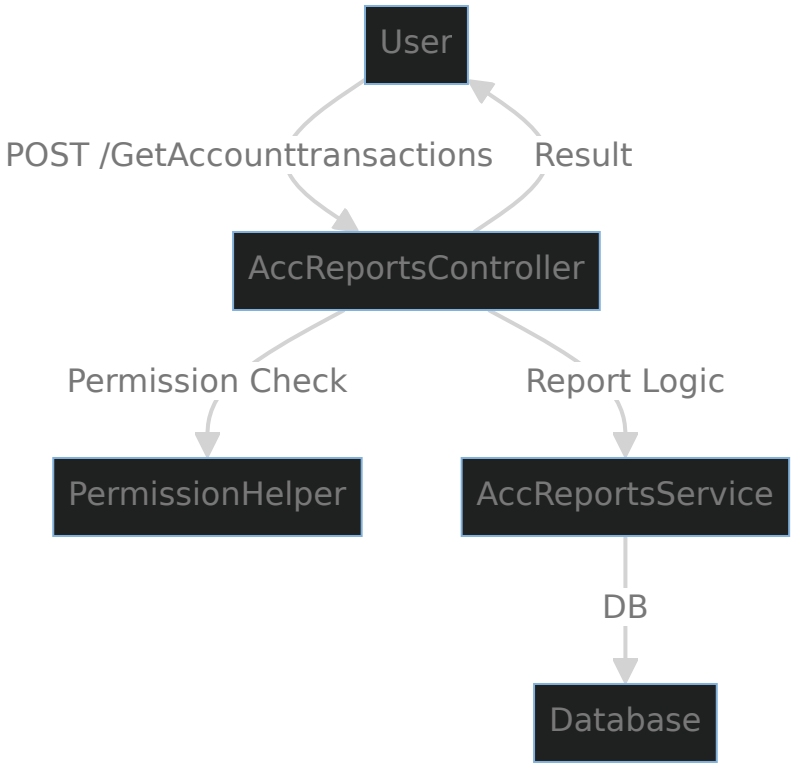
- **CRUD for cheque books**
- **Search/get by bank**
- **Advanced filtering**

□ Reports & Dashboard

AccReportsController

Handles all accounting reports:

- **Account, customer, contractor, supplier, funds, cost center, group, transaction date, trial balance, general ledger, income statement, financial center, profits/losses, cash flows, aged receivables, dashboard data**
- **Complex report filters and data aggregation**



AccDashboardApiController

Returns dashboard statistics and summaries.

- **Get accounting statistics for dashboard**
- **Get predefined accounting reports**

□ Period Closing

ClosingFinancialYearController

Assists with year-end operations.

- **Get all closing entries**
- **Create journal for closing financial year**
- **Get periods by year**

□ Shared APIs

AccSharedController

Provides shared utility endpoints:

- **Get configuration properties**
- **Get currency, customer, journal by code**
- **Validate journal details for accounting rules**
- **Dropdowns for reference types**

- **Get files/documentation for records**

□ VAT & Reference Types

SysVatGroupController

Handles VAT group management.

- **CRUD for VAT groups**
- **Get by ID, filter, search**

AcccountTypeController

Handles account reference types.

- **CRUD for reference types**
- **Get by parent or all**

□ Transfers & Settlement

AccTransferTogeneralledgerController & AccTransferFromgeneralledgerController

Manages bulk and status transfer of journal entries to/from the general ledger.

- **Get all eligible entries for transfer**
- **Bulk update transfer statuses**
- **Complex validation for posting**

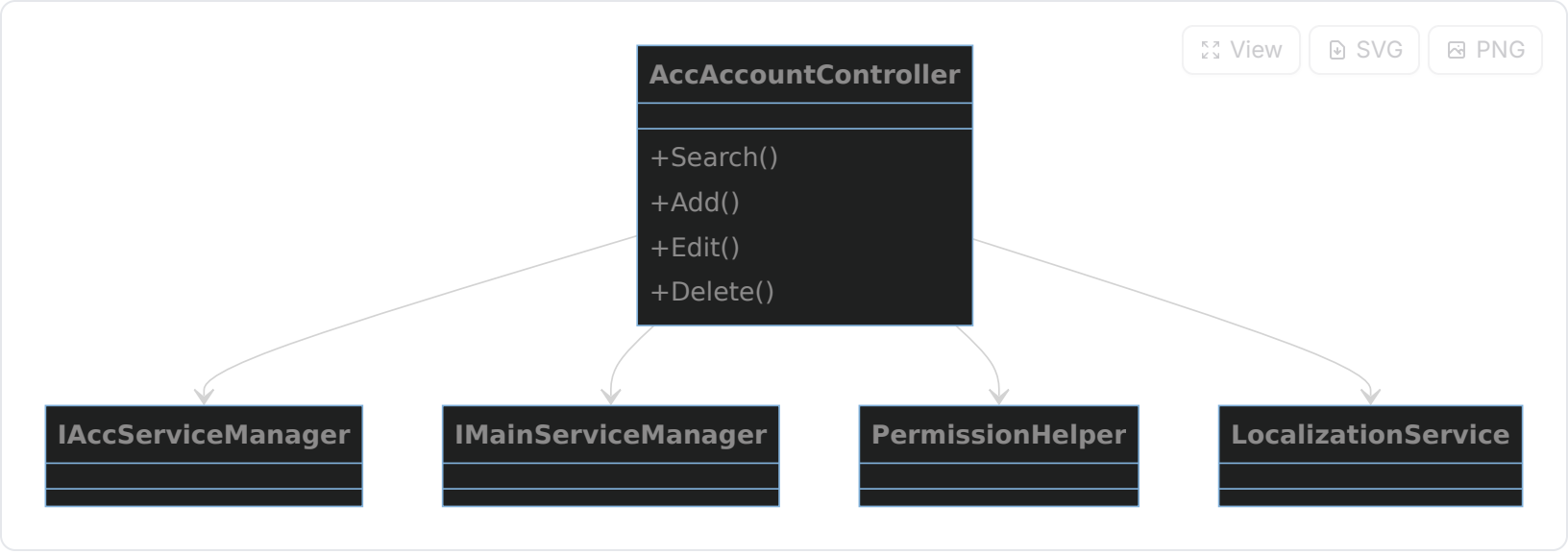
SettlementJournalController & AccSettlementScheduleApiController

Handles settlement schedules (installments, journals, etc).

- **CRUD for settlement schedules**
- **Detailed validation and get operations**
- **Create journals from settlements**

□ Code Structure at a Glance

Key Service Relationships



□ Permission & Validation

All endpoints rely on a robust permission system:

- Each action requires a specific permission code (e.g., Show, Add, Edit, Delete).
- If permissions are insufficient, access is denied with a clear message.
- Most endpoints validate input models and return localized error messages if invalid.

□ Data Flow Example

Adding a New Account

1. User sends a POST to `/api/v1/Acc/AccAccount/Add`.
2. Controller checks for "Add" permission.
3. Validates the model.
4. Calls `AccAccountService.Add()`.
5. Returns success or error result.

□ Summary Table of Main Controllers

Controller	Main Purpose	Core Endpoints
AccAccountController	Chart of accounts CRUD	Search, Add, Edit
AccAccountsCostcenter	Account-cost center mapping	Search, Add, Delete
AccAccountsLevelController	Account level structure	GetAll, Edit
AccAccountsTreeController	Account hierarchy/tree	GetAccountsTree

Controller	Main Purpose	Core Endpoints
AccBankController	Bank management	Search, Add, Edit
AccCashOnHandController	Petty cash/cash management	Search, Add, Edit
ACCFinancialYearController	Financial year management	Search, Add, Edit
ACCPeriodsController	Accounting periods	Search, Add, Edit
AccJournalApiController	General journals	Search, Add, Edit
AccJournalAutoApiController	Automated journal entries	Search, Add, Edit
AccJournalReverseApiController	Reversed journals	Search, Add, Edit
AccFirstTimeBalancesApiController	First-time/opening balances	Search, Add, Edit
AccOpeningBalanceApiController	Opening balances	Search, Add, Edit
AccExpensesController	Expenses management	Search, Add, Edit
AccIncomeController	Income management	Search, Add, Edit
AccPettyCashController	Petty cash transactions	Search, Add, Edit
AccTypesExpensesController	Petty cash/expenses types	Search, Add, Edit
SysVatGroupController	VAT group definitions	Search, Add, Edit
AccSharedController	Shared utilities/validation	Various GET/POST
AccReportsController	Full accounting reports	Multiple POST
AccSettlementScheduleApiController	Settlements/installments	Search, Add, Edit
SettlementJournalController	Journals for settlements	Search, Create
ClosingFinancialYearController	Year-end closing/journals	Search, Create
AccTransferTogeneralledgerController	GL transfers	Search, Update
AccTransferFromgeneralledgerController	GL transfers (reverse)	Search, Update

□ Final Notes

- All controllers use dependency injection, permission checks, and localization.
- Error messages are detailed and user-friendly.
- Each endpoint has corresponding API documentation.
- The system is extensible for new modules or business rules.

For further details, see the interactive API blocks and Mermaid diagrams within each section.