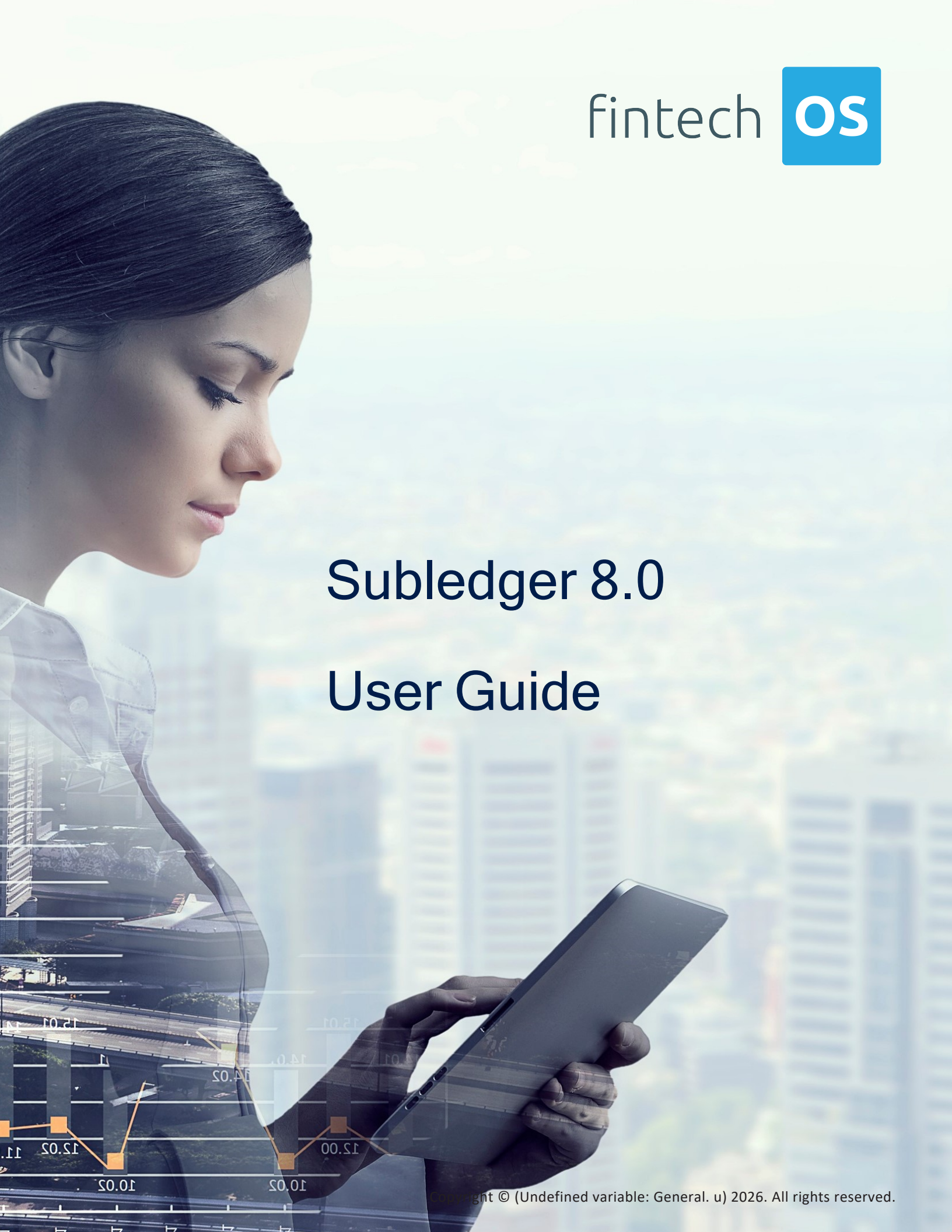


fintech OS

Subledger 8.0

User Guide

TOC

Overview 4
Operational Ledger Settings 5
Accounting Legal Entities 6
Creating Legal Entities 6
Adding an Accounting System to the Legal Entity 7
Accounting Systems 7
Creating Accounting Systems 8
Adding Information into the Accounting Chart 8
Accounting Models Scopes 9
Creating Accounting Scopes 10
Accounting Models For Transactions 10
Creating Transaction Types 11
Operational Ledger 16
Accounting Entries 16
Creating Accounting Entries 16
Transactions Values For Accounting Entries 18
Creating Operation Transactions 18
Subledger SDK 20
1. FTOS_GL_SetValues_TransactionOperation 20
2. getFinalValue 22
3. getAttributeValue 23
4. isAttributeInvariantDate 23
5. contains 24
6. validateFormulaAttribute 25

7. existsAttributeInEntity 26
8. existsAttributeInTableName 26
9. existsAttributeInSourceEntity 27
10. existsOperationTransactionForRecordId 27
11. existsOperationTransactionValue 28

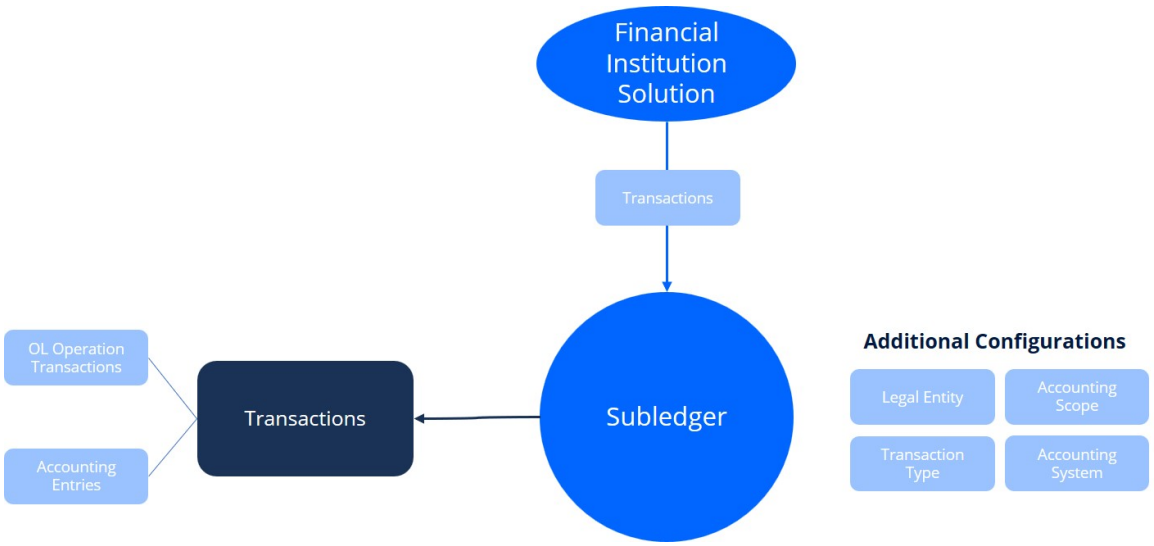
Overview

The FintechOS **Subledger** solution is a module that enables companies to effectively perform financial transactions and to gather the specific accounting information needed for ledger reports and other financial statements.

All the features in Subledger are built using the capabilities of **FintechOS Studio**, and you can access its menus when logged in **FintechOS Portal**.

Subledger comes with a transaction accounting model that reads information from the transaction and displays it in the right fields. It allows you to register transactions, store and organize financial information which is then used in the company's statements, for the creation of a balance or other operations. The double-entry system keeps the data organized for further reports and documents. You can also search for journal entries. Subledger logs, along with a company's financial transactions, specific details that build ledger entries, allowing you to aggregate financial data in a single source of truth for your analysis, reports, or financial statements.

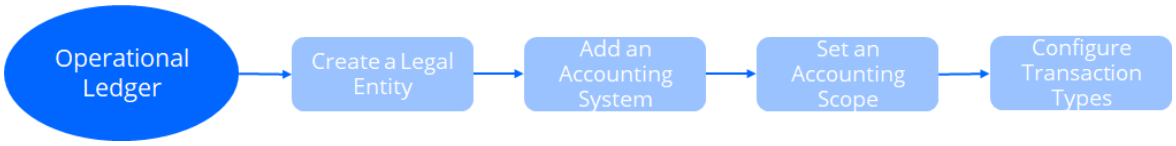
The diagram below exhibits the main features of Subledger, along with a series of configurations used to automate the accounting processes performed by the system. You can use the Subledger along with other financial institution solutions such as **Loan Management**, **Distribution Management**, **Service Insurance**, and so on to access the transaction records generated there.



Operational Ledger Settings

The **Subledger** holds legal entities with their respective accounting systems and their transactions. The inserted transaction information is organized by types into accounts, inside the chosen account chart. This way, it helps keep track of revenues, expenses, owners' equity, assets, liabilities.

In order to set up **Subledger** after installation, you should perform a series of configurations within the **FintechOS Portal**, as presented in the following diagram:



IMPORTANT!
FintechOS offers you an **Utilities** package for banking solutions, which you can use for testing purposes on your pre-production environments. This package helps you test the FintechOS banking solutions during the implementation phase of your project, using sample data. As such, you'd benefit from its content if you are a developer, a tester, a digital consultant, or a partner who works on implementing one of the FintechOS banking solutions.

The configurations are detailed in the pages below:

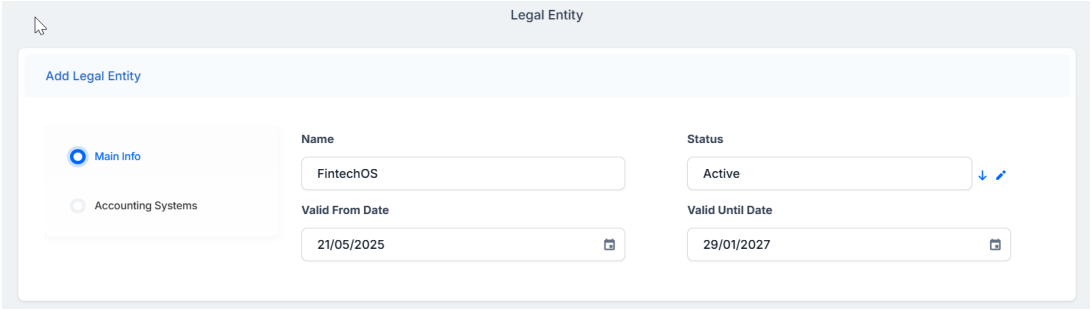
- Legal Entity
- Accounting System
- Accounting Scope
- Transaction Type

Accounting Legal Entities

The legal entity is the company for which you operate the accounting. A legal entity can have multiple accounting systems.

Creating Legal Entities

1. In **FintechOS Portal**, click the main menu > **Ledger** > **Operational Ledger Settings** > **Accounting Legal Entities**[1].
2. Click **Insert** to create a new legal entity. The **Add Legal Entity page** opens.
3. Fill in the following fields:



   - **Name:** Enter the name of the entity.
   - **Status:** Select the status of the entity. Select from the following statuses or create a new one: Draft, Active, or Inactive.
   - **Valid From Date:** Enter the date from which the entity is valid.
   - **Valid Until Date:** Enter the date from which the entity is no longer valid.
4. Click the **Save and reload** button.

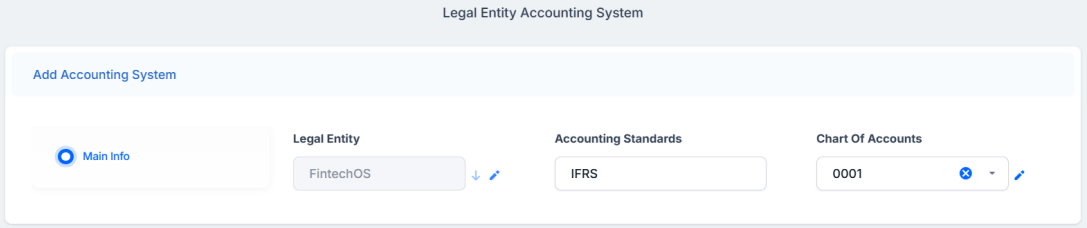
---

[1]An association, corporation, partnership, proprietorship, trust, or individual that has legal standing in the eyes of law. A legal entity has legal capacity to enter into agreements or contracts, assume obligations, incur and pay debts, sue and be sued in its own right, and to be held responsible for its actions.

Adding an Accounting System to the Legal Entity

Store data about the accounting systems used by the legal entity in the **Accounting Systems** section.

1. To add a new legal entity accounting system, click the **Insert** button under the **Accounting Systems** section. The **Add Accounting Systems** page opens.
2. Fill in the following fields:



   - **Legal Entity:** Enter the name of the legal entity.
   - **Accounting Standards:** Enter the accounting standards used by the system.
   - **Chart Of Accounts:** Enter the accounting systems used.

You can add, delete, or export Legal Entity Accounting Systems tables.

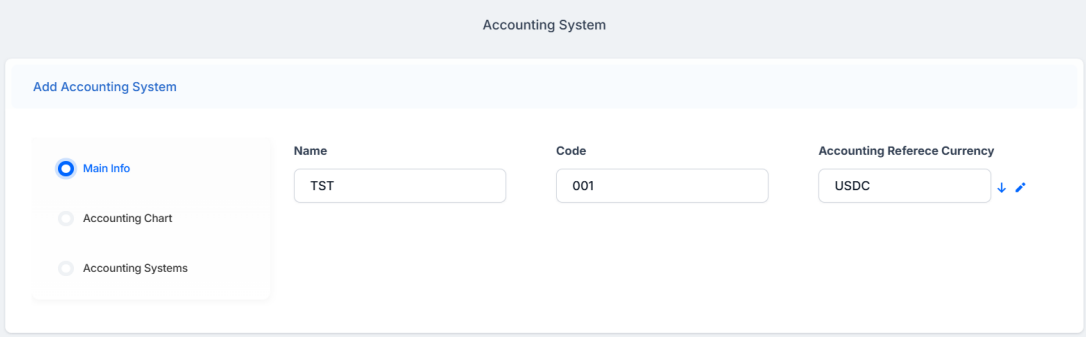
Accounting Systems

An **accounting system**[1] represents the system used for collecting and organizing the financial data. For each legal entity there is a corresponding system. An entity can have more accounting systems.

[1] The system used to manage the income, expenses, and other financial activities of a business.

Creating Accounting Systems

1. In **FintechOS Portal**, click the main menu > **Ledger** > **Operational Ledger Settings** > **Accounting System**.
2. Click the **Insert** button to create an accounting system. The **Add Accounting System page** opens.
3. Fill in the following fields:

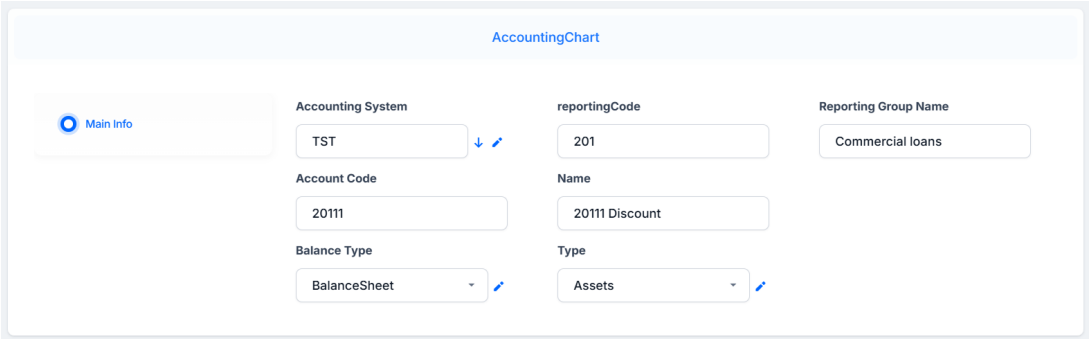


   - **Name:** Enter the official name of the accounting system.
   - **Code:** Enter the code of the accounting system.
   - **Accounting Reference Currency:** It contains every currency possible with its code and symbol. Select the currency for this specific system.
4. Click the **Save and reload** button. The **Accounting Chart** section is displayed.

Adding Information into the Accounting Chart

Store data about the account in the **Accounting Chart** section.

1. To add a new transaction item accounting configuration, click the **Insert** button under the **Accounting Chart** section. The **Accounting Chart** page opens.
2. Fill in the following fields:



- **Accounting System:** Select the accounting system of the **chart**[1] .
- **Account Code:** Enter the code of the account.
- **reportingCode:** Enter the code of the reporting.
- **Type:** Select the account type. The following options are available: **Assets, Expenses, Revenues, Liabilities** or **Others.**
- **Balance Type:** Select the account balance type. The following options are available: BalanceSheet, P&L or OffBalanceSheet.
- **Reporting Group Name:** Enter the name of the reporting group.
- **Name:** Enter the name of the account.

3. Click the **Save and close** button.

You can add, delete, or export accounting charts.

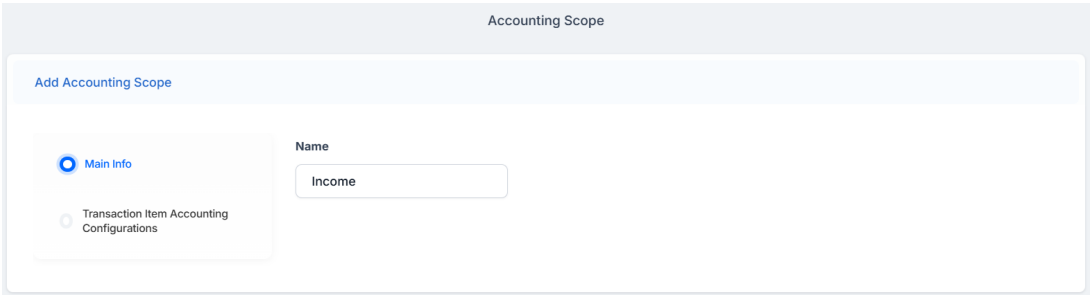
Accounting Models Scopes

You can manage relevant financial information using the **Accounting Scope** menu. This menu introduces the scope for accounting a financial transaction.

[1]An index of all the financial accounts in the general ledger of a company.

Creating Accounting Scopes

1. In **FintechOS Portal**, click the main menu > **Ledger** > **Operational Ledger Settings** > **Accounting Models Scope**.
2. Click the **Insert** button to add a new accounting scope. The **Add Accounting Scope page** opens.
3. Fill in the accounting scope **Name.**



4. Click the **Save and reload** button.

From here, you can export the transactions displayed in the **Transaction Item Accounting Configuration** sections.

> **NOTE**
> View the configuration of each transaction in the **Transaction Item Accounting Configuration** section. For more details on creating such transaction configurations, see the Transaction Item Accounting Configuration chapter.

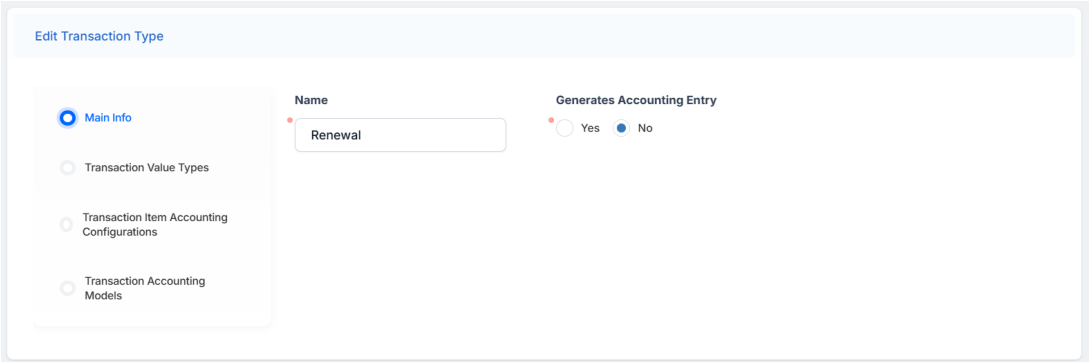
Accounting Models For Transactions

You can enter financial data transactions using the **Accounting Models For Transactions** menu, which contains the accounting model and item configuration. You can delete, or export **Transaction Types** tables.

For using transaction types in conjunction with Loan Management, see Transaction Types Used in Loan Management.

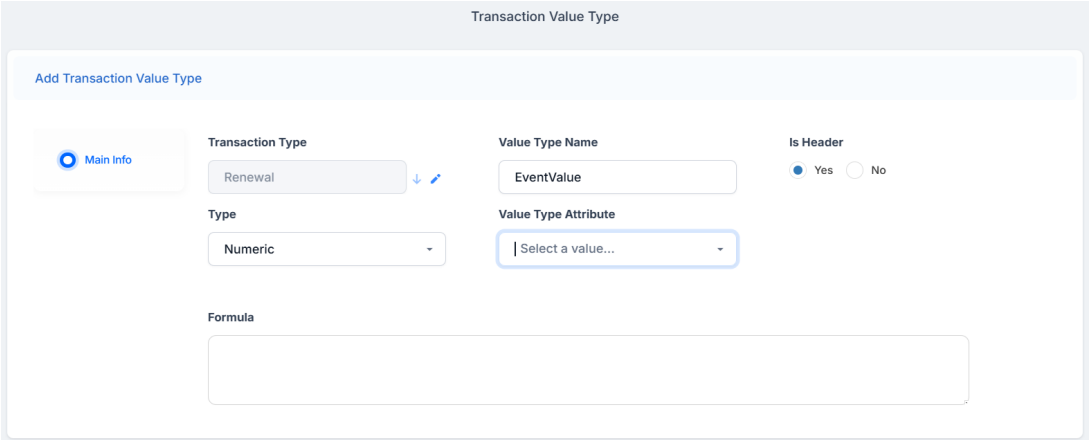
Creating Transaction Types

1. In **FintechOS Portal**, click the main menu > **Ledger** > **Operational Ledger Settings** > **Accounting Models For Transactions**.
2. Double click a transaction type to view and edit it.
3. In the newly displayed **Edit Transaction Type** page's **Main Info**, edit in the following fields:



   - **Name:** Enter the name of the transaction type.
   - **Generates Accounting Entry:** Select whether or not to create an entry in the general ledger. It generates records in the `Accounting Entry` entity.
4. Transaction value types are defined as header items or detail items. Header items are the general details of a transaction (for example date, customer, currency, and so on). The detail items are grouped into numeric or text information.

   You can create and determine the values calculated for each transaction in the **Transaction Value Type** section. The additional data from here is used in the **Transaction Accounting Models** section. In the **Transaction Value Types** section, click **Insert** to add a new type and fill in the following:



- **Value Type Name:** Enter the name of the value type.
- **Transaction Type:** Select the transaction value type.
- **Type:** Select the type of the transaction. The following options are available: Numeric or Text.
- **Value Type Attribute:** Select the value of a specific attribute from the source entity. It is a list of all the attributes defined in the **SourceEntityId** field from the `GL TransactionType` entity.
- **Is Header:** When selected, it defines the header items of the transaction.
- **Formula:** Supports only basic math operations: addition (+), subtraction (-), multiplication (*), and division (/). Input a specific formula based on the Value Type Attribute chosen.

Click the **Save and close** button.

When a transaction value type is marked as a header item, the transaction values are set into the attribute values of the `Operation Transaction` entity. If the **Value Type Name** field is not an attribute of that entity, then an error is displayed.

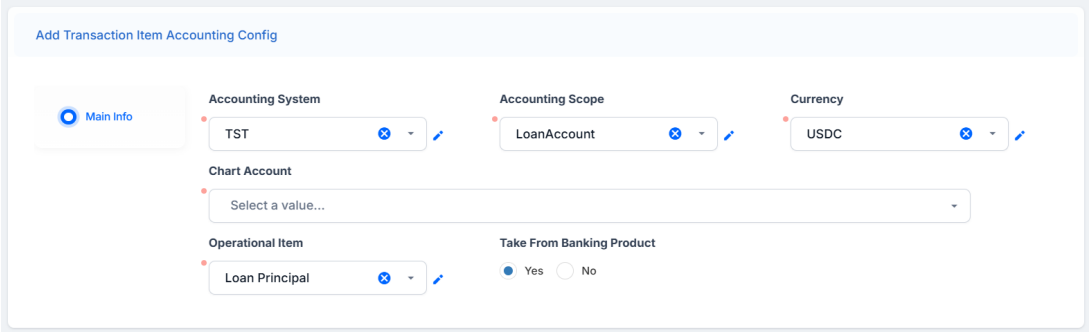
A `json` with default values is sent when using the function for setting the operation transaction values. The `json` has the following form:

```
[
  {
    attributeName: 'DescriptionText',
    value: 'Disburse 1500'
   },
   {
    attributeName: 'ProvisionAmount',
    value: '15.00'
   }
]
```

The `json` checks if there is any default value for the `attributeName`, from the **Value Type Name** field. If no values are returned, the **Formula** field is checked. When neither field returned any values, the source entity of the attribute from the **Value Type Attribute** field is checked.

5. You can define an account from the Accounting Chart in the **Transaction Item Accounting Configuration** section, which holds the configuration of each transaction. The section represents the listing of the names of the accounts for the company inserted in the **Legal Entity** menu.

   To add a new transaction item accounting configuration, click the **Insert** button in the **Transaction Item Accounting Configuration** tab. The **Add Transaction Item Accounting Config** page opens.



Fill in the following fields:

- **Accounting System:** Select the accounting system.
- **Accounting Scope:** Select the accounting scope.
- **Chart Account:** Select the accounting chart.

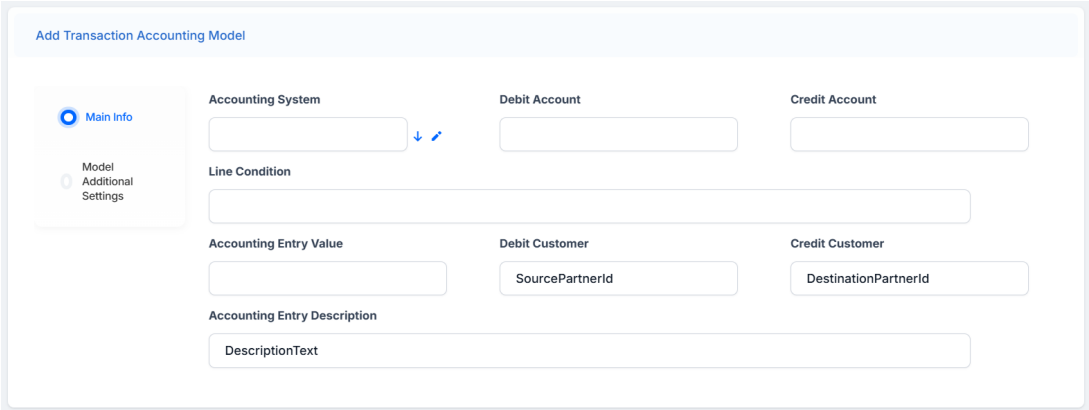
- **Operational Item:** Select the item of operations.
- **Currency:** Select the currency of the accounting entry line.
- **Take From Product:** If you select Yes, then the configurations for each transaction are inherited from the banking product level.

Click the **Save and close** button.

6. The **Transaction Accounting Models** tab holds the accounting models, all the rules used in order to generate accounting entries for each transaction.

   The details from the **Debit Account Rule** and the **Credit Account Rule** are defined by the information from the **Transaction Item Accounting Configuration** section. All other details are defined by the information from the **Transaction Value Type** section.

   To add a new transaction accounting model, click the **Insert** button under the **Transaction Accounting Models** section. The **Add Transaction Accounting Model** page opens.



Fill in the following fields:

- **Accounting System:** Select the accounting system.
- **Debit Account:** The accounting entry value of the debit account.
- **Credit Account:** Enter the credit account of the accounting entry line. It is auto-filled.
- **Line Condition:** Enter the condition applied in order to post the accounting entry line.

- **Accounting Entry Value Rule:** Enter the posted accounting entry value.
- **Debit Customer:** The rule to save the partner transaction in the debit-credit relationship.
- **Credit Customer Rule:** It is auto-filled by the destination partner ID.
- **Accounting Entry Description:** The description of the generated accounting entry. It is auto-filled.

Click the **Save and close** button.

7. In the **Model Additional Settings**, add the following:
   - **EntityId:** Enter the internal status of the record.
   - **Item:** The transaction item of the accounting entry line. It is auto-filled.
   - **Currency:** The accounting entry line currency. It is auto-filled.
   - **Transaction Id:** Enter the related contract ID of the transaction.
   - **Transaction Detail:** Enter the rule to identify and post the ID of the operational transaction detail.
   - **Transaction Value:** Select the value type of the transaction. It is defined in the **Transaction Value Type** section.

Operational Ledger

View and manage transactions by customizing transaction values in the **Subledger** menus.

The **Subledger** captures ledger details for each financial transaction in order to create ledger records according to the accounting system implemented. It also allows customers to feed the same transaction data into different accounting systems and make financial reports. For more details see the pages below:

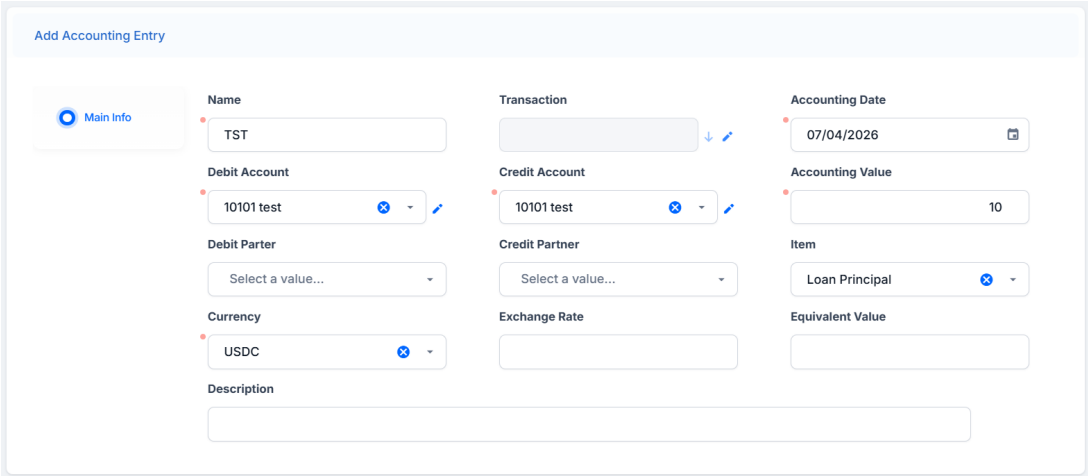
- Operation Transaction
- Accounting Entry

Accounting Entries

Manage the accounting entries generated from the operation transaction and the operation transaction value records in the **Accounting Entries** menu.

Creating Accounting Entries

1. In **FintechOS Portal**, click the main menu > **Ledger** > **Operational Ledger** > **Accounting Entries**.
2. Click the **Insert** button to create an accounting entry.
3. Fill in the following fields:



- **Name**: Enter the name of the entry.
- **Accounting Date:** Enter the date of the accounting entry line.
- **Debit Account:** Select the debit account.
- **Credit Account:** Select the credit account.
- **Accounting Value:** Enter the value of the entry.
- **Debit Partner:** Select the debit partner.
- **Credit Partner:** Select the credit partner.
- **Item:** Select the accounting registration item.
- **Currency:** Select the currency of the accounting entry line.
- **Exchange Rate:** Enter the exchange rate of the accounting entry line.
- **Equivalent Value:** Enter the value after the exchange rate has been applied.
- **Description:** Enter the description of the accounting entry.

4. Click the **Save and close** button.

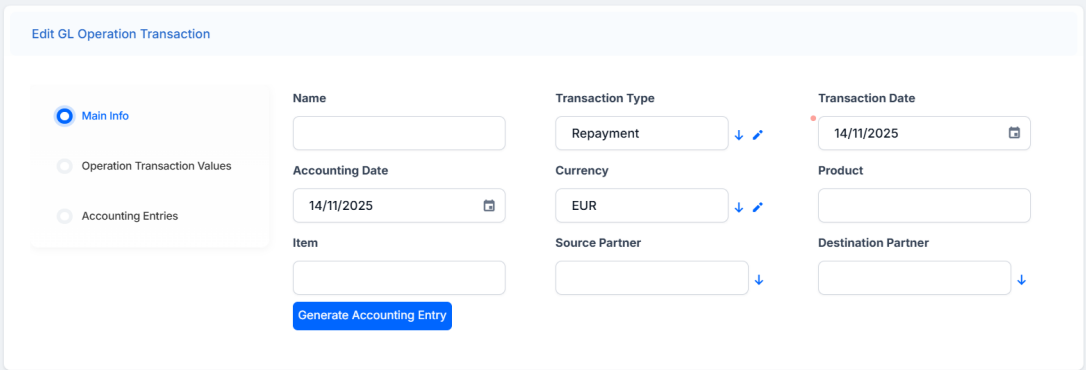
You can add, delete, or export accounting entries.

Transactions Values For Accounting Entries

If you want to search for existing transactions or create new ones, use the **Transactions Values For Accounting Entries** menu, which holds the header items of a transaction. The `Operation Transaction` entity stores all the transaction data from the ledger. The transaction value types represent the details saved into the **Operation Transaction Value** section.

Creating Operation Transactions

1. In **FintechOS Portal**, click the main menu > **Ledger** > **Operational Ledger** > **Transactions Values For Accounting Entries**.
2. Double click an item to open the operation transaction.
3. In the **Main Info** tab, fill in the following fields:

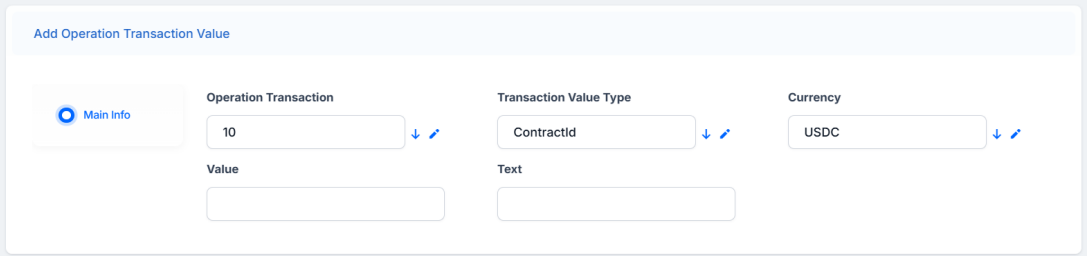


- **Name**: Enter the name of the transaction.
- **Transaction Type:** Select the type of the transaction.
- **Transaction Date:** Select the transaction date.
- **Accounting Date:** Enter the accounting date.

- **Currency:** Select the currency of the accounting entry line.
- **Product:** Enter the product of the transaction.
- **Item:** Enter the transaction item.
- **Source Partner:** Select the source partner.
- **Destination Partner:** Select the destination partner.

  To automatically generate accounting entries, click the **Generate Accounting Entries** button. The accounting entries generated are shown in the **Accounting Entries** section.

4. You can customize the details of each transaction in the **Operation Transaction Values** section. Store the detailed items of a transaction in the **Operation Transaction Values** section. To add a new operation transaction value, click the **Insert** button. Fill in the following fields:



- **Operation Transaction:** Select the operation transaction ID.
- **Transaction Value Type**: Select the value type of the transaction.
- **Currency:** Enter the currency of the accounting entry line.
- **Value:** Enter the value of the transaction.
- **Text:** Enter the description of the transaction.

5. The **Accounting Entries** tab holds the accounting entries of an operation transaction value. From here, you can add, delete, or export accounting entries. For more information on creating accounting entries, see Accounting Entry.

Subledger SDK

The FTOS_GL_OperationTRansactionHelper library server automation script and the FTOS_GL_GetAttributesForTransactionValueType server automation script list and endpoint have been implemented to help generate accounting entries in the Subledger.

The functions available for the implemented library and script are presented below. These functions allow Subledger to convert the transaction value type details into another entity that contains the actual transaction values. Then, based on certain rules, the generated accounting entries are inputted on accounting models. For more information on accounting models, see Transaction Accounting Models.

FTOS_GL_OperationTransactionHelper Library

1. FTOS_GL_SetValues_TransactionOperation

Description:

Based on the transactionTypeName function, it retrieves all the information needed from the FTOS_GL_TransactionType and the FTOS_GL_TransactionValueType entities.

Iterates through the transaction value types list and creates an object with the needed properties for each of the below entities. If a line for the same record ID is found, it updates it, if not, a new one is inserted.

- FTOS_GL_OperationTransaction
- FTOS_GL_OperationTransactionValue

For both objects, the value is retrieved by calling the getFinalValue function.

Input:

- **recordId** - the record ID for which the transaction operation is logged
- **transactionTypeName** - the transaction type (for example Loan Contract, Repayment, Disbursement, Interest Capitalization)
- **defaultValuesJson** - a stringified `json` containing a list of objects with the following form:

```
[
  {
    attributeName: 'DescriptionText',
    value: 'Disburse 1500'
   },
   {
    attributeName: 'ProvisionAmount',
    value: '15.00'
   }
]
```

Output:

No values re returned. The outcome of this function creates the `FTOS_GL_OperationTransaction` and the `FTOS_GL_OperationTransactionValue` entities.

CallExample:

```
var operationTransactionHelperLibrary = importLibrary('FTOS_
GL_OperationTransactionHelper');
var eventId = '64f1c182-d329-4d94-892e-c3cf4556d186';
var defaultJsonValues= [
  {
    attributeName: 'DescriptionText',
    value: 'Disburse 1500'
   },
   {
    attributeName: 'ProvisionAmount',
    value: '15.00'
   }
];
operationTransactionHelperLibrary.FTOS_GL_SetValues_
TransactionOperation(eventId, 'Disbursement', JSON.stringify
(defaultJsonValues));
```

2. getFinalValue

Description:

The `json` checks if there is any default value for the `attributeName`, from the **Value Type Name field**. If no values are returned, the **Formula** field is checked. When neither field returned any values, the source entity of the attribute from the **Value Type Attribute** field is checked.

Input:

- **defaultValues** - `json` object containing a list of objects with the following form:

```
{
    attributeName: 'DescriptionText',
    value: 'Disburse 1500'
  },
  {
    attributeName: 'ProvisionAmount',
    value: '15.00'
  }
]
```

- **recordId** - the record ID of the retrieved value
- **entityId** - the ID of the entity for which the value is retrieved (source entity ID)
- **valueTypeName** - the value type name defined in the Transaction Value Type section, used to check for default values
- **valueTypeAttributeName** - the name of the attribute from the source entity
- **valueTypeFormula** - the value type formula, used for getting the value

Output: the value of a specific transaction value type

Call example:

```
{
    attributeName: 'DescriptionText',
```

```
    value: 'Disburse 1500'
   },
   {
    attributeName: 'ProvisionAmount',
    value: '15.00'
   }
]
```

3. getAttributeValue

Description: a general fetch is made in order to retrieve the attribute value.

Input:

- **recordId** - the record ID of the retrieved value
- **entityId** - the ID of the entity for which the value is retrieved
- **attributeName** - the name of the attribute from the entity for which the value is retrieved

Output: the attribute value

CallExample:

```
var defaultValue = ;
var recordId = '631def4a-c0ab-4c3f-9a23-f5ffc4488f13';
var sourceEntityId= '631def4a-c0ab-4c3f-9a23-f5ffc4488f13';
var valueTypeName = 'ProvisionAmount';
var valueTypeAttributeName = '';
var formula = 'availableValue * 2';
var valueToBeSaved = getFinalValue(defaultValue, recordId,
sourceEntityId, valueTypeName , valueTypeAttributeName,
formula );
```

4. isAttributeInvariantDate

Description: it checks if the attribute value is an invariant date or not.

Input:

- **attributeId** - the attribute ID

Output: true or false.

Call example:

```
var recordId = '631def4a-c0ab-4c3f-9a23-f5ffc4488f13';
var returnedValue = getAttributeValue(recordId, 'FTOS_CB_
Contract', 'availableAmount')
```

5. contains

Description: iterates through an array and if the condition is satisfied, the object is returned, otherwise no results are returned.

Input:

- **arr** - an array of objects
- **key** - the parameter name of the object
- **val** - the value on which the condition is made

Output: if the array contains an object with that specific value for that key, it returns the object, otherwise no output is returned.

Call example:

```
var defaultJsonValues= [
  {
    attributeName: 'DescriptionText',
    value: 'Disburse 1500'
   },
   {
    attributeName: 'ProvisionAmount',
    value: '15.00'
   }
];
```

```
var value = contains(defaultJsonValues, 'attributeName',
'ProvisionAmount');
//value will be
//{
//    attributeName: 'ProvisionAmount',
//    value: '15.00'
//}
```

6. validateFormulaAttribute

Description: using a regex pattern `/[A-Za-z]+/gm`, it matches all the attribute names and it iterates through all the findings to check if it's an attribute of the source entity from the transaction type ID using the `existsAttributeInSourceEntity` function.

Input:

- **formula** - a field which validates basic math operation formulas with the attributes of a source entity
- **transactionTypeId** - the ID of the transaction type; the formula is validated against the source entity ID of the transaction type

Output: if the attribute inputted in the formula field is not part of the entity, then following error is displayed:

`Name should be an attribute name from header table FTOS_GL_ OperationTransaction.`

Call example:

```
var formula = 'availableValue * 2'
var transactionTypeId = '631def4a-c0ab-4c3f-9a23-
f5ffc4488f13';
validateFormulaAttribute(formula, transactionTypeId);
```

7. existsAttributeInEntity

Description: a fluent query is done on the attribute table conditioning on the entity ID and the attribute name.

Input:

- **attributeName** - attribute name that is checked
- **entityId** - entity ID on which the attribute is checked

Output: true or false.

Call example:

```
var blnExists = existsAttributeInEntity('currencyId',
'631def4a-c0ab-4c3f-9a23-f5ffc4488f13');
```

8. existsAttributeInTableName

Description: a fluent query is done on the attribute and the entity table conditioning on the entity table name and the attribute name.

Input:

- **attributeName** - attribute name that is checked
- **entityTableName** - table name of the entity on which the attribute is checked

Output: true or false.

Call example:

```
var blnExists = existsAttributeInTableName('currencyId',
'FTOS_CB_Contract');
```

9. existsAttributeInSourceEntity

Description:based on the `transactionTypeId` function, the source entity ID is retrieved from the `FTOS_GL_TransactionType`. Then, a fluent query is done on the attribute and the entity tables, conditioning on the source entity ID and the attribute name.

Input:

- **attributeToCheck** - attribute name that is checked
- **transactionTypeId** - the ID of the transaction type, used to retrieve the source entity ID

Output: true or false

Call example:

```
var blnExists = existsAttributeInSourceEntity('currencyId',
'631def4a-c0ab-4c3f-9a23-f5ffc4488f13');
```

10. existsOperationTransactionForRecordId

Description: a fluent query that searches in the `FTOS_GL_OperationTransaction` table for record IDs that exist in the **EntityID** column lines.

Input:

- **recordId** - the record ID that checks if an operation transaction exists for it

Output: the operation ID is returned if found. If not, no results are displayed.

Call example:

```
var blnExists = existsAttributeInSourceEntity('currencyId',
'631def4a-c0ab-4c3f-9a23-f5ffc4488f13');
```

11. existsOperationTransactionValue

Description a fluent query, that searches in the `FTOS_GL_OperationTransactionValue` table for a record with a specific operation transaction ID and transaction value type ID.

Input:

- **operationTransactionId** - operation transaction ID
- **transactionValueType** - transaction value type ID

Output:the operation transaction value ID is returned. If not, no results are displayed.

Call example:

```
var result = existsOperationTransactionForRecordId
('631def4a-c0ab-4c3f-9a23-f5ffc4488f13');
```

FTOS_GL GetAttributesForTransactionValueType Server Script and Endpoint

Description: It retrieves a list of attributes conditioned by the source entity ID from the transaction type. If the attribute value type is numeric, only numeric or whole number type attributes ore displayed. If not, all attributes are retrieved.

Input:

- **attributeValueTypeId** - the attribute value type ID which is either numeric or text
- **transactionTypeId** - the transaction type ID that is used for retrieving the source entity

Output: a list of attributes `{attributeid: '', attributeType: '', displayName: '', name: ''}`

Call example:

```
var result = ebs.callActionByNameAsync("FTOS_GL_
GetAttributesForTransactionValueType",
{attributeValueTypeId: '', transactionTypeId:''})
                .then(function(res){
                });
```