

fintech  OS

# Operational Ledger 3.3

## User Guide

# TOC

Overview .....	4
Installing Operational Ledger .....	5
Dependencies .....	5
Pre-Installation Checklist .....	6
Installation Steps .....	6
Dependencies .....	10
Installation Steps .....	11
Configurations .....	14
Legal Entity .....	14
Creating Legal Entities .....	15
Accounting System .....	16
Creating Accounting Systems .....	16
Accounting Scope .....	18
Creating Accounting Scopes .....	18
Transaction Type .....	19
Creating Transaction Types .....	19
Transactions .....	27
Operation Transactions .....	27
Creating Operation Transactions .....	27
Accounting Entry .....	30
Creating Accounting Entries .....	30
Operational Ledger SDK .....	32
1. FTOS_GL_SetValues_TransactionOperation .....	32

2. getFinalValue .....	34
3. getAttributeValue .....	35
4. isAttributeInvariantDate .....	35
5. contains .....	36
6. validateFormulaAttribute .....	37
7. existsAttributeInEntity .....	38
8. existsAttributeInTableName .....	38
9. existsAttributeInSourceEntity .....	39
10. existsOperationTransactionForRecordId .....	39
11. existsOperationTransactionValue .....	40
<b>Glossary .....</b>	<b>42</b>

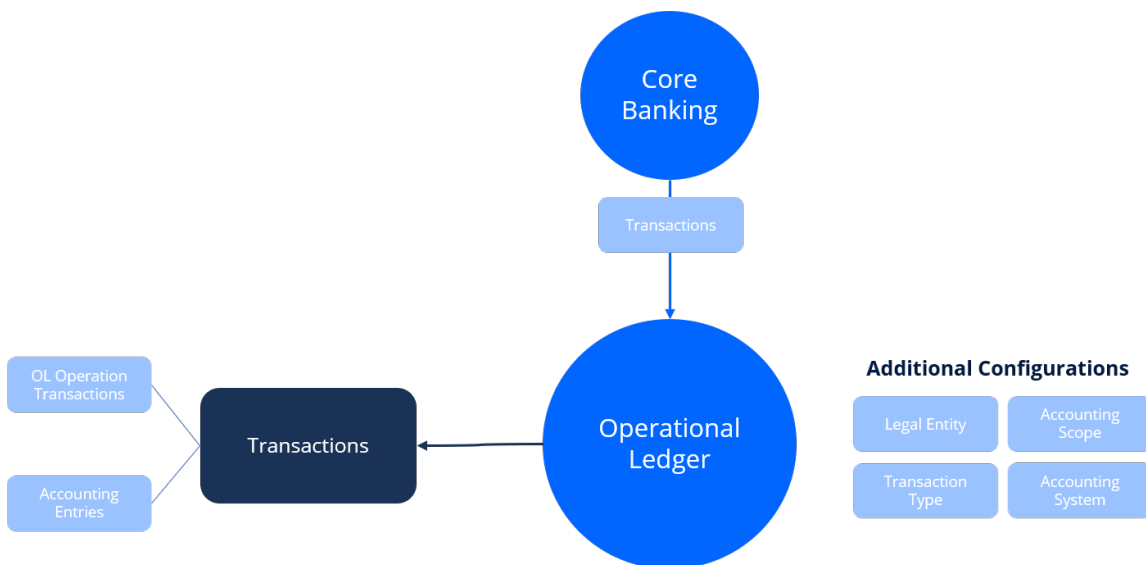
# Overview

The **FintechOS Operational Ledger** 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 Operational Ledger are built using the capabilities of **Innovation Studio**, and you can access its menus when logged in **FintechOS Portal**.

**Operational Ledger** 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. Operational Ledger 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 Operational Ledger, along with a series of configurations used to automate the accounting processes performed by the system. You can use the Operational Ledger along with **Core Banking** to access the transaction records generated there.



# Installing Operational Ledger

Operational Ledger comes in two distinct installation packages, suitable for different usage:

- **Operational Ledger**, a stand-alone product that can be installed to be used with other Lighthouse or Northstar products.
- **Operational Ledger Add-On**, an add-on to be installed on top of the **Core Banking** & Operational Ledger packages with the same version.

The installation for both packages is a process of running two `install_SysPack.bat` files on your environment.

## NOTE

You must run the scripts on the machine where Innovation Studio is installed. Make sure you have access rights to Studio's database.

This page contains the installation steps specific to the Operational Ledger packages available for the **v3.3** release:

## Installing Operational Ledger v3.3

Follow the steps described below to perform an automatic installation of the Operational Ledger v3.3.

## Dependencies

In order to install Operational Ledger v3.3, first you need to install the following:

- **Innovation Studio** minimum version **v22.1.1**

- **SySDigitalSolutionPackages** minimum version **v22.1.1000**
- **Banking Product Factory** (Project or Standard Pack) **v3.3**.

## Pre-Installation Checklist

The SysPack has a unique constraint on the FTOS\_CB\_SystemParameter entity.

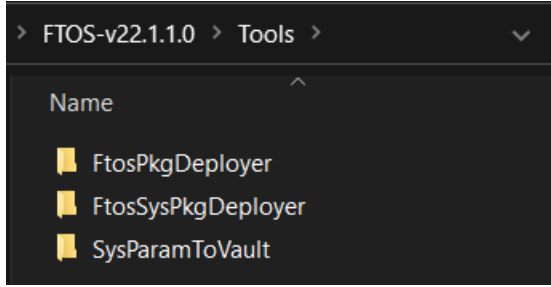
If you have already moved data using the **Configuration Data Deployment Package** menu, then you probably have already configured some unique constraints.

Before running the script, make sure you:

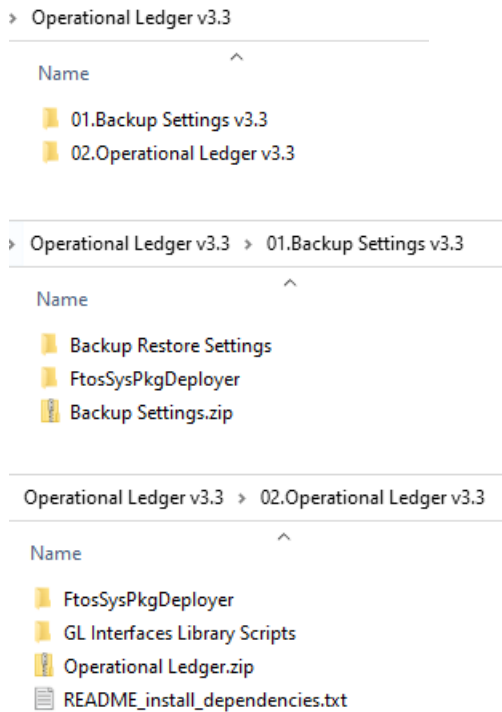
1. Disable the constraints that you have created on your environment, allowing the system to create the new one after Operational Ledger 3.3 is imported.
2. Use the new **Configuration Data Definitions** imported with the Operational Ledger 3.3 file when you export the data.

## Installation Steps

1. Unzip your **Operational Ledger v3.3** archive file.
2. Locate the *FtosSysPkgDeployer* folder in the FintechOS installation kit (the path is *<unzipped\_install\_archive>\Tools\FtosSysPkgDeployer*). You need it to install the Operational Ledger v3.3.



3. Select and copy the *FtosSysPkgDeployer* folder.
4. Navigate to the location where you have unzipped the **Operational Ledger v3.3** (let's call this location *<pkg\_deployer\_dir>*), then paste the *FtosSysPkgDeployer* folder there, within each and every zip file that comes with the package.

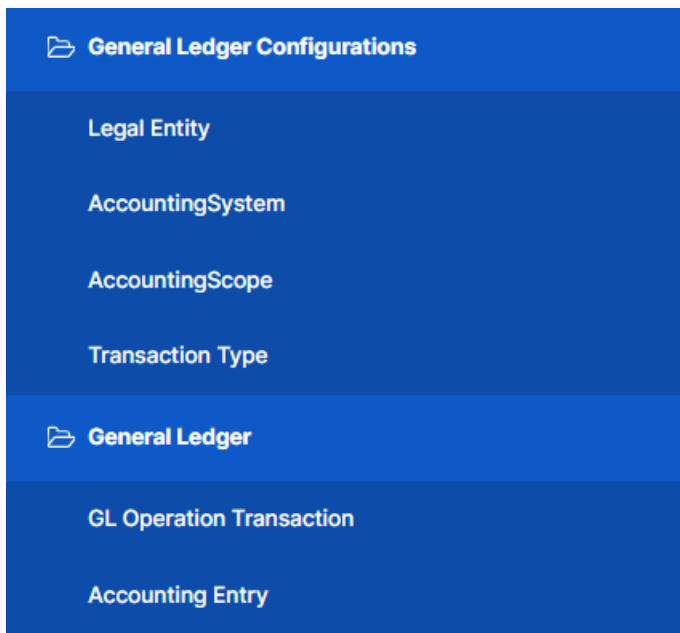


**IMPORTANT!**

Each zip file within the **Operational Ledger v3.3.zip** must be extracted and installed separately, in the given order!

5. Create a new or edit a previous `install_Syspack.bat` file. Replace the parameters described in the "[install\\_SysPack.bat Parameters Explanation](#)" on the next page section with your own values. Save and close the file.
6. Right-click `install_SysPack.bat` » **Run**, for each package.

The script start running in your Windows console. Wait for it to finish. If the parameter values were correct, the FintechOS Portal has two new menus, visible after a refresh, the **General Ledger Configurations** and the **General Ledger** menus:



The `install_SysPack.bat` file allows you to import both the data model and the data.



install\_SysPack.bat syntax for **Data Model** import

```
FtosSysPkgDeployer.exe -i -s "<StudioLink>" -u
<AdminStudioUser> -p <user_password> -z <DataBaseServer> -v
<DB_user> -k <DB_user_password> -d "<TheNameOfTheDataBase>"
-r "<syspack_path>\01 DeploymentPackages\*.zip"
```

install\_SysPack.bat syntax for **Data** import

```
FtosSysPkgDeployer.exe -g -s "<StudioLink>" -u
<AdminStudioUser> -p <user_password> -r "<syspack_path>\02
ConfigurationDataDeploymentPackages\*.zip"
```

**NOTE**

The syntax presented here is for information purposes only. Please run the actual `install_SysPack.bat` file.

## install\_SysPack.bat Parameters Explanation

- `<StudioLink>` - The web URL of the Innovation Studio installation, for example `http://localhost/ftos_studio`.
- `<AdminStudioUser>` - The username of the Innovation Studio user under which this import is executed. The user has to exist in Innovation Studio prior to this operation.
- `<user_password>` - The password for the Innovation Studio user.
- `<DataBaseServer>` - The name of the database server where the FintechOS installation database was created.
- `<DB_user>` - The username of the SQL Server user with administration rights on the FintechOS installation database.
- `<DB_user_password>` - The password for the above mentioned SQL user.

- <TheNameOfTheDataBase> - The name of the database where the Operational Ledger v3.3 is deployed.
- <syspack\_path> - The physical path to the unzipped Operational Ledger v3.3 previously downloaded.

#### HINT

For more information about the script, please run `FtosSysPackageDeployer.exe` without any arguments to see the built-in help.

#### IMPORTANT!

If you're using **SQL Server Integrated Authentication**, make sure that the Windows user used for running the script has access to the FTOS database, with read/ write rights. Run the command without the SQL username/ password parameters.

If you're using **SQL Server Build In Authentication**, make sure that the SQL Server user has read/ write access to the FTOS database. Run the command with the SQL username/ password parameters.

## Installing Operational Ledger Add-On v3.3

Follow the steps described below to perform an automatic installation of the Operational Ledger Add-On.

## Dependencies

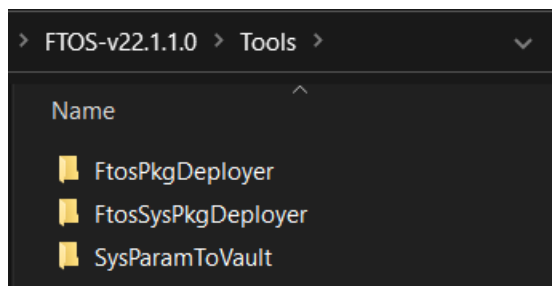
In order to install Operational Ledger Add-On v3.3, first you need to install the following:

- **Innovation Studio** minimum version **v22.1.1**

- **SySDigitalSolutionPackages** minimum version **v22.1.1000**
- **Banking Product Factory** (Project or Standard Pack) **v3.3**
- **Core Banking v3.3**
- **Operational Ledger v3.3.**

## Installation Steps

1. Unzip your **Operational Ledger AddOn v3.3** archive file.
2. Locate the *FtosSysPkgDeployer* folder in the FintechOS installation kit (the path is *<unzipped\_install\_archive>\Tools\FtosSysPkgDeployer*). You need it to install the Operational Ledger AddOn v3.3.



3. Select and copy the *FtosSysPkgDeployer* folder.
4. Navigate to the location where you have unzipped the **Operational Ledger AddOn v3.3** (let's call this location *<pckg\_deployer\_dir>*), then paste the *FtosSysPkgDeployer* folder there, within each and every zip file that comes with the package.

Operational Ledger AddOn v3.3	
Name	
01.Backup Settings v3.3	
02.Operation Ledger AddOn v3.3	
03.General Ledger Security Roles	

Operational Ledger AddOn v3.3 > 01.Backup Settings v3.3	
Name	
Backup Restore Settings	
FtosSysPkgDeployer	
Backup Settings.zip	

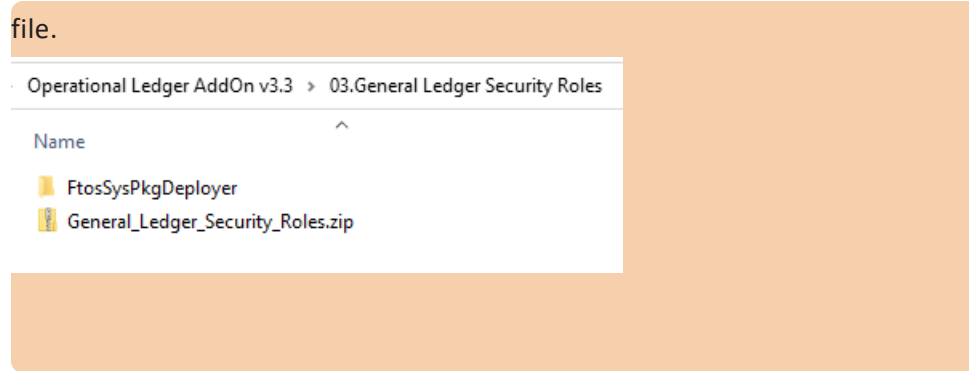
Operational Ledger AddOn v3.3 > 02.Operation Ledger AddOn v3.3	
Name	
FtosSysPkgDeployer	
GL Core Banking AddOn	
Operational Ledger AddOn.zip	
README_install_dependencies.txt	

Operational Ledger AddOn v3.3 > 03.General Ledger Security Roles	
Name	
FtosSysPkgDeployer	
General_Ledger_Security_Roles.zip	

**IMPORTANT!**

Each zip file within the **Operational Ledger AddOn v3.3.zip** must be extracted and installed separately, in the given order!  
If you decide not to use the default security roles that come with the package, simply skip the General Ledger Security Roles zip

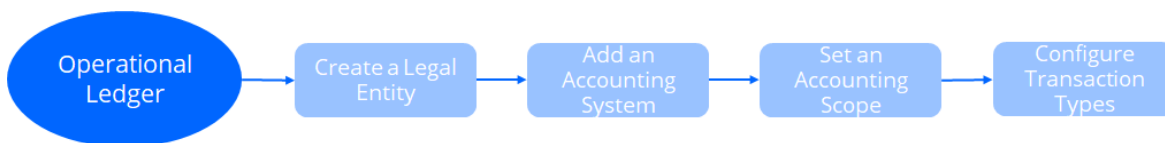


5. Create a new or edit a previous `install_Syspack.bat` file. Replace the parameters described in the "[install\\_SysPack.bat Parameters Explanation](#)" on page 9 section with your own values. Save and close the file.
6. Right-click `install_SysPack.bat` » **Run**, for each package.

# Configurations

The **Operational Ledger** 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 **Operational Ledger** after installation, you should perform a series of configurations within the **FintechOS Portal**, as presented in the following diagram:



The configurations are detailed in the pages below:

- [Legal Entity](#)
- [Accounting System](#)
- [Accounting Scope](#)
- [Transaction Type](#)

## Legal Entity

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 > **General Ledger Configurations** > **Legal Entity**<sup>1</sup> menu item to open the **Legal Entities List** page.
2. Click the **Insert** button to create a new legal entity. The **Add Legal Entity** page opens.
3. Fill in the following fields:

ADD LEGAL ENTITY

**Legal Entity**

**Name**

**Status**

**Valid From Date**

**Valid Until Date**

- **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 or Active.
- **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. The **Legal Entity Accounting Systems** section is displayed.

## Adding an Accounting System to the Legal Entity

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

---

<sup>1</sup>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.

1. To add a new legal entity accounting system, click the **Insert** button under the **Legal Entity Accounting Systems** section. The **Add Legal Entity 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.
3. Click the **Save and close** button.

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

## Accounting System

An **accounting system**<sup>1</sup> 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.

### Creating Accounting Systems

1. In **FintechOS Portal**, click the main menu > **General Ledger Configurations** > **Accounting System** menu item to open the **Accounting Systems List** page.

---

<sup>1</sup>The system used to manage the income, expenses, and other financial activities of a business.



2. Click the **Insert** button to create an accounting system. The **Add Accounting System** page opens.
3. Fill in the following fields:

ADD ACCOUNTING SYSTEM

Accounting System

Code	Name
<input type="text" value="IRS"/>	<input type="text" value="0001"/>
Accounting Reference Currency	
<input type="text" value="EUR"/>	

- **Code:** Enter the code of the accounting system.
- **Name:** Enter the official name 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:

AccountingChart

Accounting System	reportingCode	Reporting Group Name
<input type="text" value="0001"/>	<input type="text" value="201"/>	<input type="text" value="Commercial loans"/>
Account Code	Name	
<input type="text" value="20111"/>	<input type="text" value="20111 Discount"/>	
Balance Type	Type	
<input type="text" value="BalanceSheet"/>	<input type="text" value="Assets"/>	

- **Accounting System:** Select the accounting system of the **chart**<sup>1</sup>.
  - **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 Scope

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

### Creating Accounting Scopes

1. In **FintechOS Portal**, click the main menu > **General Ledger Configurations** > **Accounting Scope** menu item to open the **Accounting Scopes List** page.
2. Click the **Insert** button to add a new accounting scope. The **Add Accounting Scope** page opens.
3. Fill in the accounting scope **Name**.

---

<sup>1</sup>An index of all the financial accounts in the general ledger of a company.

ADD ACCOUNTING SCOPE

Accounting Scope

Name



---

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

From here, you can export the transactions displayed in the **Banking Product GL Accounts** and **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.

## Transaction Type

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

For using transaction types in conjunction with Core Banking, see [Transaction Types Used in Core Banking](#).

### Creating Transaction Types

1. In **FintechOS Portal**, click the main menu > **General Ledger Configurations** > **Transaction Type** menu item to open the **Transaction Types List** page.
2. Click the **Insert** button to create a new transaction type.

- In the newly displayed **Add Transaction Type** page's **Settings**, fill in the following fields:

Edit Transaction Type

**Settings**

Name: Disbursement | Transaction Code: OSB | Process Type: Disbursement | Edit Form: FTOS\_CB\_EventDisbursement | Transaction Operation Type: Disbursement | Is System Transaction:

Is Automatic Transaction:  | Real Time Process:  | Only One Draft:  | Generate New Contract Version:

Commission Type: | Return Commission Type: | Is Clawback Transaction:

**Accounting Configurations**

Generates Accounting Entry:  | OL Master Entity: FTOS\_CB\_ContractEvent

**Purge Configurations**

To Be Purged:  | Purge Number of Days: 30 | Master Purge Entity: FTOS\_CB\_ContractEvent

**Transaction Value Types**

Transaction Type	Type	Value Type Name	Is Header	Value Type Attribute
Disbursement	Numeric	EventValue	<input type="checkbox"/>	eventValue
Disbursement	Text	ContractEventId	<input type="checkbox"/>	FTOS_CB_Contr...

**Transaction Item Accounting Configurations**

Transactio...	Accounting...	Accounting...	Chart Acco...	Currency	Operational...	Take From...
Disburse...	0001	OFFBLoa...	90300 C...			
Disburse...	0001	Provision...	66211 Im...			

- Name:** Enter the name of the transaction type.
- Transaction Code:** Enter the code of the transaction type.
- Process Type:** This field associates the process type with the transaction operation type, which makes the connection with the transaction motor of bank accounts. Select one of the processes defined within the system.

**NOTE**

There is no integration at this point with a payment gateway.

- Edit Form:** Select the form used to edit this transaction type. The lookup only shows forms defined on the Contract Event entity.
- Transaction Operation Type:** The transaction operation type makes the connection with the transaction motor of bank accounts. Select one of the transaction operation types defined within the system.
- Is System Transaction:** Select this checkbox to mark the transaction as neither an account credit nor an allocation of funds but a transaction that does not affect other transactions in the account. System transactions are meant to be used only for accounting. It is used when there's a need of a

transaction for the sole purpose of generating accounting entries (accruals, provisions).

- **Is Automatic Transaction:** This checkbox marks the transaction as automatic. If you select it, then you cannot select the transaction type within the contract operations.
  - **Real Time Process:** If you select this checkbox, then the transaction is processed right away. If you do not select it, then the transaction is inserted as a [bank account transaction queue](#) record. This checkbox marks the transaction as to be processed in real-time.
  - **Only One Draft:** If you select this checkbox, there can only be one record of this transaction type in the Draft status created at the contract level.
  - **Generate New Contract Version:** If you select this checkbox, then a new contract version is generated by this transaction type.
  - **Commission Type:** Select the commission type applicable for this transaction type. Return fee commissions are filtered out and can't be selected.
  - **Return Commission Type:** Select the return fee commission type applicable for this transaction type. You can only select fee commissions.
  - **Is Clawback Transaction:** If selected, this checkbox marks the transaction as generating a clawback commission. The transaction type is to be used in third-party management invoicing activities. Default value: False.
4. In the **Accounting Configurations** section, fill in the following fields:
- **Generates Accounting Entry:** Select this checkbox to create an entry in the general ledger. It generates records in the Accounting Entry entity.
  - **OL Master Entity:** Select the entity referenced by the accounting systems.
5. In the **Purge Configurations** section, fill in the following fields:
- **To Be Purged:** This checkbox marks the transaction as purgeable. Select it so that you can purge or archive records in Draft status.
  - **Purge Number of Days:** Enter the default number of calendar days that a record can be kept in Draft status before it is purged. For additional information, see the [Core Banking System Parameters](#) page. This field is

displayed only when the To Be Purged checkbox is selected.

- **Master Purge Entity:** Select the master purge entity under which the draft records are. This field is displayed only when the To Be Purged is selected.

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

**NOTE**

If a transaction type is marked to automatically generate accounting entries (the *Generates Accounting Entry* checkbox is selected), then the following sections are displayed: **Transaction Value Type**, **Transaction Item Accounting Configuration**, and **Transaction Accounting Models**.

## Manage Transaction Value Type

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.

1. To add a new transaction value type, click the **Insert** button under the **Transaction Value Type** section. The **Add Transaction Value Type** page opens.

Transaction Value Type

Value Type Name EventValue	Transaction Type Disbursement	Is Header <input checked="" type="checkbox"/>
Type Numeric	Value Type Attribute eventValue	Formula eventValue*2

2. Fill in the following fields:

- **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.

3. 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.

## Define Transaction Item Accounting Configuration

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.

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

The screenshot shows the 'Transaction Item Accounting Config' form with the following fields and values:

- Accounting System:** 0001
- Accounting Scope:** LoanAccount
- Chart Account:** 20219 Other treasury loans
- Operational Item:** Loan Principal
- Currency:** EUR
- Take From Banking Product:**

2. 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 this checkbox, then the configurations for each transaction are inherited from the banking product level.
3. Click the **Save and close** button.

## Manage Transaction Accounting Models

This section 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.

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

2. Fill in the following fields:
  - **Transaction Type:** The transaction type. It is auto-filled.
  - **Accounting System:** Select the accounting system.
  - **Line Condition:** Enter the condition applied in order to post the accounting entry line.
  - **Debit Account Rule:** The accounting entry value of the debit account. It is auto-filled.
  - **Debit Customer Rule:** The rule to save the partner transaction in the debit-credit relationship. It is auto-filled.
  - **Accounting Entry Value Rule:** Enter the posted accounting entry value.
  - **Item Rule:** The transaction item of the accounting entry line. It is auto-filled.
  - **EntityId Rule:** Enter the internal status of the record.

- **Transaction Detail Rule:** Enter the rule to identify and post the ID of the operational transaction detail.
  - **Credit Account Rule:** Enter the credit account of the accounting entry line. It is auto-filled.
  - **Credit Customer Rule:** It is auto-filled by the destination partner ID.
  - **currencyRule:** The accounting entry line currency. It is auto-filled.
  - **DescriptionRule:** The description of the generated accounting entry. It is auto-filled.
  - **Transaction Id Rule:** Enter the related contract ID of the transaction.
  - **Transaction Value Type:** Select the value type of the transaction. It is defined in the **Transaction Value Type** section.
3. Click the **Save and close** button.

# Transactions

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

The **Operational Ledger** 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](#)

## Operation Transactions

If you want to search for existing transactions or create new ones, use the **OL Operation Transaction** 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 > **General Ledger** > **GL Operation Transaction** menu item to open the **GL Operation Transaction List** page.
2. Click the **Insert** button to create an operation transaction.
3. Fill in the following fields:

ADD GL OPERATION TRANSACTION

**GL Operation Transaction**

<p><b>Transaction Type</b></p> <input type="text" value="Accruals and Provisions"/> <p><b>Accounting Date</b></p> <input type="text" value="11/08/2022"/> <p><b>Product</b></p> <input type="text" value="e44d5b27"/> <p><b>Source Partner</b></p> <input type="text"/> <p><b>Analytic Debit Code</b></p> <input type="text" value="TL_EUR"/>	<p><b>Currency</b></p> <input type="text" value="EUR"/> <p><b>Transaction Date</b></p> <input type="text" value="23/08/2022"/> <p><b>Item</b></p> <input type="text" value="74805593"/> <p><b>Destination Partner</b></p> <input type="text"/> <p><b>Analytic Credit Code</b></p> <input type="text" value="TL_EUR"/>
---	---

- **Transaction Type:** Select the type of the transaction.
  - **Accounting Date:** Enter the accounting date.
  - **Product:** Enter the product of the transaction.
  - **Source Partner:** Select the source partner.
  - **Analytic Debit Code:** Enter the analytic debit code.
  - **Currency:** Select the currency of the accounting entry line.
  - **Transaction Date:** Select the transaction date.
  - **Item:** Enter the transaction item.
  - **Destination Partner:** Select the destination partner.
  - **Analytic Credit Code:** Enter the analytic credit code.
4. Click the **Save and reload** button. The **Operation Transaction Values** and the **Accounting Entries** sections are displayed.

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

## Adding Operation Transaction Values

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.

1. To add a new operation transaction value, click the **Insert** button under the **Operation Transaction Values** section. The **Add Operation Transaction Value** page opens.
2. Fill in the following fields:

Operation Transaction Value	
Name	Operation Transaction
<input type="text" value="Contractid"/>	<input type="text" value=""/>
Transaction Value Type	Currency
<input type="text" value="Contractid"/>	<input type="text" value="EUR"/>
Value	Text
<input type="text" value="0"/>	<input type="text" value="D02A0E8C-05B4-4E44-AC5A-920FC65D3063"/>

- **Name:** Enter the name of the transaction value.
  - **Transaction Value:** Select the value type of the transaction.
  - **Value:** Enter the value of the transaction.
  - **Operation Transaction:** Select the operation transaction ID.
  - **Currency:** Enter the currency of the accounting entry line.
  - **Text:** Enter the description of the transaction.
3. Click the **Save and reload** button. The **Accounting Entries** section is displayed.

The **Accounting Entries** section 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](#).

# Accounting Entry

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

## Creating Accounting Entries

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

<b>Accounting Entry</b>	
<b>Accounting Date</b> 06/05/2022	<b>Accounting Value</b> 1.02
<b>Debit Partner</b> BOBO - Cust 3	<b>Credit Partner</b> BOBO - Cust 3
<b>Debit Account</b> 20271 Accrued Interest	<b>Credit Account</b> 70222 Interest from term loans
<b>Analytic Debit Account Code</b> 20271.TL - RF	<b>Analytic Credit Account Code</b> 70222.TL - RF
<b>Currency</b> EUR	<b>Exchange Rate</b> 1
<b>Equivalent Value</b> 1.02	<b>Item</b> Loan Interest
<b>Description</b> EOD 06.05.2022	

- **Accounting Date:** Enter the date of the accounting entry line.
- **Accounting Value:** Enter the value of the entry.
- **Debit Partner:** Select the debit partner.
- **Credit Partner:** Select the credit partner.
- **Debit Account:** Select the debit account.

- **Credit Account:** Select the credit account.
- **Analytic Debit Account Code:** Enter the code of the analytic debit account.
- **Analytic Credit Account Code:** Enter the code of the analytic credit account.
- **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.
- **Item:** Select the accounting registration item.
- **Description:** Enter the description of the accounting entry.

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

You can add, delete, or export accounting entries.

# Operational Ledger 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 Operational Ledger.

The functions available for the implemented library and script are presented below. These functions allow Operational Ledger 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 are 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 are 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){  
        });
```

# Glossary

---

## A

---

### **Accounting**

The measurement, processing, and communication of financial and non financial information about economic entities such as businesses and corporations.

### **Accounting chart**

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

### **Accounting system**

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

## C

---

### **Circulation**

The circulation of money/ transactions in a given period.

### **Credit**

An entry recording of a received sum. It has a positive value.

## D

---

### **Debit**

An entry recording of an owed sum. It has a negative value.

I

---

**IBAN**

International Bank Account Number

L

---

**Ledger**

A book in which the monetary transactions of a business are posted in the form of debits and credits.

**Legal Entity**

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.

T

---

**Transaction**

An instance of buying or selling something.