Client Assets Protection and Reconciliations

Contents

[Background 2](#_Toc108107953)

[A Typical Real World Trade Flow 2](#_Toc108107954)

[What are Client Assets, Client Money? 2](#_Toc108107955)

[How Do Financial Institutions Manage Client Money Risks? 3](#_Toc108107956)

[The Challenge 3](#_Toc108107957)

[What is a LEDGER 4](#_Toc108107958)

[What is SWIFT 4](#_Toc108107959)

[Example of a SWIFT BIC code 5](#_Toc108107960)

[SWIFT ISO20022 XML Explained 5](#_Toc108107961)

[Bonus Questions 7](#_Toc108107962)

[Challenge Judging Points 7](#_Toc108107963)

[Appendix A 8](#_Toc108107964)

[Resources 10](#_Toc108107965)

[References 11](#_Toc108107966)

# Background

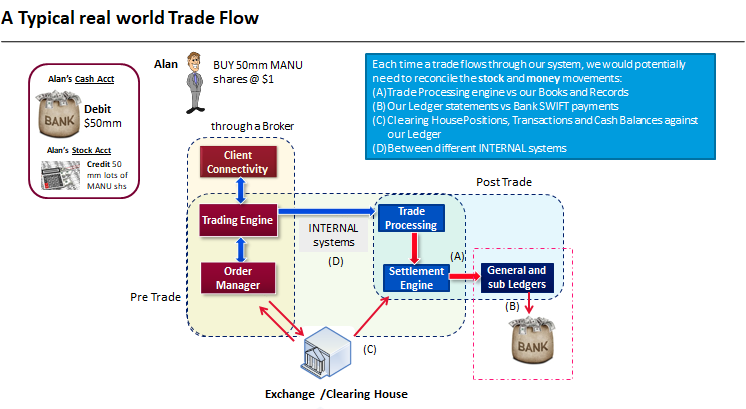
The protection of client assets is crucial to maintaining the integrity of the financial markets and securing consumers' rights and the trust they place with the Financial Institution (FI). Failure to adequately protect clients is considered a serious matter by regulators. In 2015, an investment bank was fined £126m by the Financial Conduct Authority (FCA) due to failures to adequately protect client assets.

One critical step for ensuring adequate client assets protection and compliance with financial regulation rules is to completely and accurately identify Client Assets, and report as well as remediate any Client Money issues, in a timely fashion.

Therefore, reconciliations must be performed by every FI that holds client assets.

## A Typical Real World Trade Flow

Figure 1: Alan's Order Flow



When Alan wants to buy 50 million of MANU shares, the order flows through the FI’s (aka Broker) systems and is finally recorded in the Ledger. Sometimes, the large order might be split into a series of executions (or trades) by the Broker’ algorithmic (i.e., smart order) engines. *Why might this be so?*

## What are Client Assets, Client Money?

Client assets are defined as money, securities and/or positions which are held or controlled by a Financial Institution (FI) for investment purposes on behalf of their clients.

Client money is money of any currency that a FI entity holds or receives from a client and is required to protect and segregate from its own funds.

## How Do Financial Institutions Manage Client Money Risks?

Reconciliation between the FI’s Ledger records and the SWIFT bank to customer statement is performed as an essential control step to manage client money risks. Any unmatched/mismatched transaction and/or balance records, i.e., “client money breaks”, will then need to be flagged, as this could represent client assets and/or money that needs to be properly segregated.

*Note:* The actual client money segregation rules are a lot more complicated and depends on the entity, type of products, as well as jurisdiction.

# The Challenge

For this challenge, we will only focus on (B) Our Ledger statements vs Bank SWIFT payments in Figure 1: Alan’s Order Flow. You will need to come up with a schematic Ledger vs SWIFT reconciliation between two fictional counterparties, a ‘sender counterparty’ (i.e., the Sender) and a ‘receiving counterparty’ (i.e., the Receiver). You will need to implement the reconciliation as the Receiver.

You are provided with sample input files in resources.zip under **Resources**. These are:

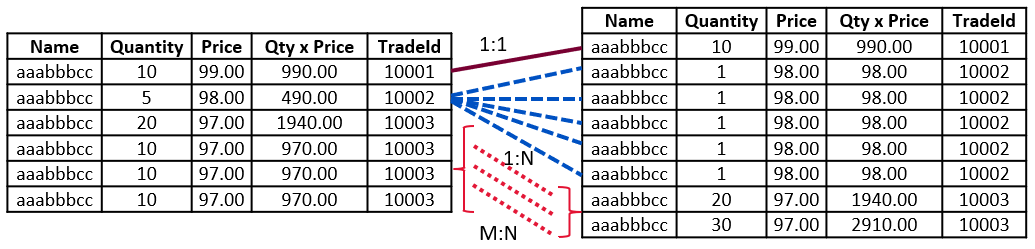
* a sample of the processed data that feeds into the Ledger in a CSV (tabular) format
  + LedgerBalance.csv
  + LedgerTransactions.csv
* a sample of the SWIFT data that is received in ISO20022 XML format
  + swift\_sample\_msg.xml

You will need to read and parse the source files, and then write a matching algorithm to completely and accurately identify “client money breaks”. Your matching algorithm would need to (minimally) be able to perform the following types of matches:

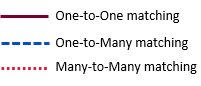
* One-to-One Matching [1:1]
* Aggregate Matching
  + One-to-Many [1:N]
  + Many-to-Many [M:N]

*Do you think the match type sequencing matters?*

Exhibit A: Matching Example



**Legend:**

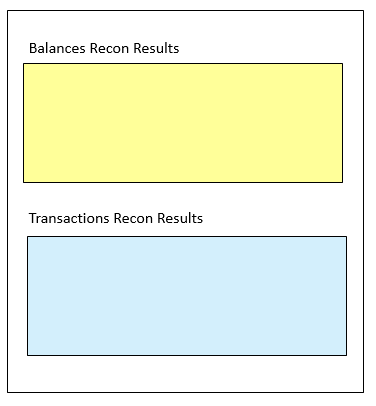


You will need to perform TWO (2) types of reconciliation checks.

**Full Integrity:** The full integrity check helps identify the root cause of balance breaks. In most cases an unmatched transaction will support the difference in the balances. We can use the unmatched transaction to explain the balance break and leave the users with just the unmatched transaction to research.

**Proofing:** This is checking the movement in balances intra / day-over-day is supported by the transactions. This ensures that the source system has sent us all the data. If the reconciliation is not in proof, it implies an issue with the source data, calling the entire reconciliation result into question.

You will need to create a simple (web) UI to display the results intuitively. A suggested reconciliation results display could be:



## What is a LEDGER

A ledger is a book or collection of accounts in which account transactions are recorded. Each account has an opening or carry-forward balance and would record each transaction as either a debit or credit in separate columns, and the ending or closing balance.

## What is SWIFT

Most international money and security transfers between financial institutions and intermediaries are performed today through the Society for Worldwide Interbank Financial Telecommunications (SWIFT) system. SWIFT is a messaging network that financial institutions use to securely transmit information and instructions through a standardized system of codes.

The Sender and Receiver each has their own Bank Identifier Code (BIC). This is a unique identifier for a specific financial institution. A BIC is composed of a 4-character bank code, a 2-character country code, a 2-character location code and an optional 3-character branch code. BIC codes are used to send money between banks to ensure money is directed to the right place.

## Example of a SWIFT BIC code

**AAAA** Bank code

**BB** Country code

**CC** Location code

**123** Branch code

We will assume that the BIC codes are as follows:

* Sender: LVERSGSGXXX
* Receiver: POOLSG3XXXX

The SWIFT payment messages are fully contained in an XML document format and transmitted via the SWIFT alliance network.

## SWIFT ISO20022 XML Explained

The bank to customer statement message is sent by the account servicer to an account owner or to a party authorised by the account owner to receive the message. It is used to inform the account owner, or authorised party, of the entries booked to the account, and to provide the owner with balance information on the account at a given point in time.

Please refer to **Resources** section for the full sample of the SWIFT ISO20022 bank to customer statement XML message included.

It consists of 3 parts:

1. **AppHdr**: This contains information about the Sender and Receiver.

<AppHdr>

<Fr>

<FIId>

<FinInstnId>

<- Sender BIC

<BICFI>LVERSGSGXXX</BICFI>

</FinInstnId>

</FIId>

</Fr>

<To>

<FIId>

<FinInstnId>

<- Receiver BIC

<BICFI>POOLSG3XXXX</BICFI>

</FinInstnId>

</FIId>

</To>

</AppHdr>

1. **GrpHdr**: This contains common information for the message, example Message ID and Message Creation Date Time.

<GrpHdr>

<MsgId>123456789</MsgId>

<CreDtTm>2022-07-04T10:57:33+00:00</CreDtTm>

</GrpHdr>

1. **Stmt**: This contains information on booked entries and balances for a cash account.

<Stmt>

<Id>100-01</Id>

<StmtPgntn>

<PgNb>1</PgNb>

<LastPgInd>true</LastPgInd>

</StmtPgntn>

<LglSeqNb>1001</LglSeqNb>

<Acct>

<Id>

<Othr>

<Id>5491-00003011-MULT</Id>

</Othr>

</Id>

<Ccy>SGD</Ccy>

</Acct>

<Bal>

<Tp>

<CdOrPrtry>

<Cd>OPBD</Cd>

</CdOrPrtry>

</Tp>

<Amt Ccy="SGD">65000000.</Amt>

<CdtDbtInd>CRDT</CdtDbtInd>

<Dt>

<Dt>2022-07-04</Dt>

</Dt>

</Bal>

<Bal>

<Tp>

<CdOrPrtry>

<Cd>CLBD</Cd>

</CdOrPrtry>

</Tp>

<Amt Ccy="SGD">15052345.27</Amt>

<CdtDbtInd>CRDT</CdtDbtInd>

<Dt>

<Dt>2022-07-04</Dt>

</Dt>

</Bal>

<Ntry>

<Amt Ccy="SGD">15000000</Amt>

<CdtDbtInd>DBIT</CdtDbtInd>

<Sts>

<Cd>BOOK</Cd>

</Sts>

<BookgDt>

<DtTm>2022-07-04T10:15:00+00:00</DtTm>

</BookgDt>

<ValDt>

<Dt>2022-07-04</Dt>

</ValDt>

<NtryDtls>

<TxDtls>

<Refs>

<EndToEndId>ALAN/MANU/PURCH-001</EndToEndId>

<UETR>50b6ef580b09448029e5f9302364d632</UETR>

</Refs>

<Amt Ccy="SGD">15000000</Amt>

<CdtDbtInd>DBIT</CdtDbtInd>

</TxDtls>

</NtryDtls>

</Ntry>s

<Ntry>…</Ntry>

<Ntry>…</Ntry>

<Ntry>…</Ntry>

</Stmt>

You may assume that the XML is valid, well-formed, and conforms to the SWIFT ISO20022 XML Schema Definition (XSD). Please refer to **Appendix A** for details of XML Element specifications for <Stmt>:

## Bonus Questions

1. What other types of SWIFT data validations can be performed?
2. How can you ensure that there are no duplicate SWIFT messages?
3. What are some things you will consider for matching performance?

## Challenge Judging Points

You can refer to the below criterion and points worth as an indication of what to focus on.

|  |  |
| --- | --- |
| **Criterion** | **Points worth** |
| Backend design and development (read/parse files and data store) | 20 pts |
| Matching algorithm | 20 pts |
| Frontend (UI) design and display | 10 pts |
| Correctness of results | 20 pts |
| Code quality and Organisation | 10 pts |
| Demonstration (to mentor) of the following:   * design considerations (inclusive of any assumptions) * solutions and challenges faced * explanations to bonus questions | 20 pts |
| Total: | 100 pts |

**Note:** Please be expected to demonstrate and explain your solution to the mentors.

Some points to consider:

* You are free to use any programming language/framework of choice.
* Your application code should show clear logic.
* Attention to code structure and code quality is important.
* Errors or exceptions should be handled appropriately.
* The UI can be simple but should display the results clearly.
* Writing unit tests is highly encouraged.

# Appendix A

|  |  |  |  |
| --- | --- | --- | --- |
| ***<XML Tag>*** | **Mult.** | **Type** | **Definition** |
| Identification *<Id>* | [1..1] | Text | Unique identification, as assigned by the account servicer, to unambiguously identify the account statement. |
| StatementPagination *<StmtPgntn>* | [0..1] | ± | Provides details on the page number of the statement. |
| PageNumber *<PgNb>* | [1..1] | Text | Page number. |
| LastPageIndicator *<LastPgInd>* | [1..1] | Indicator | Indicates the last page. |
| LegalSequenceNumber *<LglSeqNb>* | [0..1] | Quantity | Legal sequential number of the statement, as assigned by the account servicer. It is increased incrementally for each statement sent. |
| Account <Acct> | [1..1] |  | Unambiguous identification of the account to which credit and debit entries are made. |
| Identification <Id> | [0..1] | ± | Unique and unambiguous identification for the account between the account owner and the account servicer. |
| Other <Othr> | [1..1] | ± | Denotes generic (non-IBAN) account identification. |
| Identification <Id> | [1..1] | Text | Identification assigned by an institution. |
| Currency <Ccy> | [0..1] | CodeSet | Identification of the currency in which the account is held. |
| Balance *<Bal>* | [1..\*] |  | Set of elements used to define the balance as a numerical representation of the net increases and decreases in an account at a specific point in time. |
| Type *<Tp>* | [1..1] | ± | Specifies the nature of a balance. |
| CodeOrProprietary *<CdOrPrtry>* | [1..1] |  | Coded or proprietary format balance type. |
| Code *<Cd>* |  |  | Balance type, in a coded format (we will assume there are only 2 types). See below for meanings. OPBD = OpeningBooked. Book balance of the account at the beginning of the account reporting period. It always equals the closing book balance from theprevious report.  CLBD = ClosingBooked. Balance of the account at the end of the pre-agreed account reporting period. It is the sum of the opening booked balance at the beginning of the period and all entries booked to the account during the pre-agreed account reporting period. |
| Amount *<Amt>* | [0..1] | Amount | Amount of money of the cash balance. |
| CreditDebitIndicator *<CdtDbtInd>* | [1..1] | CodeSet | Indicates whether the balance is a credit or a debit balance. CRDT = Credit. Operation is an increase. DBIT = Debit. Operation is a decrease. |
| Date *<Dt>* | [1..1] | ± | Indicates the date (and time) of the balance. |
| Date *<Dt>* | [1..1] | Date | ISO date. A particular point in the progression of time in a calendar year expressed in the YYYY-MMDD format. |
| Entry *<Ntry>* | [0..\*] |  | Specify an entry in the statement. |
| Amount *<Amt>* | [1..1] | Amount | Amount of money in the cash entry. |
| CreditDebitIndicator *<CdtDbtInd>* | [1..1] | CodeSet | Indicates whether the entry is a credit or a debit entry. |
| Status *<Sts>* | [1..1] | ± | Status of an entry on the books of the account servicer. |
| Code *<Cd>* | [1..1] | CodeSet | Entry status, in a coded form.  BOOK = Booked. Means that the transfer of money has been completed between account servicer and account owner. |
| BookingDate *<BookgDt>* | [0..1] | ± | Date and time when an entry is posted to an account on the account servicer's books. |
| DateTime *<DtTm>* | [1..1] | DateTime | ISO DateTime. A particular point in the progression of time defined by a mandatory date and a mandatory time component, expressed in either UTC time format (YYYY-MM-DDThh:mm:ss.sssZ), local time with UTC offset format (YYYY-MM-DDThh:mm:ss.sss+/-hh:mm), or local time format (YYYY-MMDDThh:mm:ss.sss). |
| ValueDate *<ValDt>* | [0..1] | ± | Date and time at which assets become available to the account owner in case of a credit entry, or cease to be available to the account owner in case of a debit entry. |
| EntryDetails *<NtryDtls>* | [0..\*] |  | Provides details on the entry. |
| TransactionDetails *<TxDtls>* | [0..\*] |  | Provides information on the underlying transaction(s). |
| References *<Refs>* | [0..1] |  | Provides the identification of the underlying transaction. |
| EndToEndIdentification *<EndToEndId>* | [0..1] | Text | Unique identification, as assigned by the initiating party, to unambiguously identify the transaction. This identification is passed on, unchanged, throughout the entire end-to-end chain. |
| UETR *<UETR>* | [0..1] | IdentifierSet | Universally unique identifier to provide an end-to-end reference of a payment transaction. |
| Amount *<Amt>* | [0..1] | Amount | Amount of money in the cash transaction. |
| CreditDebitIndicator *<CdtDbtInd>* | [0..1] | CodeSet | Indicates whether the transaction is a credit or a debit transaction. |
| Batch *<Btch>* | [0..1] |  | Provides details on batched transactions. |
| MessageIdentification *<MsgId>* | [0..1] | Text | Point to point reference, as assigned by the sending party, to unambiguously identify the batch of transactions. |
| PaymentInformationIdentification *<PmtInfId>* | [0..1] | Text | Unique identification, as assigned by a sending party, to unambiguously identify the payment information group within the message. |
| NumberOfTransactions *<NbOfTsxs>* | [0..1] | Text | Number of individual transactions included in the batch. |
| TotalAmount *<TtlAmt>* | [0..1] | Amount | Total amount of money reported in the batch entry. |
| CreditDebitIndicator *<CdtDbtInd>* | [0..1] | CodeSet | Indicates whether the batch entry is a credit or a debit entry. |

# Resources



# References

* <https://en.wikipedia.org/wiki/Ledger>
* <https://www.investopedia.com/articles/personal-finance/050515/how-swift-system-works.asp>
* <https://www.swift.com/standards/iso-20022/iso-20022-programme/iso-20022-programme-document-centre>
* <https://www.iso20022.org/catalogue-messages/additional-content-messages/external-code-sets>