API Integration Guide for Frontend Developers

This document outlines how to interact with the financial reporting microservice API. It covers the available endpoints, required request formats, and expected responses for various operations related to entities, ledgers, accounts, transactions, and financial statements.

Base URL: http://127.0.0.1:8008/report_microservice/api/

All API requests should be made to this base URL, followed by the specific endpoint path.

Endpoints Overview

The API endpoints are categorized by their primary resource:

- **Entities:** Operations related to managing organizations or businesses.
- **Ledgers:** Operations related to accounting ledgers within an entity.
- **Accounts:** Operations related to individual accounts within a ledger.
- **Transactions:** Operations related to recording financial transactions.
- **Financial Statements:** Endpoints to retrieve summarized financial reports.

1. Entity Endpoints

1.1 Create Entity

- **Endpoint:** POST /create-entity/
- **Description:** Creates a new entity in the system.
- **Request Body (JSON):**

```
{
    "name": "string",
    "address_1": "string",
    "address_2": "string",
    "path": "string",
    "depth": integer,
    "admin": integer,
    "city": "string",
    "state": "string",
    "zip_code": "string",
    "country": "string",
    "email": "string (email format)",
    "website": "string (URL format)",
    "phone": "string",
```

```
  "hidden": boolean,
  "accrual_method": boolean,
  "fy_start_month": integer (1-12),
  "last_closing_date": "YYYY-MM-DD",
  "meta": {},
  "managers": []
}
```


Example Request:


```
{
  "name": "My New Business",
  "address_1": "456 Main St",
  "address_2": "Suite 200",
  "path": "my-new-business-path",
  "depth": 0,
  "admin": 1,
  "city": "Anytown",
  "state": "CA",
  "zip_code": "90210",
  "country": "USA",
  "email": "contact@mynewbusiness.com",
  "website": "https://mynewbusiness.com",
  "phone": "+19876543210",
  "hidden": false,
  "accrual_method": true,
  "fy_start_month": 1,
  "last_closing_date": "2024-01-01",
  "meta": {},
  "managers": []
}
```


- **Success Response (201 Created):**


```
{
  "uuid": "string (UUID of the created entity)",
  "name": "string",
  "address_1": "string",
  "address_2": "string",
  "path": "string",
  "depth": 0,
```

"admin": 1,
"city": "string",
"state": "string",
"zip_code": "string",
"country": "string",
"email": "string (email format)",
"website": "string (URL format)",
"phone": "string",
"hidden": boolean,
"accrual_method": boolean,
"fy_start_month": integer (1-12),
"last_closing_date": "YYYY-MM-DD",
"meta": {},
"managers": []
}

- **Error Responses:**
  - 400 Bad Request: If validation fails (e.g., missing required fields, invalid data types).
  - 500 Internal Server Error: For unexpected server issues.

1.2 Get Entity Detail

- **Endpoint:** GET /entities/{entity_uuid}/
- **Description:** Retrieves details for a specific entity.
- **URL Parameters:**
  - entity_uuid: The **UUID** of the entity to retrieve.
- **Success Response (200 OK):**

{
"uuid": "string (UUID)",
"name": "string",
"address_1": "string",
"address_2": "string",
"path": "string",
"depth": integer,
"admin": integer,
"city": "string",
"state": "string",
"zip_code": "string",
"country": "string",

```
    "email": "string",
    "website": "string",
    "phone": "string",
    "hidden": boolean,
    "accrual_method": boolean,
    "fy_start_month": integer,
    "last_closing_date": "YYYY-MM-DD",
    "meta": {},
    "managers": []
}
```


- **Error Responses:**
  - 404 Not Found: If the entity_uuid does not exist.

1.3 List Entities

- **Endpoint:** GET /list-entities/
- **Description:** Retrieves a list of all entities in the system.
- **Success Response (200 OK):**


```
[
  {
    "uuid": "string (UUID)",
    "name": "string",
    "address_1": "string",
    "address_2": "string",
    "path": "string",
    "depth": integer,
    "admin": integer,
    "city": "string",
    "state": "string",
    "zip_code": "string",
    "country": "string",
    "email": "string",
    "website": "string",
    "phone": "string",
    "hidden": boolean,
    "accrual_method": boolean,
    "fy_start_month": integer,
    "last_closing_date": "YYYY-MM-DD",
    "meta": {},
```

```
    "managers": []
  },
  // ... more entities
]
```

- **Error Responses:**
  - 500 Internal Server Error: For unexpected server issues.

2. Ledger Endpoints

2.1 Get Entity Ledgers

- **Endpoint:** GET /entity/{entity_uuid}/ledgers/
- **Description:** Retrieves a list of all ledgers associated with a specific entity.
- **URL Parameters:**
  - entity_uuid: The **UUID** of the entity.
- **Success Response (200 OK):**

```
[
  {
    "uuid": "string (UUID)",
    "ledger_name": "string",
    "entity": "string (entity UUID)",
    "posted": boolean,
    "locked": boolean,
    "hidden": boolean,
    "additional_info": {}
  },
  // ... more ledgers
]
```

- **Error Responses:**
  - 404 Not Found: If the entity_uuid does not exist.

2.2 Create Ledger

- **Endpoint:** POST /entity/{entity_uuid}/create-ledger/
- **Description:** Creates a new ledger for a given entity.
- **URL Parameters:**
  - entity_uuid: The **UUID** of the entity for which to create the ledger.
- **Request Body (JSON):**

```
{
  "ledger_name": "string",
```

```
    "posted": boolean,
    "locked": boolean,
    "hidden": boolean,
    "additional_info": {}
}
```

Note: The entity field is derived from the URL path.
Example Request:

```
{
    "ledger_name": "My Company's Main Ledger",
    "posted": false,
    "locked": false,
    "hidden": false,
    "additional_info": {}
}
```

- **Success Response (201 Created):**

```
{
    "uuid": "string (UUID of the created ledger)",
    "ledger_name": "string",
    "entity": "string (entity UUID)",
    "posted": boolean,
    "locked": boolean,
    "hidden": boolean,
    "additional_info": {}
}
```

- **Error Responses:**
  - 400 Bad Request: If validation fails (e.g., missing required fields, invalid data types).
  - 404 Not Found: If the entity_uuid does not exist.

2.3 Create Chart of Accounts

- **Endpoint:** POST /entity/{entity_uuid}/create-chart-of-accounts/
- **Description:** Creates a default chart of accounts for a specified ledger within an entity.
- **URL Parameters:**
  - entity_uuid: The **UUID** of the entity.
- **Request Body (JSON):**

```
{
  "ledger_name": "string"
}
```

Example Request:

```
{
  "ledger_name": "SMARTINNO_LEDGER_MVP1"
}
```

- **Success Response (201 Created):** (The exact response may vary, but it should indicate successful creation, perhaps with a list of created accounts.)

```
{
  "message": "Chart of accounts created successfully for ledger 'SMARTINNO_LEDGER_MVP1'",
  "accounts_created": [
    {
      "code": "1000",
      "name": "Cash",
      "uuid": "..."
    },
    // ... list of created accounts
  ]
}
```

- **Error Responses:**
  - 400 Bad Request: If the ledger_name is missing or invalid, or if a chart of accounts already exists for that ledger.
  - 404 Not Found: If the entity_uuid does not exist or the ledger_name does not belong to the entity.

3. Account Endpoints

3.1 View Chart of Accounts (Ledger Specific)

- **Endpoint:** GET /entity/{entity_uuid}/ledgers/{ledger_name}/chart-of-accounts/
- **Description:** Retrieves the chart of accounts for a specific ledger within an entity.
- **URL Parameters:**
  - entity_uuid: The **UUID** of the entity.
  - ledger_name: The name of the ledger.
- **Success Response (200 OK):** (The structure might be nested by account type or flat.)

```
[
  {
    "uuid": "string (UUID)",
    "account_name": "string",
    "account_code": "string",
    "account_type": "string (e.g., Asset, Liability, Equity, Revenue, Expense,
COGS)",
    "ledger": "string (ledger UUID)",
    "initial_balance": "decimal",
    "current_balance": "decimal",
    "description": "string",
    "parent_account": "string (UUID, optional)",
    "status": "string",
    "meta": {}
  },
  // ... more accounts
]
```

- **Error Responses:**
  - 404 Not Found: If the entity_uuid or ledger_name does not exist.

3.2 View All Accounts (Global)

- **Endpoint:** GET /all-accounts/
- **Description:** Retrieves a list of all accounts across all entities and ledgers. **(Use with caution for large datasets.)**
- **Success Response (200 OK):**

```
[
  {
    "uuid": "string (UUID)",
    "account_name": "string",
    "account_code": "string",
    "account_type": "string",
    "ledger": "string (ledger UUID)",
    "entity_name": "string (associated entity name)",
    "initial_balance": "decimal",
    "current_balance": "decimal",
    "description": "string",
    "parent_account": "string (UUID, optional)",
    "status": "string",
```

```
    "meta": {}
  },
  // ... more accounts
]
```

- **Error Responses:**
  - 500 Internal Server Error: For unexpected server issues.

3.3 View Accounts (Entity and Ledger Specific)

- **Endpoint:** GET /entity/{entity_uuid}/ledgers/{ledger_name}/accounts/
- **Description:** Retrieves a list of accounts within a specific ledger of an entity.
- **URL Parameters:**
  - entity_uuid: The **UUID** of the entity.
  - ledger_name: The name of the ledger.
- **Success Response (200 OK):**

```
[
  {
    "uuid": "string (UUID)",
    "account_name": "string",
    "account_code": "string",
    "account_type": "string",
    "initial_balance": "decimal",
    "current_balance": "decimal",
    "description": "string",
    "parent_account": "string (UUID, optional)",
    "status": "string",
    "meta": {}
  },
  // ... more accounts
]
```

- **Error Responses:**
  - 404 Not Found: If the entity_uuid or ledger_name does not exist.

4. Transaction Endpoints

4.1 Create Transaction

- **Endpoint:** POST /entity/{entity_uuid}/ledgers/{ledger_name}/create-transaction/
- **Description:** Records a new financial transaction within a specific ledger.
- **URL Parameters:**

- entity_uuid: The **UUID** of the entity.
- ledger_name: The name of the ledger.

- **Request Body (JSON):**

```
{
  "account_uuid": "string (UUID of the account to debit/credit)",
  "amount": decimal,
  "description": "string",
  "tx_type": "string (either 'dr' for debit or 'cr' for credit)",
  "entity_unit_uuid": "string (UUID, typically a unique identifier for the transaction within the entity's context)",
  "corresponding_account_uuid": "string (UUID of the offsetting account)"
}
```

Example Request (Debit Transaction):

```
{
  "account_uuid": "c95f056b-005c-4bf5-8930-1b6c0a1a6552",
  "amount": 100.00,
  "description": "Payment for services",
  "tx_type": "dr",
  "entity_unit_uuid": "a1b2c3d4-e5f6-7890-1234-567890abcdef",
  "corresponding_account_uuid": "b45487e6-926d-4b5e-a671-2a2d6c60c1a0"
}
```

- **Success Response (201 Created):**

```
{
  "uuid": "string (UUID of the created transaction)",
  "account_uuid": "string",
  "amount": "decimal",
  "description": "string",
  "tx_type": "string",
  "entity_unit_uuid": "string",
  "corresponding_account_uuid": "string",
  "timestamp": "YYYY-MM-DDTHH:MM:SS.fZ"
}
```

- **Error Responses:**
  - 400 Bad Request: If validation fails (e.g., missing required fields, invalid tx_type, invalid UUIDs).
  - 404 Not Found: If the entity_uuid, ledger_name, account_uuid, or

corresponding_account_uuid does not exist.

4.2 View Transactions

- **Endpoint:** GET /entity/{entity_uuid}/ledgers/{ledger_name}/transactions/
- **Description:** Retrieves all transactions for a specific ledger.
- **URL Parameters:**
  - entity_uuid: The **UUID** of the entity.
  - ledger_name: The name of the ledger.
- **Success Response (200 OK):**

```
[
  {
    "uuid": "string (UUID)",
    "account_uuid": "string",
    "amount": "decimal",
    "description": "string",
    "tx_type": "string",
    "entity_unit_uuid": "string",
    "corresponding_account_uuid": "string",
    "timestamp": "YYYY-MM-DDTHH:MM:SS.fZ"
  },
  // ... more transactions
]
```

- **Error Responses:**
  - 404 Not Found: If the entity_uuid or ledger_name does not exist.

5. Financial Statement Endpoints

These endpoints provide aggregated financial data.

5.1 View Balance Sheet

- **Endpoint:** GET /entity/{entity_uuid}/ledgers/{ledger_name}/balance-sheet/
- **Description:** Generates and retrieves the balance sheet for a specified ledger.
- **URL Parameters:**
  - entity_uuid: The **UUID** of the entity.
  - ledger_name: The name of the ledger.
- **Success Response (200 OK):**

```
{
  "assets": [
    {
```

```
            "code": "string",
            "name": "string",
            "balance": "decimal",
            "uuid": "string (UUID)"
        },
        // ...
    ],
    "total_assets": "decimal",
    "liabilities": [
        {
            "code": "string",
            "name": "string",
            "balance": "decimal",
            "uuid": "string (UUID)"
        },
        // ...
    ],
    "total_liabilities": "decimal",
    "equity": [
        {
            "code": "string",
            "name": "string",
            "balance": "decimal",
            "uuid": "string (UUID)"
        },
        // ...
    ],
    "total_equity": "decimal",
    "total_liabilities_and_equity": "decimal"
}
```


- **Error Responses:**
  - 404 Not Found: If the entity_uuid or ledger_name does not exist.
  - 500 Internal Server Error: If there's an issue generating the report.

5.2 View Income Statement

- **Endpoint:** GET /entity/{entity_uuid}/ledgers/{ledger_name}/income-statement/
- **Description:** Generates and retrieves the income statement for a specified ledger.

- **URL Parameters:**
  - entity_uuid: The **UUID** of the entity.
  - ledger_name: The name of the ledger.
- **Success Response (200 OK):**

```
{
  "revenues": [
    {
      "code": "string",
      "name": "string",
      "balance": "decimal",
      "uuid": "string (UUID)"
    },
    // ...
  ],
  "total_revenues": "decimal",
  "cogs": [
    {
      "code": "string",
      "name": "string",
      "balance": "decimal",
      "uuid": "string (UUID)"
    },
    // ...
  ],
  "total_cogs": "decimal",
  "gross_profit": "decimal",
  "expenses": [
    {
      "code": "string",
      "name": "string",
      "balance": "decimal",
      "uuid": "string (UUID)"
    },
    // ...
  ],
  "total_expenses": "decimal",
  "net_income": "decimal"
}
```

- **Error Responses:**
  - 404 Not Found: If the entity_uuid or ledger_name does not exist.
  - 500 Internal Server Error: If there's an issue generating the report.

5.3 View Cash Flow Statement

- **Endpoint:** GET /entity/{entity_uuid}/ledgers/{ledger_name}/cash-flow-statement/
- **Description:** Generates and retrieves the cash flow statement for a specified ledger.
- **URL Parameters:**
  - entity_uuid: The **UUID** of the entity.
  - ledger_name: The name of the ledger.
- **Success Response (200 OK):** (The structure of a cash flow statement can be complex; this is a general example.)

```
{
  "cash_from_operating_activities": "decimal",
  "cash_from_investing_activities": "decimal",
  "cash_from_financing_activities": "decimal",
  "net_increase_in_cash": "decimal",
  "beginning_cash_balance": "decimal",
  "ending_cash_balance": "decimal",
  "operating_activities_details": [
    {
      "item": "string",
      "amount": "decimal"
    }
    // ...
  ],
  "investing_activities_details": [
    // ...
  ],
  "financing_activities_details": [
    // ...
  ]
}
```

- **Error Responses:**
  - 404 Not Found: If the entity_uuid or ledger_name does not exist.
  - 500 Internal Server Error: If there's an issue generating the report.

5.4 View Balance for All Accounts (within a ledger)

- **Endpoint:** GET /entity/{entity_uuid}/ledgers/{ledger_name}/balance-for-accounts/
- **Description:** Retrieves the current balance for all accounts within a specific ledger, categorized by type. This endpoint is particularly useful for building financial statements or for a general overview of account balances.
- **URL Parameters:**
  - entity_uuid: The **UUID** of the entity.
  - ledger_name: The name of the ledger.
- **Success Response (200 OK):**

```
{
  "assets": [
    {
      "uuid": "string (UUID)",
      "code": "string",
      "name": "string",
      "balance": "decimal"
    },
    // ...
  ],
  "liabilities": [
    {
      "uuid": "string (UUID)",
      "code": "string",
      "name": "string",
      "balance": "decimal"
    },
    // ...
  ],
  "equity": [
    {
      "uuid": "string (UUID)",
      "code": "string",
      "name": "string",
      "balance": "decimal"
    },
    // ...
  ],
  "revenues": [
```

```
        {
            "uuid": "string (UUID)",
            "code": "string",
            "name": "string",
            "balance": "decimal"
        },
        // ...
    ],
    "cogs": [
        {
            "uuid": "string (UUID)",
            "code": "string",
            "name": "string",
            "balance": "decimal"
        },
        // ...
    ],
    "expenses": [
        {
            "uuid": "string (UUID)",
            "code": "string",
            "name": "string",
            "balance": "decimal"
        },
        // ...
    ]
}
```

- **Error Responses:**
    - 404 Not Found: If the entity_uuid or ledger_name does not exist.

5.5 Fetch Specific Accounts by Code

- **Endpoint:** GET /entity/{entity_uuid}/ledgers/{ledger_name}/balance-for-accounts/
- **Description:** (This isn't a separate endpoint, but a way to filter the results of balance-for-accounts on the frontend). This endpoint returns all accounts with their balances, categorized by type. The frontend should then filter this response based on the desired account codes.
- **URL Parameters:**
    - entity_uuid: The **UUID** of the entity.

    - ledger_name: The name of the ledger.
- **Frontend Implementation (Example Logic):**
    1. Make a GET request to /entity/{entity_uuid}/ledgers/{ledger_name}/balance-for-accounts/.
    2. Receive the JSON response containing assets, liabilities, equity, revenues, cogs, and expenses arrays.
    3. Iterate through all these arrays (or combine them into a single list) to find accounts whose code matches the codes_to_fetch array.

Example Frontend Filtering (JavaScript Pseudocode):

```
async function fetchSpecificAccounts(entityUuid, ledgerName, codesToFetch) {
  const BASE_URL = "http://127.0.0.1:8008/report_microservice/api/";
  const url = `${BASE_URL}entity/${entityUuid}/ledgers/${ledgerName}/balance-for-accounts/`;
  try {
    const response = await fetch(url);
    if (!response.ok) {
      throw new Error(`HTTP error! status: ${response.status}`);
    }
    const data = await response.json();

    const allAccounts = [
      ...(data.assets || []),
      ...(data.liabilities || []),
      ...(data.equity || []),
      ...(data.revenues || []),
      ...(data.cogs || []),
      ...(data.expenses || [])
    ];

    const filteredAccounts = allAccounts.filter(account =>
      codesToFetch.includes(account.code)
    );

    return filteredAccounts;
  } catch (error) {
    console.error("Error fetching specific accounts:", error);
    return null;
  }
}

// Usage example:
const myEntityUuid = 'bd6cf457-d67f-41e6-884c-6c30e8ce4488'; // Replace with an actual UUID
```

```
const myLedgerName = 'SMARTINNO_LEDGER_MVP1'; // Replace with an actual
ledger name
const desiredCodes = ["2040", "2030", "6110"];

fetchSpecificAccounts(myEntityUuid, myLedgerName, desiredCodes)
    .then(accounts => {
        if (accounts) {
            console.log("Fetched specific accounts:", accounts);
        }
    });
```

General Considerations for Frontend

- **Error Handling:** Always implement robust error handling for API calls. Check the response.status_code and parse error messages from the response body.
- **Asynchronous Operations:** All API calls are asynchronous. Use async/await or Promises for managing these operations in your JavaScript code.
- **UUIDs:** Be mindful that many endpoints use UUIDs (Universally Unique Identifiers) in the URL path or request body. Ensure these are correctly formatted strings.
- **Data Types:** Pay attention to the expected data types for request bodies (e.g., decimal for amount, boolean for hidden).
- **CORS:** Ensure your backend is configured to allow Cross-Origin Resource Sharing (CORS) from your frontend's domain, especially during development (http://127.0.0.1:8008 is local, but this will be crucial for deployment).
- **API Versioning:** (Future consideration) As the API evolves, consider implementing API versioning (e.g., /v1/report_microservice/api/) to manage changes without breaking existing frontend integrations.
- **Documentation Updates:** Inform the backend team of any changes or new requirements for API endpoints.