Nasdaq Calypso

MarkitWire Integration Guide

Version 10.14.0

Revision 43.0
December 2024
Approved

Document History

Revision	Published	Summary of Changes
1.0	January 2014	First edition for version 14.0 Suite.
2.0	October 2014	Updates for version 14.1 of Core Calypso.
3.0	October 2015	Updates for version 5.2.0.
4.0	February 2016	Updates for version 5.3.0.
5.0	April 2016	Updates for version 5.4.0.
6.0	June 2017	Updates for version 6.2.0.
7.0	August 2017	Updates for version 6.3.0.
8.0	April 2018	Updates for version 7.2.1.
9.0	January 2019	Updates for version 7.1.6.
10.0	March 2019	Updates for version 7.4.0.
11.0	August 2019	Updates for version 7.4.4.
12.0	March 2020	Updates for version 7.4.5 – Added CCP Trade Division and Netting.
13.0	April 2010	Updates for version 7.5.2.
14.0	June 2020	Updates for version 8.2.0.
15.0	September 2020	Updates for version 8.2.1.
16.0	October 2020	Updates for version 8.4.0.
17.0	December 2020	Updates for version 8.5.1.

Revision	Published	Summary of Changes
18.0	February 2021	Updates for version 8.7.0.
19.0	April 2021	Updates for version 8.9.2.
20.0	October 2021	Updates for version 8.13.0, 8.14.1 – Added support for Equity Share Swap, Equity Share Options, Equity Index Swap, Equity Index Options.
21.0	February 2022	Updates for version 9.0.1 – Technical release only – Version 17.0 compatibility.
22.0	September 2022	Updates for version 9.6.0
23.0	October 2022	Added support for Global UTI, Global Prior UTI, and Global Block UTI fields from MarkitWire as trade keywords.
24.0	December 2022	Added support for multiple broker and counterparty BIC codes of MW. Added support for MarkitWire API version 19.2.1
25.0	February 2023	Added support for OffMarkitwire trades
26.0	March 2023	Added support for MarkitWire API version 20.0.1
27.0	April 2023	Added support for MarkitWire API version 20.0.2
28.0	May 2023	Added support for multiple broker and counterparty BIC codes.
29.0	June 2023	Added support for MarkitWire API version 20.1
30.0	July 2023	Added MarkitWire SEF trades Allocation performance improvement
31.0	August 2023	Added support for the UPI code from MarkitWire
32.0	September 2023	Added support for MarkitWire API version 20.2
33.0	October 2023	Added support of Discrepancy Clause Details on Cancellable IRS and OIS in incoming and outgoing mode
34.0	February 2024	Added support for Report Tracking Number, PTRR ID, PTRR, PTRR Technique & PTRR Service Provider for UPI.
35.0	March 2024	Added support for MarkitWire API version 20.2.1, 21.0, and 21.0.1
36.0	May 2024	Added support for MarkitWire API version 21.0.2
37.0	June 2024	Added CCP interest
38.0	July 2024	Added support for MarkitWire API version 21.1.0
39.0	August 2024	Added support for Calypso mapping for MW holiday USGS would be FRBNY.

Revision	Published	Summary of Changes
		Added support for MarkitWire API version 21.1.1.
40.0	September 2024	Added support for MarkitWire API version 21.2.0
41.0	October 2024	Removed MWProcessState|PickedUp, MWProcessState|Done, MWProcessState|Affirm, MWProcessState|Pending from SwapswireSchemaData.xml
42.0	November 2024	Added support for Fixed-Fixed Single Currency Swap in MarkitWire Interface
43.0	December 2024	Added support for MarkitWire API version 21.2.1
44.0	January 2025	Added support for MarkitWire API version 21.2.2

This document guides you through the setup required for importing Interest Rate Derivatives-related data from MarkitWire into Calypso with the MarkitWire interface.

Calypso supports the keywords and features found in this document. Undocumented keywords and features are subject to removal without notice.

NOTE: The Calypso License to use this Calypso Integration Module does not include a license for any third-party data services to which this module can interface. Clients are responsible for contracting with the appropriate third-party data service(s) prior to using this Calypso Integration Module.

Table of Contents

Introduction

1.1 Overview

MarkitWire is an electronic deal confirmation platform used by financial counterparties to enter, receive and affirm trades and other lifecycle trade events. These trades and trade events can be imported into Calypso with the MarkitWire interface described in this document.

A MarkitWire bidirectional module can also be used to submit trades and lifecycle events from Calypso to MarkitWire. Please contact your account manager for information on this feature, which is separately licensed and is covered in a separate document.

MarkitWire automates several business processes:

- Trade Capture
- Broker Confirmation
- ISDA Confirmation
- Clearing through LCH SwapClear
- Clearing to CCPs such as LCH and CME
- Trade division and Netting synchronization where supported by CCPs

Note: All functionality is not supported on all Calypso versions.

MarkitWire can be used when both parties to the trade are on MarkitWire, or when only one party is on (in this case, for broker confirmation and/or trade capture only). When only one side of the trade is on MarkitWire, this is called a single-sided deal.

The trade lifecycle within MarkitWire is modeled on current industry practice: traders have authority over the financial structure of deals in their books; they negotiate and strike deals with other market Participants directly, or via Brokers, and complete the tickets detailing the terms of the transactions. The Middle Office or Operations monitor the flow of transaction records from the trading desk. Operations maintain the institution's official books and records.

With MarkitWire, the bilateral trade confirmations traditionally exchanged between documentation teams are not required (except for single sided deals).

Please contact your account manager for information about licensing the clearing functionality available in the MarkitWire interface.

1.2 Software Requirements

1.2.1 Supported JDK Versions

The MarkitWire module was developed based on the core Calypso Java version. Please check the core version for JDK compatibility.

1.2.2 Microsoft Visual Studio Runtime Version

Clients need to install the Microsoft Visual Studio 2010 Runtime package.

- Please refer to the MarkitWire website for further information.

1.2.3 Supported MarkitWire API Version

- Please refer to the MarkitWire Release Notes for latest supported versions.

MarkitWire Calypso Interface

The MarkitWire-Calypso Interface enables trades from MarkitWire to automatically flow to Calypso via the Data Uploader and performs the following tasks:

- Converts incoming messages into BO Messages, where they are validated before being processed into Trades and uploaded to the database.
- The Calypso MarkitWire Trade Engine listens for and processes MarkitWire Dealsink notifications.
- The **SWValidateUpdate** TradeRule and **SWErrorUpdate** TradeRule workflow rules notify the Calypso MarkitWire Trade Engine to update MarkitWire whether trades have been validated or not.
- A workflow rule such as “CheckAmend” can be used to disallow modification of Calypso trades imported from MarkitWire.
- The UpdateTermination trade workflow rule should be added to your Terminate action (usually located between your Verified and Terminated status) to handle the rolling of External Reference IDs from the parent trade to the child trade.
- Each Calypso trade imported from MarkitWire is saved with a set of trade keywords containing information such as the trade’s MarkitWire **Deal ID**, etc.
- The Calypso Mapping table provides the infrastructure to map MarkitWire values to the equivalent internal representation.

The Calypso interface supports importing the following products from MarkitWire:

- Basis Swap
- Cross Currency Swap
- Equity Index Options, Equity Index Swap, Equity Share Options, Equity Share Swap – They are created as EquityLinkedSwap in Calypso
- Fixed-Fixed Cross Currency Swap
- Fixed-Fixed Single Currency Swap
- FRA
- Interest Rate Cancellable Swap
- Interest Rate Cap/Floors
- Interest Rate FRAs
- Interest Rate Swaps
- Interest Rate Swaps for Non-deliverable currencies (NDS)
- Interest Rate Swaps with Amortization
- Interest Rate Swaptions
- Non Deliverable OIS
- OIS

- OIS Basis Swap
- Zero Coupon Inflation Swap

Calypso supports the following trade lifecycle events from MarkitWire:

- Support Clearing Lifecycle states with LCH SwapClear - Clear and Declear
- Released trade capture
- Trade amendment - unilateral and bilateral
- Cancellation (Termination in Calypso terminology)
- Novation (Termination via Novation in Calypso terminology)
- Swaption exercise
- Partials (Termination or Novation)
- Exit - Exit reasons (SWExitReason) are stored as a trade attribute
- Allocation - Imports the MarkitWire allocated trades into Calypso
- Trade Division – Division of trades into alpha and beta trades post clearing
- Netting Synchronization – Synchronizing the netting results from MarkitWire
- Cancellable Exercise – Exercising Cancellable Swaps in MarkitWire

Calypso's MarkitWire integration provides limited supported for Allocations at this time. Block trades are not imported. Also, allocations done subsequent to a trade release are not captured. Additional support for Allocations is forthcoming.

The interface also supports the following:

- PrimeBrokered deals
- Single-sided deals
- Dealer deals
- Hedge fund/Dealer scenario
- Broker Confirmation
- SEF executed deals

2.1 Known Issues

▶ Users should also refer to the "Known Issues" section of the Calypso Data Uploader Integration Guide for additional information.

- Backloading via Deal matcher and Backloading report is deprecated.
- The scheduled task MW_RECON_REPORT is deprecated.

- For the trades which are part of Trade division, Unilateral Amends on the Alpha trade post clearing are not supported.
- For any trade we do not support Unilateral Amends on the prior version of trade in MarkitWire if the current version in Calypso is higher than the incoming message.
- Because core Calypso does not support **Compounding Method - Spread Exclusive**, neither does the Calypso MarkitWire Module. Spread Exclusive trades are mapped to Spread with a warning in the Task Station.
- In the event of an incorrect Book mapping, SWML messages are rejected outright without creating a BO Message. For other cases, the application creates BO Messages with ability to correct the setup and reprocess the failed SWML messages. Also we do not send out acknowledgements in case of such cases. Once the book is mapped and messages are correctly reprocessed then we send out the acknowledgements.
- If this occurs, release the Trade from MarkitWire once more to fetch the rejected trade. Note, using the Scheduled Task to fetch rejected trades may create duplicate BO Messages. To avoid duplicates, ensure that you use the attribute, MARKITWIRE_DEAL_IDS, on the SW_DO_RECOVERY. The attribute should contain a comma separate list of Deal IDs to retrieve from MarkitWire. If this attribute is empty, then Schedule Task will fetch all error trades (possibly resulting in duplicates), as in earlier releases. See related details.
- For amends and exercises action to function, you must either define AMEND or EXERCISE trade workflow actions or define custom actions. Please refer to the **UploadAmendAction** and **UploadExerciseAction** domain values in the Data Uploader.
- When a Basis Swap is Partially Terminated in Calypso, the first **Rate** for the newly-created residual trade (i.e., the child trade) is cleared. This is a core Calypso issue and not a problem specific to the MarkitWire interface.
- When reprocessing multiple blocked messages for the same trade, the status is sometimes updated incorrectly against the wrong version of the trade. This is apparently a bug in the MarkitWire API.
- During reprocessing of messages with Validation Workflow rules enabled, the trade status is sometimes shown as Saved even though the workflow is executed and **SWValidated** keyword is set to True. This occurs because the Validate status message is updated before the Saved message is sent to MarkitWire.
- Because MarkitWire treats a Partial Termination as an amend, when a Swaption/CapFloor is partially terminated multiple times in MarkitWire, the Premium is not propagated to a child trade because the premium date occurs before the Termination Effective Date. Thus, Premium Fees as applicable to the terminated amount are applied to terminated trade.

Calypso uses a common termination API to process all terminations and novations. The API expects origination fees on the parent trade and it will propagate partial amounts as per configuration.

Workaround for Partial Termination

Calypso suggests bringing both the partially terminated MarkitWire trade and the Calypso verified trade into sync by bilaterally amending the trade and removing/modifying fees in MarkitWire as per business functionality.

2.2 Installation Instructions

2.2.1 Upgrading from the Swapswire Module

When upgrading to the MarkitWire module from the Swapswire module, please note the following:

> **Note: MarkitWire requires the Calypso Data Uploader module. Please install the Data Uploader with the MarkitWire Module.**

- Every trade stored in Calypso is now assigned a unique External Key based on the MarkitWire ID and is unique for every MarkitWire trade across Positions.
- Previously, the Swapswire Additional Field was used to store Calypso Trade IDs. A Swapswire Additional Field now stores the External Key as assigned by the MW_EXT_REF_FLD environment property. Please note that the default for MW_EXT_REF_FLD is AdditionalField4. If you currently map this field to a Keyword, please set the value of MW_EXT_REF_FLD to a different Additional Field.
- During migration, you can use the MW_MIGRATE Scheduled Task (which can be used once) to convert existing Verified trades to the new format. Only Verified trades are modified. Calypso provides SQL migration scripts for Oracle (`sw_oracle_migration.sql`) and Sybase (`sw_sybase_migration.sql`) to move mappings:
  - From OptionExerciseTimeZone to TimeZone.
  - From OptionExerciseLocation to Location.
  - From Index Tenor to RateIndex. If there are no Tenor-specific mappings, there will be no change.

The migration script will also delete unused Keywords and Exception Types:

- Delete SWLoginHandle and SWDealHId, provided they are not used in any trades.
- Delete the EX_SWAPSWIRE_EXCEPTION and EX_SWAPSWIRE_INFORMATION exception types.
- Delete SWNovationAction.

2.2.2 MW_MIGRATE Scheduled Task

> **Note: Prior to running the MW_MIGRATE task, ensure that the SwapswireTradeEngine is *not* running. During the migration, there can only be one connection to MarkitWire.**

The MW_MIGRATE scheduled task provides a method to import custom trade statuses. To enable this feature, you must specify in the MWMigrateTradeStatus domain, all trade statuses that you wish to import. MW_MIGRATE also provides support for multiple login configurations.

> **Note: For multiple login configuration migration, please retain the sequence of user name and password during the migration as was used in the previous execution. Otherwise, this may result in exceptions.**

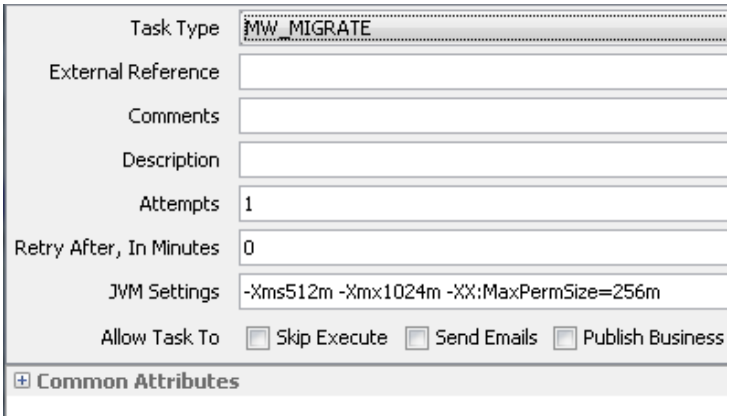
During migration, the application examines every trade for which the SWDealID trade keyword is not null (i.e., all trades from MarkitWire). In addition, if AUTO_FEED_EXTERNAL_REF is true, trades with existing External Reference IDs are not migrated.

The migration status of each trade is logged in the Task Station (via the EX_GATEWAY event).

Do not specify the TERMINATED status in this domain name to avoid identical external references on multiple trades. If no values are specified in MWMigrateTradeStatus, VERIFIED trades are migrated.

Configuring MW_MIGRATE scheduled task does not actually require MarkitWire DLLs or *nix libraries. Therefore, any user with sufficient permissions can configure the task using Configure > Scheduled Task from the Calypso Navigator. However, to run the MW_MIGRATE scheduled task, the *nix libraries must be added to LD_LIBRARY_PATH or the Windows DLLs must be in the Windows PATH.

Please note that the MW_MIGRATE scheduled task has no attributes.



General Notes on MW_MIGRATE

- Stop the SwapswireTradeEngine prior to running MW_MIGRATE.
- MW_MIGRATE considers all Calypso trades whose SWDealId keyword value is not null.
- MW_MIGRATE overwrites existing external reference IDs on migrated trades.
- The Task Station logs the migration status of all trades if the EX_GATEWAY event type is configured.
- If you are using single login through engine, the SWLoginHandle keyword on existing trade must be 1901. If the SWLoginHandle keyword is different, please update it to 1901.
- If MW_MIGRATE is unsuccessful, for example because of a configuration issue which has been addressed, then the Calypso Navigator and the Calypso Scheduler should be restarted prior to running the MW_MIGRATE scheduled task again to avoid caching issues.

2.2.3 Data Uploader Installation

▶ Refer to the Calypso Data Uploader Integration Guide for installation and configuration information.

You must install and configure the Calypso Data Uploader prior to configuring the MarkitWire Integration. The use of the Calypso MarkitWire Module requires the Data Uploader.

2.2.4 MarkitWire Components

Step 1 - Obtain the MarkitWire client and the Swapswire API from MarkitWire. The client file contains a number of Windows .DLL files, a Unix.lib file, and example files. The API contains numerous example and header files.

Step 2 - Uncompress the SW*.zip files to separate directories on the machine where you will run the Swapswire Engine.

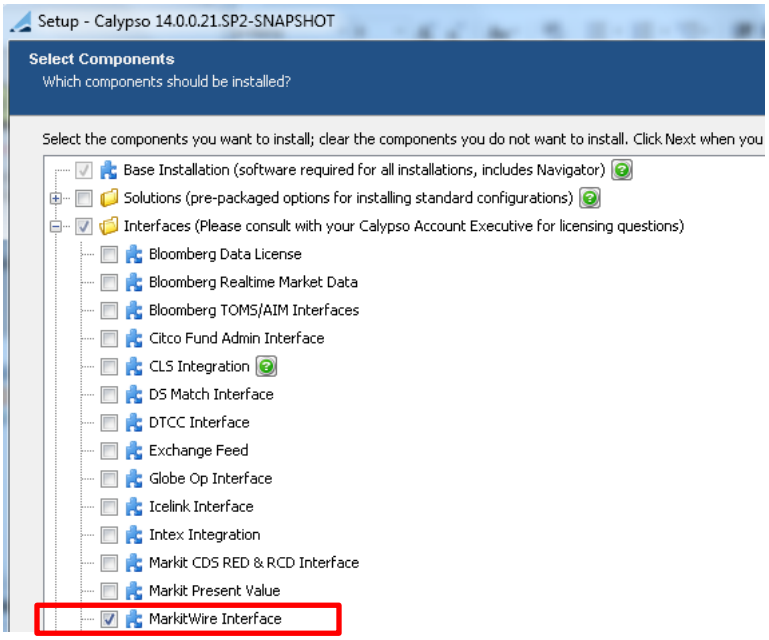
- If the machine running the Swapswire Engine is a 32-bit Windows platform, add the `PATH\TO\SW_X_X_XXXXX_api_win_32\lib` directory to the Windows PATH.
- If the machine running the Swapswire Engine is a 64-bit Windows platform, add the `PATH\TO\SW_X_X_XXXXX_api_win_64\lib` directory to the Windows PATH.
- If the Swapswire Engine machine is a *nix platform, add the appropriate libraries (*.so) the LD_LIBRARY_PATH.
- If the Swapswire Engine machine is on the Windows platform, ensure that the Swapswire DLLs are in the Path.

Step 3 - The DLL/so file versions must be equal to or less than MarkitWire server versions.

Step 4 - Obtain and install the Visual Studio 2010 redistributable package from the Microsoft website as described in MarkitWire's documentation.

2.2.5 Calypso Components

MarkitWire is installed as part of the Calypso Installer when you select the interface "Markitwire Interface":



▶ Please refer to the Calypso Installation Guide for complete details on the Calypso Installer.

If you are installing a Calypso Upgrade package instead, the instructions are also in the Calypso Installation Guide.

2.2.6 Firewall Requirements

Ensure that any firewall is configured to allow connections between your site and http://*.swapswire.com, ports 9009 and 443. If you require specific IP addresses, refer to http://www.markitserv.com/assets/jsp/mw-status.jsp. You will need your MarkitWire website login credentials to access the page.

2.2.7 Setup and Data Configuration Checklist

Integration Setup Checklist

- Environment Properties - Provide basic login information and configuration for single or multiple Dealsink users/listeners.

  ▶ See the "Environment Properties" section.

- Data Model Synchronization - Use ExecuteSQL to populate the following domains: tradeKeyword, eventClass, dsInit, engineName, exceptionType, leAttributeType, UploadMessageFormatTypes, and workflowRuleTrade.

  ▶ See the "Data Model Synchronization" section.

- Access Permissions - Set the required Access Permissions for the Calypso Mapping window.

  ▶ See the "Access Permissions" section.

- Trade Keywords - Use the Domain Values application to add any additional Trade Keywords required by your implementation.

  ▶ See the "Trade Keywords" section.

- Engine Configuration – Configure the SwapswireTradeEngine.

  ▶ See the "Engine Configuration" section.

- Legal Entities - Set the **SwapswireParticipant** LE Attribute for each MarkitWire counterparty.

  ▶ See the "Legal Entities" section.

- Books - Map the MarkitWire Book to the Calypso Book.

  The SwapswireTradeEngine only traps messages for mapped books.

  ▶ See the "Books" section.

- Broker

  ▶ See the "Broker" section.

- Reference Bank

  ▶ See the "Reference Bank" section.

- Workflow Configuration - Setup the **SWValidateUpdate** TradeRule and **SWErrorUpdate** TradeRule in the PSEventTrade Event Class.

  ▶ See the "Workflow Configuration" section.

MarkitWire to Calypso Mapping Checklist

Mappings using the Calypso Mapping Application:

- Fee Types - Cancellation, Cash Exercise, Premium, swBrokerageAmount, swSalesCredit, and PartialTermination.
- Holiday Codes - Two-letter Country/City codes to Calypso Holiday code.
- Exercise and Cash Settlement - Settlement Location and Time Zone.
- Reference (Rate) Indices - Note the difference between the Calypso separator (~) and the MarkitWire separator (-).
- Trader - Swapswire Trader name to the Calypso Trader name.
- Workflow Actions - Workflow actions are performed by the Data Uploader. Refer to the “Installation and Configuration” section of the Calypso Data Uploader Integration Guide.

2.3 MarkitWire API version

This section will describe the enhancements that are added in the API version for Markitwire.

MarkitWire API version 21.2.2

- Added support for the keyword for capturing the value for “Cancellation with Forward Premium” flag: <swForwardPremium>true</swForwardPremium>

MarkitWire API version 21.2.1

- Added support for Initial and Variation margin collateral portfolio code as trade keyword in incoming and outgoing mode

MarkitWire API version 21.2.0

- Added support for CZK CZEONIA (transition from PRIBOR to CZEONIA)
- Addition of ICESWAP Rate to EUR RFR Swaption template

MarkitWire API version 21.1.1

- Added support for the SAR (Saudi Arabian Riyal) currency on the IRS product.
- Added support for the FRO ‘ZAR-ZARONIA-OIS Compound’ on RFR swaption and RFR Cap Floor products.
- Added support for new ‘AMV’ (Australian Market Venue) Execution Venue type.

MarkitWire API version 21.1.0

- Added support for the AED (United Arab Emirates Dirham) currency on IRS and FRA products
- Added support for the FRO 'MXN-TIIE ON-OIS Compound' on RFR Swaption and RFR CapFloor products.

MarkitWire API version 21.0.2

- Added support 'Front and Back' Stubs on Fixed Fixed Cross Currency Swap
- Cross Currency Swap product enhancements.

MarkitWire API version 20.2.1 and 21.0

- Added support for Novation workflow on CapFloor.
- Added support for new MarkitWire API zip.

MarkitWire API version 20.2.1 and 21.0

- Added support for Amortizing/Accreting functionality on CapFloor: The CapFloor template will be enhanced with an 'Amortizing/Accreting' tab in order to support variable notional and strike rate (i.e. cap/floor rate) schedules.
- Added support for the following unilateral Counterparty Data fields which are required to be included in the trade state reports to ESMA:
  - Nature of Counterparty
  - Corporate Sector
  - Clearing Threshold
  - Reporting Obligation
  - Added support for implementing a new optional unilateral field to capture the Custom Basket Code.
    - The Custom Basket Code submitted on the trade will be carried through the lifecycle of the trade.
    - Where a Custom Basket has been provided for the trade, it will be included in the trade state reports submitted for ESMA where applicable.

MarkitWire API version 20.2

- Added support for sending value of ClearedTradeUPI for CCP mode as part of accept acknowledgements.
- Added support for capturing the keyword value ClearedTradeUPI on cleared Swaps with value provided by CCP as part of Clearing.
- Added support for capturing Underlying Swap UPI as a keyword and also able to allege for Swaption trade.
- Added support of Discrepancy Clause Details on Cancellable IRS and OIS in incoming and outgoing mode.

- MarkitWire has enhanced the Cancellable IRS and OIS template to capture two additional fields on the MarkitWire GUI.

  Discrepancy Clause Title - It refers to bespoke Discrepancy Clause language which both parties have previously established and agreed upon in the DMS portal on a LE to LE basis. It can then be incorporated by reference on a per transaction basis in MarkitWire. This language needs to be approved by both parties in DMS. The MarkitWire transaction screen field would refer to a title/name given to the clause itself.

  Agreement Date – It refers to the date on which the Discrepancy Clause language was agreed to by both parties. This field would be static driven and read only on the transaction screen.

- Additionally, users will be able to view the Discrepancy Clause Language in the GUI by selecting the "View Clause" button. Please note the full clause language itself will not be output in the SWML or Deal Ticket.

MarkitWire API version 20.1

- Addition of 'ILS-SHIR-OIS Compound' and 'ILS-SHIR' FROs
- Addition of 'MYR-MYOR-OIS Compound' and 'MYR-MYOR' FROs
- EMTA-ISDA Market Practice on Incorporation of SOFR as a Risk-Free Rate into Latin American Currency Non-Deliverable Cross Currency Swap Transactions – GUI Improvement.

MarkitWire API version 20.0.2

- Addition of "RUB-RUONIA-OIS Compound" and "RUB-RUONIA" FRO support to OIS, Cross Currency IRS and Cross Currency Basis Swap
- Support for different initial notional on the fixed and floating legs on OIS and IRS.

MarkitWire API version 20.0.1

- Addition of "USD-SOFR CME Term" FRO to USD Cap Floor & Swaption - Handled via Calypso Mapping.
- Addition of PLN-WIRON and PLN-WIRON-OIS Compound FROs - Handled via Calypso Mapping.
- Updates to certain DKK transaction defaults and availability of DESTR as a specified discount rate - MW GUI change.
- OIS trades with an invalid IMM End Date will be restricted - MW GUI change.
- CCP Libor Transition - Added TransitionFrom keyword to link original trade id on new migrated trade.

MarkitWire API version 20.0

- Swaption Physical Exercise: Ability to elect to exercise USD LIBOR Swaption into SOFR OIS.
- Swaption Physical Exercise: Change in how Underlying Swap Trade Date is calculated.

- Support Negative Fixed Amount on ZC IRS, ZC OIS and ZC Inflation Swap

MarkitWire API version 19.2.1

- MarkitWire has increased the decimal place precision support for the following fields: Fixed Rate, Spread over floating, Init 1st Fixing Rate, Strike (cap/floor), Reportable Price keyword.
- Added support for ReportingToTV keyword in incoming and outgoing mode.
- Added support for separate UTI for Cap / Floor leg of CapFloor Straddle and Payer/Receiver leg for Swaption Straddle

MarkitWire API version 19.2

- Extended support for overnight RFR averaging/compounding to Cross-Currency Basis Swap and Cross-Currency IRS product templates.

MarkitWire API version 19.1.1

- Ability to set Break defaults for Cash Settlement method and Prescribed Documentation adjustment.
- Deprecation of 2m and 9m DKK CIBOR tenors.
- Allow Auto Send for Clearing on a Swaption Exercise.

MarkitWire API version 19.1

- Compounding and Averaging Provisions on Overnight FRO
- Danish Krone IBOR Transition (Upcoming changes to CIBOR, Tom/Next, DESTR)
- Norwegian Krone (NOK) CapFloor support expansion

MarkitWire API version 19.0.

- MarkitWire will be enhancing the Cancellable Option tab for IRS and OIS. The proposed changes are the following:

  Change GUI naming convention for First Cancellation to Optional Early Termination Date - No change in Calypso.

  Change GUI naming convention for Exercise Lag to Expiration Date - No change in Calypso.

  Allow users to edit holiday centers related to the Cancellable/Callable provision - No change in Calypso

  This field is currently not editable and linked to float roll holiday centers

  Add support for Calculation Agent field - We need to add support for this in our interface

For D2D, the default will be "As Specified in Master Agreement"

For D2C the dealer role in the transaction will default as the Calc Agent (i.e., Buyer or Seller based on Dealer Buy/Sell of the IRS/OIS)

- The values "MYOR" and "SWESTR" are being added as available choices in the Cash Collateral Interest Rate and Agreed Discount Rate fields on the Break(2) tab for OETs and METs. Additionally, as available choices in the Swaption Agreed Discount Rate field. We would not need any changes for these, and these will be stored in Calypso as received.

MarkitWire API version 18.2.2

- SOFR First Phase 3 - Holiday defaults to change in MarkitWire UI from Option Exercise tab.
- Changes to USD SOFR Swaptions - ICE Swap Rate to be supported for Collateralized Cash Price and Cleared Physical Settlement.
- Changes to JPY TONA Swaptions - Tokyo Swap Reference Rate to be renamed as Tokyo Swap Rate (MarkitWire UI only change and has no impact on the incoming or outgoing messages).
- Fallbacks Mechanism Unilateral Field: New unilateral Fallback field is introduced to MarkitWire to allow firms to capture which distinct Fallback mechanism each non-cleared MarkitWire trade is reliant upon in the processing tab having values - ISDA/Bespoke. We will be adding a trade keyword for this field.

MarkitWire API version 18.0.

- Added support for Risk-Free Rates on Swaptions incoming and outgoing mode.
- Added support for different Payment Lags for Cross-Currency Basis Swaps (e.g. SONIA vs SOFR, EuroSTR vs SOFR).
- Added support for “ICESWAP Rate” as Settlement rate source for Swaptions.
- Fixed issues with Bidirectional Allege for Cash Settled Swaptions for settlement methods - "Collateralized Cash Price" and "Par Yield Curve - Unadj".

2.4 Integration Setup

2.4.1 Environment Properties

Swapswire Dealsink Password

When the SwapswireTradeEngine starts, it checks the user and system properties file for plain text passwords (40 characters, max.), and if found, it then encrypts the password(s). The **AutoEncryptPassword** flag in the `calypso_SW_config.properties` controls the operation of the auto-encryption feature. When set to true, the system auto-encrypts plain text passwords. If the properties file is not read-only and if the user who launched the SwapswireTradeEngine also has write permissions to the user/system properties file, then Calypso will store the password in the encrypted form. If the user who launched the SwapswireReferenceServer does not have write access to the user/system properties file, a warning is logged, and the process continues without encryption.

When **AutoEncryptPassword** is set to false, the password is left as plain text.

Clients can, if desired, encrypt the Dealsink password held in the properties file using a client-written utility that makes use of Calypso's encryption module. Refer to the Security User Guide and to the Calypso Developer's Guide for further information on working with the Calypso Encryption API.

Single Dealsink User

Start the Edit User Env or System Env application and add the following Environment Properties:

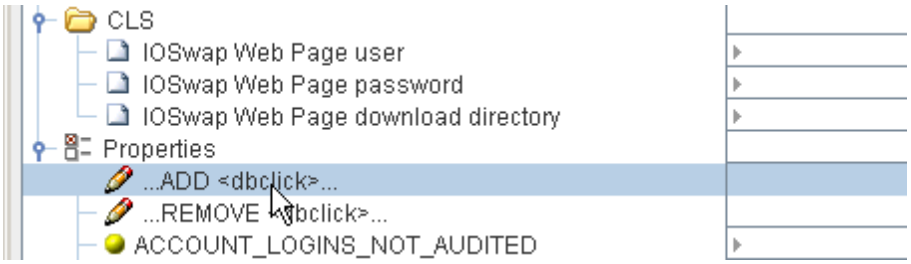
- SWAPSWIRE_CONCURRENT_LOGIN_NO= <Number of dealsink logins>
- SWAPSWIRE_LOGIN_ATTEMPTS = 1 (suggested)
- SWAPSWIRE_LOGIN_INTERVAL = 10000 (suggested)
- SWAPSWIRE_PASSWORD = <your dealsink login password>
- SWAPSWIRE_SERVER = <IP address of Swapswire server>
- SWAPSWIRE_TIMEOUT = 60000 (suggested)
- SWAPSWIRE_USER = <your dealsink login>

Multiple Dealsink Users

Certain implementations may necessitate the organization listening to the output of more than a single Swapswire Dealsink. In this situation, you can define additional Swapswire Users to capture trade output from more than a single book.

You can use a text editor to manually create the additional user/password pairs and to modify the value of **SWAPSWIRE_CONCURRENT_LOGIN_NO** in your Calypso User Properties file or use the User Env application. The following example uses the User Env application.

Step 1 - Open User Env and double click **Add**:



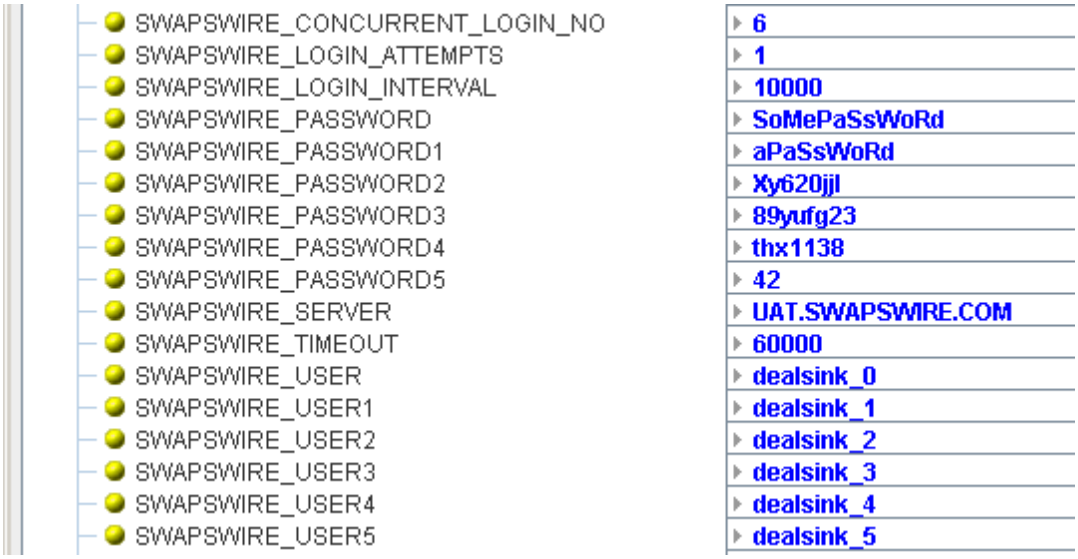
Step 2 - Enter **SWAPSWIRE_USER1**:

Step 3 - Repeat Steps 1 and 2 to add SWAPSWIRE_PASSWORD1.

Repeat this procedure to add the required number of user/password properties, incrementing the trailing number for each pair.

When you have added the number of additional users required for your implementation, then update SWAPSWIRE_CONCURRENT_LOGIN_NO to the total number of concurrent users.

The image below shows the Edit User Env application with six concurrent Dealsink listeners.



Remember that each SWAPSWIRE_USER listens to a single Dealsink. In the Book Window, set the Swapswirebook attribute for the Calypso book that should receive the trades. That procedure is covered in the "Books" section. You must not create multiple mappings to a single Swapswire book.

Dealsink Users' pool

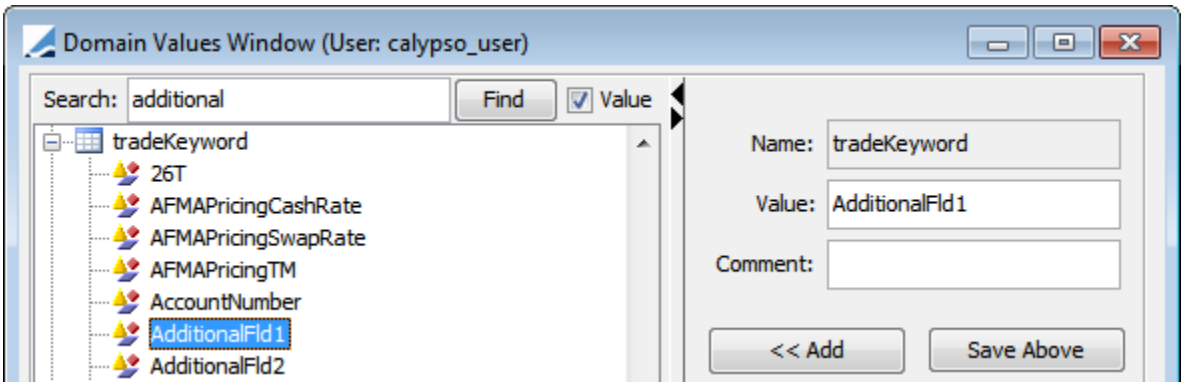
The SwapswireTradeEngine can be configured such that multiple dealsink users can be used simultaneously, but each active dealsink user may have a specific role. For instance, certain master dealsink users can be configured to only listen to deal notification callbacks and do-recovery (Master dealsink user), and some other dealsink users will be used to send requests to fetch deal XMLs and sending acknowledgements (Pooled dealsink user). There can be

one-to-many relationship between Master dealsink users and Pooled dealsink users; there is no limit to how many pooled dealsink users can be configured for a given master dealsink user. Given that a dealsink user can be used to send only one request at a time, having multiple pooled dealsink users will enable the SwapswireTradeEngine to send multiple requests to the swapswire server in parallel.

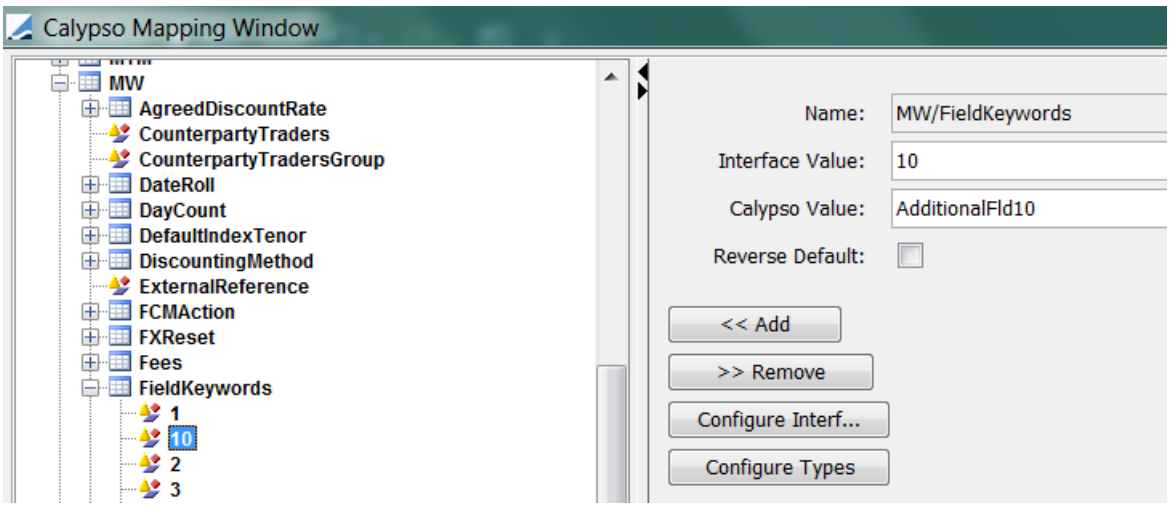
The SwapswireTradeEngine will be configured in dealsink users' pooling mode (hereafter referred to as pooling mode) if and only if the file calypso_sw_dealsink_config.xml is found in the Engine Server's classpath. If the SwapswireTradeEngine is configured in pooling mode it will ignore all Swapswire-related properties in Environment Properties (mentioned in 'Single Dealsink User' and 'Multiple Dealsink Users above.) For further details on the pooling mode refer to section 'calypso_sw_dealsink_config.xml Settings'.

Applicable to All Configurations: Importing Additional Fields from MarkitWire

Users can import additional trade keywords from MarkitWire using **AdditionalFld**x. To configure this feature, ensure that the tradeKeywords domain contains AdditionalFld1 through AdditionalFldx (default is 1-5):



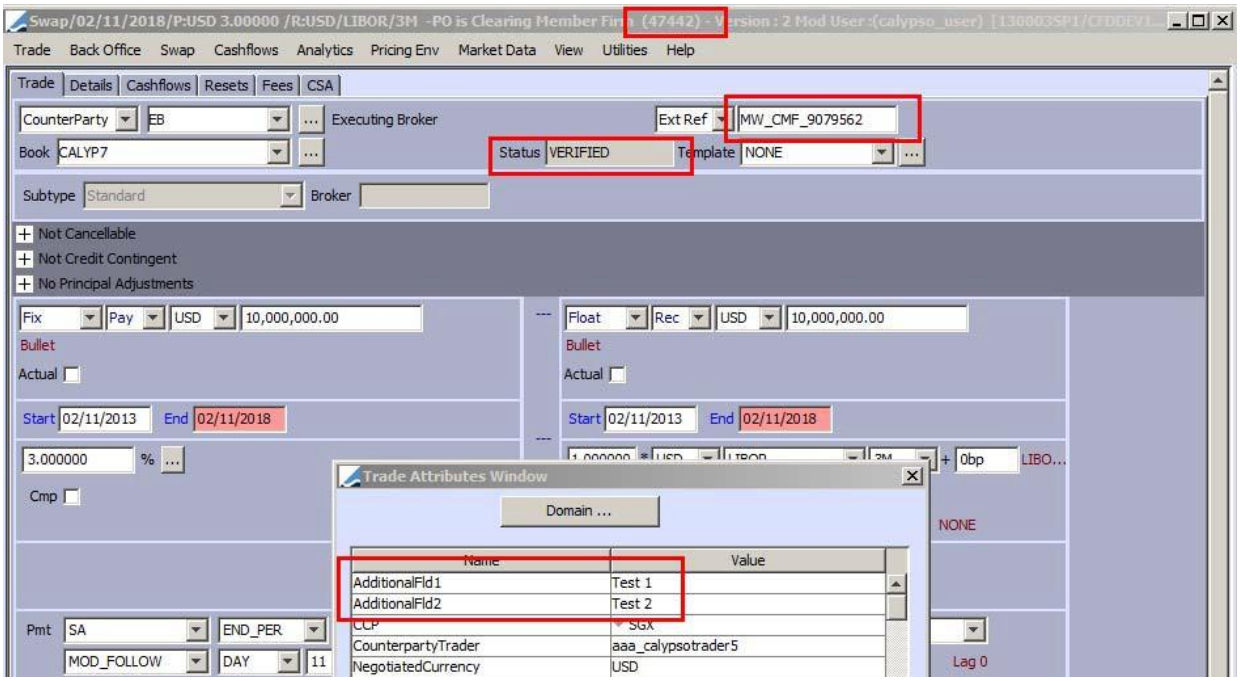
Next, using the Calypso Mapping window, ensure that the fields 1–x are included under the MW interface as FieldKeywords:



The Calypso Value should be the name of the trade keyword.

The MarkitWire interface passes Additional Fields to Calypso as fields 1–x. This mapping assigns them to the AdditionalFld1–x trade keywords.

These trade keywords contain the values imported from the **Additional Fields** on the **Internal** tab of the MarkitWire MarkitSERV application.



To obtain specific data from MarkitWire and apply them to the trade imported into Calypso, you need to set environment properties for each MarkitWire field and set their value to the Interface Value of the FieldKeywords.

Example:

- MW_EXT_REF_FLD - External Reference in Calypso. Controls which of the five MarkitWire Additional Fields will store the Calypso External Trade Reference. If this environment property is not defined, the application uses AdditionalField4.
  - If there is a conflict, i.e., if an Additional Field mapped to a trade keyword and the same field is also used to store the ExternalKey, then the application ignores that keyword mapping. An Additional Field used to store the ExternalKey always takes precedence over a keyword mapping.
  - In the event that you currently map AdditionalField4 to a trade keyword, please set MW_EXT_REF_FLD environment property to a different Additional Field value.
- MW_STATUS_FLD - Trade Status in Calypso (populated when SWStatusUpdate rule is present in the prior trade workflow transition), stored in AdditionalField3 in this example.
- MW_TRADE_ID_FLD - Trade ID in Calypso, stored in AdditionalField5 in this example.

2.4.2 "calypso_SW_config.properties" Settings

Copy the file "<calypso home>/client/resources/calypso_SW_config.properties.sample" to "<calypso home>/tools/calypso-templates/resources", and remove the ".sample" extension.

You will then need to deploy the files to your applications servers.

▶ Please refer to the Calypso Installation Guide for details.

Session Timeout

The MarkitWire API requires a session timeout parameter while establishing connection with MarkitWire. This timeout is configurable via a setting in the calypso_SW_config.properties file.

```
#Timeout for MW connection
session_timeout=360
```

If not specified, the application defaults to 360.

Upload mode

The MarkitWire interface supports the Local upload mode and the BOMessage* upload mode (*for CPP Clearing only). The following properties need to be configured:

```
uploadMode= [<Blank>, Local, BOMessage*.
persistMessages= [<Blank>, Failure, All.
```

If the properties are not configured, the default upload mode is Local.

Please refer the Calypso Data Uploader documentation for more information on the Local upload mode.

The MarkitWire interface allows to delay the notification callback event creation. The following properties need to be configured:

```
#value in milliseconds.
EventCreationLag=100
```

The MarkitWire interface allows to delay the clearing notification callback event creation. The following properties need to be configured:

```
#value in milliseconds.
NewClearingEventCreationLag=100
```

The MarkitWire interface allows to delay the reconnect post disconnection of session. The following properties need to be configured:

```
#value in milliseconds.
```

```
delayFromDisconnectToReconnect=200
```

Custom SWML Debug File Location

You can customize the location of the SWML debug file using the following properties.

- logMessages = true/false. If set to true (default), debug files are generated, else no file is generated.
- messageFileDir = <user defined location>. If messageFileDir is not provided then the debug files are generated in <user home>/Calypso/markitwire/ by default.

Message Processing Delay

There is a default delay of 1 second when processing messages coming from MW. You can configure the delay using the property EventCreationLag.

2.4.3 "calypso_sw_dealsink_config.xml" Settings

Copy the file "`<calypso home>/client/resources/calypso_sw_dealsink_config.xml.sample`" to "`<calypso home>/tools/calypso-templates/resources`", and remove the ".sample" extension.

You will then need to deploy the files to your applications servers.

▶ Please refer to the Calypso Installation Guide for details.

Following is a sample of what a "calypso_sw_dealsink_config.xml" could look like. All values shown here are only for the purposes of illustration:

```
<?xml version="1.0" encoding="UTF-8"?>
<calypso-sw-dealsink-pool-config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <sw-server>https://mw.uat.api.markit.com</sw-server>
    <sw-login-attempts>1</sw-login-attempts>
    <sw-login-interval>10000</sw-login-interval>
    <sw-reconnect-attempts>1</sw-reconnect-attempts>
    <sw-reconnect-interval>10000</sw-reconnect-interval>
    <sw-api-init-file></sw-api-init-file>
    <calypso-sw-dealsink-pool>
        <master-dealsink-user bidir="true">master_dealsink1</master-dealsink-user>
        <master-dealsink-password>password</master-dealsink-password>
        <dealsink>
            <user>pool_dealsink1</user>
            <password>password</password>
        </dealsink>
        <dealsink>
            <user>pool_dealsink2</user>
            <password>password</password>
        </dealsink>
        <dealsink>
            <user>pool_dealsink3</user>
            <password>password</password>
        </dealsink>
    </calypso-sw-dealsink-pool>
    <calypso-sw-dealsink-pool>
        <master-dealsink-user bidir="false">master_dealsink2</master-dealsink-user>
        <master-dealsink-password>password</master-dealsink-password>
        <dealsink>
            <user>pool_dealsink4</user>
            <password>password</password>
        </dealsink>
        <dealsink>
            <user>pool_dealsink5</user>
            <password>password</password>
        </dealsink>
        <dealsink>
            <user>pool_dealsink6</user>
            <password>password</password>
        </dealsink>
        <dealsink>
            <user>pool_dealsink7</user>
            <password>password</password>
        </dealsink>
    </calypso-sw-dealsink-pool>
</calypso-sw-dealsink-pool-config>
```

Following is a description of all tags and attributes and tags in “calypso_sw_dealsink_config.xml”:

Sr. No.	Tag name	Description	Recommended value
1	calypso-sw-dealsink-pool-config	Indicates the start and end of the XML document.	NA
2	sw-server	IP address of the Swapswire server.	NA
3	sw-login-attempts	The number of re-attempts the SwapswireTradeEngine will make to login for any given login credential after a failed first attempt to login. (The value shown in tag is the recommended value.)	1
4	sw-login-interval	The time interval in milliseconds after which the SwapswireTradeEngine will attempt to login, for any given login credential, after a failed previous attempt. (The value shown in tag is the recommended value.)	10000
5	sw-reconnect-attempts	The number of re-attempts the SwapswireTradeEngine will make to connect after a failed first attempt to connect to the Swapswire server. (The value shown in tag is the recommended value.)	1
6	sw-reconnect-interval	The time interval in milliseconds after which the SwapswireTradeEngine will attempt to connect after a failed previous attempt to connect to the Swapswire server. (The value shown in tag is the recommended value.)	10000
7	sw-api-init-file	The location of the sw_client_api.ini	NA
8	calypso-sw-dealsink-pool	These tags can be used to specify a pool of dealsink users. There can be multiple such pools; each pool can have exactly one master dealsink user and multiple dealsink users.	NA
9	master-dealsink-user (attribute: bidir="true" / ”false”)	The Swapswire login id for master dealsink user. The 'bidir' attribute can be used to specify whether do-recovery is to be carried out for the master dealsink user(true) or not(false). The attribute is not mandatory and defaults to false is not specified.	NA
10	master-dealsink-password	The password for the master dealsink user.	NA

Sr. No.	Tag name	Description	Recommended value
11	dealsink	These tags are used to specify the pooled dealsink user.	NA
12	user	The Swapswire login id for pooled dealsink user.	NA

2.4.4 Data Model Synchronization

When you run Execute SQL as part of your installation, the Markitwire files will be already loaded. You just need to check the “markitwire” checkbox.

This procedure creates the mapping table and provides the required mappings between MarkitWire and Calypso values. See the “Data Mapping Between MarkitWire and Calypso” section.

Populated Domains

The following domains are populated during synchronization:

Domain	Populated Values
tradeKeyword	BlockUSIPrefix, BlockUSIValue, CCP, CCPAccount, CCPClearedDate, CCPClearedVer, CCPClearingBroker, CCPClientTradeType, CCPFCM, CCPFund, CCPGroupId, CCPMessageTimeStamp, CCPOriginCode, CCPRejectReason, CCPStatus, CCPTradeID, ClientTradeID, DeliveryDate, DeliveryDateConvention, DeliveryDateHolidays, InitialMarginAmount, InitialMarginCcy, InitialMarginType, IS_CLIENT, LinkedTo, Platform, PlatformCP, PlatformPO, PlatformTradeId, PriorUSIPrefix, PriorUSIValue, ReportingBlkEventID, ReportingClearingException, ReportingClearingMandatory, ReportingCollateralized, ReportingConfirm, ReportingEventID, ReportingExecutionTime, ReportingJurisdiction, ReportingLocationBroker, ReportingLocationDesk, ReportingLocationsSales, ReportingLocationsTrader, ReportingNonStandard, ReportingOffPlatform, ReportingParty, ReportingPET, ReportingPostTrade, ReportingPriceFormingVenue, ReportingRegulatoryReportable, ReportingRT, ReportingToTV, ReportingVenue, SWAmendmentType, SWAutoSendForClearing, SWBookState, SWBrokerTradeId, SWCancellationType, SWClearingStatus, SWClientClearingDeal, SWContractState, SWContractualDefinitions, SWContractVer, SWDealId, SWEligibleForClearing, SWExitReason, SWPBGiveupDealID, SWGiveUpTradeId, SWLoginHandleIdentifier, SWMasterAgreementType, SWPBMirrorDealID, SWModificationEffectiveDate, SWModificationTradeDate, SWOriginalCounterparty, SWOriginalDealType,

Domain	Populated Values
	SWOriginalPayStartDate, SWOriginalRecStartDate, SWPrivateVer, SWProcessState, SWSendForClearing, SWSendForClearingTimestamp, SWSide, SWSingleSided, SWValidated, TradeSource, USIPrefix, USIValue
engineEventPoolPolicyAliases	SwapswireTradeEngine
eventClass	PSEventSwapswire, PSEventSwapswireScheduledTask, PSEventRepublish
eventType	EX_MWGATEWAYMSG, EX_MWGATEWAYMSG_ERROR, EX_MWGATEWAYMSG_REJECT, EX_MWGATEWAYMSG_WARNING
exceptionType	MWGATEWAYMSG, MWGATEWAYMSG_ERROR, MWGATEWAYMSG_REJECT, MWGATEWAYMSG_WARNING
function	ModifyMWMappings, ViewMWMappings
engineEventPoolPolicies	tk.util.SwapswireTradeEngineSequencePolicy
engineEventPoolPolicyAliases	SwapswireTradeEngine
engineName	SwapswireTradeEngine
eventFilter	SwapswireTradeEngineEventFilter
leAttributeType	CCPClientBook, CCPHouseBook, SwapswireParticipant, SwapswireBroker, SwapswireLongName, SwapswireParent,
MsgAttributes	CCPRejectReason, Rejected
messageType	MWGATEWAYMSG
MWUploadMessageType	MWGATEWAYMSG
scheduledTask	MW_MIGRATE, SW_DO_RECOVERY
UploadMessageFormatTypes	MW
workflowRuleTrade	ApplyLinkedTradeAction, ChangeFullCoupon, DeclearAction, MWExit, SWErrorUpdate, SWStatusUpdate, SWValidateUpdate, UpdateLinkedToKeyword, UpdateAllocationChild
workflowRuleMessage	ApplyLinkedMsgAction, ForceAmend, ForceNew, ReMap, UpdateLinkedToAttribute

Unpopulated Domains

The following Domains are created, but not populated:

Domain	Populated Values
MWEnforceEffectiveDate	For enabling MarkitWire amend validation.
MWContractState.PreRelease	Specifies the new Contract state filter for the SwapswireTradeEngine.
MWExitKeywords	
MWProcessState	Specifies the new State Filter from MarkitWire.
MWMigrateTradeStatus	Domain name for all trade statuses to migrate from the old to new version of interface.

Inserted Domain Data

The Book attribute **SwapswireBook** is inserted.

2.4.5 Access Permissions

Required Access Permissions for using the Calypso Mapping window are found in the "Calypso Mapping" section.

▶ Refer to the Security User Guide or to the **Working with Group Access Permissions** topic in the User Access section of the Calypso Help System for instructions on enabling functions

2.4.6 Trade Keywords

MarkitWire Field	Calypso Keyword Name	Description
See note, below.	ReportingJurisdiction	If flagged, keyword is set to 'CFTC'
See note, below.	ReportingParty	Calypso keyword is set to the short name of an existing Legal Entity in Calypso
See note, below.	ReportingPriceForming	True/False
See note, below.	ReportingOffPlatform	Blank/Electronic/NonElectronic
See note, below.	ReportingExecutionTime	2012-10-23T18:01:29Z
See note, below.	ReportingVenue	Blank/SEF/DCM/Off-Facility
See note, below.	ReportingRT	No/YesIfRCP
See note, below.	ReportingPET	No/Yes/YesIfDF/YesIfRCP

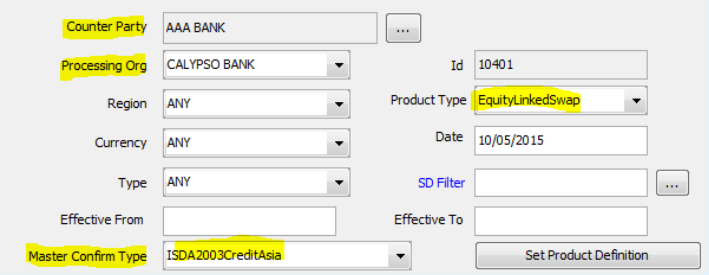
MarkitWire Field	Calypso Keyword Name	Description
See note, below.	ReportingConfirm	No/Yes/YesIfDF.YesIfRCP
See note, below.	ReportingPostTrade	No/Yes/YesIfDF.YesIfRCP
See note, below.	USIPrefix and USIValue	Existing keyword
See note, below.	PriorUSIPrefix, PriorUSIValue	Existing keyword
See note, below.	BlockUSIPrefix, BlockUSIValue	Existing keyword
See note, below.	ReportingLocationBroker	3-digit country code
See note, below.	ReportingLocationDesk	3-digit country code
See note, below.	ReportingLocationsTrader	3-digit country code
See note, below.	ReportingLocationsSales	3-digit country code
See note, below.	ReportingEventID	ID number
See note, below.	ReportingBlkEventID	ID number
See note, below.	ReportingCollateralized	Blank/Fully/One-Way/Partially/Uncollateralized
See note, below.	ReportingNonStandard	True/False
See note, below.	ReportingClearingMandatory	True/False
See note, below.	ReportingClearingException	True/False
See note, below.	ReportingRegulatoryReportable	True/False
See note, below.	ReportingToTV	Yes/No

Full List of MarkitWire-Related Calypso Trade Keywords

Trade Keyword	Description
BlockUSIPrefix	
BlockUSIValue	
CCP (clearing)	Short code for Clearing House Legal Entity. Populated with the Clearing House name on trade inception if

Trade Keyword	Description
	"Exclude From Clearing" on the Trader/Tracker Clearing tab.
CCPAccount (clearing)	When the trade is against the LCH House Account, CCPAccount = House. When the trade is against the LCH Client Account, CCPAccount = Client.
CCPClearedDate (clearing)	The date the trade is registered with the clearing house.
CCPClearedVer (clearing)	Populated for CCP role only to save the cleared version of the trade.
CCPClearingBroker (clearing)	The clearing broker (when available in the trade).
CCPClientTradeType (clearing)	If the cleared trade originates directly from MarkitWire, CCPClientTradeType = Primary. If the cleared trade is a clone, CCPClientTradeType = Secondary.
CCPExecutionSource	
CCPFCM (clearing)	This trade keyword defines whether or not the ClearingBroker is FCM or not (true | false).
CCPFund (clearing)	Populated when acting as CCP in a client clearing trade. Stores the client fund of the trade.
CCPGroupId (clearing)	Populated for CCP role to save the group Id of the parent trade.
CCPMessageTimeStamp (clearing)	Time stamp of last clearing message.
CCPOriginCode (clearing)	Set to HOUSE for Direct trades, and CLIENT for Client Clearing trades.
CCPRejectReason (clearing)	To send MarkitWire a send Clear Reject / Declear Reject notification on trade transition, populate the rejection reason on the **CCPRejectReason** Trade Keyword. Calypso sends this reject reason MarkitWire in the Reject notification.
CCPStatus (clearing)	Clearing status of trade sent for clearing.
CCPTradeID (clearing)	The clearing Trade ID registered with clearing house.
ClientTradeID (clearing)	

Trade Keyword	Description
CounterpartyRejectReason	Rejection Reason entered by the counterparty in response to a new trade or trade amendment initiated by the Processing Org.
BlockUSIPrefix	
BlockUSIValue	
CCP (clearing)	Short code for Clearing House Legal Entity. Populated with the Clearing House name on trade inception if “Exclude From Clearing” on the Trader/Tracker Clearing tab.
CCPAccount (clearing)	When the trade is against the LCH House Account, CCPAccount = House. When the trade is against the LCH Client Account, CCPAccount = Client.
CCPClearedDate (clearing)	The date the trade is registered with the clearing house.
DeliveryDate	These keywords map to the MarkitWire Independent Amount tab. These fields may become a Fee in a future version of Calypso’s MarkitWire Integration. The fields are accessible via the Swapswire Mapping Window.
DeliveryDateConvention	
DeliveryDateHolidays	
InitialMarginAmount	
InitialMarginCcy	
InitialMarginType	
IS_CLIENT (clearing)	Set to true if trade is related to client activity or false otherwise.
LinkedTo (clearing)	After applying UpdateLinkedToKeywordTradeRule in an initial trade transition, Calypso searches for other trades having same SWDealId keyword value and the same SWContractVer keyword value, then adds the Trade ID of that trade as a value of the LinkedTo keyword. Calypso also adds the LinkedTo keyword on the other trade, as well. This links two trades having same SWDealId keyword value and same SWContractVer keyword value.
MasterConfirmationDate	Date of associated master confirmation for outgoing Equity Linked Swaps – Format yyyy-mm-dd.
MasterConfirmationType	Type of associated master confirmation for outgoing Equity Linked Swaps.

Trade Keyword	Description
	Example: 
Platform (clearing)	Execution Source Identifier. When trades are SEF executed, MarkitWire now passes the Execution Source in this keyword in the Clearing XML. Replaces CCPExecutionSource from previous versions.
PlatformCP	Original CounterParty in MarkitWire.
PlatformPO	Original Processing Org party in MarkitWire.
PlatformRejectReason	Rejection Reason by the Processing Org: In bidirectional mode, this keyword must be populated prior to submitting a rejection to a new trade or a trade amendment initiated by the counterparty.
PlatformTradeId	
PriorUSIPrefix	Prior Unique Swap Identifier prefix (after a USI change).
PriorUSIValue	Prior Unique Swap Identifier suffix (after a USI change).
ReportingBlkEventID	ID number
ReportingClearingException	True/False
ReportingClearingMandatory	True/False
ReportingCollateralized	Blank/Fully/One-Way/Partially/Uncollateralized
ReportingConfirm	No/Yes/YesIfDF.YesIfRCP
ReportingEventID	ID number
ReportingExecutionTime	2012-10-23T18:01:29Z
ReportingJurisdiction	If flagged, keyword is set to 'CFTC'
ReportingLocationBroker	3-digit country code

Trade Keyword	Description
ReportingLocationDesk	3-digit country code
ReportingLocationsSales	3-digit country code
ReportingLocationsTrader	3-digit country code
ReportingNonStandard	True/False
ReportingOffPlatform	Blank/Electronic/NonElectronic
ReportingParty	True or false depending on whether the Processing Org reports to the trade repository. (In 3.0.4 and below, this Trade Keyword was ReportingCounterparty.)
ReportingParty	Calypso keyword is set to the short name of an existing Legal Entity in Calypso.
ReportingPET	No/Yes/YesIfDF/YesIfRCP
ReportingPostTrade	No/Yes/YesIfDF.YesIfRCP
ReportingPriceForming	True/False
ReportingRegulatoryReportable	True/False
ReportingRT	No/YesIfRCP
ReportingVenue	Blank/SEF/DCM/Off-Facility
SWAmendmentType	Stores the provided amendment type. (Dodd-Frank Compliance)
SWAutoSendForClearing	Yes/No depending upon “Auto Send For Clearing” checkbox in MarkitWire.
SWBookState	
SWBrokerTradeId	
SWCancellationType	Stores the provided cancellation type. (Dodd-Frank Compliance)
SWClearingStatus	The reason for rejection by the clearing house.
SWClientClearingDeal	
SWContractState	Contract of trade in MarkitWire.

Trade Keyword	Description
SWContractualDefinitions	The value of the Definition field from the Processing tab in MarkitWire.
SWContractVer	Swapswire contract version
SWCorrelationID	Trade Pair Id at CCP
SWDealId	Swapswire Deal ID; the existence of this keyword can be used to determine whether this trade has been imported from Swapswire.
SWEligibleForClearing	Is the trade eligible for clearing?
SWExitReason	
SWGiveUpTradeId	
SWLoginHandlerIdentifier	Stores the Deal Sink ID used to retrieve the trade. This field is used in lieu of SWLoginHandle (which is not needed).
SWMasterAgreementType	The value of the Master Agreement field.
SWModificationEffectiveDate	Stores the Modification Effective Date from MarkitWire.
SWModificationTradeDate	Stores the Modification Trade Date from MarkitWire.
SWOriginalCounterparty	Shows the original counterparty on a cleared trade.
SWOriginalDealType	Populated for the Executing Broker role when upgrading a trade from bilateral to trilateral, with values “Bilateral” or Trilateral”.
SWOriginalPayStartDate	Original Paying Leg Start Date in MarkitWire. Populated when an adjustment occurs on the Paying Leg Start Date.
SWOriginalRecStartDate	Original Receiving Leg Start Date in MarkitWire. Populated when an adjustment occurs on the Receiving Leg Start Date.
SWPBGiveupDealID	
SWPBMirrorDealID	
SWPrivateVer	Swapswire private version.
SWProcessState	Stores the New State from MarkitWire

Trade Keyword	Description
SWSchedulingMethod	Must be set to ListDateEntry for outgoing Equity Linked Swaps.
SWSendForClearing	Was the trade sent for clearing?
SWSendForClearingTimeStamp	The date and time the trade was sent for clearing.
SWSide	Swapswire deal side.
SWSingleSided	Swapswire single-sided deal. (MWMigrateTradeStatus domain)
SWValidated	Swapswire trade is validated. (MWMigrateTradeStatus domain)
TradeSource	Platform Source. The domain “UploadMessageSourceTypes” should contain valid trade sources. If the trade keyword TradeSource is empty or invalid, you can use the trade workflow rule MWSource to set TradeSource = MW.
USIPrefix	Unique Swap Identifier - Namespace prefix.
USIValue	Unique Swap Identifier - Transaction Identifier suffix.
ConditionPrecedentBond	Cancellable Swap Bond availability
ConditionPrecedentBondCodeType	Cancellable Swap Bond code type (ISIN/CUSIP)
ConditionPrecedentBondCodeValue	Cancellable Swap Bond code value
ConditionPrecedentBondMaturityDate	Cancellable Swap Bond Maturity Date (yyyy-mm-dd)
DiscrepancyClause	Cancellable Swap Bond discrepancy clause
FollowUpConfirmation	Cancellable Swap Condition follow up confirmation
ESMAFrontloadingCategory	Frontloading Category values – Category1/Category2/None
ESMAClearingExemption	Indicating if exempt from ESMA mandatory clearing. Values - True/False
PricedToClearCCP	Field mapped to MW PricedToClearCCP dropdown. Contains a valid Calypso LE code mapped to MW CCP BIC code

Trade Keyword	Description
SWBulkAction	It is set in MW while Termination/Novation in processing tab. Values - True/False
PlatformBackload	Indicated whether backloading deal. Values – True/False

MIFID Keywords

Trade Keyword	Description
ReportingMIFIDCounterpartyType	Specifies the type of reporting counterparty as selected in MarkitWire. The type selected could be We/Cpty/Venue. In all cases there should be a legal entity in Calypso with the attribute SwapswireParticipant with the value of the ReportingCounterpary BIC code.
ReportingMIFIDPreferenceTransparencyReporting	The value will be as selected in MarkitWire.
ReportingMIFIDPreferenceTransactionReporting	The value will be as selected in MarkitWire.
ReportingMIFIDTransparencyReportable	The value will be as selected in MarkitWire.
ReportingMIFIDTransactionReportable	The value will be as selected in MarkitWire.
ReportingMIFIDDestinationAPA	Populated from static. APA - Approved Publication Arrangement
ReportingMIFIDDestinationARM	Populated from static. ARM - Approved Reporting Mechanism
ReportingMIFIDTransactionIdentifier	Generated or user submitted per transaction
ReportingMIFIDShortSaleIndicator	User submitted per transaction
ReportingMIFIDOTCPostTradeIndicator	Selectable flags submitted per transaction
ReportingMIFIDWaiverIndicator	Selectable flags submitted per transaction
ReportingInvestmentDecisionMaker	User submitted per transaction, can be defaulted based upon user logged in
ReportingInvestmentDecisionMakerLocation	One or more locations as selected in MarkitWire
ExecutionVenueMIC	MICs taken from static, selectable on a transaction

Trade Keyword	Description
InstrumentISIN	Derived from transaction details if possible, can be overridden on transaction
InstrumentCFI	We capture the value passed from MarkitWire.
InstrumentFullName	We capture the value passed from MarkitWire.
ReportingMIFIDJurisdiction	True if MIFID reporting is applicable
ReportingMIFIDCounterparty	Calypso Legal entity for the corresponding party selected in MarkitWire as the Reporting Party
ReportingMIFIDToTV	Set to true if selected in MarkitWire.
ReportingTradingCapacity	A default value for trading capacity, can be overridden on transaction
ReportingTraderName	Trade Name
ReportingTraderLocation	One or more locations as selected in MarkitWire
ReportingToTV	Set it to Yes if selected in MarkitWire

Order Details Keywords

Trade Keyword	Description
TypeOfOrder	As entered in MarkitWire
OrderTotalConsideration	As entered in MarkitWire
OrderRateOfExchange	As entered in MarkitWire
OrderClientCounterparty	As entered in MarkitWire
OrderTotalCommissionAndExpenses	As entered in MarkitWire
OrderClientSettlementResponsibilities	As entered in MarkitWire
OrderTransmission	Boolean. True if selected in MarkitWire
OrderBuyer	As entered in MarkitWire
OrderSeller	As entered in MarkitWire

2.4.7 Modification Effective Date and Modification Trade Date

MarkitWire V7.1 introduced two fields to support interoperability with the DSMatch system. The fields are Modification Effective Date, and Modification Trade Date, which are used for deal Amendments and Terminations. They provide a means to apply a modification to a trade that is effective from a specific date forward. For example, amending the notional amount one month into a 3-month swap. Calypso stores these fields as the trade keywords, SWModificationEffectiveDate and SWModificationTradeDate.

By default, amendments in Calypso affect the entire life of the deal. Therefore, Calypso's MarkitWire validation only allows an Amendment (or Exit) having a Modification Effective Date equal to or earlier than the Trade Start Date. The Amendment (except if caused by a Partial Termination) is then processed by the Calypso API.

In Calypso, the default behavior when receiving an amendment (not caused by a Termination or Novation) that has an Effective Date after the Start Date of the trade appears, is to log the Amendment as an error in the Task Station and cease processing.

However, by adding a domain, the user can configure Calypso to support MarkitWire's DSMatch interoperability (i.e., allow amendments whose Effective Date occurs after the Start Date). In addition, after configuring, any Amendments stuck in the exception queue can then be re-processed and will flow through without error. This also applies to the Exit action.

To enable support for DSMatch interoperability, use the Domain Value application to add MWEnforceEffectiveDate to the domainName domain. Set MWEnforceEffectiveDate to true. If MWEnforceEffectiveDate is missing or not set, the default Calypso behavior applies: The Modification Effective Dates must be on or before the trade Start Date.

2.4.8 Pre-Mapped Values

Data Synchronization adds the following values to the Calypso Mapping Table:

Columns

- Daycount
- Fees
- FieldKeywords
- FXReset
- Holidays
- IndexTenor
- Location
- MWBookingState
- RateIndex
- TimeZone
- Traders
- ProductType

Daycounts

Interface Value	Calypso Value
30E/360.ISDA	30E*/360
ACT/365.FIXED	ACT/365
ACT/365.ISDA	ACT/365
ACT/365L	ACT/365L
ACT/ACT.AFB	ACT/ACT29
ACT/ACT.ICMA	ACTB/ACTB
ACT/ACT.ISDA	ACT/ACT
BUS/252	BU/252

Fees

Interface Value	Calypso Value
AmendmentFees	FEE
Cancelation	TERMINATION_FEE
CashExercise	EXERCISE_FEE
FixedAmount	FIXED_AMOUNT
NovationStepIn	TERMINATION_FEE
NovationStepOut	TERMINATION_FEE
PartialTermination	TERMINATION_FEE
Premium	PREMIUM
swBrokerageAmount	BRK
swSalesCredit	FEE
UnclassifiedFee	FEE
UpfrontFee	UPFRONT_FEE

The FixedAmount MarkitWire fee should be mapped to a specific and distinct fee type in Calypso, such as FIXED_AMOUNT, and be used exclusively as a way to represent the ZC swap final payment.

Field Keywords

Interface Value	Calypso Value
1	AdditionalFld1
2	AdditionalFld2
3	AdditionalFld3
4	AdditionalFld4
5	AdditionalFld5

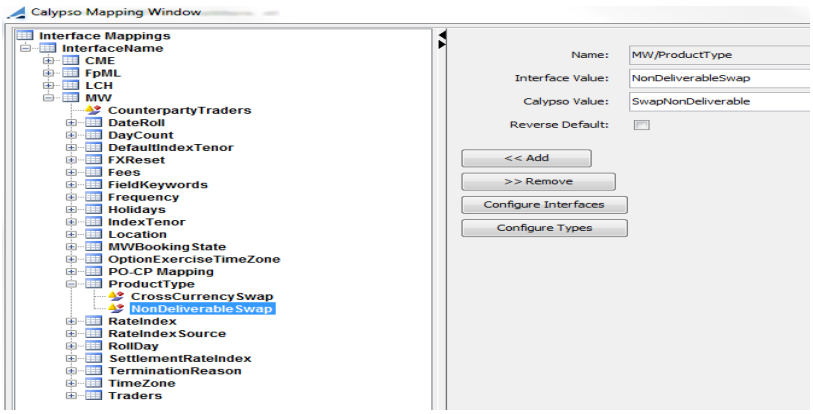
MarkitWire Booking States

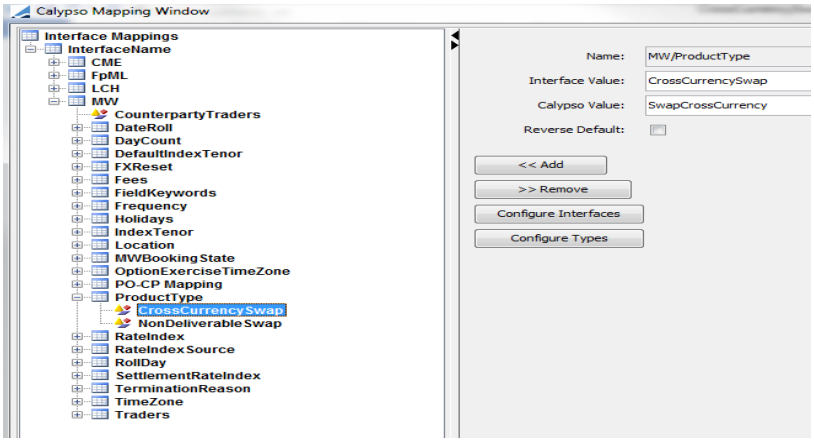
Values in the MWBookingState column:

- Error
- Saved
- Validated

ProductType

The import of cross-currency Swaps and Non deliverable swaps in IRS window as SwapCrossCurrency and SwapNonDeliverable respectively can be configured in the “ProductType” mapping. The support is added in all modes the interface supports. Please note that the product old types are deprecated and the new types are recommended to be used – Swap Non Deliverable and Swap Cross Currency. The same can be mapped via following in the Calypso mapping window:





For getting the trades booked with old product type calypso value should be same as the interface value.

Note: If the mappings are available, and we book a CrossCurrencySwap in MW with either leg having a non-deliverable currency, it will get saved in calypso with product type SwapNonDeliverable as per core calypso standard. In bidirectional mode, we check if the legs have different currencies, we Allege trade with the MarkitWire product as CrossCurrencySwap.

2.4.9 Engine Configuration

The Swapswire Trade engine is configured in the Engine Manager of Web Admin: event subscription and engine parameters.

It subscribes to: PSEventTrade, PSEventRepublish, PSEventSwapswire, and PSEventSwapswireScheduleTask.

Add the SwapswireTradeEngineEventFilter for the SwapswireTradeEngine to the list Event Filters.

You may need to add this engine if it is not available for configuration: Create a new engine called SwapswireTradeEngine with class name `com.calypso.engine.advice.SwapswireTradeEngine`.

▶ Please refer to Calypso Web Admin documentation for complete details.

Engine Parameters

You can set the engine parameters using the Engine Manager in Web Admin.

EVENT_POOL_POLICY - Select SwapswireTradeEngine.

2.4.10 Using Multiple SwapswireTradeEngine

You can use multiple engines in markitwire and connect to separate dealsinks for parallel processing.

To enable this, an instance of the engine must be provided with an additional argument -engineName denoting the unique name of the engine; for example, -engineName=SwapswireTradeEngineOne.

You also need to set the environment property:

SWAPSWIRE_ENGINE_MULTI_MODE=true

Multiple SwapswireTrade engines can be created as follows.

Configure the engines in "`<calypso home>/tools/local-jboss-deployer/config/deployLocalConfig.properties`":

Please note that numbers/digits and underscores cannot be used to name an engine (e.g. SwapswireEngine1 will not work).

Sample Engine Configuration:

```
SwapswireTradeEngineOne_SWAPSWIRE_USER=dealsink1
SwapswireTradeEngineOne_SWAPSWIRE_PASSWORD=pwd1
SwapswireTradeEngineOne_SWAPSWIRE_CONCURRENT_LOGIN_NO=1

SwapswireTradeEngineTwo_SWAPSWIRE_USER=dealsink2
SwapswireTradeEngineTwo_SWAPSWIRE_PASSWORD=pwd2
SwapswireTradeEngineTwo_SWAPSWIRE_CONCURRENT_LOGIN_NO=1
```

In the environment properties, the deal sink user id (SWAPSWIRE_USER), deal sink password (SWAPSWIRE_PASSWORD) and the concurrent login count (SWAPSWIRE_CONCURRENT_LOGIN_NO) must be modified to <engine name>_SWAPSWIRE_USER, <engine name>_SWAPSWIRE_PASSWORD and <engine name>_SWAPSWIRE_CONCURRENT_LOGIN_NO.

Similarly, in "calypso_SW_config.properties", properties related to do-recovery need to be changed; performDoRecovery, doRecoveryStartDate and doRecoveryEndDate become <engine name>_performDoRecovery, <engine name>_doRecoveryStartDate and <engine name>_doRecoveryEndDate respectively.

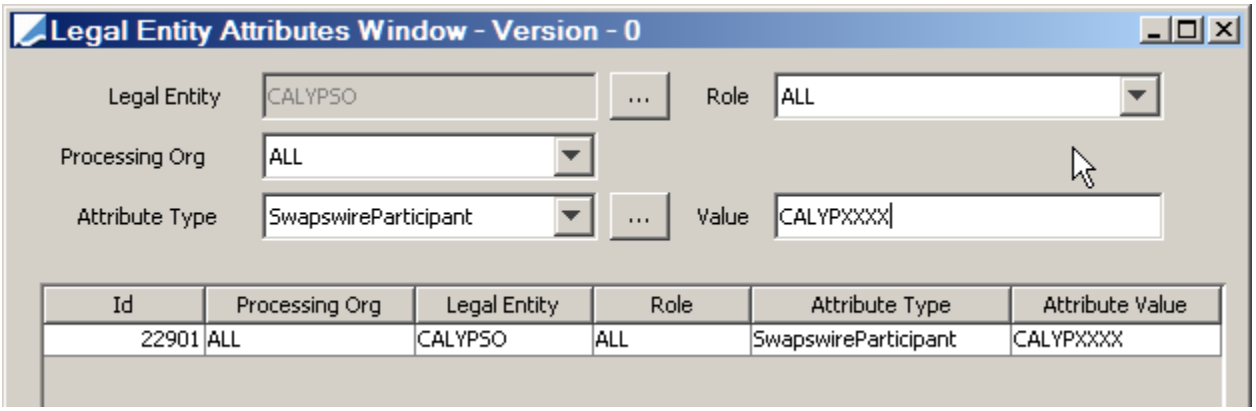
2.4.11 Legal Entities

Map legal entities in Calypso with attribute "SwapswireParticipant" set to participants' IDs as sent by MarkitWire, and SwapswireBroker set to brokers' participant IDs.

Clearing House Legal Entities should have roles: Clearer, CounterParty, and MarketPlace.

This is an example of the required mapping of a MarkitWire counterparty to Calypso **Legal Entity**. The required **Participant ID** is the value of the <partyId> in a trade's SWML file. For example, in the following excerpt, CALYPXXXX is how MarkitWire refers to counterparty in question:

```
<party id="partyA">
<partyId>CALYPXXXX</partyId>
<partyName>Calypso Bank</partyName>
</party>
```



Support for multiple broker and counterparty BIC codes of MarkitWire

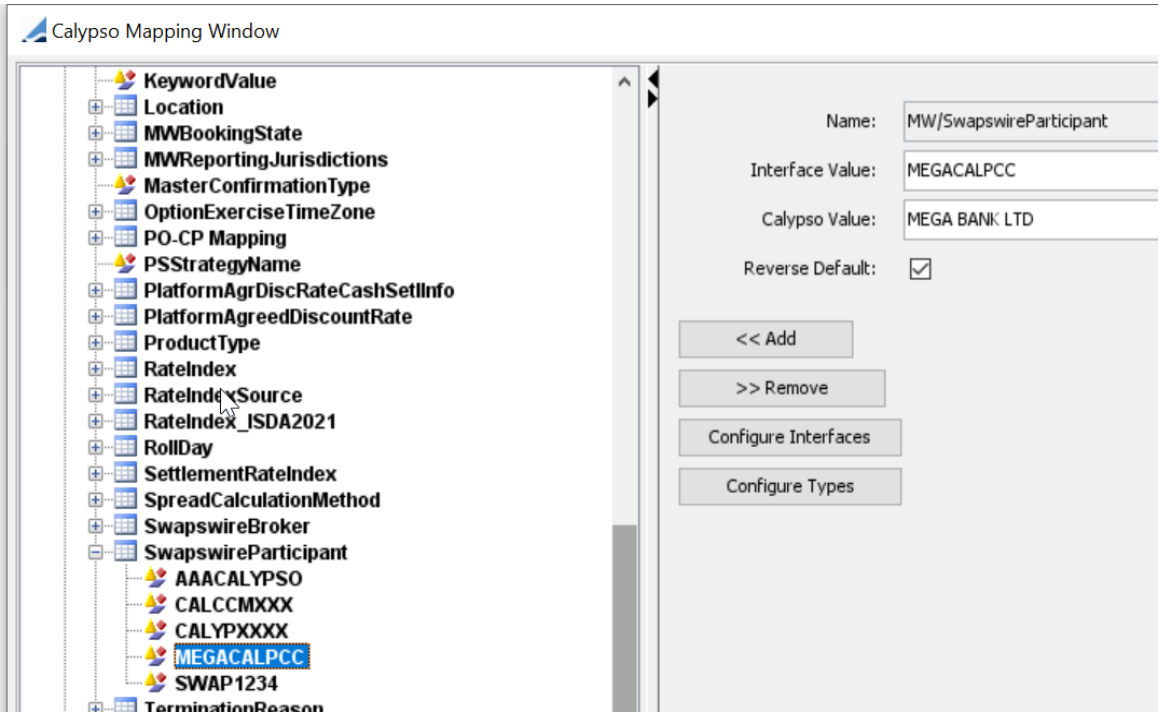
If more than a single code for the broker and counterparty needs to be mapped the only option to store different settings is to use different constellations of Processing Org and/or Role. As the business is getting instantly more complex we are currently running into a situation where for the broker/counterparty has multiple codes need to be mapped. But all the codes belong to the same broker/counterparty.

Therefore we need a configuration option where we can map multiple codes for one Broker/Counterparty.

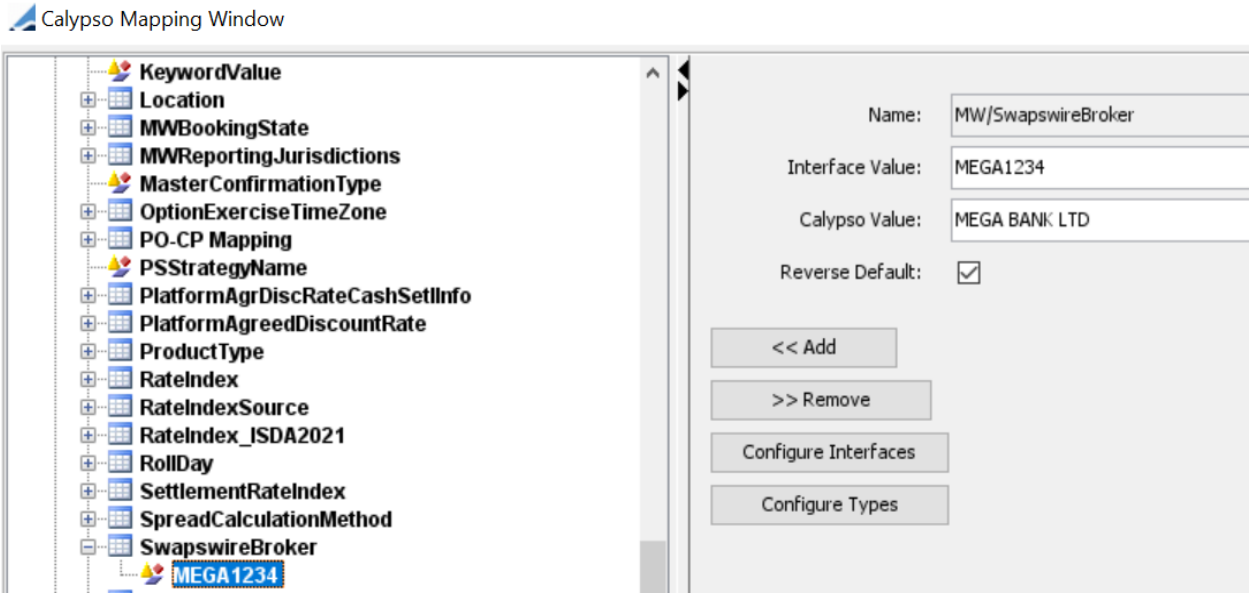
Let's take below example, in Calypso mappings, define multiple PO Swapswireparticipant required for booking a trade from MW and link to one legal entity PO in calypso. PO CALYPSO Bank Legal Entity is mapped with multiple swapswireparticipants as shown below. In calypso mappings, define multiple clearing broker bicode required for booking a trade from MW and link to one legal entity cpty in calypso.

NOTE: The same process can be followed for the Broker as well. This is available only for incoming mode.

Here, the Interface value - Swapswireparticipant of MW cpty and Calypso Value - cpty Legal entity in Calypso.



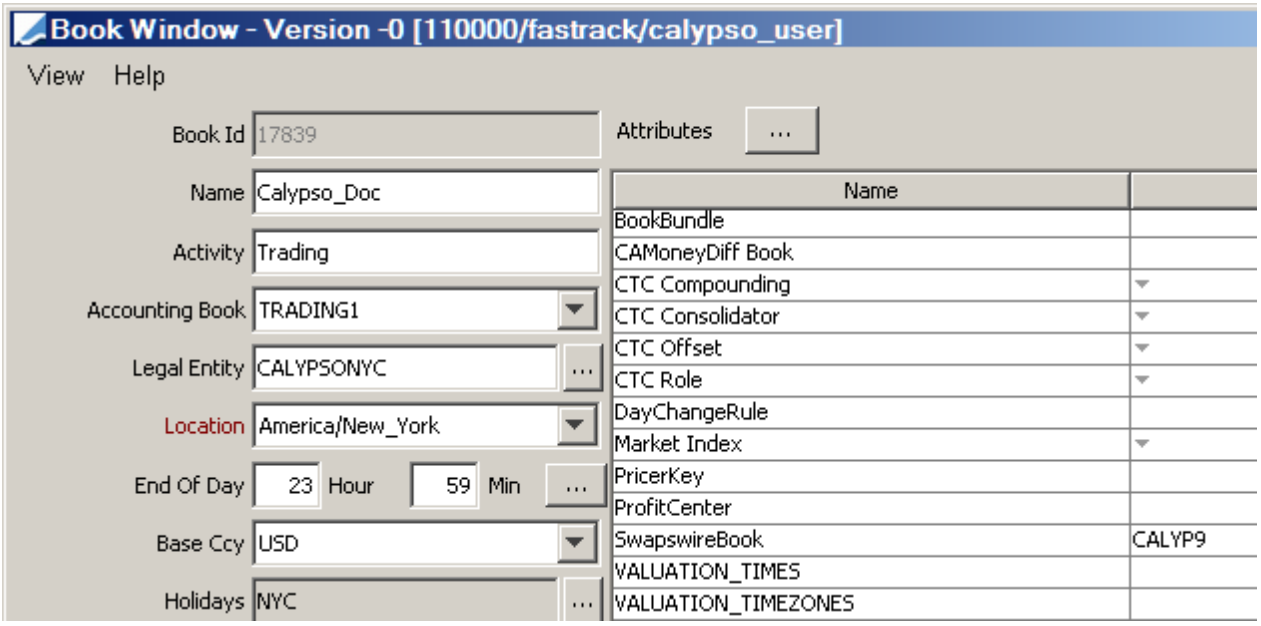
Interface value - Swapswirebroker of MW cpty and Calypso Value - broker Legal entity in Calypso



If there is a mapping provided in Calypso mapping window for a MW BIC code against Calypso LE that will be given preference and if for a certain BIC code it is not provided then it will fallback to existing logic of checking for LE having an attribute with this BIC code.

2.4.12 Books

Set the SwapswireBook attribute (create this attribute if it is not present) for each book that will receive MarkitWire trades. The SwapswireBook attribute is the MarkitWire Book name.



In the example above, the Calypso book, “Calypso_Doc” has the SwapswireBook attribute value, “CALYP9.” MarkitWire trades booked against the CALYP9 book will be entered against the Calypso Book, “Calypso_Doc.” If desired, you can create Calypso book names identical to those on MarkitWire, however, the mapping must still be made.

Note: The SwapswireBook name is always from the MarkitWire point of view.

Multiple MarkitWire Users

If your implementation uses multiple concurrent MarkitWire Users (see the “Multiple Dealsink Users” section), ensure that you have also set the SwapswireBook attribute of each Calypso book as appropriate.

BuySide Case

The domain “MWUserRole” is used to identify BUY or SellSide. It is SellSide by default. It should contain BuySide to identify BuySide.

In case of BuySide:

Incoming messages - If parent LE BIC code (SwapswireParticipant) matches with swTradingBookId field from SWML, the book is set using SwapswireBook attribute on the allocation Fund-LE instead of swTradingBookID from SWML.

Outgoing messages – Set the Fund-LE party id (LE having SwapswireParticipant = SwapswireBook attribute.

In BuySide mode, orders executed by Tradeweb in Calypso are linked to trades incoming from Markitwire following the scenario below:

- Order is created in Calypso and sent to Tradeweb.
- Tradeweb executes order and sends message to Calypso.
- Calypso executes order and creates trade (PlatformTradeId = Tradeweb Trade Id).
- Tradeweb sends message to Markitwire.
- Markitwire sends trade incoming notification message to Calypso.
- Calypso links existing trade created from Tradeweb execution with incoming Markitwire trade using SWBrokerTradeId.
- Post trade lifecycle is supported via standard Markitwire integration.

2.4.13 Broker

You must map the SwapswireBroker Legal Entity Attribute to Broker in MarkitWire.

2.4.14 Reference Bank

You must map the SwapswireLongName Legal Entity Attribute to the Break Clause Reference Bank in MarkitWire.

2.4.15 Workflow Configuration

Note: Ensure that you have first imported the most current version of the `MWGATEWAYMSG.wf` and `UPLOADSOURCEMSG.wf` message workflows from the `<calypso home>/client/resources` directory. These files were extracted from the Service Pack jar that you are installing.

Workflow rules associated with MarkitWire only affect trades uploaded from an external source, i.e., those trades with the TradeSource Trade Keyword.

Adding Domain Data

Use the Domain Values window to add the following actions to the indicated Domains:

Domain	Action
UploadExitAction	MWEXIT
tradeAction	MWEXIT

Note that you can use any valid TradeAction for a customized workflow action. When an Action is not provided, Calypso defaults to the MWEXIT Action.

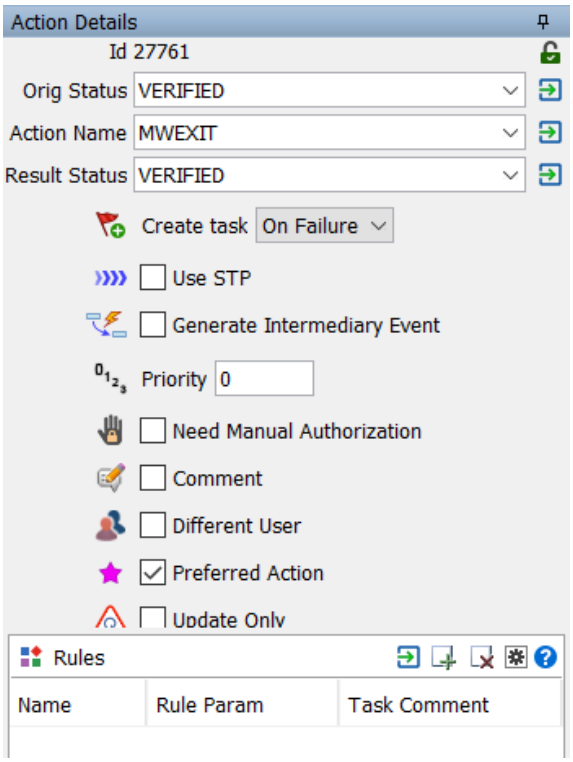
Please remember to setup workflow rules for any custom actions used here.

Removing MarkitWire Keywords

In the event that you wish to remove MarkitWire Keywords from a trade, follow the procedure below:

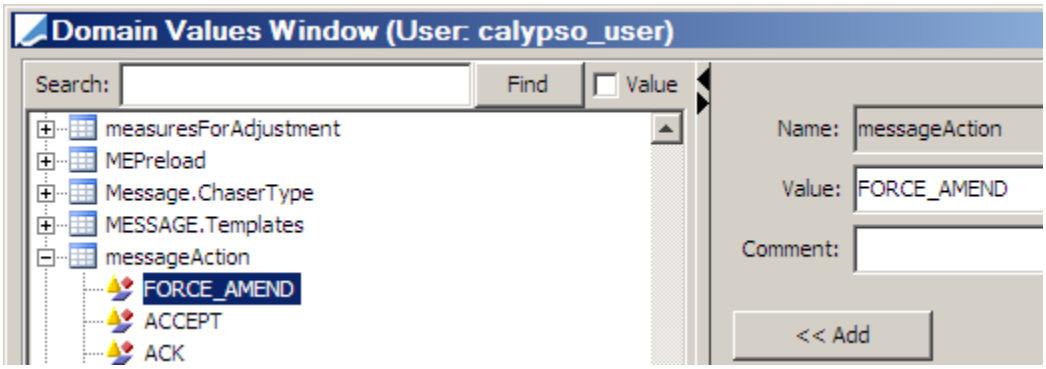
Step 1 - Add the MarkitWire Keywords to remove under the MWExitKeywords domain using the Domain Values window.

Step 2 - Add a trade workflow rule for the exit action. The action name should match with the value specified in the UploadExitAction domain (i.e., MWEXIT).



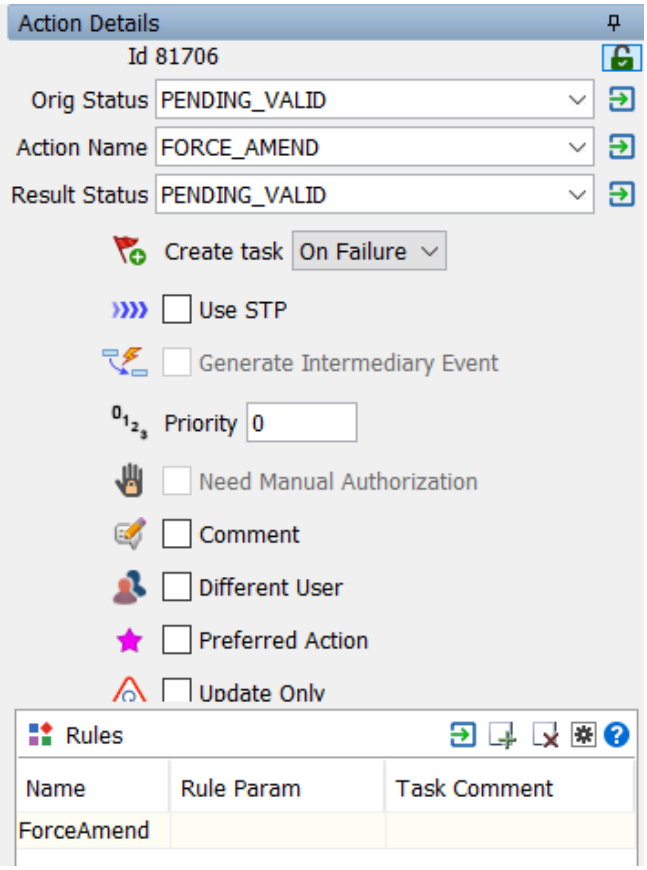
ForceAmend

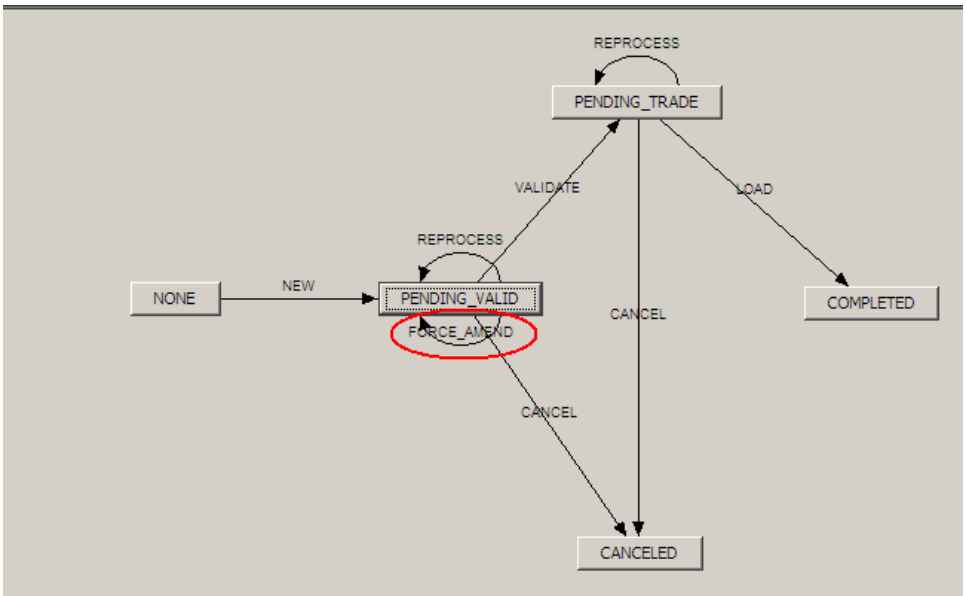
ForceAmend allows you to force an amend on a trade using the new message workflow rule for the scenario where the interface is down and the trade was entered with a manual link to an existing MarkitWire trade in Calypso. The User must add the FORCE_AMEND action under the messageAction domain:



Add a New Action to the GatewayMsg Workflow

Insert the FORCE_AMEND action and ForceAmend workflow rule. The Original Status is PENDING_VALID and Result Status is PENDING_VALID:



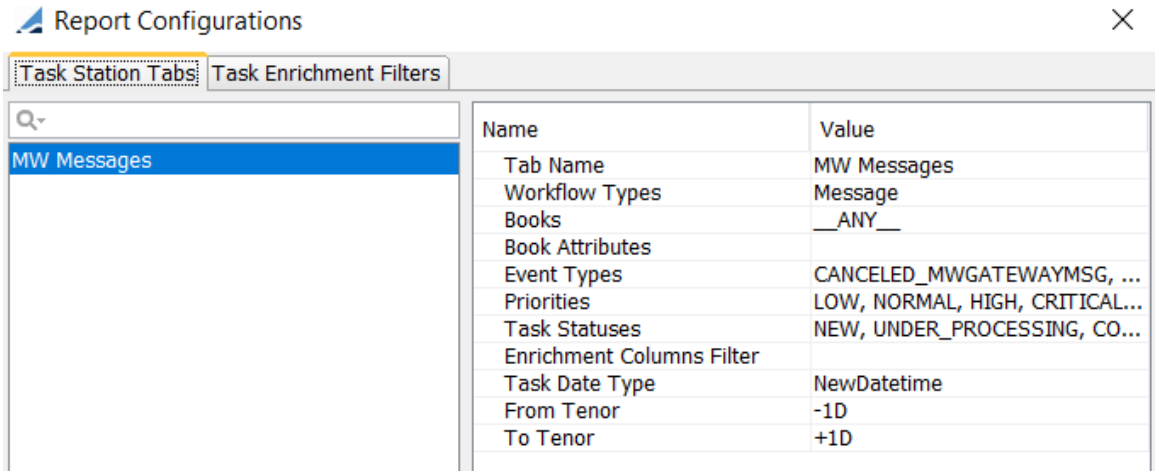


If a trade with given external reference already exists in Calypso, then a new trade from MarkitWire is not uploaded to Calypso. The SwapswireTradeEngine throws the exception task: “Trade Already exists with External Reference: MW_POName_MarkitWireDealID”, where POName is the Processing Org name and MarkitWireDealId is the ID of the MarkitWire deal.

The user can replace the existing trade with the MarkitWire trade by manually applying the FORCE_AMEND action on the gateway message. Doing so will force the Calypso trade to change from NEW to the default amend action and MarkitWire trade is then uploaded to Calypso with the Amend action.

2.4.16 Task Station Configuration

You should configure the Task Station with specific reports to monitor messages and exceptions generated when processing MarkitWire trades. The image and table below provide the suggested configuration:



Suggested Reports	Event
MW Messages	BACKLOAD_MWGATEWAYMSG (provided the BACKLOAD action is configured – See Backloading Support section)
	CANCELED_MWGATEWAYMSG
	PENDING_TRADE_MWGATEWAYMSG
	PENDING_VALID_MWGATEWAYMSG
MW Warnings	EX_MWGATEWAYMSG_WARNING
MW Exceptions	EX_MWGATEWAYMSG
	EX_MWGATEWAYMSG_ERROR
	EX_MWGATEWAYMSG_REJECT

Suggested Reports	Event
UploadSource Messages	UploadSource Messages
CANCELED_UPLOADSOURCEMSG	CANCELED_UPLOADSOURCEMSG
	PENDING_UPLOADSOURCEMSG
	RECEIVED_UPLOADSOURCEMSG
	TRANSLATED_UPLOADSOURCEMSG
UploadSource Warnings	EX_UPLOADSOURCEMSG_WARNING
UploadSource Exceptions	EX_UPLOADSOURCEMSG
	EX_UPLOADSOURCEMSG_ERROR
	EX_UPLOADSOURCEMSG_REJECT

2.4.17 MarkitWire Module Date Adjustments

This section describes how the Calypso MarkitWire module handles Date Adjustments.

Start Dates

A trade's **Start Date** is adjusted in the Calypso GUI if the **Adj Start** checkbox in the MarkitWire GUI is selected (checked). The **Start Date** is left unadjusted, otherwise. Any adjustment of the **Start Date** must occur in the Calypso GUI to ensure the proper calculation of coupon amounts. Calypso never changes the payment date of the first cash flow, regardless of the Coupon Accrual Type. Calypso stores the **Start Date** of each leg of the trade in **SWOriginalPayStartDate** and in **SWOriginalRecStartDate** if there is a difference in date after an adjustment occurs.

End Dates

End Dates in the Calypso GUI are always unadjusted. However, the coupon payment periods are adjusted according to the **Accrual Type** selected in Calypso's GUI.

Trade Keywords

- SWOriginalPayStartDate - The original Paying Leg Start Date in MarkitWire. This Keyword is populated when an adjustment occurs on the Paying Leg Start date.
- SWOriginalRecStartDate - The original Receiving Leg Start Date in MarkitWire. This Keyword is populated when an adjustment occurs on the Receiving Leg Start Date.

2.5 Data Mapping Between MarkitWire and Calypso

Note: If your system uses Access Permissions, users who require access to the Calypso Mapping Window must have the appropriate permissions. See the "Access Permissions" section.

From the Calypso Navigator, navigate to Processing > Tools > Calypso Mapping to open the Calypso Mapping window.

The mapping table provides the infrastructure to map MarkitWire values to the equivalent Calypso representation.

The essential mappings are simply Value/Name. These are the items for which mappings must be made to allow the SwapswireTradeEngine to correctly import trades. The sample data provides some basic mappings for currency, holidays, tenors, workflows, etc. As the user, you will need to create additional mappings appropriate to your implementation.

The creation of the Mapping table and the domain population was accomplished in the "Data Model Synchronization" section.

You will need to create additional mappings.

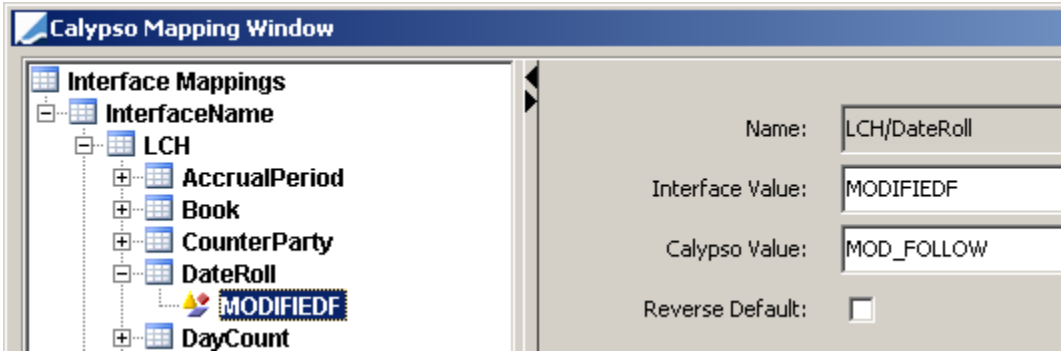
Upgraders: Prior to using the Calypso Mapping Window with MarkitWire, you must migrate any existing Swapswire mappings to the Calypso Mapping table. This is accomplished during the Execute SQL step of the MarkitWire setup of the Calypso Data Uploader.

2.6 Mapping Based on Source

When the attribute "Source" is present in the root tag of the CalypsoUploadDocument:

```
<CalypsoUploadDocument Source="LCH" Version="1" ClientCode="LCH" UploadDate="05/ 13/2011">
```

then the application calls the Mapper Component, which is an extensible API that receives an XML document in the form of JAXB and the Source name. Using this Source Name the application maps the data from its Source-specific form to a Calypso-specific form:



In LCH, the DateRoll is defined as MODIFIEDF whereas in Calypso its value is MOD_FOLLOW, once this is defined, the mapper component then changes data from Source form to Calypso form before calling the Data Uploader.

The Source Mapper Component is used before actual upload of the object and during the REPROCESS workflow action. The Source Mapper Component is product specific, which means that the user must create custom code for the components. For example:

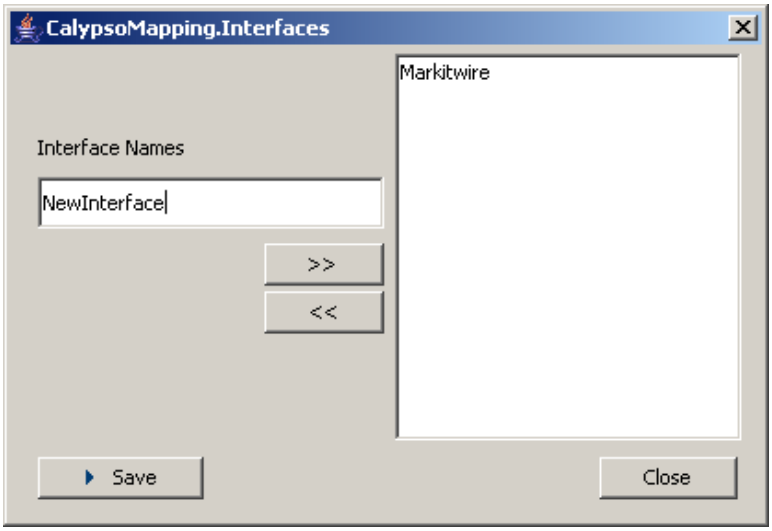
```
SOURCEInterestRateSwapMapper.java

SOURCECreditDefaultSwapMapper.java

SOURCECalypsoLegalEntity.java

If a Source-based mapper is not found, then the uploader framework attempts to find the default mappers:

InterestRateMapper.java

CreditDefaultSwapMapper.java

CalypsoLegalEntityMapper.java
```

If neither the default, nor the Source Mapper Components are found, then the Calypso Mapper is called and the application proceeds to upload.

2.6.1 Adding a New Interface Name

Synchronizing the schema files will create the required Interface Name and Interface types in the domain for MarkitWire.

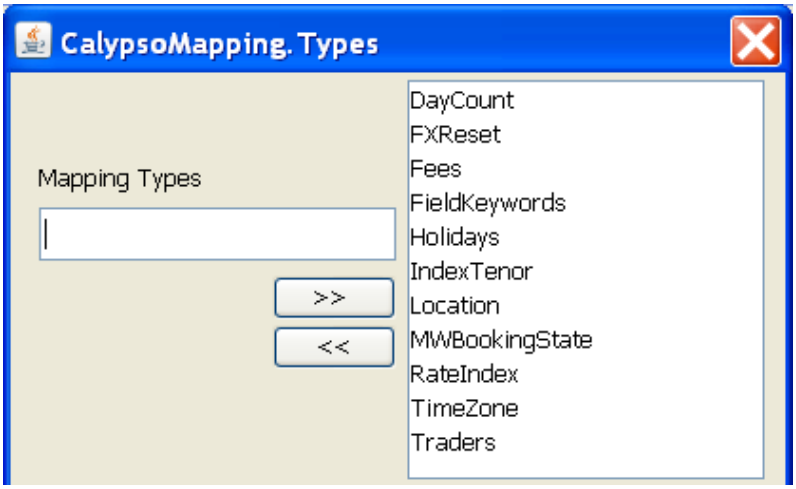
To manually add a new Interface, click **Configure Interfaces** on the Calypso Mapping Window to display the Calypso Mapping Interfaces dialog where you can add a new Interface Name to the system:



2.6.2 Adding a New Interface Type

Synchronizing the schema files will create the required Interface Name and Interface types in the domain for MarkitWire.

To manually add a new Interface Type, click **Configure Types** on the Calypso Mapping window to display the Calypso Mapping Types dialog where you can add a new Interface Type to the system:



2.6.3 Fee Types

Calypso provides most fee mappings via the data model synchronization process. The following fees are predefined:

Swapswire Fee	Calypso Value
AmendmentFee	FEE
Cancellation	TERMINATION_FEE
CashExercise	EXERCISE_FEE
FixedAmount	FIXED_AMOUNT
NovationStepIn	TERMINATION_FEE
NovationStepOut	TERMINATION_FEE
PartialTermination	TERMINATION_FEE
Premium	PREMIUM
UnclassifiedFee	FEE
UpfrontFee	UPFRONT_FEE
swBrokerageAmount	BRK
swSalesCredit	FEE
CancellablePremium	PREMIUM

Notes on Fee Mapping and Trade Exercise

- If the user does not specify the Cash Exercise fee on MarkitWire, Swaptions are then exercised in Calypso without the fee.
- If the user specifies a fee on MarkitWire, and no corresponding mapping is found on Calypso, the trade is not exercised in Calypso and an exception is then generated on the Task Station.

- If the user specifies a fee on MarkitWire and corresponding mapping is found on Calypso, the Swaption on Calypso is exercised and the fee is added.

Manual Fees

Calypso provides users the option to manually add fees to a Calypso trade that are not amended by newer versions of the MarkitWire trade.

Users can add fees to the "**UploadPreserveFee**" domain to prevent their being amended. Fee types defined in this domain are considered "Manual Fees." These fees follow the same logic as that of "**propagateFees"**. If the fee is defined in the domain and if the **Fee Date** is greater than the **Transfer Date**, then the fee propagates to the child trade.

Fee Types entered in the "**UploadPreserveFee"** domain may not be the same as any MarkitWire Fee Types defined in Calypso Mapping Window.

Fee types that should not be sent to MW while alleging trades from Calypso should be added to the domain "**PlatformIgnoreFees**". A fee type defined in domain "**PlatformIgnoreFees**" must also be defined in domain "**UploadPreserveFee**" so that the fee is preserved for the incoming updates coming from MW on the alleged trades.

Example:

PlatformIgnoreFees contains Value = COMMISSION

UploadPreserveFee contains Value = COMMISSION

COMMISSION fee is not sent to MW while alleging trades from Calypso and is preserved on the Calypso trade when incoming updates are coming from MW on alleged trades.

Handling of Reappearing Fees

Calypso defines a "Reappearing Fee" as a fee that has the same type, amount, and date as a previously defined fee on any of the ancestral trades.

Any Fee having the same type, amount, and date present in parent trade does not propagate to child trade if the **Transfer Date** (i.e., the **Trade Effective Date**) is greater than the **Fee Date**.

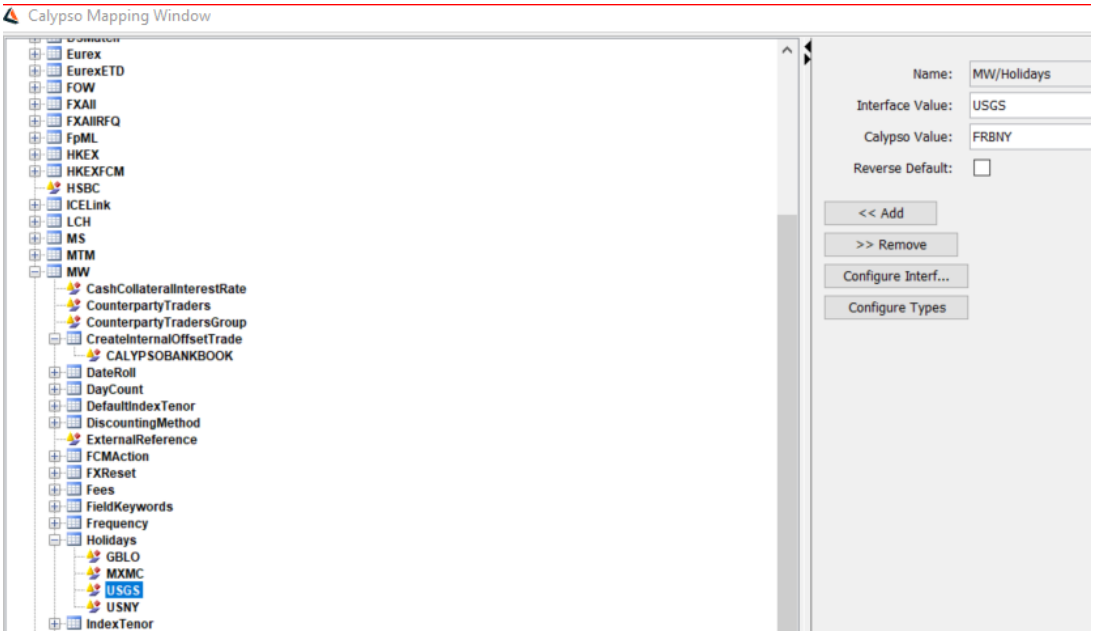
Fees from MarkitWire propagate to child trades in Calypso if:

1. The fee is defined in the **propagateFees** domain; and,
2. The **Fee Date** is greater than the **Transfer Date** of the trade.

2.6.4 Holiday Codes

MarkitWire holiday codes use the code convention common to ISO and FpML, which is the of 2-letter ISO country code followed by a 2-letter city code. For example, the Swapswire USNY code is the equivalent of the Calypso NYC code.

The Swapswire USGS code is the equivalent of the Calypso FRBNY code.



2.6.5 Exercise/Settlement Location

Calypso uses the Exercise/Settlement **Location** field populated from the value of the **Location** field on MarkitWire Compose Trade window's **Break** tab. Locations are mapped in the same manner as Holiday mappings. To map an exercise/settlement **Location**, open the Calypso Mapping, and add the **Location** domain to the list of ValueNames. Next, map the Cash Settlement Locations.

2.6.6 FX Reset Mapping

The Calypso FXReset mapping handles the FX rate reference for MTM XCCy swaps from MarkitWire to Calypso. When importing MTM XCCy swaps from MarkitWire to Calypso, the FX rate reference (next to the **Adj** checkbox) will be at appropriate rate at that trade level. Users can make these mappings these in Calypso Mapping Window.

For example:

MW Rate: USD-EUR-Reuters-FXFIX (Pay leg currency-receive leg currency-Rate Source-Source Page)

Calypso rate: USD.EUR LNB (name of calypso rate as defined in FX Rate Definitions Window (custom names)

FXReset is made available by executing `SwapswireSchemaBase.xml`. No action is required beyond adding your mappings.

2.6.7 DayCount

Users can map DayCounts using Calypso Mapping Window. Executing `SwapswireSchemaBase.xml` and `SwapswireSchemaData.xml` adds all MW DayCounts to the mapping table.

Please note that all standard DayCounts supported by MarkitWire and Calypso are present in the interface and require no action by the user. You can add additional DayCount mappings if the available set lacks a particular DayCount required in your implementation.

2.6.8 Tenors

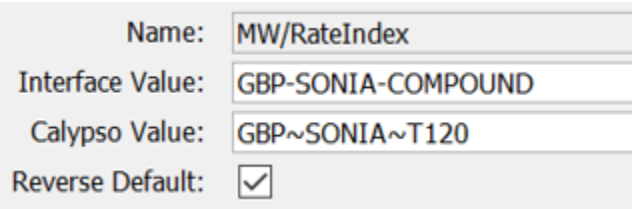
Calypso's MarkitWire implementation supports custom tenors. For example, the pre-populated 28D MarkitWire tenor equates to 4W in Calypso. Using the Calypso Mapping Window, you can enter custom tenors as required.

All standard tenors supported by Calypso and MarkitWire are already available (via `SwapswireSchemaData.xml`) and require no action on your part. Only custom tenors need be entered.

2.6.9 Reference Indices

MarkitWire Rate Index codes follow ISDA conventions. Map the Rate Index codes that you will use to the equivalent Calypso currency/rate index/source names. Note that MarkitWire uses the hyphen (-) as a separator, while Calypso uses the tilde (~).

Sample mapping:



Rate Index Changes

[Important Note: These enhancements are part of the Libor Reform package. The Libor Reform package requires a specific license agreement..

Agreed Discount Rate

The Agreed Discount Rate for swaptions can be set using field / trade keyword PlatformAgreedDiscountRate.

It can be mapped as follows:

Name = MW/PlatformAgreedDiscountRate

Interface Value = EFFR

Calypso Value = FEDFUNDS

If mapping is not provided, the trade keyword is set to the value coming from MarkitWire.

Cash Collateral Interest Rate

The Cash Collateral Interest Rate for swaptions can be set using field / trade keyword CashCollateralInterestRate.

It can be mapped as follows:

Name = MW/CashCollateralInterestRate

Interface Value = EONIA

Calypso Value = EUR/EONIA/1D/T247

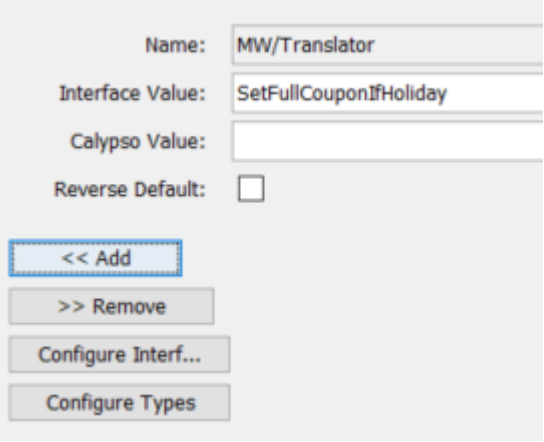
USD LIBOR Trade with Short End Stub

To set the full coupon else, the SetFullCouponIfHoliday is used to map explicitly.

It can be mapped as follows:

Name = MW/Translator

Interface Value = SetFullCouponIfHoliday



Note: The mapping will be visible under FpML/Translator by default, and it would need to be manually added in MW else it will be applicable for all Interfaces which are using the FpML format like LCH/CME/Eurex.

Linking Canceled Trades with Replacement Trades

The following trade keywords have been added.

Canceled trade with old rate index:

- PlatformTransitionTradeId - MW trade id of New trade (with new index)
- PlatformTransitionTradeIdType - MW trade id type - MarkitWireId
- PlatformTransitionReason - Transition reason = IndexTransitionReplacedByTrade
- TransitionTo - Calypso trade id of New trade (with new index)
- TransitionReason - Calypso value for index transition reason = IndexTransition

New trade with new trade index:

- PlatformTransitionTradeId - MW trade Id of Cancelled trade (with old index)

- PlatformTransitionTradeIdType - MW trade id type - MarkitWireId
- PlatformTransitionReason - Transition reason = IndexTransitionReplacedTrade
- TransitionFrom - Calypso trade id of Cancelled trade (with old index)
- TransitionReason - Calypso value for index transition reason = IndexTransition

Netting Sync Approach (Offsetting/Compensation trades) for Cleared Trades

For Cleared trades, MW will work with the CCPs to provide new trades - “Cash Compensation” and “Risk Compensation” trades with the new rate index, using the Netting Sync approach. The new netting type - “IBOR Compensation” and netting sub types - “Cash Compensation/Risk Compensation” are introduced for such trades. There will not be any termination messages. Once the CCP releases these messages to MarkitWire, there will be new trades also sent to dealers/end-users who are also a party on the trade.

- Cash Compensation trades – An amount that can be attributed to the change in discounting method and It will be added as an UPFRONT_FEE on the trade.

  The trades will be Fix-Float Swaps with old Index and Tenor of 1D and payment frequency ZC and notional as 1.0.

  These will have the CCPNettingType - IBOR Compensation, CCPNettingSubCategory - “Cash Compensation”, CCPNettingId as - “NOT_APPLICABLE” and Netting_Count as 0.

  The UPFRONT_FEE will be available on the trade.

- Risk Compensation trades - New trades booked to restore firms’ positions to their original risk profiles. These will come with the Netting Sub type - “Risk Compensation”.

  The trades will be Float-Float Basis Swap trades with FedFunds vs SOFR index.

  These will have the CCPNettingType - IBOR Compensation, CCPNettingSubCategory - “Risk Compensation”, CCPNettingId as - “NOT_APPLICABLE” and Netting_Count as 0.

The following trade keywords have been added:

- SWCounterpartyCorporateSector (incoming and outgoing mode)
- Reporting<Jurisdiction>ReportForCounterparty (Incoming mode only as it is not available in outgoing mode)

Transition To OIS

[Important Note: This enhancement is part of the Libor Reform package. The Libor Reform package requires a specific license agreement..

When you apply the “Transition To OIS” action in Markitwire, it can be processed as an AMEND or a RATEINDEX_UPDATE in Calypso based on the following mapping:

Name = MW/Translator

Interface Value = TransitionToOISAsRateIndexUpdate

Calypso Value = false/true

If false or not set, the AMEND action is applied. If true, the RATEINDEX_UPDATE action is applied.

AMEND Action

The LIBOR trades are modified with the FRF index.

RATEINDEX_UPDATE Action

When this action is applied, there is creation of replacement trades in the RFR index and termination of the initial trades in LIBOR index.

It needs to be added to the Trade workflow:

VERIFIED - RATEINDEX_UPDATE – TERMINATED with workflow rule UpdateTerminationTradeRule.

TerminationReason is set to RateIndex_Update.

It requires the additional mapping:

Name = MW/Translator

Interface Value = TransitionToOISAction

Calypso Value = RATEINDEX_UPDATE

ISDA 2021 Rate Index Config

Mapping category RateIndex_ISDA2021 to handle the index mappings from ISDA2021.

For incoming mode, the ISDA version is taken from the FpML tag:

```
<documentation>
<masterAgreement>
<masterAgreementType masterAgreementTypeScheme= "http://www.swapswire.com/spec/2001/master-agreement-type-1-0">ISDA</masterAgreementType>
</masterAgreement>
<contractualDefinitions>ISDA2021</contractualDefinitions>
</documentation>
```

In outgoing mode, you need to set the trade keyword PlatformContractualDefinition = ISDA2021.

Sample mapping:

If not available, the FpML mapping is used instead, if defined.

Interp Flag on Floating Rate Stubs

By default, the “Interp” flag on the trades is set based on the rate index “No Auto Interp” field.

If you want to clear the “Interp” flag, regardless of the rate index “No Auto Interp” field, then set the following mapping entry to true.

Name = FpML/Translator

Interface Value = UndetStubInterpolation

Calypso Value = true

2.6.10 Traders

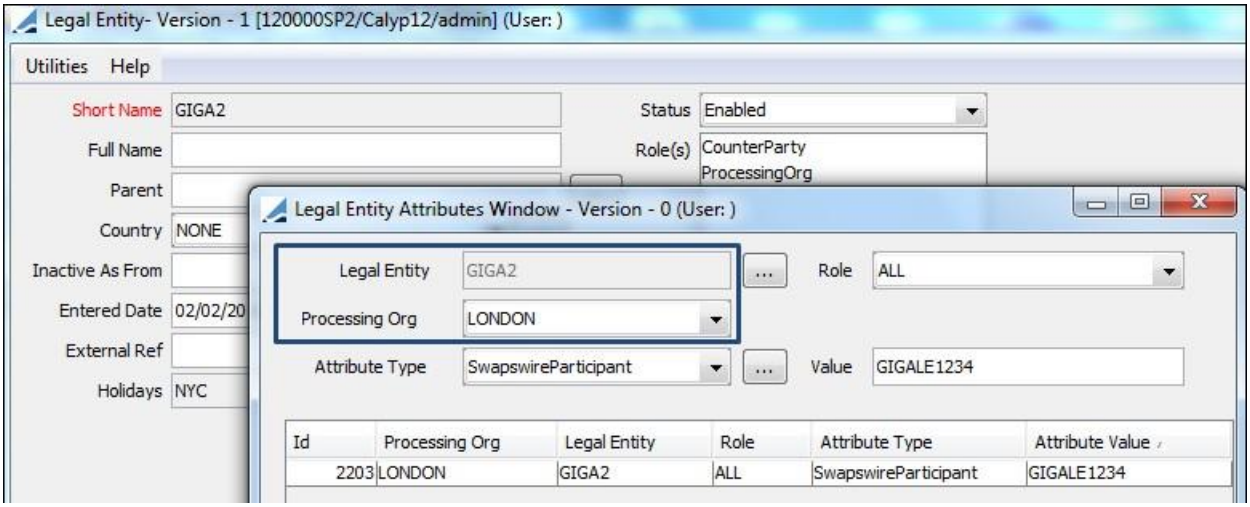
Map Trader names from MarkitWire to their Calypso Trader names.

2.6.11 PO-CP Mapping

Calypso uses the Legal Entity Attribute window to determine the counterparty of a trade based on the processing organization. The Calypso Mapping window is not used.

Configuration and maintenance at the level of the Legal Entity is more efficient and provides more flexibility for entry and reporting.

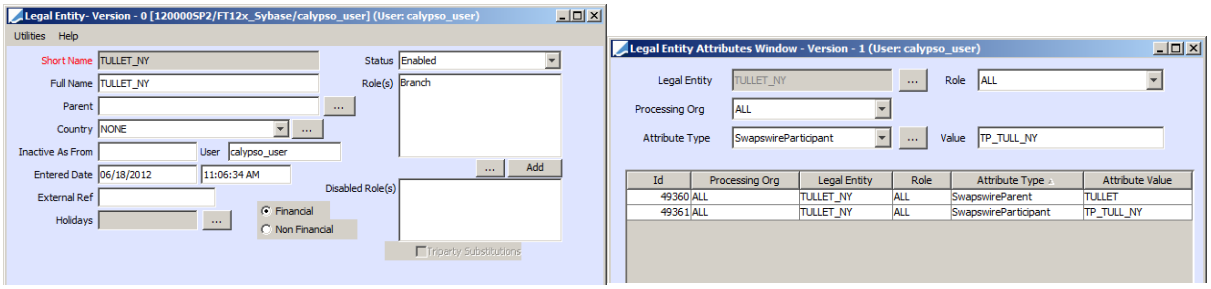
For example, if the Processing Organization is NewYork then counterparty is Giga. If Processing Org is London then counterparty will be Giga2:



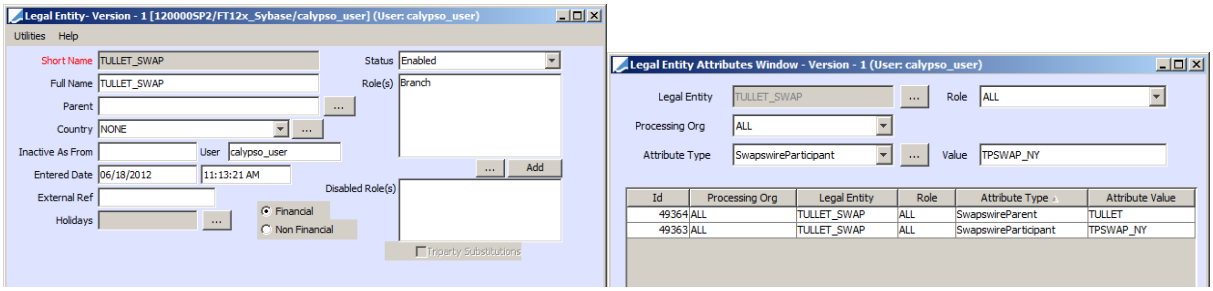
2.6.12 BIC Mappings

It is possible to map multiple BIC codes for child entities to a single, parent Calypso Legal Entity, whether that parent entity is a counterparty or processing organization. Two or more child entities having different BIC codes (specified in SwapswireParticipant) can be mapped to a single legal entity in Calypso by making use of an identical SwapswireParent attribute (i.e., the BIC code of the parent entity).

The images below show the first of two child entities that will be mapped to a single legal entity:



The images below show the second entity:



Notice that the SwapswireParticipant BIC ID differs for each child entity, while the SwapswireParent BIC ID is the same for both.

Mapping for multiple broker and counterparty BIC codes

Both the broker and counterparty mapping for the Markitwire interface is done via the Legal Entity Attributes SwapswireBroker and SwapswireParticipants. If multiple BIC code needs to be mapped, then different constellations of Processing Org and/or Role must be used.

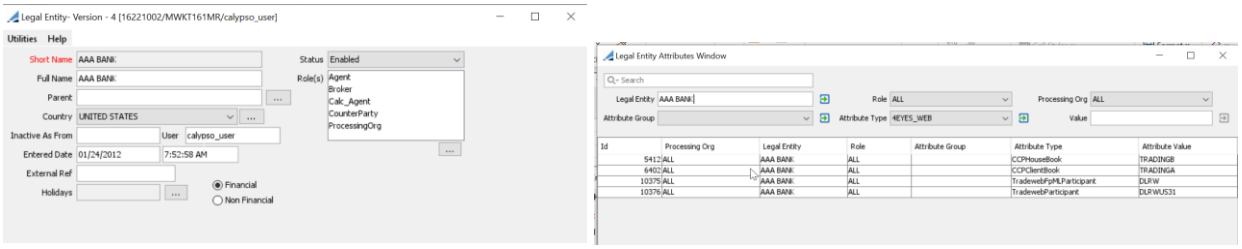
As the business is getting instantly more complex, we are currently running into a situation where for the broker Icap more than 4 codes need to be setup (ICAP, ICap_LN, ISWAP_LN, RSBROK, and RESET). All codes belong to the same broker Icap Europe Limited. We also have very similar constellations for counterparties who use multiple codes.

Therefore, we need a configuration option where we can map more than 4 codes for one Broker. We view an ideal solution as a coma separated configuration within the current Processing Org and Role framework in Calypso Mapping window.

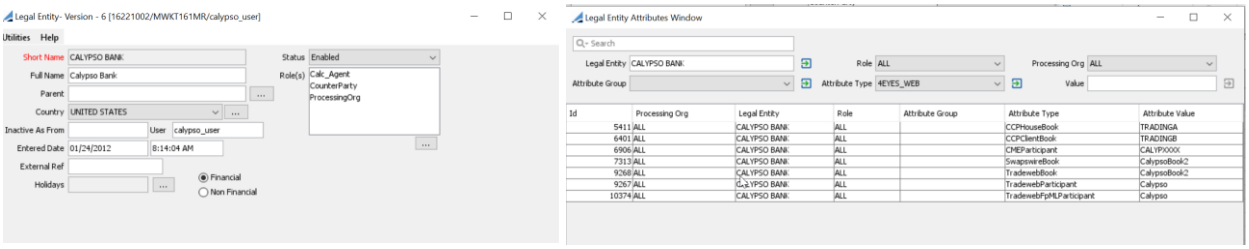
Example of Multiple Mappings:

Delete all the Swapswire participant or Swapswirebroker codes from the attributes of cpty and PO legal entity as below:

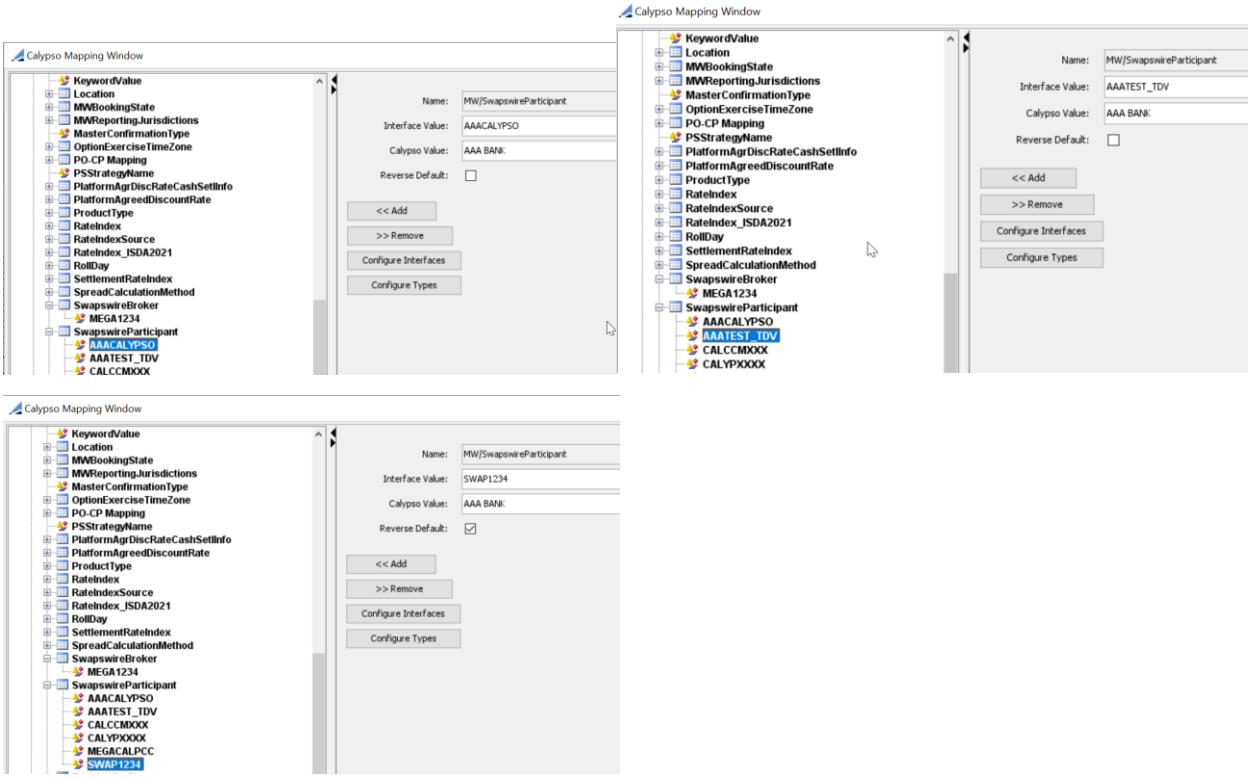
CPTY



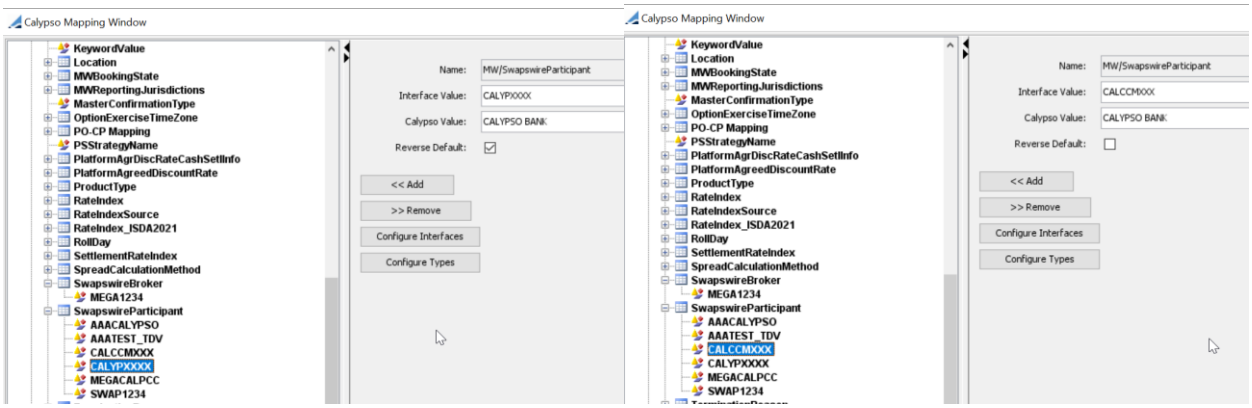
Processing Org (PO)



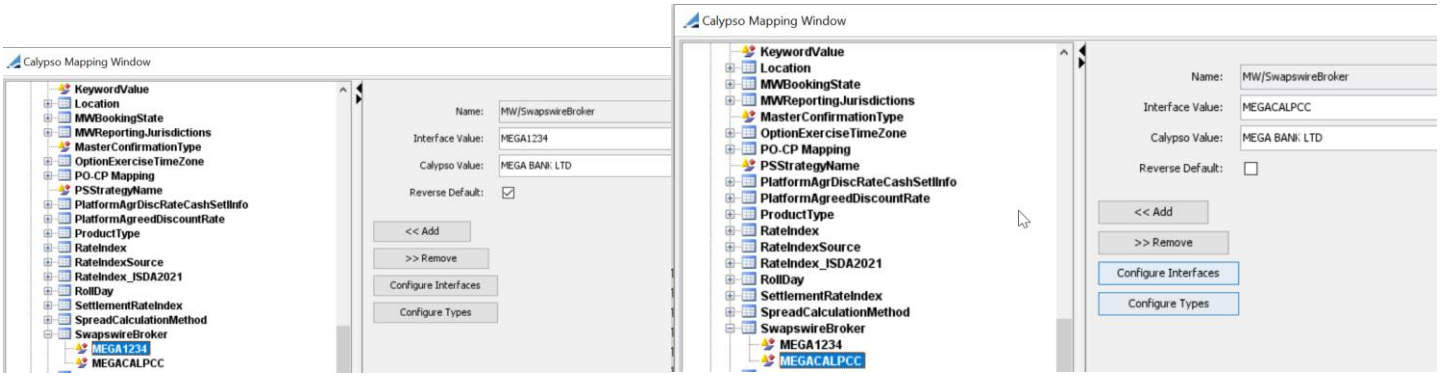
In Calypso mappings, define multiple cpty Swapswireparticipant required for booking a trade from MW and link to one legal entity cpty in Calypso. Here, Cpty AAA Bank Legal Entity is mapped with multiple swapswireparticipants as below:



In Calypso mappings, define multiple PO Swapswireparticipant required for booking a trade from MW and link to one legal entity PO in Calypso. Here, PO CALYPSO Bank Legal Entity is mapped with multiple swapswireparticipants as below:



In Calypso mappings, define multiple clearing broker bicode required for booking a trade from MW and link to one legal entity cpty in Calypso. The below image shows the link to one legal entity:



2.7 Starting the SwapswireTradeEngine

The Swapswire Trade engine can be started from the Engine Manager in Web Admin.

▶ Please refer to Calypso Web Admin documentation for complete details.

2.8 Importing Pre-Release Trades

Using an Event filter based on states contained in the MWProcessState domain, you can filter and process trades according to their notification state. Processing different notification states allows your implementation to handle multiple notifications for the same contract and version.

Calypso processes the first notification for the NEW contract state as per the contract state functionality. Subsequent notifications for the same contract version are processed as a Unilateral Amend. Add the notification states that the SwapswireTradeEngine should process to the MWProcessState domain. For example, Cancelled, Done, Sent, Released, etc.

If MWProcessState contains no values, the application defaults the trade state to “Released”.

Lifecycle events from MarkitWire on a trade only occur if the MWProcessState is “Released”. All other states only update trade keywords.

Processing Multiple Notifications for the Same Contract/Version

For example, assume that SwapswireTradeEngine supports notifications for Sent, Done, and Released. A notification for a trade with the status NEW arrives with same Contract/Version:

- The application processes the Sent notification and saves the trade as NEW with Private Version 1 and Contract Version 1.
- A Done notification is processed as an AMEND, using Contract Version 1 and Private Version 2.
- Similarly, a Released notification is also processed as an AMEND using Contract Version 1 and Private Version 3.

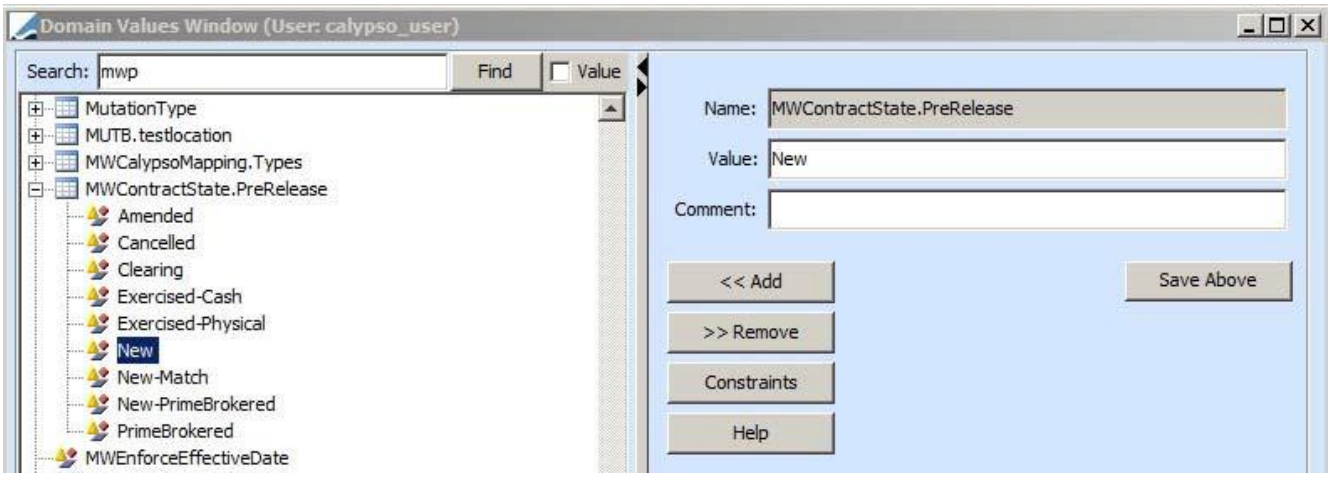
Handling Counterparty Trade Rejections

Because the SwapswireTradeEngine processes notifications before the counterparty affirms the deal, deals rejected by the counterparty are processed as Withdrawn. The SwapswireTradeEngine processes notifications for the "withdrawn" booking status by changing the state of the trade to cancelled, following the logic below:

- Contract state is NEW and the new state is Cancelled - The trade is cancelled in Calypso.
- Termination or novation and the new state is Cancelled - Calypso applies the UNDO_TERMINATE action. The domain value UploadRejectAction specifies the action to execute.
- Amend and Exercise - No action is taken. Calypso creates a Task Station warning entry. The user must manually process amends and exercises.

Processing Based on Contract State

Calypso uses values in the MWProcessState and MWContractState.PreRelease domains as the basis for processing Pre-Release trade notifications.



The SwapswireTradeEngine processes notifications for Contract States listed in the MWContractState.PreRelease domain, for trades whose current Process State is listed in the MWProcessState domain.

Lifecycle actions, other than "New", that alter the economics of a trade (amendments, novations, terminations, clearing, etc.) are not applied until the trade is released.

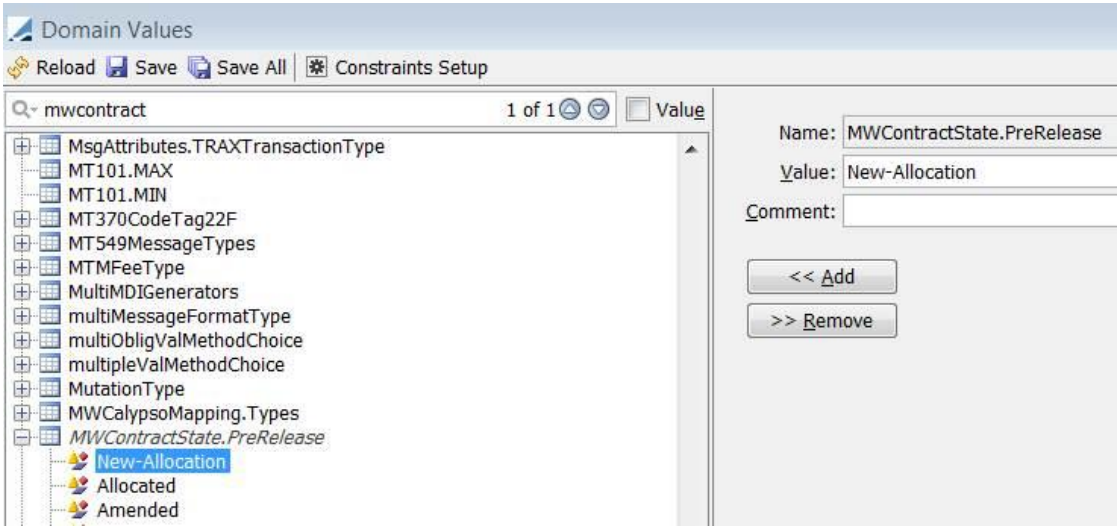
Values other than "New" present in the MWContractState.PreRelease domain trigger an update of the Trade's keywords when needed, but the Trade's terms will only change on Trade release.

Allocation Support

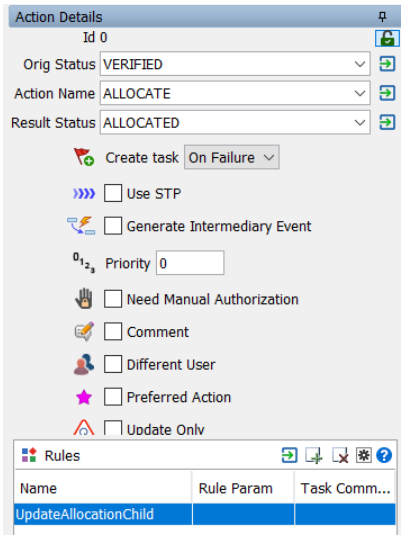
It is now possible to import block and child trades from MarkitWire to Calypso using the out-of-the-box Calypso Allocation API. If you are using the bidirectional mode, it is also possible to Allocate the trade in Calypso and forward the Allocation details to MarkitWire.

Please note that in MarkitWire, the Executing Broker does not see the incoming Funds selected by the Client and sees the counterparty (Block entity) on the child trades instead.

Step 1 - Use the Domain Values window to add Allocated and New-Allocation to the domain "**MWContractState.PreRelease**", and Released to the domain "**MWProcessState"**.



Step 2 - Add the UpdateAllocationChild rule between VERIFIED and ALLOCATED as shown below:

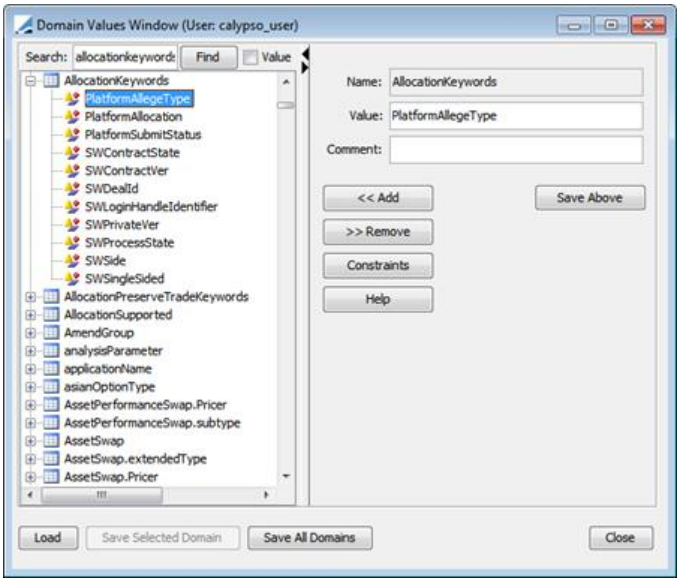


Step 3 - Ensure that your Calypso trade workflow is configured to handle the allocation of a block trade, along with the generic lifecycles for block and child trades.

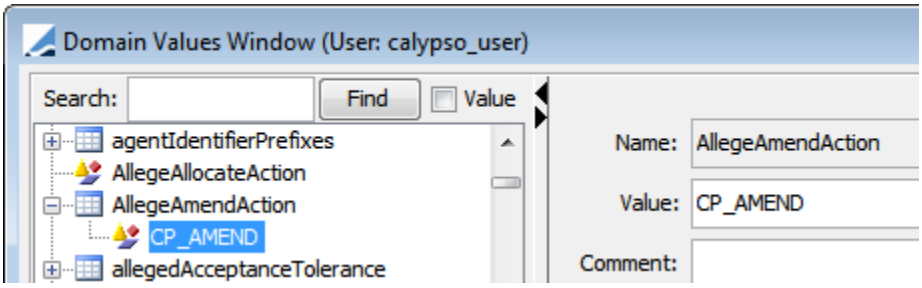
Step 4 - Ensure that the Amend transition (as well as all actions in the UploadAmendAction domain) is available to Calypso trades having the status Verified and Allocated.

Step 5 - Using the generic interface, the block trade must be released in MarkitWire before child trades are created.

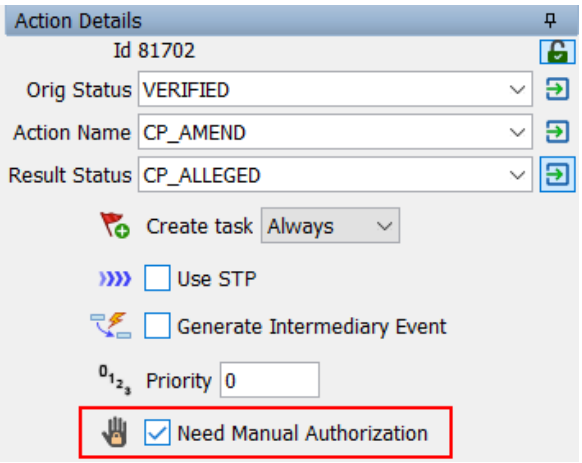
Step 6 - Populate the AllocationKeywords domain with Calypso keywords that you do not wish to propagate from the block trade to the child trades.



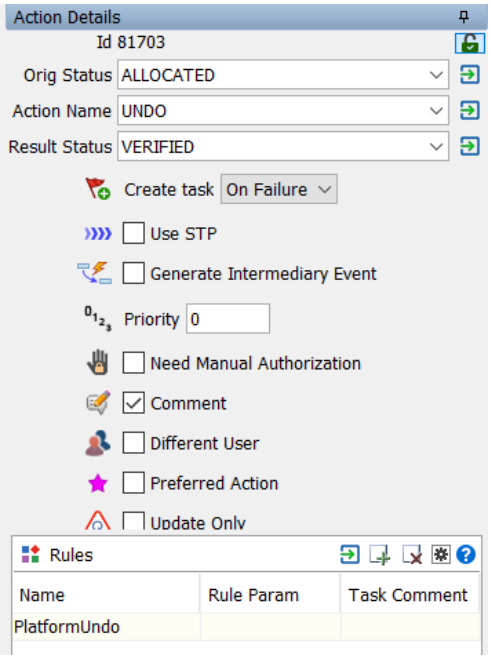
Step 7 - You can add a specific trade workflow transition to the **AllegeAmendAction** domain to validate and authorize incoming allocations submitted by the counterparty prior to amending an existing Calypso trade:



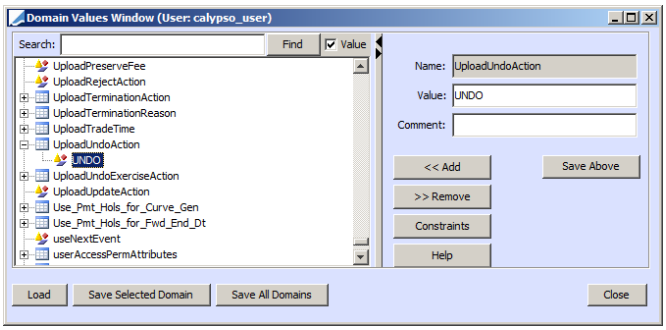
Step 8 - Modify the CP_AMEND Workflow Action by selecting **Needs Manual Authorization**:



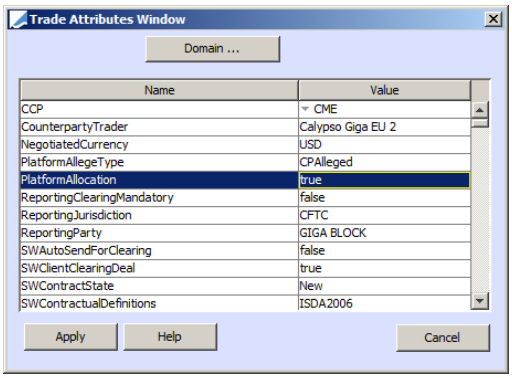
Step 9 - In bidirectional mode, a workflow transition can be created between ALLOCATED and VERIFIED to handle counterparty rejection of outgoing allocations and cancel the child trades:



Step 10 - UNDO must be present in the **UploadUndoAction** domain:



Step 11 - The system sets the PlatformAllocation keyword to true if the trade is being allocated:

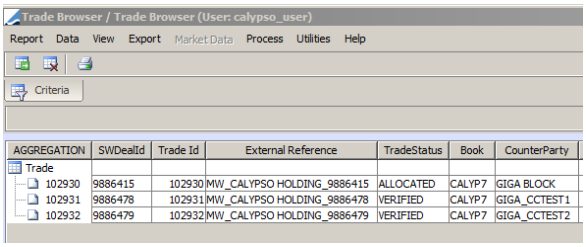


Step 12 - When using bidirectional mode, if an allocation is alleged by the client, the Task Station displays a warning:

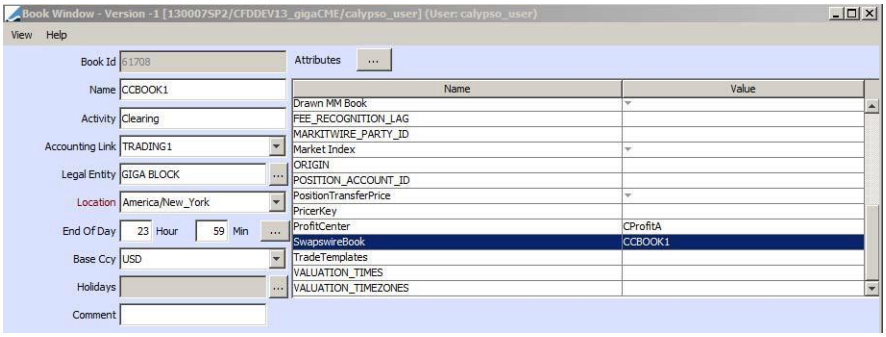
Step 13 – While importing the child allocation trade, if the block (parent) trade is not found then we check the domain "SkipAllocatedParentExistsValidation". If it contains Value = true, we save the allocation child trade as a new trade in Calypso. Otherwise, an error indicates that the parent trade cannot be found, and the child trade is not saved.

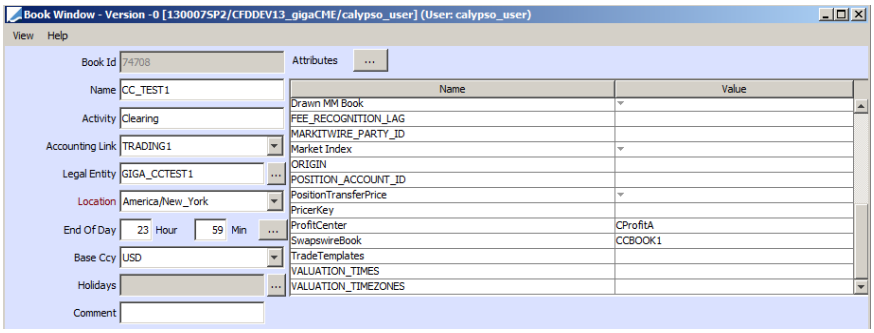
If the parent trade is found, the trade keyword SWBlockDealId is set on the child trade to the parent MW deal id.

Step 14 - The Allocation can be performed as Executing Broker, in which case, a Legal Entity Allocation will take place to mirror an external allocation to the counterparties entities. You will need to populate the SwapswireParticipant attribute of each counterparty entity, (block and fund) with their BIC code so they appear as counterparty in the trade as in the example below:

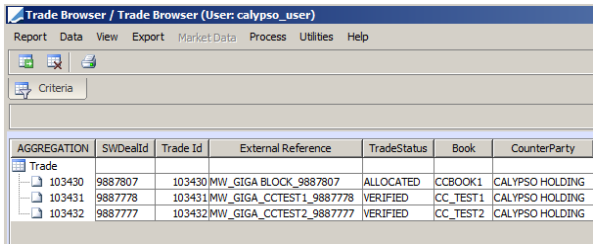


The Allocation can also be performed with a Client role. In that case a Book allocation will be performed to mirror an internal Fund allocation. You will need to map Calypso books for each book from MarkitWire. If the MarkitWire book is the same for both the block trade and the child trades, then the book in Calypso will be chosen based on the incoming Fund BIC Code:





Trades will be allocated as in the example below:



If you are using the bidirectional mode, the allocation should be submitted from the Calypso allocation GUI. The resulting Allocated trade should then be alleged to MarkitWire by re-saving it using a transition containing the PlatformAllege rule. Once the Counterparty has affirmed the allocation, (SWProcessState is Done on the block trade), then the trade should be released using a transition containing the PlatformRelease rule for the child trades to be updated with MarkitWire trade IDs.

- Please refer to the Data Uploader documentation to get more details on the allocation feature.

BackLoading Support

Note: The CSV format back loading using the back loading report and Deal Matcher is deprecated because it is no longer supported by MarkitWire. It will be removed in the subsequent release..

We have added support for importing New-Match trades from MarkitWire.

BackLoading is being done as follows:

Step 1 - Create a trade in Calypso manually.

Step 2 - Update the Calypso trade id in the MarkitWire trade in Internal Data Tab in field – "Internal trade ID".

Step 3 - MW sends a New-Match message and we create a separate trade in Calypso and update that until released.

Step 4 - Once released we perform the backloading action which includes:

a. Novate the manually created trade with the incoming data.

b. Cancel the existing trade that is created as part of Step (2).

c. Subsequent Amends will be applied on Novation-Child trade created as part of (3). (a).

Alternatively check the "Backloading" flag in the MarkitWire "Processing" tab and set the originating event as "Backload". So the trade will be imported in Calypso and will have these values set in the trade keywords.

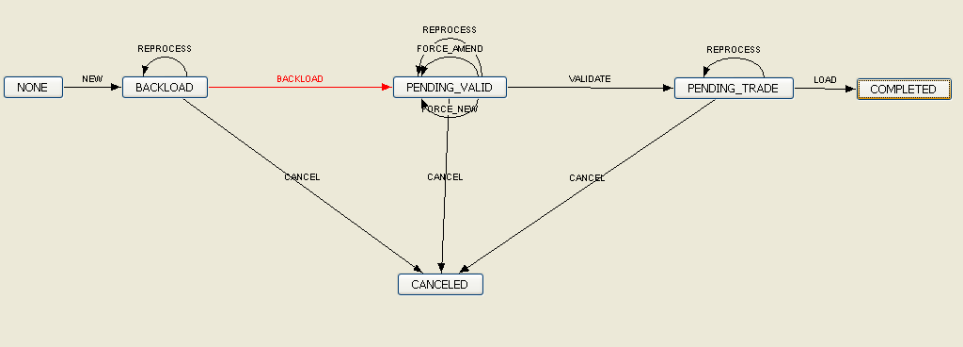
▶ Please refer to MarkitWire Backloading API Cookbook.doc for installation and execution instructions. This document can be downloaded from the MarkitWire website (https://secure.swapswire.com/login.asp) using your login credentials.

4.1 Message Workflow Configuration

You must add a new Transition state for BackLoading to work as shown in the picture and workflow configuration below.

Note: You must remove the CheckLink message rule from PENDING_VALID-VALIDATE-PENDING_TRADE and add a DataBackLoad rule for the BackLoading process.

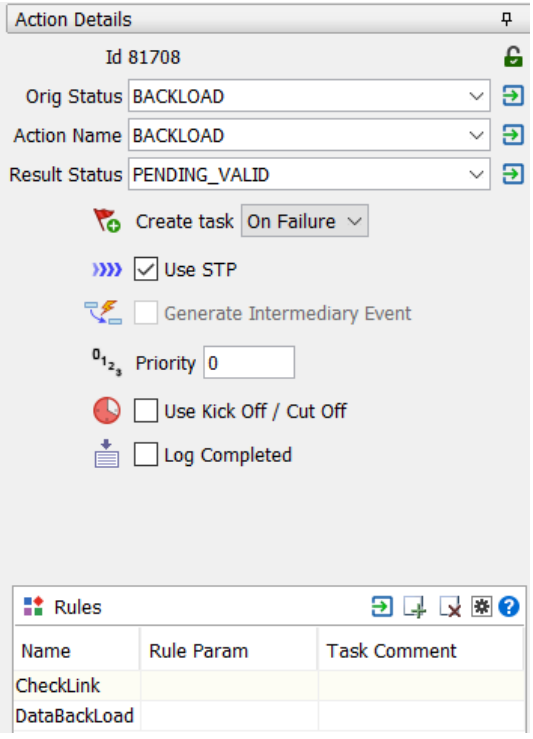
The following is a sample message workflow to support BackLoading included in "`<calypso home>/client/resources/GATEWAYMSG_BACKLOAD.wf`".



Ensure that the CheckLink, DataBackLoad message rule sequence is present:

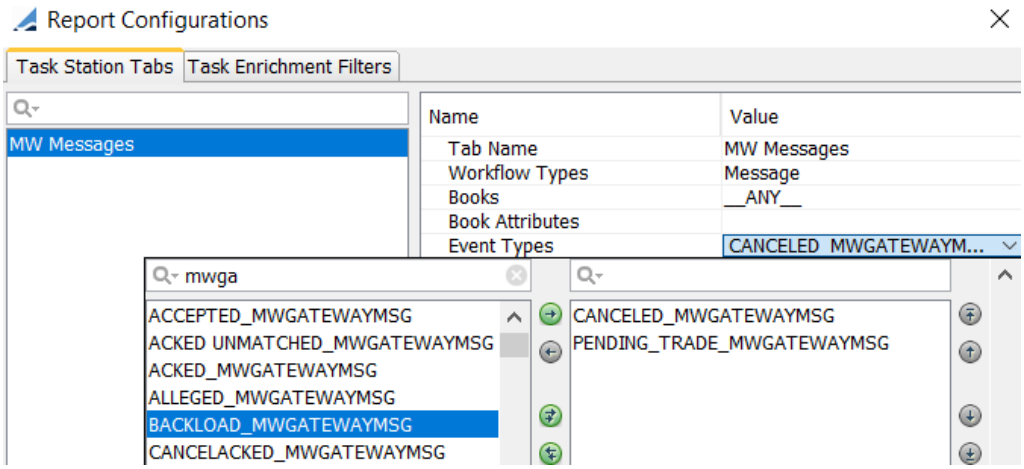
Id	Orig Status	Action	Resulting Status	Different User	Use STP	Log	Subtype	Product Type	Rules	Processing Org	Kick Off/ Cut Off
4702	BACKLOAD	BACKLOAD	PENDING_VALID		✓		GATEWAYMSG	ALL	CheckLink,DataBackLoad	ALL	
5102	BACKLOAD	CANCEL	CANCELED				GATEWAYMSG	ALL	CancelCleanUp	ALL	
5101	BACKLOAD	REPROCESS	BACKLOAD				GATEWAYMSG	ALL	ReMap	ALL	
4701	NONE	NEW	BACKLOAD		✓		GATEWAYMSG	ALL		ALL	
1002	PENDING_TRADE	CANCEL	CANCELED				GATEWAYMSG	ALL	CancelCleanUp	ALL	
1003	PENDING_TRADE	LOAD	COMPLETED		✓		GATEWAYMSG	ALL	CheckLink,Loader	ALL	
1004	PENDING_TRADE	REPROCESS	PENDING_TRADE				GATEWAYMSG	ALL	ReMap	ALL	
1005	PENDING_VALID	CANCEL	CANCELED				GATEWAYMSG	ALL	CancelCleanUp	ALL	
5301	PENDING_VALID	FORCE_AMEND	PENDING_VALID				GATEWAYMSG	ALL	ForceAmend	ALL	
3501	PENDING_VALID	FORCE_NEW	PENDING_VALID				GATEWAYMSG	ALL	ForceNew	ALL	
1006	PENDING_VALID	REPROCESS	PENDING_VALID				GATEWAYMSG	ALL	ReMap	ALL	
1007	PENDING_VALID	VALIDATE	PENDING_TRADE		✓		GATEWAYMSG	ALL	ValidateMap	ALL	✓

Ensure that the CheckLink,DataBackload Workflow action is present:



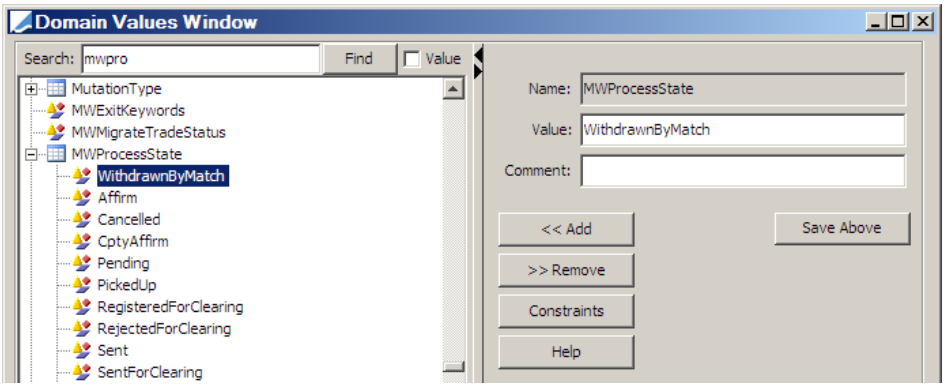
4.2 Task Station Configuration

You can add the BACKLOAD_GATEWAYMSG event type to the Task Station Markitwire report.



4.3 Pre-Release Notification

If you listen to pre-release deals, you must add the **WithdrawnByMatch** status to the **MWProcessState** domain. Deals that are withdrawn because of matching a processState notification are known as **WithdrawnByMatch**.



Also, ensure that the **New-Match Contract State** is present in the **MWContractState.PreRelease** domain.

4.4 General Notes Important Points

- Users are responsible for creating all trade filters and static data filters for report data.
- All cancel transitions must have a **Cancel** message rule.

Clearing Support

Direct Clearing with LCH is supported for initial trade entry and lifecycle with client and executing broker roles.

Client Clearing with LCH is supported for client, clearing broker, and executing broker using the LCH trilateral booking model or bilateral FCM model.

Clearing with CME is supported for initial trade registration as subsequent lifecycle actions are not available in MarkitWire for CME.

Direct and client clearing processing for CCP using the bilateral model is supported.

Note: Use of the Clearing Module requires a specific Clearing License. Please contact your account representative for further information.

5.1 Scope

The following products available for LCH clearing are supported:

- IRS
- ZC IRS
- Basis Swap
- OIS
- FRA
- Adjusting Notional Swaps

In addition, all products supported by MarkitWire for CME Client clearing are supported:

Model	Products	Role			
		Client	Clearing Broker	Executing Broker	Exchange
OTC Bilateral	Imported for Rate products (IRS, NDS, Cap/Floor, FRA, Swaption, Basis Swap, OIS, ZC Swap) via the MarkitWire Interface.	Full Support	Full Support	Full Support	Not needed.
LCH Trilateral	Clearing for MarkitWire for Rate products.	Full Support	Full Support	Full Support	No Support
CME Bilateral		Full Support	Not needed	Full Support	Full Support

Model	Products	Role			
LCH FCM Bilateral	LCH: IRS, OIS, ZC IRS, Basis swaps, Adjusting Notional Swaps, FRA CME: IRS	Full Support	Not needed	Full Support	No Support

5.2 Setup Instructions

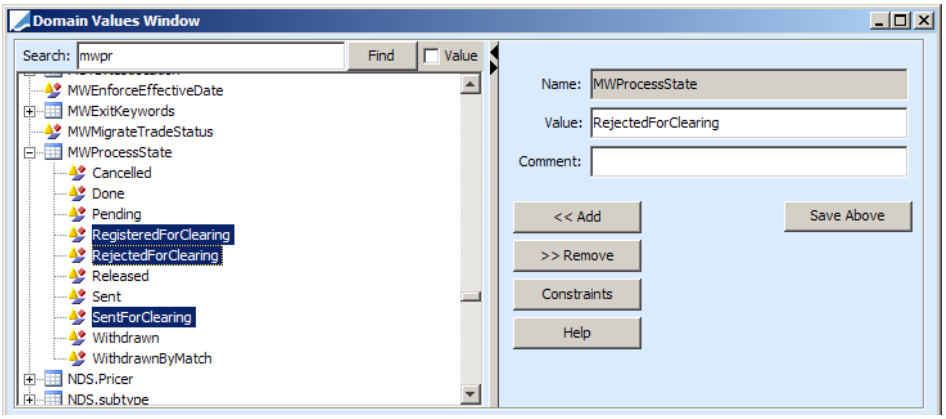
Note: SchemaData synchronization is required to add keywords and additional data.

Note: Following domain value has been removed from the SwapswireSchemaData.xml: MWProcessState|PickedUp, MWProcessState|Done, MWProcessState|Affirm, MWProcessState|Pending. Users can add these values manually in domain values, in case needed.

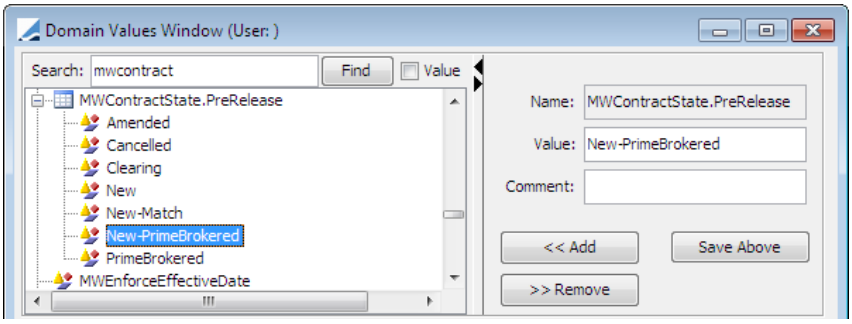
5.2.1 MWProcessState and MWContractState.PreRelease

For clearing to work you must add the following three process states to MWProcessState domain:

- RegisteredForClearing
- RejectedForClearing
- SentForClearing



Additionally, to process pre-release notification for clearing, you need to add “Clearing” value in the MWContractState.PreRelease domain:

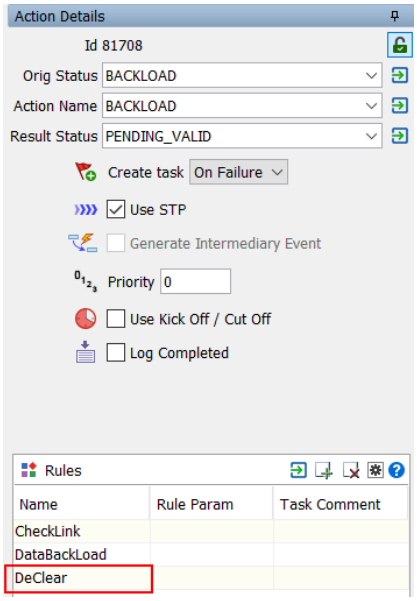


If your implementation does not require processing for Pre-Release notifications, simply add **RegisteredForClearing** in **domainName**.

5.2.2 Message Workflow Configuration

Applying a lifecycle event on a Cleared trade in MarkitWire declears the deal, making it once more a deal between the original counterparty and the processing organization. The next action is then applied on the decleared trade. Calypso's version of processing a lifecycle event on Cleared trade (i.e., declearing) is done by novating the trade from the clearing house back to the original counterparty trade, and then applying the lifecycle action on trade.

Calypso requires the **DeClear** message rule in the BACKLOAD- BACKLOAD-PENDING_VALID Transition. Use the Workflow Action window to accomplish adding the **DeClear** message rule:



Trade Keywords for Clearing

The following Calypso Trade keywords support MarkitWire clearing functionality:

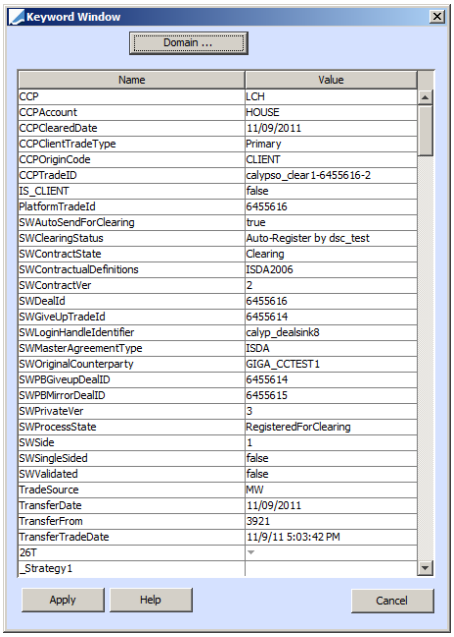
- SWMasterAgreementType
- SWContractualDefinitions

- SWAutoSendForClearing
- SWEligibleForClearing
- SWSendForClearing
- SWSendForClearingTimeStamp
- SWClearingStatus
- SWOriginalCounterparty

The following Clearing-related keywords are populated by MarkitWire:

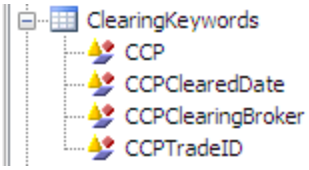
- CCP - Identifies the Clearing House.SWContractualDefinitions.
- CCPAccount - Identifies the account type at the CCP (CLIENT or HOUSE).
- CCPClearedDate - Date of clearing registration.
- CCPClientTradeType - Set to “Primary” if the first novated trade resulting from clearing or “Secondary” if a cloned trade in LCH booking model.
- CCPOriginCode - Set to HOUSE for Direct trades and CLIENT for Client Clearing trades.
- CCPTradeID - Trade ID at the Clearing House.
- CCPClearingBroker - Clearing broker when present in CME workflow.
- IS_CLIENT - Set to true is trade is related to client activity or false, otherwise.
- CCPStatus - Clearing status of trade sent for clearing.
- CCPMessageTimeStamp - Time stamp of last clearing message.

It is possible to track the clearing process state using the SWProcessState trade keyword:



5.2.3 Clearing Keyword Handling

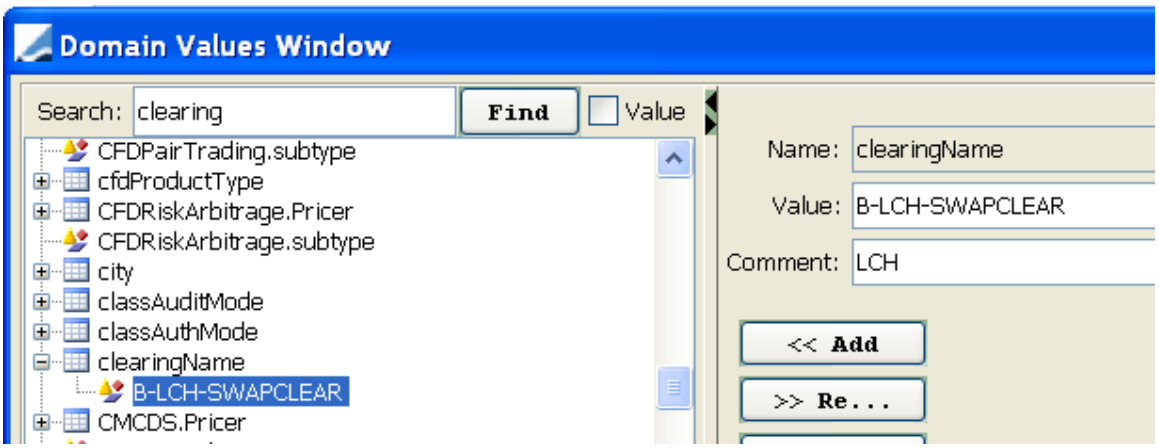
Keywords in the **ClearingKeywords** domain are not populated on bilateral trades even after clearing and are blanked out if the bilateral trade originates from a cleared trade, after a declear, for example.



5.2.4 Configurable CCP Keywords

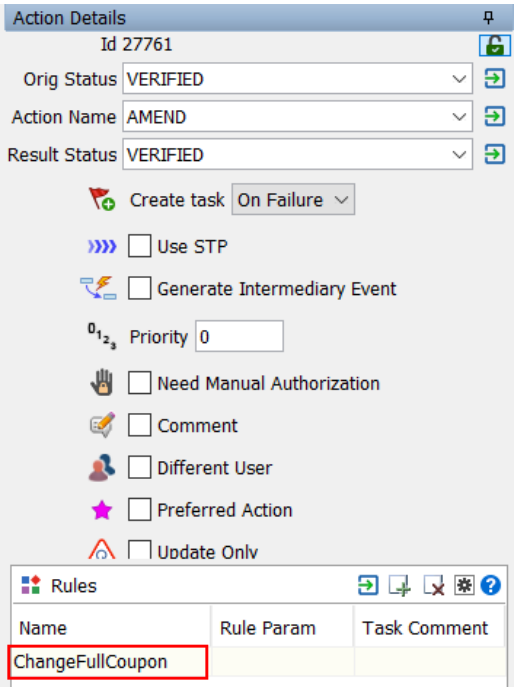
The interface populates the CCP keyword value based on value of the **clearingName** domain. You must map short name of the Clearing House (i.e., the Calypso legal entity) to clearing house abbreviations (LCH, CME, etc.). This is required for MarkitWire to coexist with Calypso's clearing offering - in particular the “Clearing Member Module”.

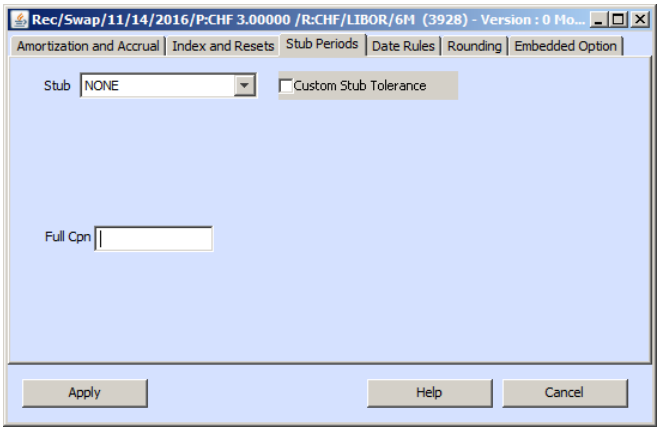
If the domain value is missing, the application uses the value of the Legal Entity short name.



5.3 ChangeFullCoupon Rule

To reset the **Full Cpn** field of an OTC trade to blank after an amendment and declear of a cleared trade, you must add the ChangeFullCoupon rule to your AMEND transition as shown below:





This Workflow rule is useful when the field is populated during the clearing novation.

5.4 CCP interest

There is a specific interest settlement process for Swap trades that have been submitted for clearing to the CCP:

For T+1 currencies

- if Trade is Cleared in T and Interest is settling in T, then the interest should remain on the bilateral trade that is being terminated.
- if Trade is Cleared in T and Interest is settling in T+1 or after, then the interest should settle on the Cleared trade.

For T+2 currencies

- if Trade is Cleared in T and Interest is settling in T, then the interest should remain on the bilateral trade that is being terminated.
- if Trade is Cleared in T and Interest is settling in T+1, then the interest should remain on the bilateral trade that is being terminated.
- if Trade is Cleared in T and Interest is settling in T+2 or after, then the interest should settle on the Cleared trade.

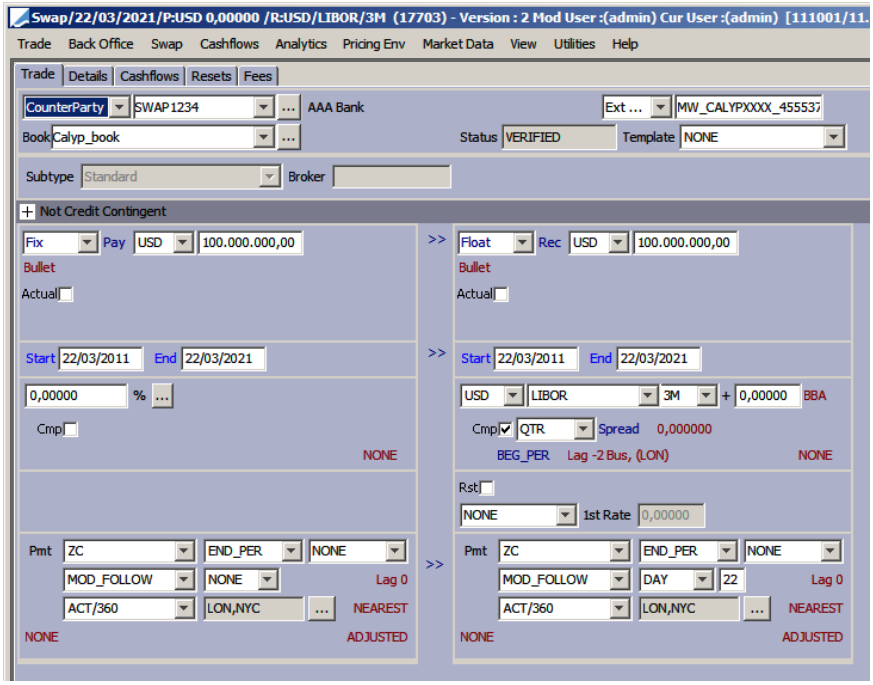
Added a new keyword TransferInterestToClearedTrade that is used by the termination window. If set to true, the last interest will settle on the Cleared trade and if it is set to false the interest will settle on the bilateral trade.

5.5 Sample Trade Flow

This section describes direct clearing deal entry in MarkitWire and the corresponding trade changes that occur in Calypso. It also describes the manual process of sending deal for clearing. Selecting the AutoSendForClearing checkbox performs these steps automatically.

Step 1 - A trade is affirmed and released by both counterparties.

Step 2 - This trade is saved in Calypso with the existing interface:



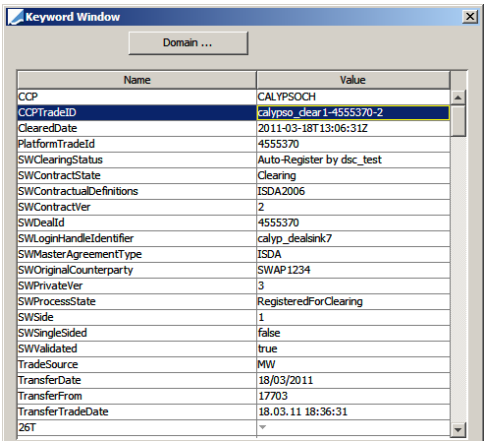
Step 3 - At this point, both counterparties submit the trade for clearing by selecting the Clearing choice from the MarkitSERV GUI.

Step 4 - Once both parties have sent the trade for clearing, the original trade is amended in Calypso with the S**WProcessState** keyword set to **SentForClearing**. At this point, a second version of the trade is created in MarkitWire against the Clearing House. However, the deal status is “Pending” and the deal has not been accepted yet by the Clearing House.

Step 5 - Once the Clearing House accepts the deal, the Booking State changes to “Released” in MarkitSERV.

Step 6 - Because the Clearing House has accepted the deal, Calypso performs a counterparty novation with the Clearing House as the new counterparty. Calypso terminates the original trade.

Step 7 - At this point, the trade keywords match the new Processing field values in the SWML file for the second version of the trade:



Step 8 - Calypso stores the Original counterparty (Novation Transferor) from the parent trade in the **SWOriginalCounterparty** keyword in the child trade.

Step 9 - Fees on the child trade are propagated from the parent trade if the fee type is in the **propagateFees** domain.

Step 10 - If the deal is rejected by the clearing house, the novation will not occur in Calypso and the booking state in MarkitWire changes to Withdrawn. In addition, the original trade is then amended and the SWProcessState keyword in Calypso now has the value Withdrawn. The **SWClearingStatus** keyword describes the reason that the clearing was rejected.

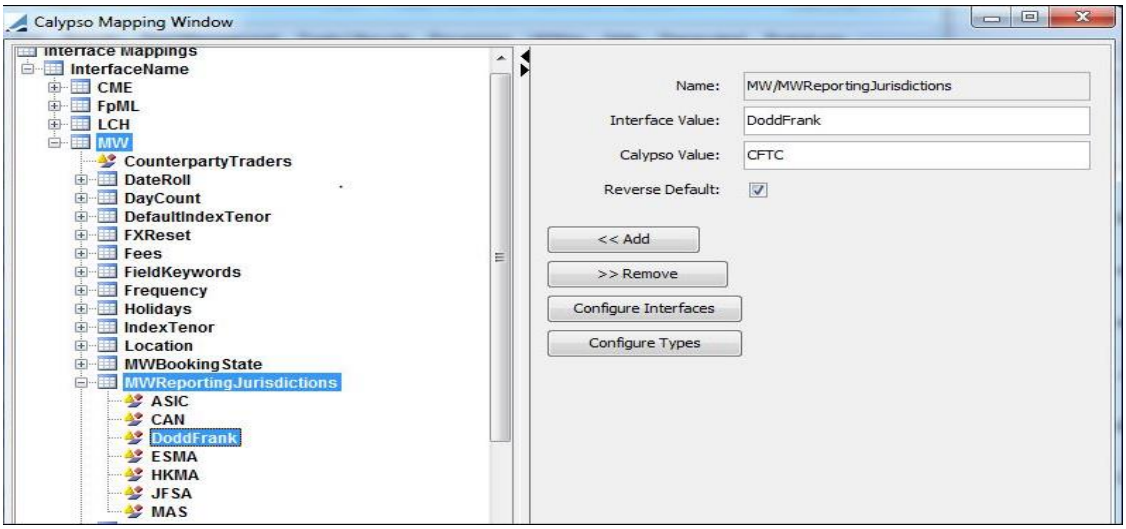
5.6 Regulatory Support

5.6.1 Support for the Reporting Jurisdictions

We support the following jurisdictions from MarkitWire for regulatory reporting perspective:

- CFTC Commodity Futures Trading Commission
- ESMA European Sales and Marketing Association
- HKMA Hong Kong Monetary Authority
- ASIC Australian Securities and Investments Commission
- MAS Monetary Authority of Singapore
- JFSA Japan Financial Services Agency
- CAN Canadian jurisdiction

For adding this new Jurisdiction in Calypso, mapping has to be provided in Calypso Mapping Window (screenshot added below). Default values get populated after running Execute SQL.



Calypso also supports separate UTI for Cap / Floor leg of CapFloor Straddle and Payer/Receiver leg for Swaption Straddle. This UTI keyword will be based on reporting Jurisdictions. For example, if it is CFTC jurisdiction then the keywords will be ReportingCFTCLeg1UTIPrefix, ReportingCFTCLeg1UTIValue, and so on. Similarly, if it is ESMA jurisdiction then the keywords will be ReportingESMALeg1UTIPrefix, ReportingESMALeg1UTIValue, and so on.

5.6.2 MarkitWire's Cleared Trade

MarkitWire's Cleared Trade Unique Swap Identifier (USI) is used in the Trader Tracker GUI and is assigned by the CCP. In the Trader Tracker, the USI field of trade both before and after clearing does not change. The USI provides a means to track a swap transaction throughout its lifecycle.

MarkitWire's Cleared Trade Global Unique Trade Identifiers (UTI) will be populated with the Global UTI if it is sent by the CCPs.

Note: To begin populating reporting keywords/attributes such as USIPrefix and USIValue in Calypso, you must contact MarkitWire support to request that they enable "output reporting tags" for the appropriate User ID(s).

Also, ensure that the IncludeTRReportingInfo parameter in `calypso_SW_config.properties` is set to true.

```
#This flag is used when user needs DF Reporting in SWML.

IncludeTRReportingInfo=true
```

Calypso will support the following Global UTI keywords:

- GlobalUTI
- GlobalBlockUTI
- GlobalPriorUTI

The above values would come from the Trades which will be received by the CCP when MarkitWire sends the trade for Clearing. As part of the CLEAR accept the CCP would need to add the value in the ClearedTradeGlobalUTI keyword, which will be sent to MarkitWire as part of the Clearing acknowledgement.

NOTE: The existing USI Trade keywords will also be supported along with Global UTI support.

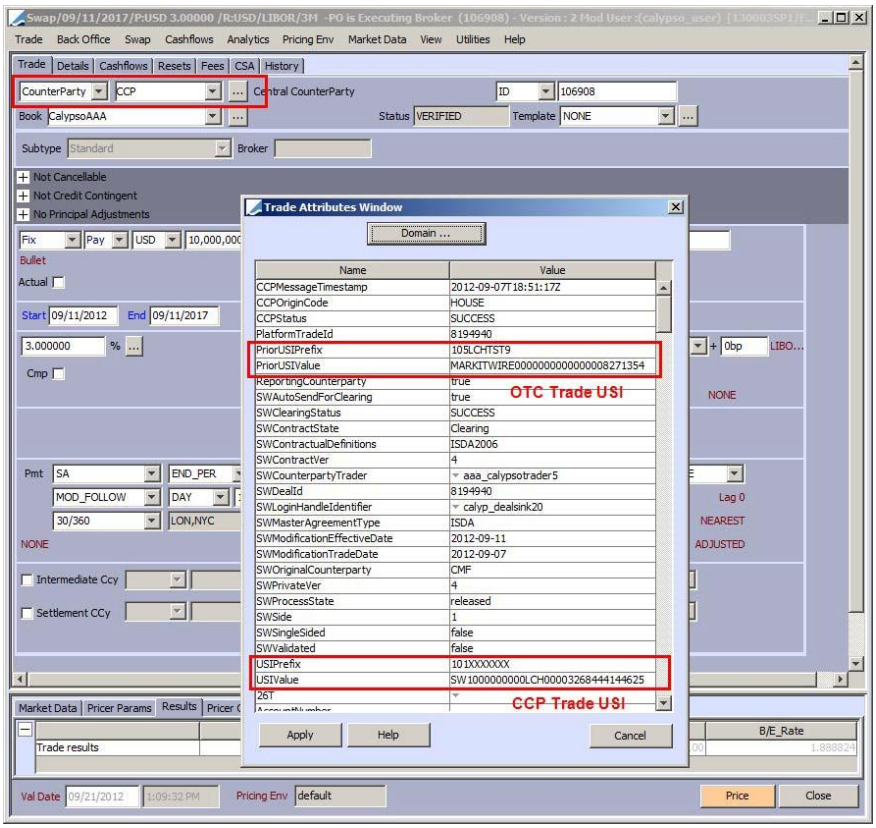
The following Trade Keywords are imported by Calypso:

- USIPrefix - The first component of the USI generated by the SEF for an OTC trade.
- USIValue - The second component of the USI generated by the SEF for an OTC trade.
- PriorUSIPrefix - The first component of a USI for a previously associated trade (for example after an OTC novation or compression).
- PriorUSIValue - The second component of a USI for a previously associated trade (for example after an OTC novation or compression).

In the Trader Tracker application, the USI assigned by the CCP is located on the DF Reporting tab in the Report Identifiers group. The USI does not change after clearing.

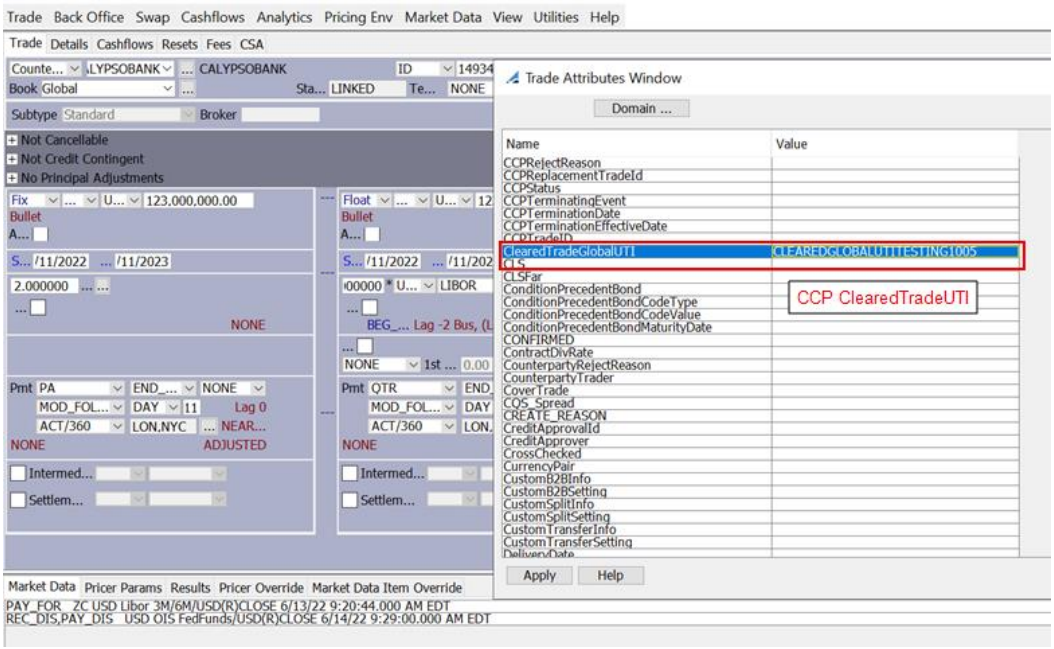
5.6.3 Cleared Trade USI

Calypso always populates the “USI” Trade Keywords with the current and latest USI version of the trade. Calypso populates the “Prior USI” Trade Keywords with the trade's USI prior to novation:



The Cleared Trade Global UTI will be copied to the Cleared Trade USI and Cleared Trade UTI fields and the corresponding fields on the Trade Window. Where CCPs continue to send a Cleared Trade USI or UTI to MarkitWire.

If a CCP does not provide a Global UTI then the field will remain empty, and the existing Cleared USI and Cleared UTI fields will show any values provided by the CCP.



5.6.4 Notes

Upon Clearing and Declearing lifecycles, the USI keywords always show the current and latest USI of the trade. Calypso's Prior USI keywords always show the USI values before novation. In other words:

- For a cleared trade, the USI keywords in Calypso are the latest USI values assigned by the CCP. The Prior USI keywords contain the USI from the original OTC trade.
- For an OTC trade (either new or subsequent to declearing or reclearing), USI keywords in Calypso are the latest USI values assigned by the SEF. Prior USI keywords contain the values present in the "prior USI" fields in MarkitWire, when present. However, if MarkitWire's prior USI field is empty. Calypso populates its own prior USI keywords with the USI value of the previously cleared version of the trade.

FCM/Clearing Broker Business Flow

6.1 FCM Pre-clearing Business Flow

6.1.1 Overview

Certain clearing houses require clearing broker acceptance prior to that same trade being submitted to them. Clearing Broker is known as FCM. Clearing take up prior to submission does exist on the trilateral client clearing workflow, it is required that this functionality is covered in the bilateral workflow.

- This specification only covers bilateral client clearing
- If trade not allocated then there can be up to 2 FCMs, but if the trade is allocated there will be up to 2 FCMs for each allocation split
- All products that are currently cleared via HKEX Rates will be eligible
- The workflow will support EB submission of the trade as well as Broker submissions
- New Contract State will be supported for Clearing Take Up flow
- FCMs should receive trades or splits id trade is allocated upon Client affirmation. The request to accept this trade will be done via the GUI
- FCM will only have the ability to 'Pickup',' Transfer', 'Accept', or 'Reject' trades via GUI
- FCM will only have the ability to 'Pickup',' Transfer', 'Accept', or 'Reject' via API
- If FCM is rejecting a trade – the entry of a status (rejection) message is mandatory

Out of Scope:

- Package Clearing
- Allocation
- Netting compression

6.1.2 FCM Pre-clearing Business Flow

1. Bi-lateral trades done between the Bank and international bank.

2. Clearing request submitted by both the Bank and international bank via MarkitWire.

3. Clearing broker checks the credit limits before accepting the request.

4. Once accepted, MarkitWire sends the matched record to OTC Clear for product, margin and credit checks.

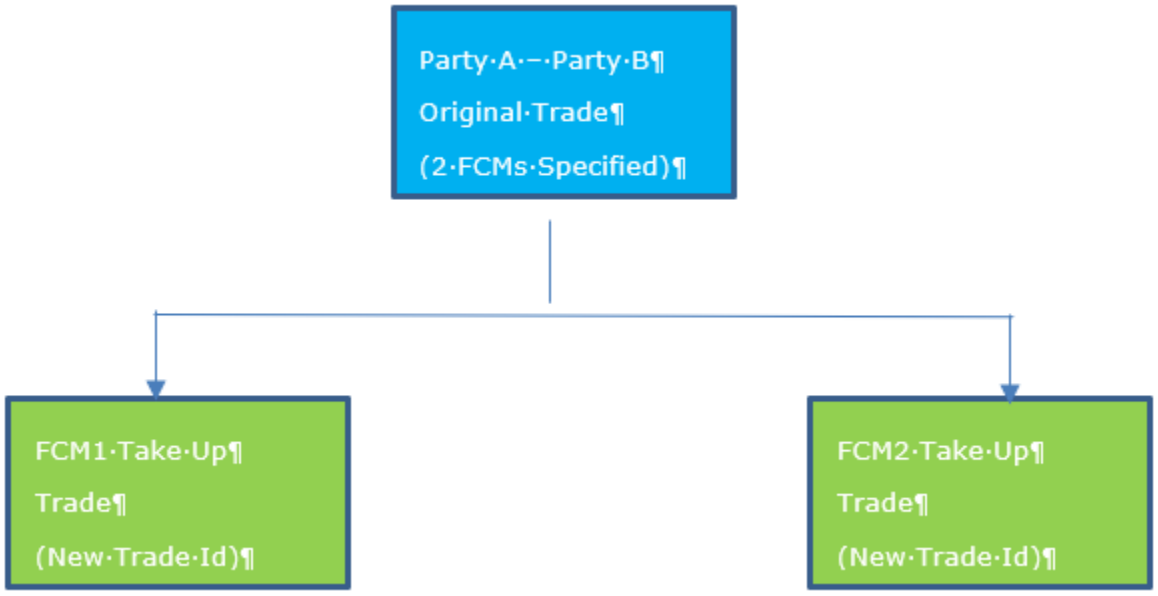
5. After the trade is accepted by registration, OTC Clear will inform Clearing Broker, Bank and International Bank about the trade clearing status through Markitwire.

6.1.3 Trade Flow

Following Lifecycle Events are supported:

- Take Up Trade
- Accept
- Reject

The Original trade between Party A and Party B will be separated into child trades in the following way: If only one FCM is specified– there will only be one new trade created. If there are 2 FCMs specified a trade will go to both FCMs and there will two new trades created.



Accept Trade Flow:

- Trade is separated into child Clearing Take Up trades
- Original trade (A) is in Clearing Pending status
- For each Clearing Take Up trade the new status will be Clearing Take Up Pending
- Once the trade is accepted by one of the FCMs the trade will stay as Clearing Take Up Pending
- Once all B and C have accepted A is sent to Clearing
- The Original Trade is updated to a new status of Clearing Released
- Once the original trade is cleared, B and C (the Clearing Take Up trades) have a new status Clearing Take Up Done

For Accept trades the following statuses are expected to be seen

Contract State	Booking State	Trigger	Trade
Clearing	Pending	Original Trade will stay in this state until a response is received from Clearing House	A
Clearing Take Up	Pending	When new trades are created	B, C
Clearing Take Up	Pending	Once one of the FCM trades has accepted	B
Clearing Take Up	Pending	Once both of the FCM trades has accepted	B, C
Clearing Take Up	Released	Once Original Trade receives a message from Clearing House	A
Clearing Take Up	Done	Once a message is received from Original Trade A	B, C
Clearing Take Up	Released	Once the FCM trade is Released	B, C

Reject Trade Flow:

- Original trade is separated into child Clearing Take Up Trades
- Original trade (A) is in Clearing Pending status
- For each Clearing Take Up trade (B, C) there will be a new trade ID and the new status will be Clearing Take Up Pending
- If the trade is accepted by one of the FCMs the trade (B, C) will stay as Clearing Take Up Pending
- If the trade is rejected by any of the FCMs the trades (B, C) will move into Withdrawn- Clearing Take Up (Message will be sent to other party's FCM)
- A is not sent to Clearing
- The Original Trade (A) is updated to new status of Clearing Withdrawn

For Reject trades the following statuses are expected to be seen

Contract State	Booking State	Trigger	Trade
Clearing	Pending	Original Trade will stay in this state until a response is received from Clearing House	A
Clearing Take Up	Pending	When new trades are created	B, C
Clearing Take Up	Pending	Once one of the FCM trades has been accepted	B, C
Clearing Take Up	Withdrawn	Once FCM A or B rejects trade (counterparty FCM will be	B, C

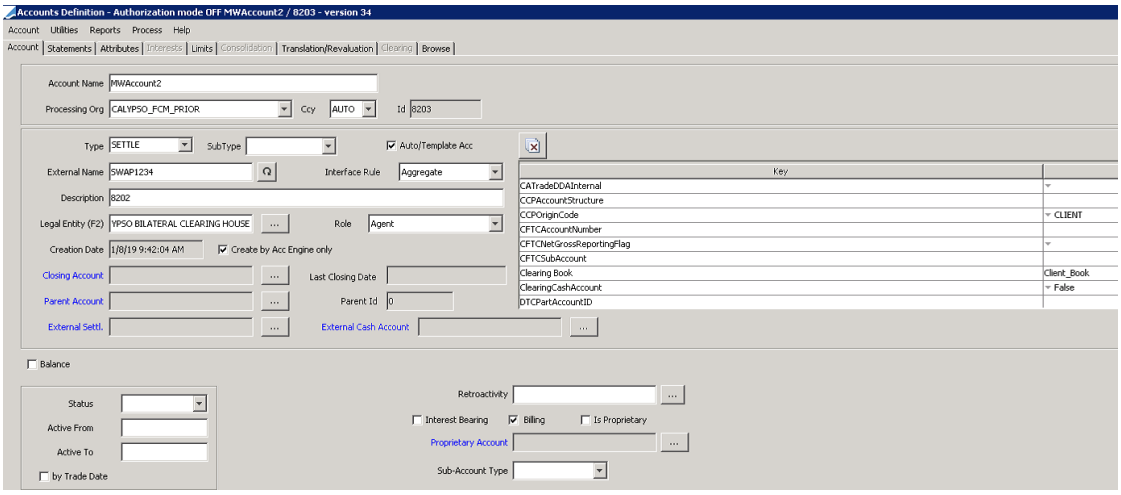
Contract State	Booking State	Trigger	Trade
		notified and both will be rejected)	

6.1.4 Configuration and Setup

Step 1 – Accounts and attribute setup

1. CCP facing account:

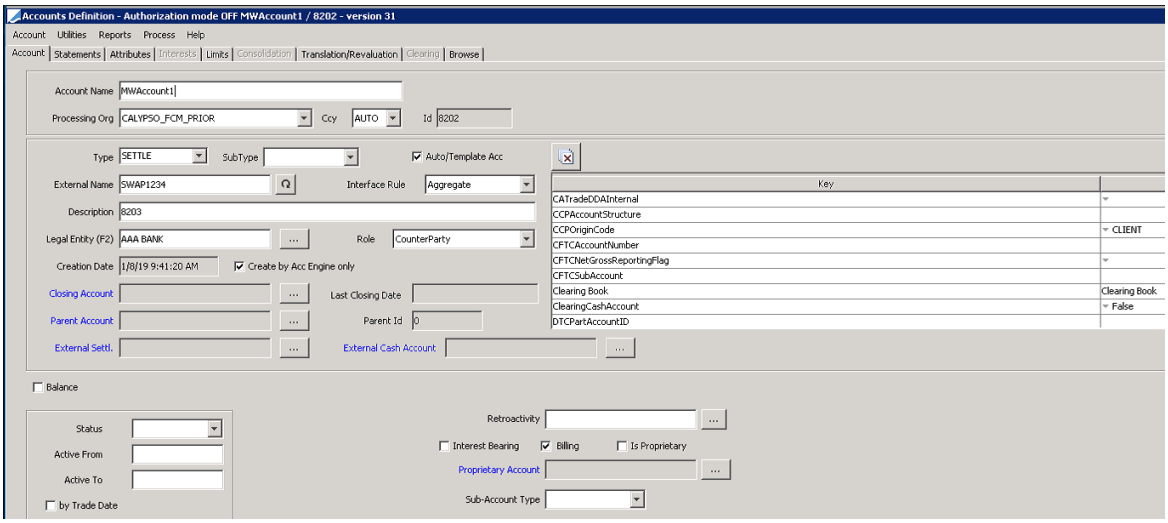
- **Processing Org**: Legal Entity configured in calypso with FCM BIC code, see STEP2 -1
- **External name**: Must be the client BIC code for which this FCM instace receives deals
- **Legal Entity**: Legal Entity configured in calpso with HOUSE BIC code, see STEP2- 3
- **Type**: SETTLE
- **Role**: Agent
- **Description**: Mirror account ID facing the client
- **Attributes**: CCPOriginCode = “CLIENT”, Clearing Book = <BOOK_NAME> (more details for this attribute go to STEP4)



2. Client facing account:

- **Processing Org**: Legal Entity configured in calypso with FCM BIC code, see STEP2 -1
- **External name**: Must be the client BIC code for which this FCM instace receives deals
- **Legal Entity**: Legal Entity configured in calpso with HOUSE BIC code, see STEP2- 2
- **Type**: SETTLE

- **Role**: Counterparty
- **Description**: Mirror account ID facing the House
- **Attributes**: CCPOriginCode = “CLIENT”, Clearing Book = <BOOK_NAME> (more details for this attribute go to STEP4)

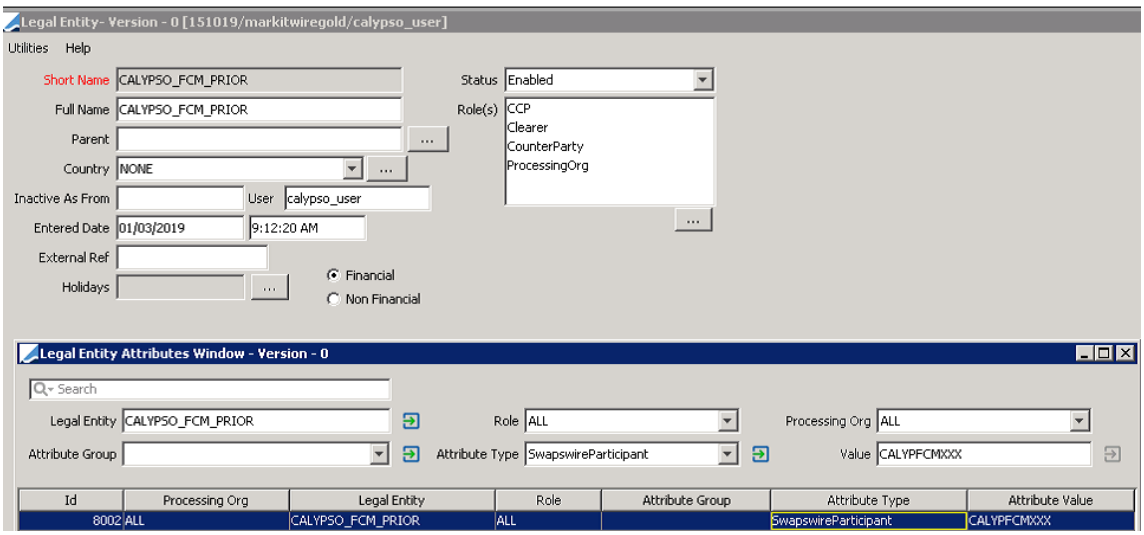


Step 2 – Legal Entity and attribute setup

1. FCM Legal Entity: This LE is used in CCP facing account.

Attributes:

- SwapswireParticipant: The FCM BIC code must be configured here



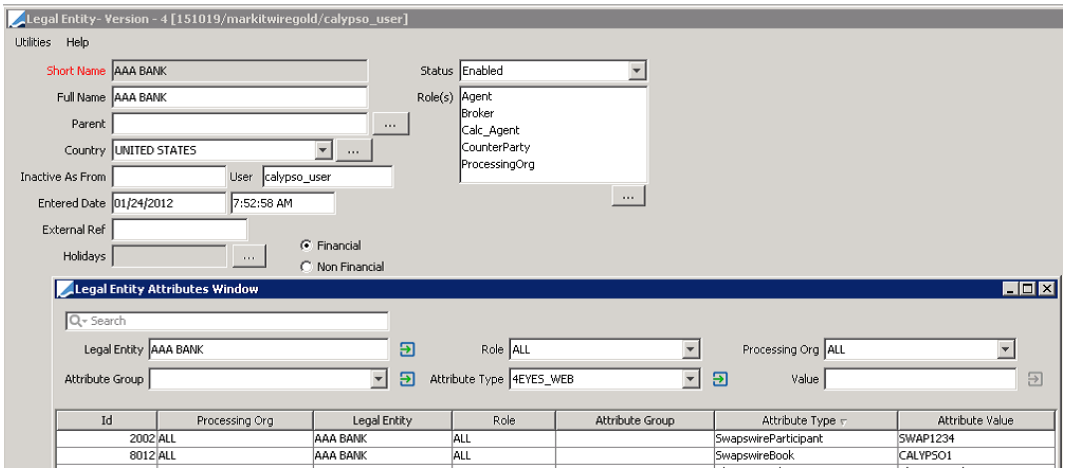
2. Account Legal Entity: This LE act as a client submitting its deal to FCM account.

Attributes:

- SwapswireParticipant: The client BIC code must be configured which comes under tag

```
<swClearingTakeup>
  <swClient>
    <partyId>SWAP1234</partyId>
    <partyName>AAA Bank</partyName>
    <tradeId>44445555</tradeId>
  </swClient>
  <swClearingHouseTradeId/>
</swClearingTakeup>
```

- SwapswireBook: MarkitWire book BIC code needs to be configured which is used when submitting deal to FCM

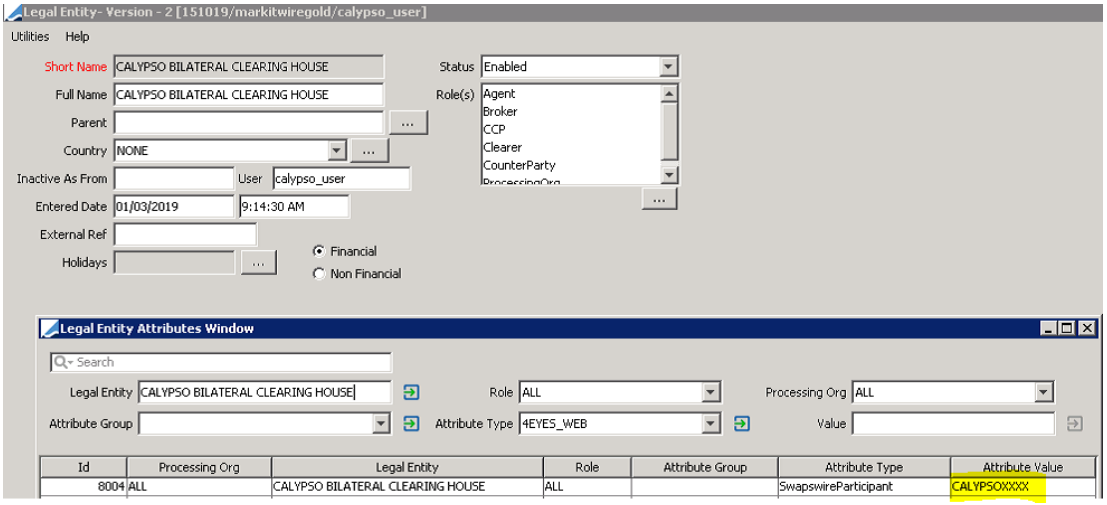


3. House Legal Entity: This LE acts as a clearing house which will perform clearing on once the deals are sent to it by FCM

Attributes:

- SwapswireParticipant: The house BIC code must be configured which comes under tag

```
<party id="partyA">
  <partyId>CALYPSOXXXX</partyId>
  <partyName>Calypso Bilateral Clearing House</partyName>
</party>
<party id="partyB">
  <partyId>CALYPFCMXXX</partyId>
  <partyName>Calypso_FCM_Prior</partyName>
</party>
</FpML>
```

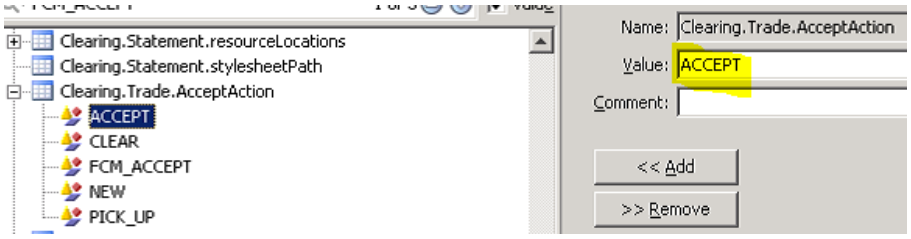


Step 3 – Domain setup

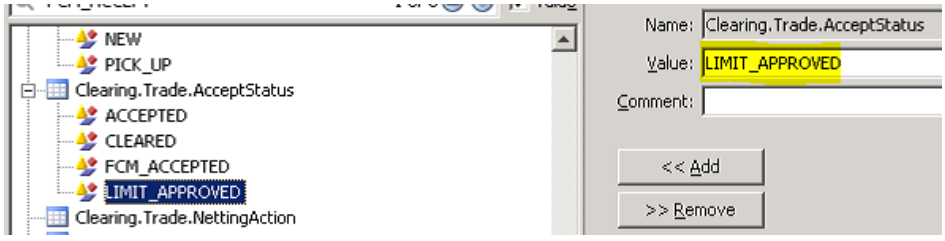
Acceptance, Rejection, and Release notifications to FCM/CCP are triggered by a combination of trade actions and the resulting trade status in Calypso. Configure the following domains as per the workflow followed for trades with the statues and actions to be applied during Accept/Reject/Release of trades from FCM.

Specimen values need to be added to the following domains:

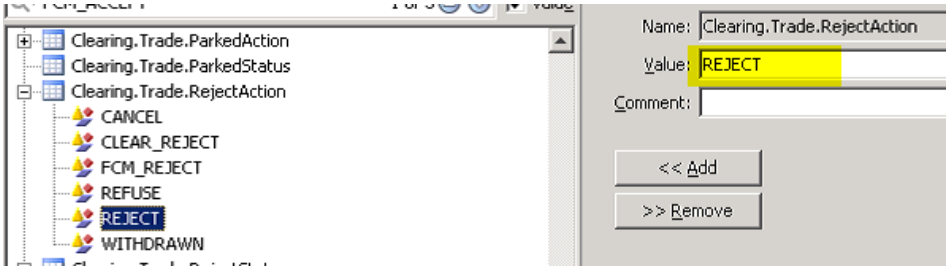
- Clearing.Trade.AcceptAction: ACCEPT, CLEAR



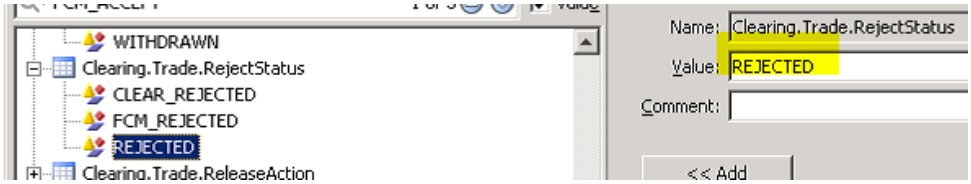
- Clearing.Trade.AcceptStatus: LIMIT_APPROVED



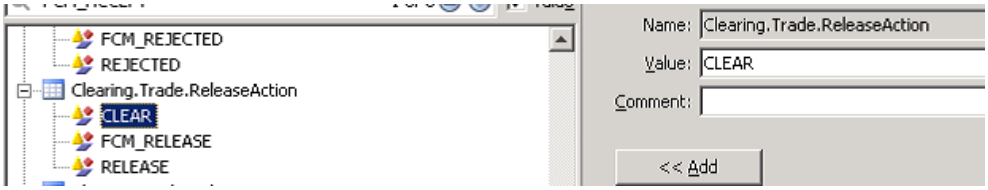
- Clearing.Trade.RejectAction: REJECT, REFUSE (this action can be further customized as shown in STEP5)



- Clearing.Trade.RejectStatus: REJECTED



- Clearing.Trade.ReleaseAction: RELEASE, CLEAR (this action can be further customized as shown in STEP5)



- Clearing.Trade.ReleaseStatus: VERIFIED

Step 4 – Book selection logic

Trading book selection logic is as follows:

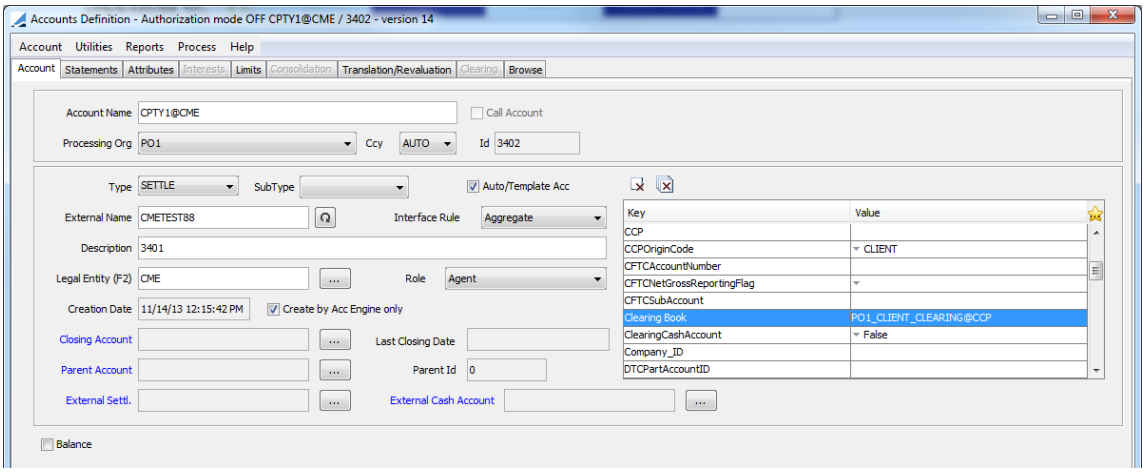
This applies to cleared trades as they come into Calypso from the CCP, Clearing Transfer Trades generated in the EOD process, and trades related to collateral management such as Collateral Exposures and Margin Calls.

The design is based on iteration of places to check for the Book Attribute to be defined. This hierarchy is described below.

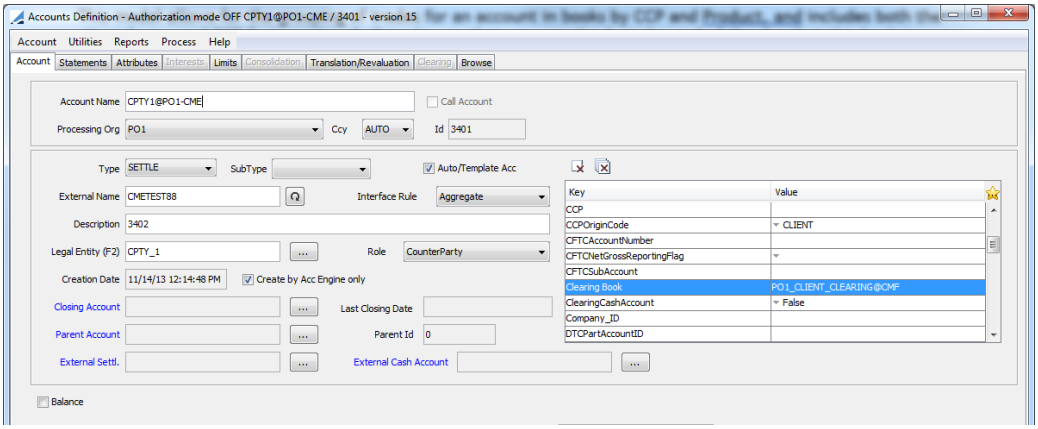
First System Checks for Clearing Account

This model allows for the grouping of trades for an account in books by CCP and Product, and includes both the client facing and CCP facing sides of all trades, so that you can check at the end of the day that this book is flat.

Create a "Clearing Book" attribute on the Clearing Account which expects a Book Name as the value. Use this book for any trades whose origin is associated with this account. **This attribute should be set on both the client facing and the CCP facing (mirror) clearing accounts.** When configured this way, it will include both the client facing and the mirrored CCP-facing versions of the cleared trades and clearing transfers as well as collateral related trades. If no attribute value is found on this account, go to next step.



Mirror Account Facing to Client can have different book defined:

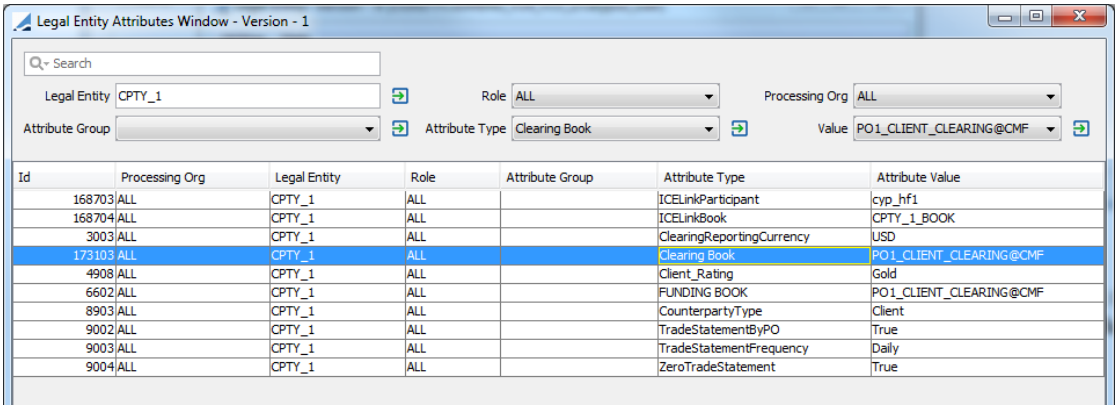


Second System Checks for Legal Entity

This model groups one account's trades across all CCP's and Products into a single book. It puts all CCP facing client trades for all accounts into another book, and all CCP facing house trades in another. It reduces the number of books.

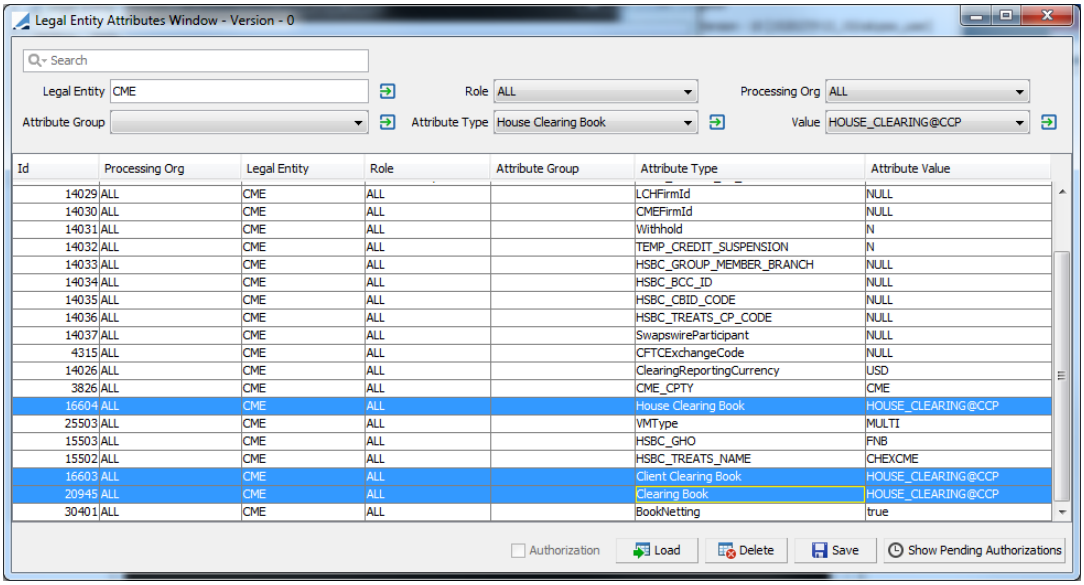
Use the existing Legal Entity Clearing Book attribute to determine the book for all client facing trades across all CCPs. Use the Client Clearing Book and House Clearing Book attributes for the Processing Org Legal Entity to determine the book for the CCP facing sides of all trades. If PO doesn't have attribute populated go to next step

Client/House LE:



Id	Processing Org	Legal Entity	Role	Attribute Group	Attribute Type	Attribute Value
168703	ALL	CPTY_1	ALL		ICELinkParticipant	cyp_hf1
168704	ALL	CPTY_1	ALL		ICELinkBook	CPTY_1_BOOK
3003	ALL	CPTY_1	ALL		ClearingReportingCurrency	USD
173103	ALL	CPTY_1	ALL		Clearing Book	PO1_CLIENT_CLEARING@CMF
4908	ALL	CPTY_1	ALL		Client_Rating	Gold
6602	ALL	CPTY_1	ALL		FUNDING BOOK	PO1_CLIENT_CLEARING@CMF
8903	ALL	CPTY_1	ALL		CounterpartyType	Client
9002	ALL	CPTY_1	ALL		TradeStatementByPO	True
9003	ALL	CPTY_1	ALL		TradeStatementFrequency	Daily
9004	ALL	CPTY_1	ALL		ZeroTradeStatement	True

CCP LE:



Id	Processing Org	Legal Entity	Role	Attribute Group	Attribute Type	Attribute Value
14029	ALL	CME	ALL		LCHFirmId	NULL
14030	ALL	CME	ALL		CMEFirmId	NULL
14031	ALL	CME	ALL		Withhold	N
14032	ALL	CME	ALL		TEMP_CREDIT_SUSPENSION	N
14033	ALL	CME	ALL		HSBC_GROUP_MEMBER_BRANCH	NULL
14034	ALL	CME	ALL		HSBC_BCC_ID	NULL
14035	ALL	CME	ALL		HSBC_CBID_CODE	NULL
14036	ALL	CME	ALL		HSBC_TREATS_CP_CODE	NULL
14037	ALL	CME	ALL		SwapswireParticipant	NULL
4315	ALL	CME	ALL		CFTCExchangeCode	NULL
14026	ALL	CME	ALL		ClearingReportingCurrency	USD
3826	ALL	CME	ALL		CME_CPTY	CME
16604	ALL	CME	ALL		House Clearing Book	HOUSE_CLEARING@CCP
25503	ALL	CME	ALL		VMType	MULTI
15503	ALL	CME	ALL		HSBC_GHO	FNB
15502	ALL	CME	ALL		HSBC_TREATS_NAME	CHEXCME
16603	ALL	CME	ALL		Client Clearing Book	HOUSE_CLEARING@CCP
20945	ALL	CME	ALL		Clearing Book	HOUSE_CLEARING@CCP
30401	ALL	CME	ALL		BookNetting	true

Third System Should Check CCP Legal Entity

This allows the CCP facing trades in the second model to be divided by CCP.

Use the existing House Clearing Book and Client Clearing Book attributes for the CCP Legal Entity to drive the book that the CCP facing trades are put into. A different book can be set for each CCP.

Assumptions:

A user will choose a consistent model across all accounts. If they configure the book on the clearing accounts for one account, we should assume that they will do it for all. If one account attribute is missing, we will go to the second check and put the trades into a book across all CCPs and there will be inconsistency due to user configuration error.

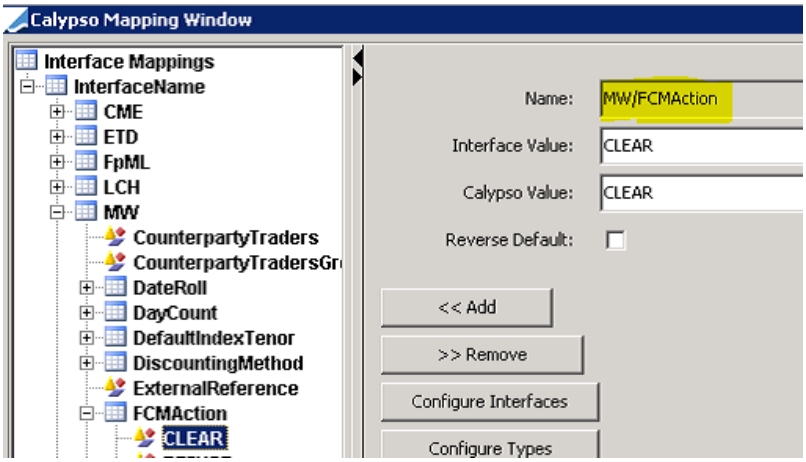
If a book is found in step 1, we will ignore any other book configuration under any of the LE's. If a book is not found in step one, we go to step 2, and stop there if the proper configuration is found. If not found in either check, we finally go to the third step.

Step 5 – Mapping Window

Following entries are added in mapping window to which can be used to add custom trade work-flow actions when clearing notification comes from CCP to the FCM calypso instance.

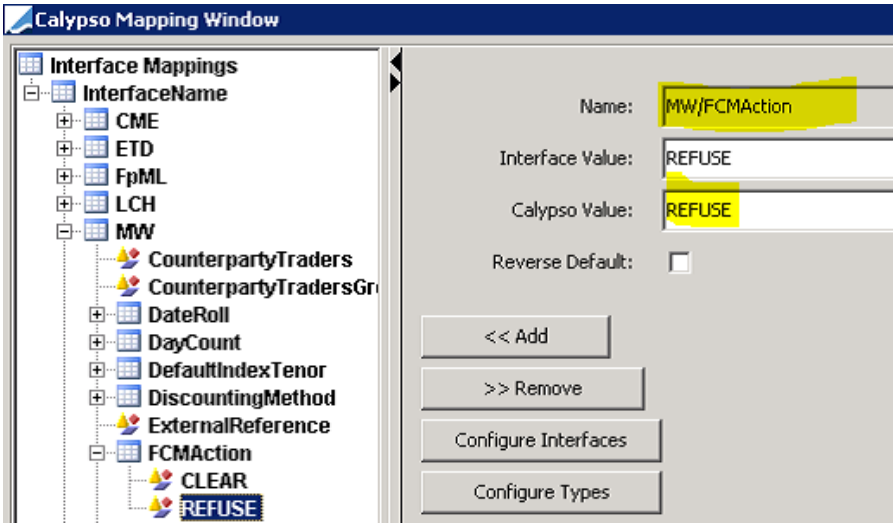
1. CCP clears bilateral trades:

FCM receives clearing notification and by default **CLEAR** action is applied however if we need to apply any custom action same need to be added into Calypso value of following mapping:



2. CCP reject bilateral trades:

FCM receives clearing withdrawn notification and by default **REFUSE** action is applied however if we need to apply any custom action same need to be added into Calypso value of following mapping:



6.1.5 FCM Trade keywords

KeywordName	KeywordValue
BusinessFlow	Taken from property BusinessFlow in “calypso_SW_config.properties”, or defaults to FCM if not set.
CCP	CALYPSO BILATERAL CLEARING HOUSE
CCPAccountReference	SWAP1234
CCPClearedDate	01-11-19
CCPMessageTimestamp	11-01-19 14:04
CCPOriginatingEvent	Trade
CCPOriginCode	CLIENT
CCPStatus	Cleared
CCPTradeID	33931
CCPRejectReason	FCM rejects deal comment
ClientTradeID	44463287
IS_CLIENT	FALSE
SentBy	SWAP1234
SentTo	CALYPFCMXXX
SWContractState	Clearing-Takeup
SWContractVer	1
SWDealId	44463293
SWEligibleForClearing	FALSE
SWLoginHandleIdentifier	calypso_fcm_dealsink1

KeywordName	KeywordValue
SWMasterAgreementType	ISDA
SWOriginalCounterparty	AAA BANK
SWPayLegSwapStreamId	fixedLeg
SWPrivateVer	5
SWProcessState	Released
SWRecLegSwapStreamId	floatingLeg
SWSide	2
SWSingleSided	TRUE
SWValidated	FALSE
TradeSource	MW

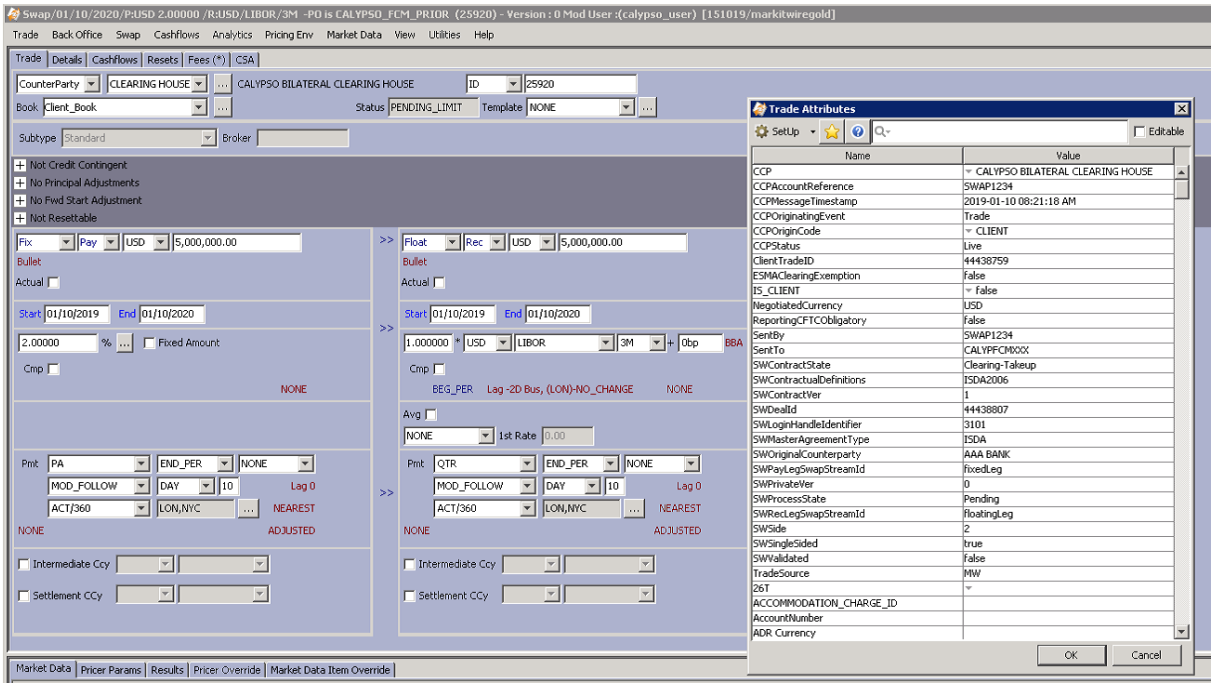
6.1.6 Calypso Trade for FCM Lifecycles

Note: These are specimen Calypso Trade representation with reference to FCM action performed at single side of a bilateral deal. The trade actions and status can differ based on the domain and trade work-flow setup.

FCM Incoming

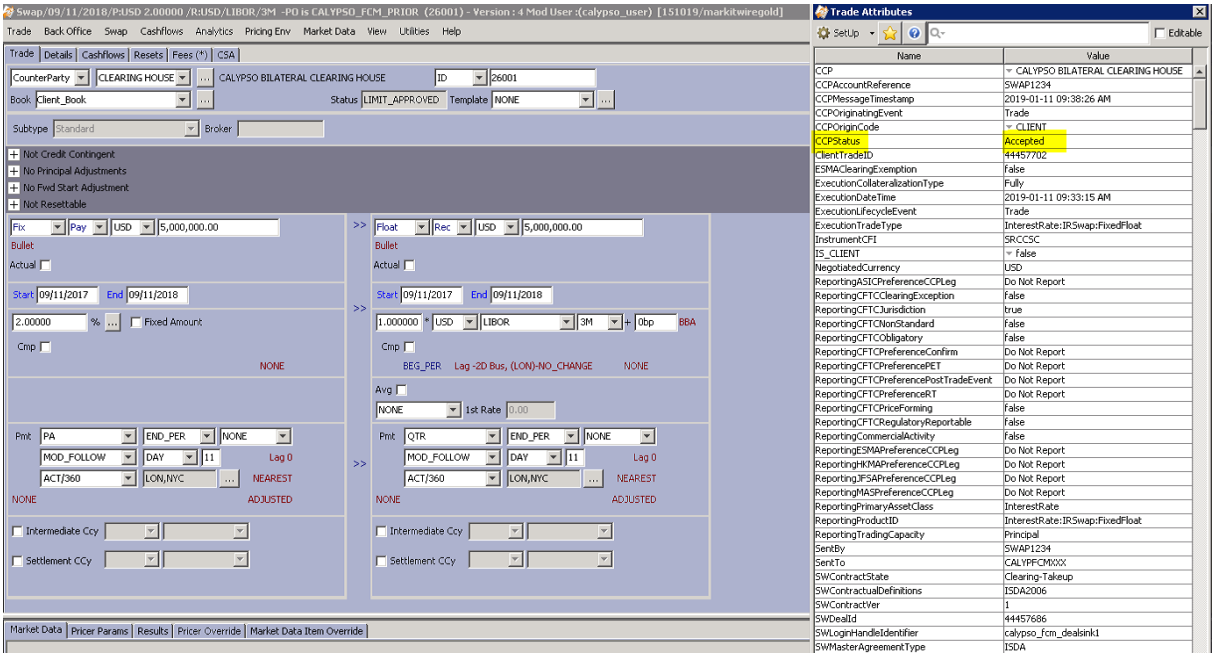
Once bilateral trades are affirmed and released by their corresponding dealer MarkitWire account FCM receives clearing notification.

The Calypso instance running listening to this FCM deal-sink will receive and process this notification. Trade representation for same in Calypso is as follows:



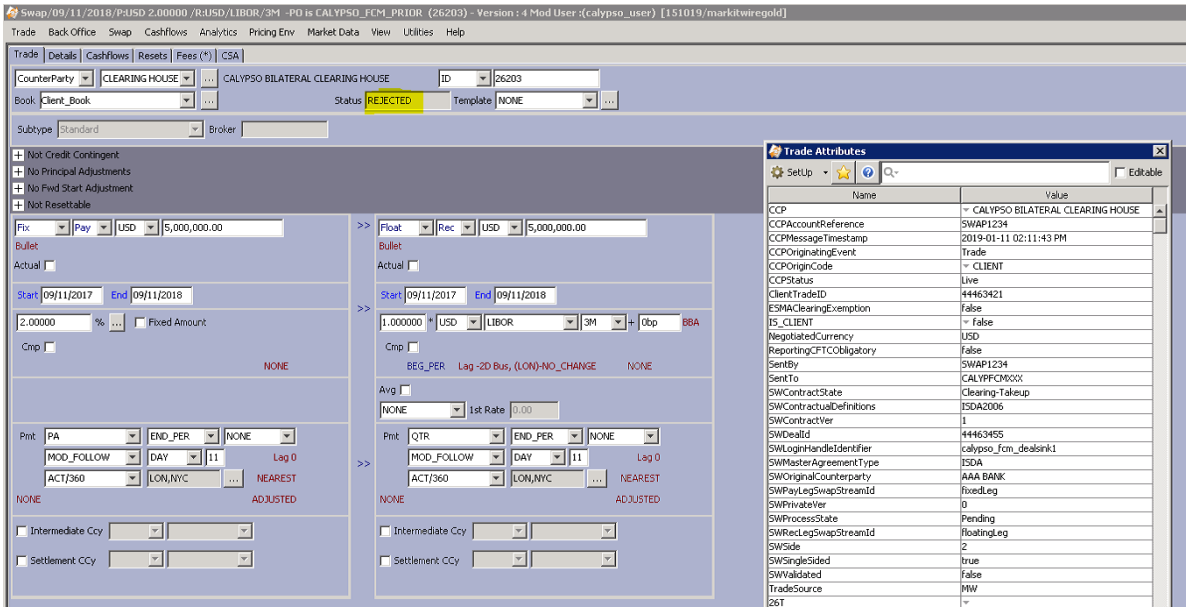
FCM Accept

Post receiving and processing the initial notification, when FCM perform ACCEPT action a notification is sent to bilateral trades stating the trades are accepted at FCM end and trades in calypso receives an acknowledgement updating the trade as below:



FCM Reject

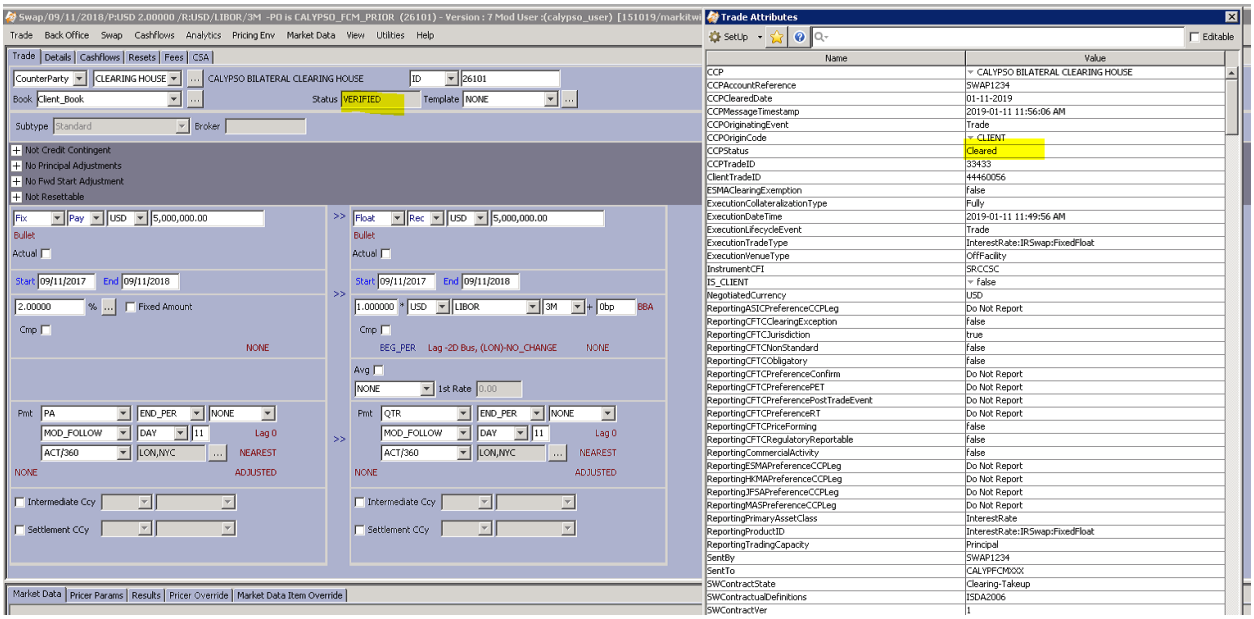
Post receiving and processing the initial notification, when FCM perform REJECT action a notification is sent to bilateral trades stating the trades are rejected at FCM. Provision of adding reject reason is provided by keyword **CCPRejectReason, when this keyword value is not provided default reason is Deal Rejected**. Trade representation



Known issue: As of now for FCM Reject we are not receiving acknowledgement from MarkitWire and we are actively working to get it working.

FCM Accept- ClearingHouse Accept

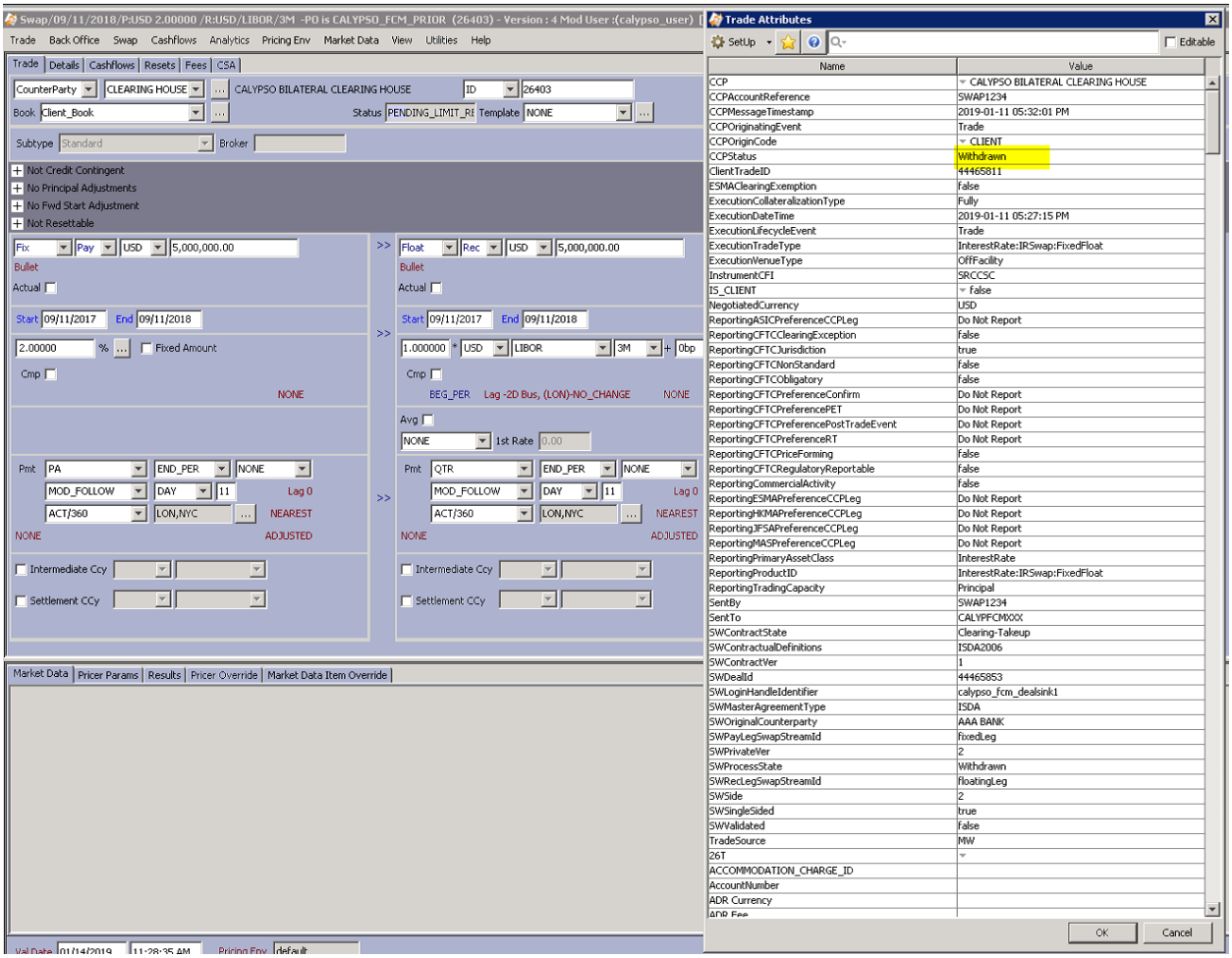
Post FCM accept from Calypso the bilateral trades are sent to ClearingHouse for clearing through MakritWire. Once Clearing House clear these bilateral trades FCM receives clearing notification and same get updated on the Calypso side of corresponding FCM as below:



The trade action to be applied when clearing house clears the deal is by default **CLEAR.** This action can be further customized by providing mapping as mentioned in section 1.1.4 STEP5. Also, the action should be part for the trade Work-Flow transition to clear trade and domain **UploadAllowedAmendActions.**

FCM Accept- ClearingHouse Reject

Post FCM accept from Calypso the bilateral trades are sent to ClearingHouse for clearing through MakritWire. If Clearing House rejects these bilateral trades FCM receives rejection notification and same get updated on the Calypso side of corresponding FCM as below:



The trade action to be applied when clearing house refuses the deal is by default **REFUSE.** This action can be further customized by providing mapping as mentioned in section 1.1.4 STEP5. Also, the action should be part for the trade Work-Flow transition to refuse trade and domain **UploadAllowedAmendActions.**

6.2 FCM Post-clearing Business Flow

6.2.1 Overview

Calypso supports the LCH FCM T2 model introduced by LCH and supported by MarkitWire for the Clearing Broker/FCM role.

Trades booked between an Executing Broker (EB) and a Client can be imported by a FCM (Futures Commission Merchant) in Calypso after CCP submission as a “Clearing Take-Up” trade.

Trade Consent/Rejection notifications from the FCM to LCH are triggered by actions performed on the trade in Calypso. No action is necessary from the MarkitWire GUI to handle the FCM role.

6.2.2 FCM Post-clearing Business flow

The business workflow follows this pattern:

1. A trade is initiated by an Executing Broker against a Client.

2. The Client affirms the trade by entering an FCM broker in the MarkitSERV Clearing tab.

3. The Executing Broker and Client submit the trade for clearing by LCH FCM.

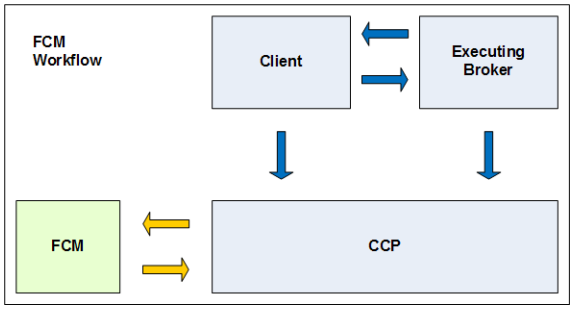
4. LCH sends a Request Consent notification to the FCM for the trade to be accepted or rejected. The trade appears in the FCM’s Clearing Takeup blotter.

5. Using the FCM dealsink, the trade between FCM and CCP is automatically created in Calypso.

6. The trade can be analyzed in Calypso for a limit check.

7. The trade is then accepted or rejected (withdrawn) from Calypso and the appropriate notification is sent to LCH.

8. After action in Calypso, the Trade status is updated in MarkitWire.



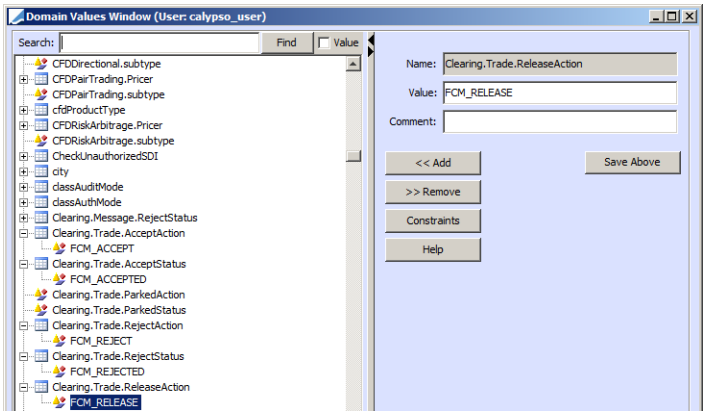
6.2.3 Configuration for FCM Clearing

After importing all jars from the Service Pack and applying the upgrade scripts.

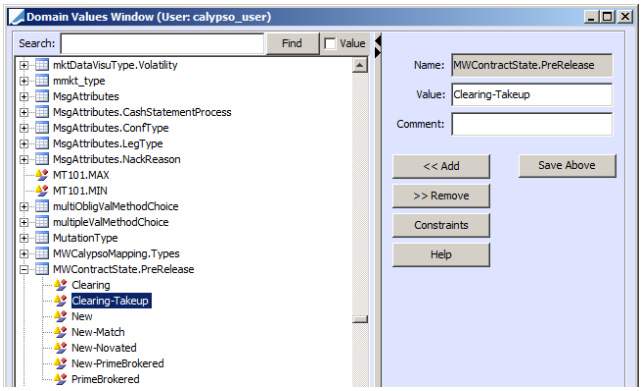
Step1 - Acceptance, Rejection, and Release notifications to LCH are triggered by a combination of trade actions and the resulting trade status in Calypso. Add the values to the following domains:

- Clearing.Trade.AcceptAction: FCM_ACCEPT

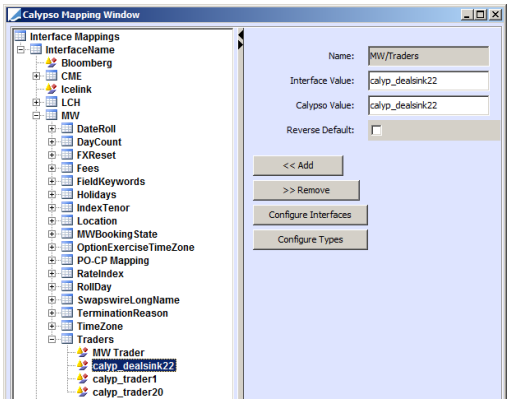
- Clearing.Trade.AcceptStatus: FCM_ACCEPTED
- Clearing.Trade.RejectAction: FCM_REJECT
- Clearing.Trade.RejectStatus: FCM_REJECTED
- Clearing.Trade.ReleaseAction: FCM_RELEASED



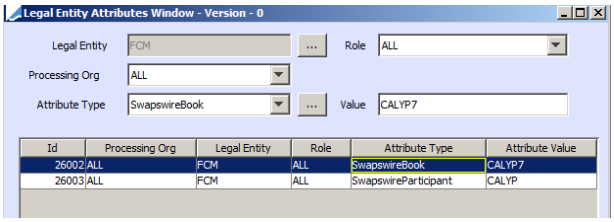
Step 2 - Add Clearing-Takeup to the MWContractState.PreRelease domain:



Step 3 - Using the Calypso Mapping Window, map the MarkitWire FCM Trader to the Calypso trader:



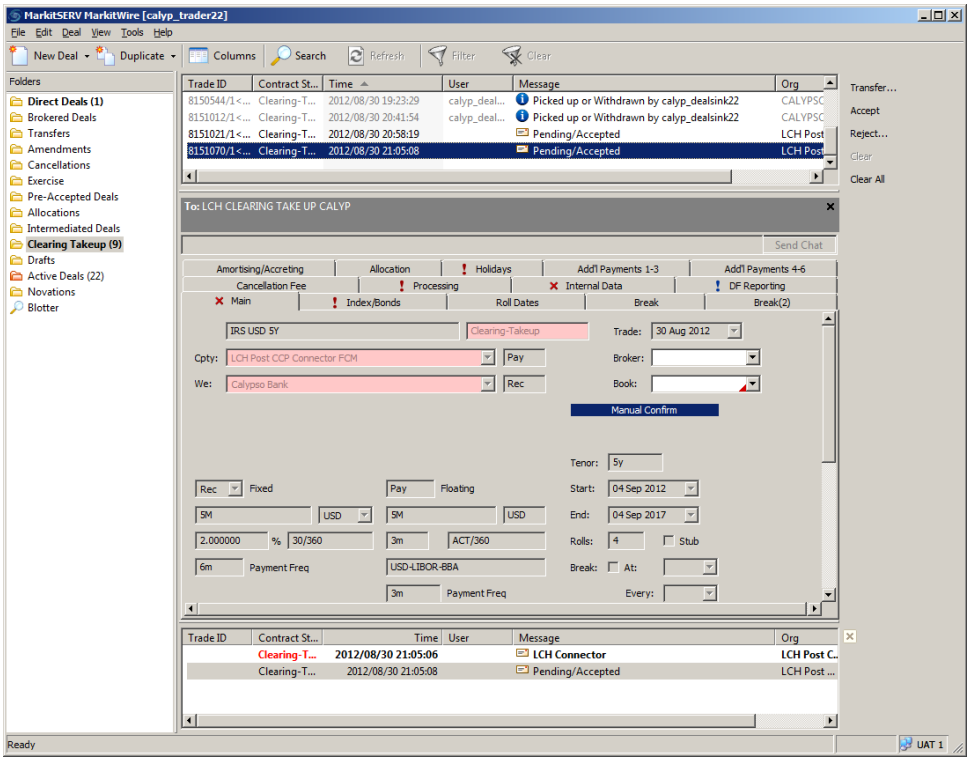
Step 4 - Populate FCM (Processing Org) attributes SwapswireBook (this will be the default book for incoming trades) and Swapswire Participant:

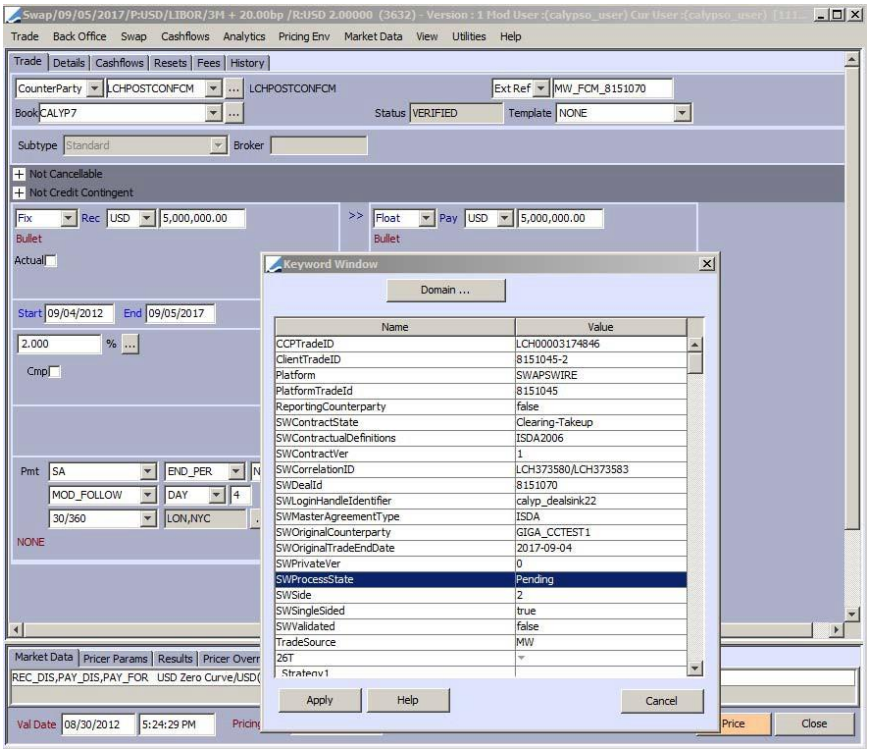


Step 5 - Configure your MarkitWire dealsink and FCM logins to auto release trades.

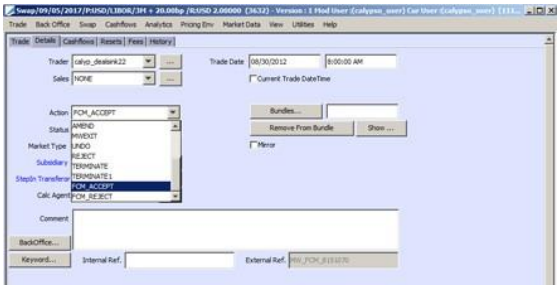
Step 6 - Submit the trade to the FCM trader. The trade appears in the Clearing Takeup folder or the MarkitWire GUI. That trade is automatically imported into the Calypso database.

The Trade status in Calypso is then directed according to the trade workflow based on keyword filtering.





Step 7 - Select action FCM_ACCEPT or FCM_REJECT and resave the trade (names provided as an example, you can use your own workflow names):



Step 8 - Trade will be updated in MarkitWire after action performed in Calypso.

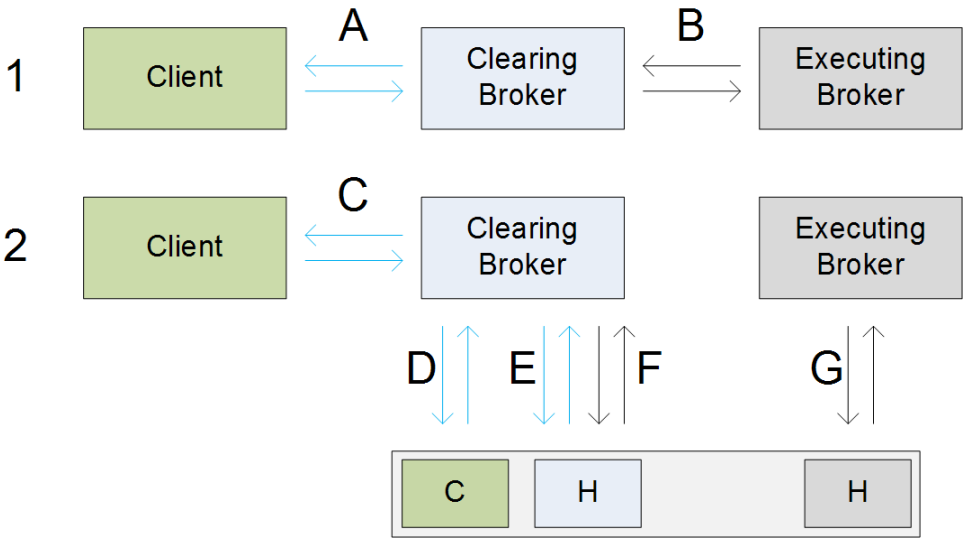
LCH Trilateral Clearing Support

Note: This is deprecated and may be removed in a future release based on LCH support.

The Calypso MarkitWire interface supports client clearing functionality using the LCH trilateral booking model. Configuration is covered in the "Configuration for LCH Trilateral Trades" section.

7.1 LCH Booking Model as Clearing Broker

In the workflow below, a trade using the trilateral booking mode is initiated by the Executing Broker (EB) at step 1 against the client using the Clearing Broker (CB) as an intermediary:



Trades A and B are created with the status, "New-PrimeBrokered" and are imported into Calypso upon affirmation by all parties.

Upon release by the Clearing Broker, Trade A is novated as Trade E to LCH and Trade B is novated as Trade F. In addition, **CCPClientTradeType** is set to "Primary" and **CCPAccount** is set to "House" in Trade E. In Trade F, **CCP-ClientTradeType** is blank and **CCPAccount** is set to "House."

Examination of the two SWML files reveals which trade is E and which is F, based on the fact that on the Client side of Trade E, one party is not a clearing member and on the Executing Broker side of Trade F, both parties are clearing members.

Upon creation of Trade E, with External Reference **MW_PO_SwapswireId**, two additional trades are created in Calypso. Trades C and D are required by the LCH booking model. They segregate "House" account trades and Client account trades at LCH.

Trade C is a copy of Trade E, except that the counterparty from the Clearing Broker's point of view is the original client rather than LCH. All Trade Keywords in Trade E are propagated to Trade C, including TransferDate, TransferTradeDate, and TransferFrom. The CCPClientTradeType Trade keyword is set to "Secondary," and the external reference is set to MW_PO_SwapswireId_Client in Trade C.

The Clearing Broker is the payer (to LCH) in Trade E, and is also the payer (to the client) in Trade C. The Clearing Broker is receiver in both Trade F and Trade C.

Trade D is also a copy of Trade E, but its direction is the opposite of Trade E. The counterparty (LCH) remains unchanged. All Trade keywords in Trade E are propagated to Trade D, including TransferDate, TransferTradeDate, and TransferFrom.

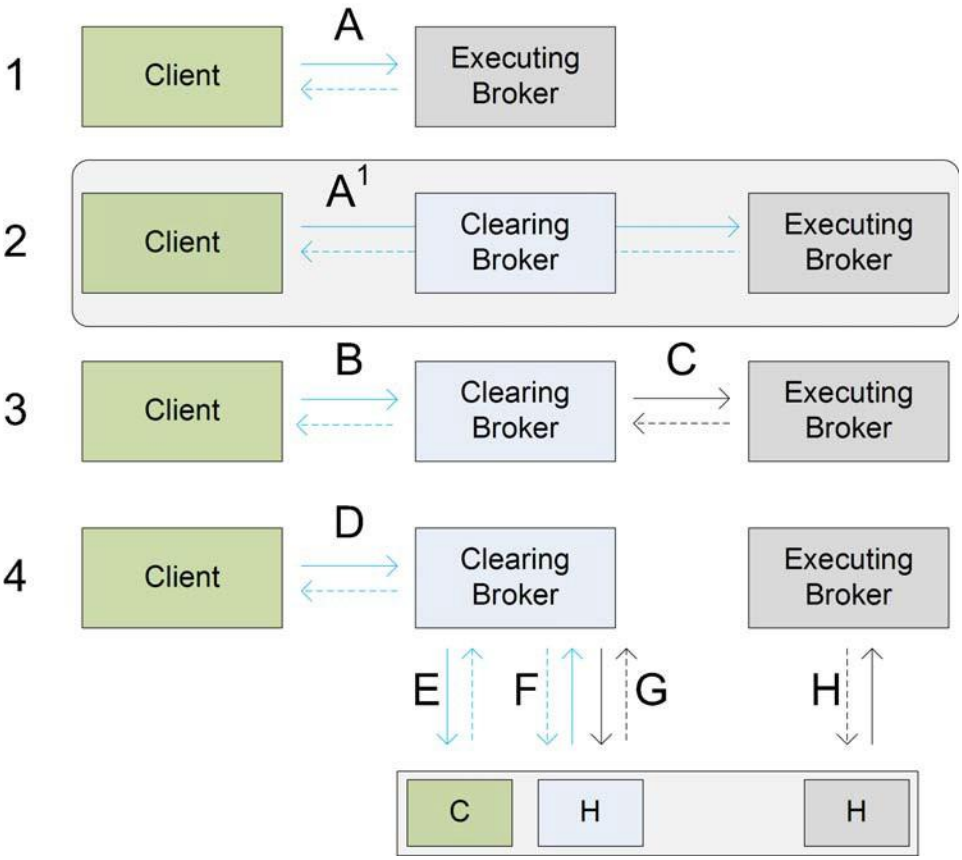
In Trade D, the CCPClientTradeType trade keyword is set to "Secondary", the CCPAccount trade keyword is set to "Client," and the external reference is set to MW_PO_SwapswireId_ClientAccount.

If the Clearing Broker is the payer to LCH in Trade E, then the Clearing Broker is the receiver to LCH in Trade D. Likewise, if the Clearing Broker is the receiver (from LCH) in Trade E, then the Clearing Broker is the payer (to LCH) in Trade D.

The user's Entity Role (whether Executing Broker or Clearing Broker) is determined by examination of the swml. When the user's Entity Role is Executing Broker, Trade G is created between the Executing Broker and LCH in the same manner as a direct clearing trade.

7.2 Bilateral to Trilateral Amendment

Based on the SWML content, your Entity Role is Executing Broker. The original trade was initiated in MarkitWire (through backloading, in this example) and has been imported into Calypso.



After using MarkitWire's Client Clearing Update function to change the trade to a Client Clearing PrimeBrokered trade (A1), the system will perform the following actions:

Upon receiving the PENDING notification for Trade A1, Calypso does not import the trade as it already exists in the system. The existing Trade A, which is a bilateral trade, is identified in the database via Swapswire ID and is amended.

At this point, Trade A can be redirected in the trade workflow to any desired status using a custom filter or rule based on the Trade Keywords.

When Calypso receives the Trade A1 'Released' notification (and after affirmation by all parties), Calypso attempts to TERMINATE Trade A and creates Trade C. If the TERMINATE action is not available for Trade A in the trade's workflow, Calypso then attempts to CANCEL the trade. If neither trade workflow action is possible from the current trade status, a warning message is sent to the Task Station.

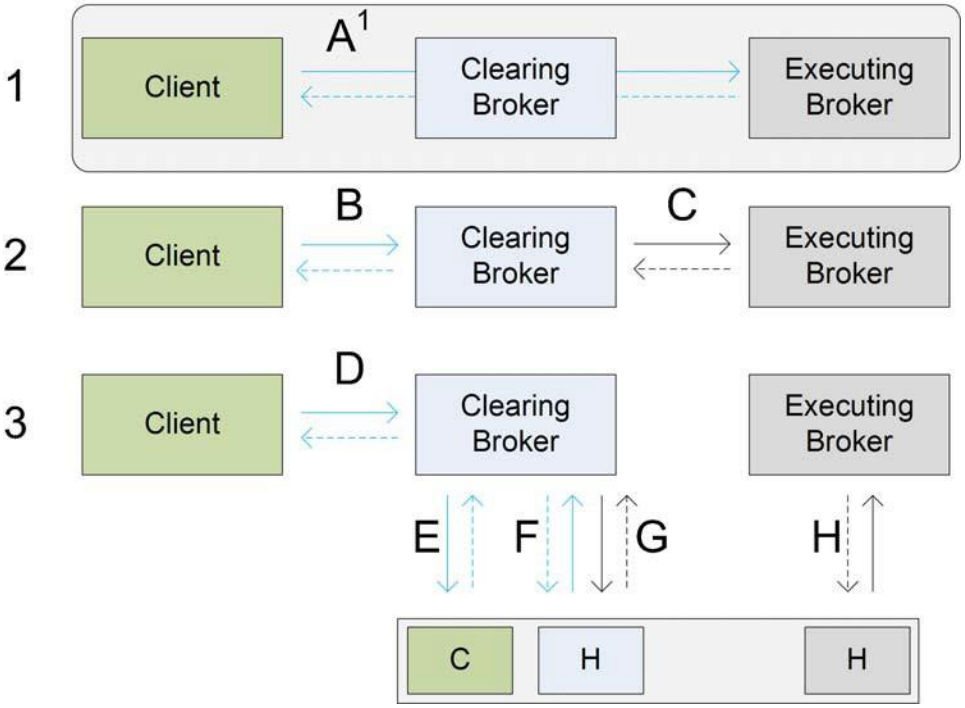
If a counterparty rejects Trade A1 and Calypso receives a 'Withdrawn' notification for Trade A1, Calypso attempts to terminate Trade A and does not create Trade C. If the TERMINATE action is not available for Trade A, Calypso then

attempts to CANCEL the trade. If neither trade workflow action is possible from the current trade status, a warning message is sent to the Task Station.

If neither trade workflow action is possible from the current trade status, a warning message is sent to the Task Station.

7.3 Trilateral Trade Creation

Based on the SWML content, your Entity Role is Executing Broker. The trade was initiated in MarkitWire as a trilateral Trade A1 and has not been created in Calypso.



After entering the trade (as the Executing Broker), when Calypso receives the Pending notification for Trade A1 and after checking that a bilateral trade with this Swapswire ID does not already exist, Calypso uses the counterparty details from the SWML to create a new bilateral trade (A) between the client and the Executing Broker.

Trade A can be directed in the trade workflow to any desired status using a custom filter or rule based on the keywords of the trade.

Once all parties have Affirmed Trade A1 and Calypso receives the Released notification, Calypso Terminates Trade A and creates Trade C.

Upon trade A1 affirmation by all parties resulting in trade A1 'Released' notification being received, trade A will be terminated, and trade C will be created in Calypso. If the TERMINATE action is not available for Trade A, Calypso then attempts to CANCEL the trade. If neither trade workflow action is possible from the current trade status, a warning message is sent to the Task Station.

If a counterparty rejects Trade A1 and Calypso receives a 'Withdrawn' notification for Trade A1, Calypso attempts to terminate Trade A and does not create Trade C. If the TERMINATE action is not available for Trade A, Calypso then attempts to CANCEL the trade. If neither trade workflow action is possible from the current trade status, a warning message is sent to the Task Station.

7.4 LCH Booking Model as Executing Broker and Clearing Broker

Your organization can act as Executing Broker or Clearing Broker.

The interface handles the following case: Two organization users log in using different login IDs, one as the Executing Broker and one as the Clearing Broker. These two users will be associated with different Dealsink IDs.

The two users will also use different books and Processing Organizations having different BIC codes.

In this scenario, the Swapswire engine must be set up to listen to the two Dealsink user notifications. Refer to the "Multiple Dealsink Users" section for instructions to setup multiple dealsink users.

Calypso creates trades for the Executing Broker and Clearing Broker with different External References since they are booked in different Processing Organizations. The processing of these Clearing Broker and Executing Broker trades follows the behavior described above.

7.5 Lifecycle Actions with the Clearing Broker or Executing Broker Role

Lifecycle processing will revert the different processes described in the preceding paragraphs as follows

7.5.1 Lifecycle Handling with the Clearing Broker Role

Upon receipt of the first two New-PrimeBrokered notifications, consecutive to the declear acceptance by LCH:

- Trades D and E are terminated.
- Trades F and G are terminated subsequent to their novation to a trilateral trade comprised of two New-PrimeBrokered trades (Trades B1 and C1 created by the novation). Their external references will be changed by appending the novated Calypso Trade ID.

7.5.2 Lifecycle Handling with Executing Broker Role

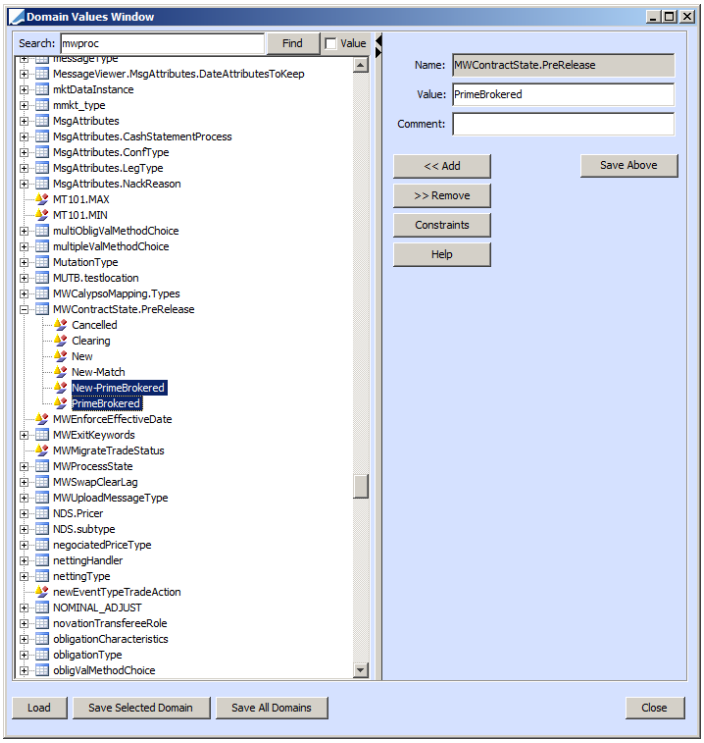
Trade H is terminated subsequent to its novation to Trade C[1] upon receipt of New-PrimeBrokered child SWMLs in the new trilateral trade. Its External Reference will be changed by appending the novated Calypso Trade ID.

7.6 Configuration for LCH Trilateral Trades

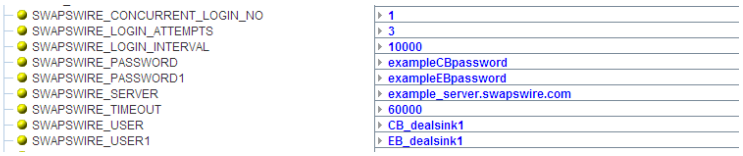
The standard upgrade XML scripts that accompanied the release jar populates the CCPAccount and **CCPClientTradeType** in the **tradeKeyword** domain.

Step 1 - Using the Domain Values window, add the following contract states to the MWContractState.PreRelease:

- New-PrimeBrokered
- PrimeBrokered



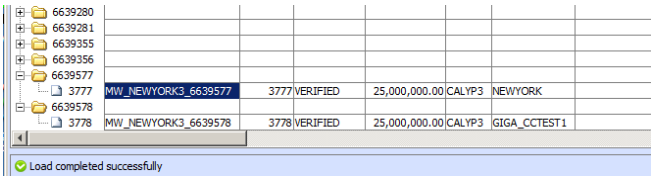
Step 2 - Configure multiple logins with appropriate the Dealsink IDs and passwords if you need to listen to Clearing Broker and Executing Broker notifications. Refer to the "Multiple Dealsink Users" section for complete instructions:



7.7 Examples

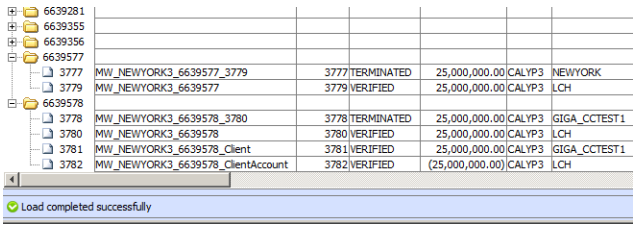
Step 1 - Trade booked as an Executing Broker. Trade affirmed as the Client and Clearing Broker.

Step 2 - New-PrimeBrokered trades are created in Calypso as you are the Clearing Broker in this example:



Step 3 - Release the New-PrimeBrokered trades in MarkitWire and send the trades to clearing.

Step 4 - Calypso creates the linked Trades:



Clearing for CCPs

This section describes the system and business configuration for the MarkitWire interface to be used as a Clearing House solution.

A Clearinghouse Legal Entity (hereinafter referred to as a CCP) can import SentForClearing trades sent by MarkitWire into Calypso after validation and handle incoming and outgoing notifications to MarkitWire to enable the process of trades.

The Calypso MarkitWire interface for CCP supports support IRS, Basis Swaps in the bilateral (Agency) model only.

8.1 Exchange Clearing House Configuration

8.1.1 Legal Entity Role for CCP

The MarkitWire interface supports the models where the CCP is defined as Processing Organization or Counterparty.

When using the processing organization model, trades are imported in Calypso so that the CCP is the Processing Organization and the dealers are counterparties.

When using the counterparty model, trades are imported so that the CCP is the counterparty and the dealers are Processing Organizations.

This choice can be defined using the CCP_IS_PO property in the calypso_SW_config.properties file.

The CCP_IS_PO property is ignored if EXCHANGE_CLEARING is set to false which is the default value.

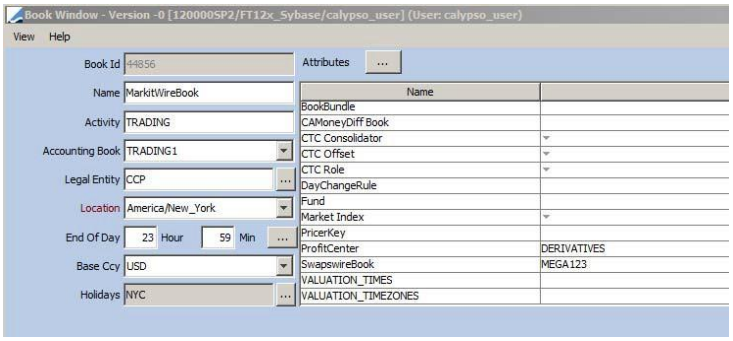
EXCHANGE_CLEARING must be set to true when acting with the CCP role and false when acting with a dealer or client role.

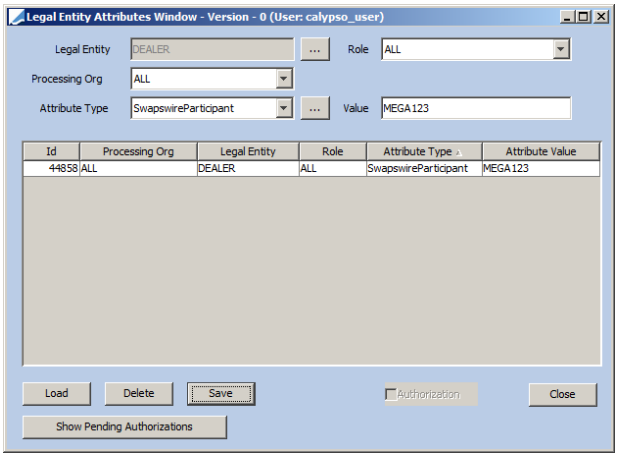
```
#Mark EXCHANGE_CLEARING as true for Exchange Clearing solution
EXCHANGE_CLEARING=true

#Flag indicating if CCP is Processing Org or Counterparty
CCP_IS_PO=true
```

8.1.2 Book and Client Mapping in the Counterparty Role Model

When acting as CCP, there is no book in the SWML notification from MarkitWire as on the Dealer side.

In the CCP = counterparty model, the Calypso book will be inferred from the SwapswireBook attribute which will contain the BIC code of the dealer.

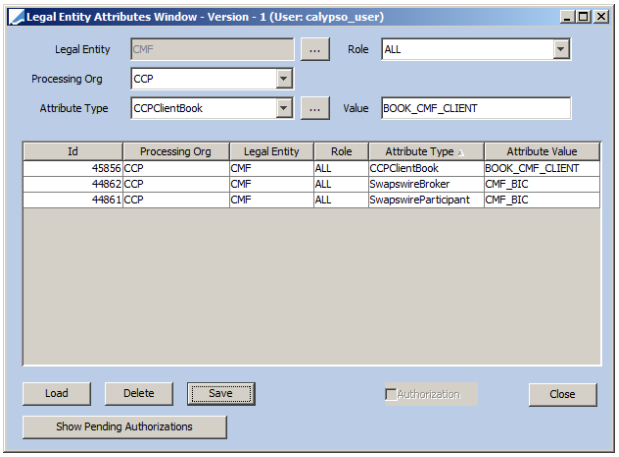




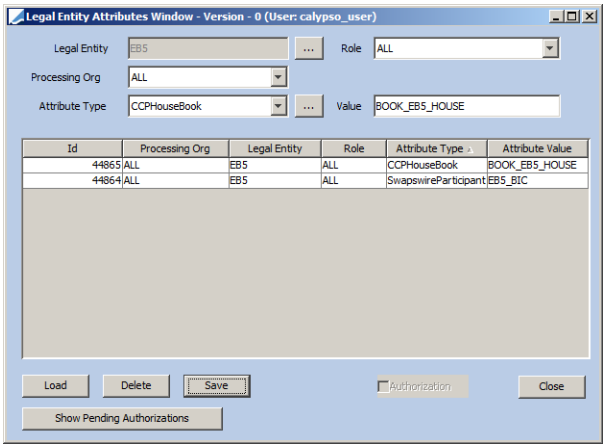
8.1.3 Book and Client Mapping in the Processing Role Model

In the CCP = Processing Organization model, the book is inferred from the CCPClientBook or CCPHouseBook attribute that contains the BIC code of the dealer.

As Clearing Broker (Client Book):

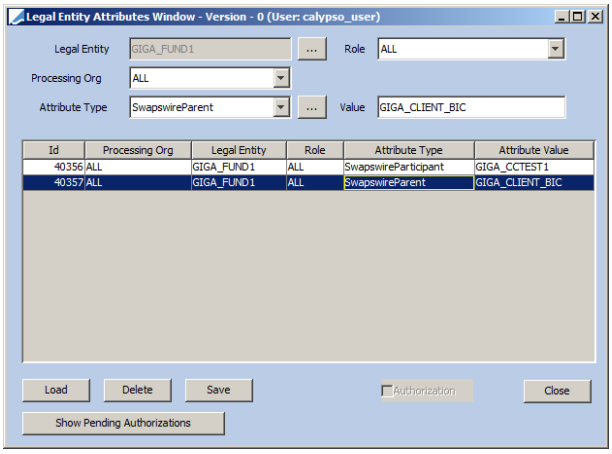


As Executing Broker (House Book):

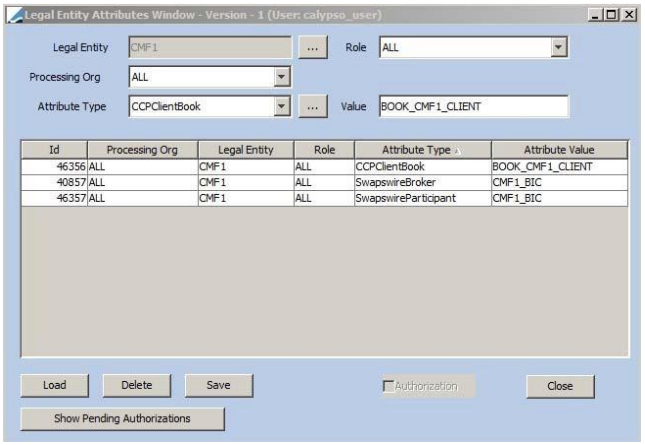


When receiving a client clearing trade code with a fund as party in the swml, the book and the client will be inferred from a combination of CMF and Fund as follows:

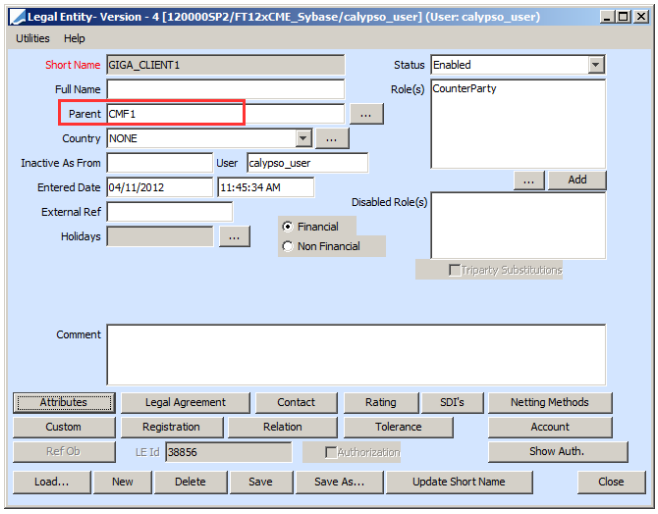
Step 1 - We will determine the Fund Legal Entity in Calypso based on the BIC code present in the SWML matching its SwapswireParticipant attribute:

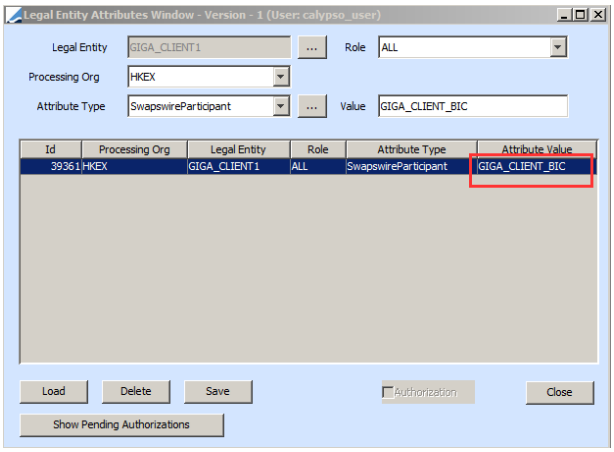


Step 2 - We will determine the CMF Legal Entity in Calypso based on the BIC code present in the SWML matching its SwapswireParticipant attribute:



Step 3 - The Book in the trade will be determined by the CCPHouseBook attribute (for a House trade) or CCPClientBook attribute (for a Client trade) of the CMF.





Step 4 - The Counterparty in the trade is the Client Legal Entity whose Swapswire participant BIC code matches the SwapswireClient attribute from the fund previously found, and whose parent is the CMF found above.

8.1.4 Domain Data for MarkitWire Exchange Notifications

To support Exchange Clearing, Calypso uses eight domainNames to control the logic for sending various notifications (ClearAccept, ClearReject, DeclearAccept, DeclearReject, Parked) to MarkitWire. The notification sent is based on a combination of the trade action and the trade status of the trade workflow. The domains are:

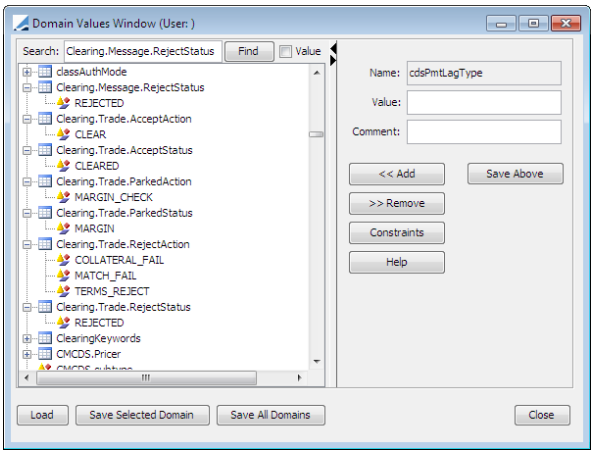
- Clearing.Trade.RejectAction
- Clearing.Trade.RejectStatus
- DeClearing.Trade.RejectStatus
- DeClearing.Trade.RejectAction
- Clearing.Trade.Accept
- Clearing.Trade.AcceptAction
- DeClearing.Trade.AcceptStatus
- DeClearing.Trade.AcceptAction

If your implementation must send Parked notifications to MarkitWire, you must manually create the Parked Action and Status. Refer to the "Support for Parked Status" section for further information.

- Clearing.Trade.ParkedAction
- Clearing.Trade.ParkedStatus

Populate each "Action" domain and each "Status" domain to create the combinations that will trigger Calypso to send the notification to MarkitWire. A combination of Trade Action and Trade Status is necessary to ensure that notifications are sent only when proper.

For example:



Refer to the image above: When a trade reaches the REJECT trade status via the COLLATERAL_FAIL, MATCH_FAIL, or TERMS_FAIL trade actions, Calypso will send the ClearReject notification to MarkitWire.

8.1.5 Legal Entity and Book Mapping

You must map dealers (traders) as Processing Organizations and the Clearing House as Counterparties. Map DealerId (e.g. MEGACALPCC) as SwapswireBook attribute in the book. The Clearing House counterparty must have an LE Attribute named SwapswireParticipant & SwapswireBroker and its value should be the value of the swClearingHouseId tag of the Clearing XML.

Note that in the Client Member Firm (CMF) and Client usage of MarkitWire, SwapswireBook contains the Book from MarkitWire. In Exchange usage, SwapswireBook contains the Swapswire Participant ID. This difference is because the Clearing XML contains no Book information.

8.1.6 SwapswireTradeEngine Configuration Changes

To enable the sending of notifications to MarkitWire based on a particular trade status and message status in a trade workflow, the SwapswireTradeEngine must listen to PSEventTrade and PSEventMessage, respectively.

8.1.7 MarkitWire Process and Contract states

In addition to the MarkitWire Process and Contract states specified in other sections of this guide, you must add the following process states to the MWProcessState domain:

- SentForClearing
- SentForClearingUpdate

The MWContractState.PreRelease domain should have all possible contract states to which the Exchange must listen. A Clearing House listens to pre-release notifications and, depending on Clearing House confirmations, the deals are released. The Clearing House should listen to combination of the above two process states and all possible Contract States:

- Clearing
- Amended
- Novated
- Novated-Partial
- Cancelled

8.1.8 Workflow Rules

The following trade rules specifically support Exchange Clearing:

- UpdateLinkedToKeywordTradeRule - This rule links two trades created from single MarkitWire Deal ID, that is, two trades having same SWDealId keyword value and same SWContractVer keyword value. The value of the LinkedTo trade keyword is populated with the Trade ID of the other trade. This rule should be applied at some initial status of the workflow.

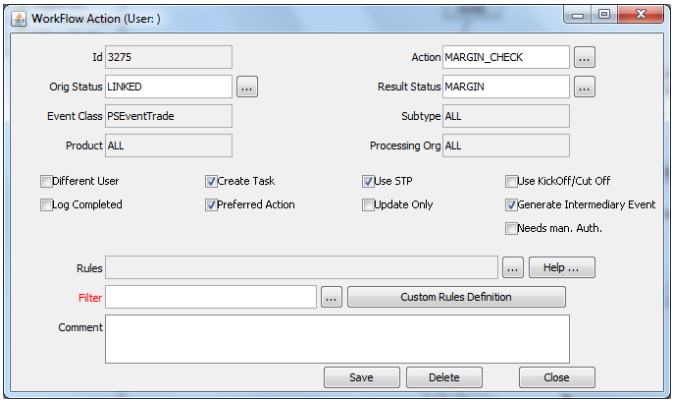
- ApplyLinkedTradeActionTradeRule - Add this rule on any transition of trade workflow to cause Calypso to apply the action to the linked trade before then applying it to the original trade. This allows both trades to move to next status simultaneously. You should only apply this rule in transitions wherein the trades should transit to next status simultaneously. For example, if the trade status of a linked trade differs from that of the original trade, then both trades remain in the old trade status (i.e., neither trades changes to the next status). If there is no linked trade for the "original" trade, then the "original" trade does not change status.
- DeclearActionTradeRule - This rule adds the Trade ID to the External Reference. Apply this rule on a Declear accept transition or Clear Reject transition to prevent an external reference conflict when the trade is again presented for clearing.

Note: The following workflow rules ApplyLinkedTradeActionTradeRule, UpdateLinkedToKeywordTradeRule, ApplyLinkedMsgActionMessageRule and UpdateLinkedToAttributeMessageRule are designed to work only with the out-of-the-box modules (markitwire and dsmatch) for CCP solution. They are not supported for individual use.

8.2 Support for Parked Status

The Exchange Clearing function sends a Parked Notification to MarkitWire after the trade is saved and before the Margin/Collateral check is applied. Once a message workflow successfully finishes, Calypso creates a trade for both the parties. Based on configuration of the Status and Action as shown in the image in the "Domain Data for MarkitWire Exchange Notifications" section, the interface sends a Parked notification to MarkitWire.

Parked Notification support in the STP workflow requires the application to generate events for the SwapswireTradeEngine. To enable these events, select the Generate Intermediary Event checkbox:



Note: Parked Notification support in STP workflows requires Initial Action and an Intermediary Action in the Clearing.Trade.ParkedAction domain.

When the SwapswireTradeEngine receives this trade event, it sends a Parked notification to MarkitWire. The Parked notification is sent for the linked trade (there are two trades) having the higher Trade ID.

8.3 Property File Changes

8.3.1 calypso_SW_config.properties

Add the EXCHANGE_CLEARING property as true if you are acting as a CCP. By default, this value is false.

When acting as CCP, set the CCP_IS_PO property to indicate whether the CCP is acting as PO or counterparty. By default, the value is set to false. This property is only used when acting with CCP role. By default, CCP_IS_PO is false.

```
# doRecovery is used to fetch trades released from Markitwire but not yet imported in Calyp
# Only perfrom the doRecovery if the flag below is set to true
performDoRecovery=false

# Do Recovery Start Date (Must be in YYYYMMDD format). This requests trades from Markitwire
# For normal operations, it must be kept blank.
#doRecoveryStartDate=20091113

# This flag is used to generate XML files from the SWML Messages. Two XML files are generat
# message. One file is the SWML message itself, the other is Calypso Data Upload XML (which
# representation of the SWML. Please refer to documentation for more details.
# The files are generated in USER_HOME\Calypso\markitwire
DEBUG_MW_XML=true

# Password Encryption flag
# Set this mandatorily otherwise engine will not startup
AutoEncryptPassword=false

#Timeout for MW connection
session_timeout=360

#Mark EXCHANGE_CLEARING as true for Exchange Clearing solution
EXCHANGE_CLEARING = false

#In exchange clearing, this flag is used for CCP as a PO model if set to true
CCP_IS_PO = false

#This flag is used to start engine in Test mode
#TEST_MODE = true
```

8.3.2 gatewayservice.properties

Change MWPublishers from “MW” to “MWClearing” to use exchange clearing.

gatewayservice.properties

```
# Gateway Service Properties

# All BO Messages need a sender and a receiver. Use these properties for that
# GatewayServiceClientName,GatwayServiceHostName: Indicate the BOMessage Receiver & Sender
GatewayServiceClientName=CLIENT
GatwayServiceHostName=CALYPSO

# Gives list of BO Messages states that are used to link pending messages
# These messages are blocked messages due to some validation error.
BOMessageIncompleteStates=PENDING_VALID,PENDING_TRADE,BACKLOAD

# For each type of input, there can be a list of publishers
# The publishers are chained together so that multiple publishers can be called
# List of one or more publishers depending on source. File, MW, IceLink
# UploaderXMLPublishers,UploaderCSVPublishers,MwPublishers,IceLinkPublishers:
UploaderXMLPublishers=File
# For Exchange Clearing keep MWPublisher as MWClearing for normal one keep it as MW
MWPublishers=MWClearing
UploaderCSVPublishers=
```

8.4 Trade Processing

This section describes how a trade flows from MarkitWire into the Calypso instance of a CCP.

A trade is SentToClearing by both counterparties via the MarkitWire GUI. The MarkitWire engine (in the Exchange) receives two SWML messages from MarkitWire which are translated from the Clearing XML format to Calypso's Upload XML.

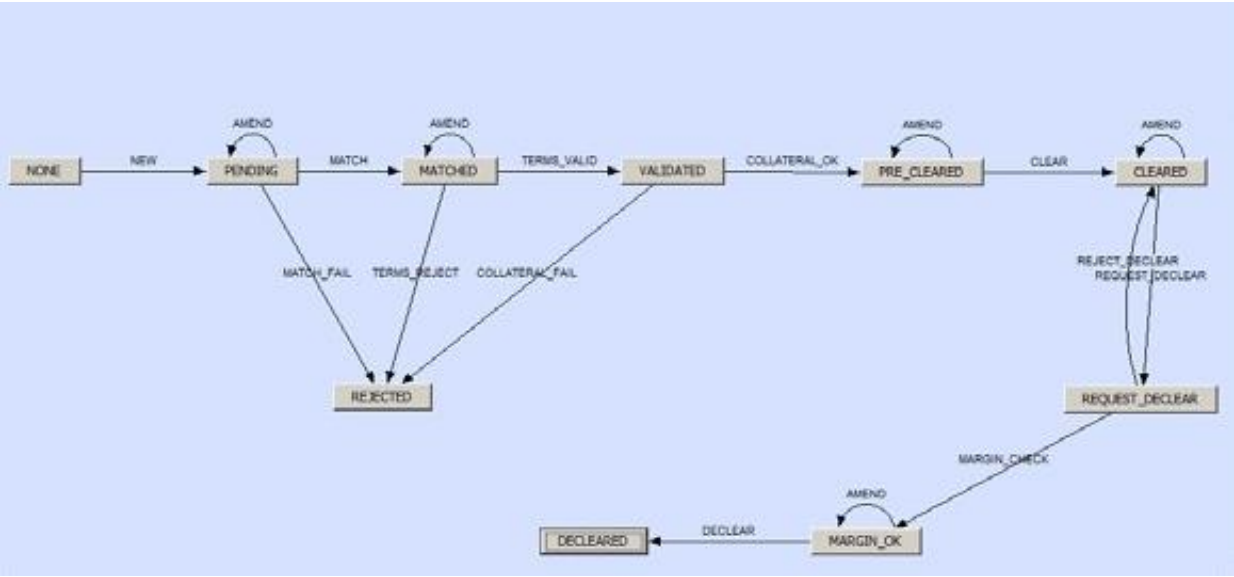
If the translation to ClearingSWML format fails, an exception is generated and displayed in the User Task Station as a task to be addressed.

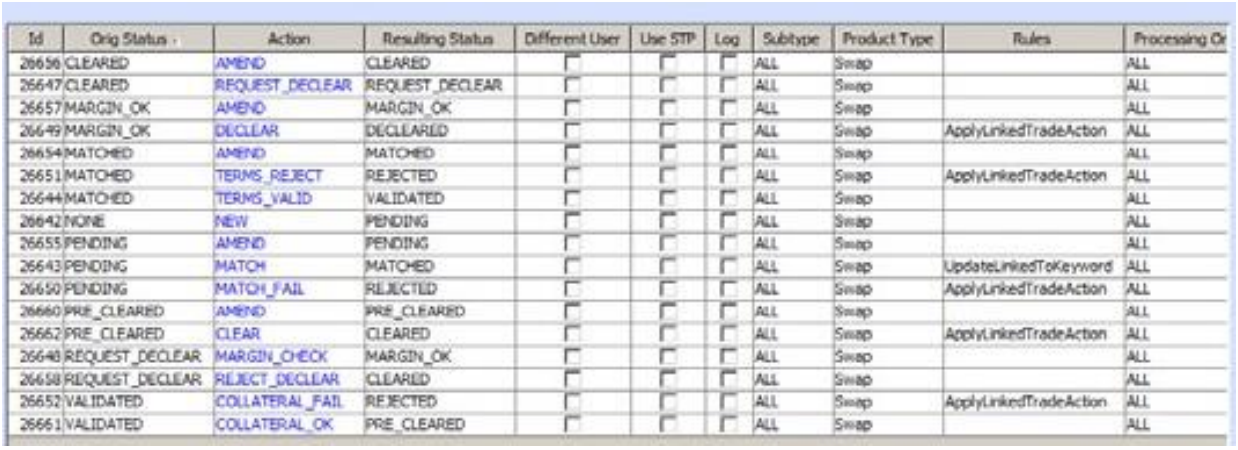
When the translation is completed, the two resulting Uploader XML Messages are then validated.

If the validation fails, which occurs when for example a mapping is missing, an exception is generated. The exception can be displayed in the Task Station.

When the issue is addressed, the two messages can be reprocessed, and the two corresponding trades will then be imported in the trade workflow.

The following describes a sample CCP workflow to illustrate the process.





Id	Orig Status	Action	Resulting Status	Different User	Use STP	Log	Subtype	Product Type	Rules	Processing Or
26656	CLEARED	AMEND	CLEARED				ALL	Swap		ALL
26647	CLEARED	REQUEST_DECLEAR	REQUEST_DECLEAR				ALL	Swap		ALL
26657	MARGIN_OK	AMEND	MARGIN_OK				ALL	Swap		ALL
26649	MARGIN_OK	DECLEAR	DECLEARED				ALL	Swap	ApplyLinkedTradeAction	ALL
26654	MATCHED	AMEND	MATCHED				ALL	Swap		ALL
26651	MATCHED	TERMS_REJECT	REJECTED				ALL	Swap	ApplyLinkedTradeAction	ALL
26644	MATCHED	TERMS_VALID	VALIDATED				ALL	Swap		ALL
26642	NONE	NEW	PENDING				ALL	Swap		ALL
26655	PENDING	AMEND	PENDING				ALL	Swap		ALL
26643	PENDING	MATCH	MATCHED				ALL	Swap	UpdateLinkedToKeyword	ALL
26650	PENDING	MATCH_FAIL	REJECTED				ALL	Swap	ApplyLinkedTradeAction	ALL
26660	PRE_CLEARED	AMEND	PRE_CLEARED				ALL	Swap		ALL
26662	PRE_CLEARED	CLEAR	CLEARED				ALL	Swap	ApplyLinkedTradeAction	ALL
26640	REQUEST_DECLEAR	MARGIN_CHECK	MARGIN_OK				ALL	Swap		ALL
26658	REQUEST_DECLEAR	REJECT_DECLEAR	CLEARED				ALL	Swap		ALL
26652	VALIDATED	COLLATERAL_FAIL	REJECTED				ALL	Swap	ApplyLinkedTradeAction	ALL
26661	VALIDATED	COLLATERAL_OK	PRE_CLEARED				ALL	Swap		ALL

In this example workflow, the trade could enter the workflow and go STP to PENDING status after message validation.

The two trades are matched (by Swapswire ID) and, when found, move together to the MATCHED status. The **LinkedTo** keyword for each trade is populated with the Trade ID of the other trade. The two trades are linked to each other using the **UpdateLinkedToKeywordTradeRule** and can further traverse together using **ApplyLinkedTradeActionTradeRule**.

The terms of the two trades (notional, currency, index, tenor) can be validated at this point against the CCP. Again, this is only an example workflow and not intended for direct use in production.

If terms are invalid, the two trades can move together to REJECTED status and Calypso will send a Rejected notification to MarkitWire, with the corresponding Clearing Status.

If however, the trade terms are valid, the trades can move to the VALIDATED status.

Collateral and limit checks could be performed at this point

If successful, the trades can move to the PRE-CLEARED status where internal processing can take place prior to sending the trades to the CLEARED status, otherwise the two trades both move to REJECTED status in this sample workflow.

When the trades are sent to CLEARED status, Calypso sends a **RegisteredForClearing** notification to MarkitWire. Although MarkitWire sent a notification for each trade, only one notification from Calypso is expected when both trades have cleared.

If however, at any point, the trades are moved to the REJECTED status, Calypso sends two **RejectedForClearing** notifications to MarkitWire.

Upon a Declear request, the database is queried for the two individual trades by their External Reference ID. Upon receipt of a Declear notification a custom action stored in **UploadDeClearAction** domain will be applied on the trade. If no custom action is specified in this domain then, by default, the DECLEAR action is applied. The Trade workflow should have the DECLEAR action on the Cleared trade. The **UploadDeClearAction** domain should contain a custom action name. The U**ploadDeClearAction** domain should not have actions like terminate/cancel. These actions can be further applied on the trade.

A workflow rule can be used to validate the Declear and check the collateral before affirming the Declear, which results in Calypso terminating the two trades and sending the Declear notifications to MarkitWire.

Client Clearing Processing with the Client Role

This section describes support in Calypso's MarkitWire Integration Module for client clearing processing for rate product trades entered in MarkitWire with a client role.

9.1 CME Bilateral Model

When clearing as client, the trade initiated by the Executing Broker and affirmed by the client after populating the Clearing Broker in the Clearing tab.

At this point, the two parties can request that this trade be cleared in the MarkitWire GUI.

When the trade is Sent For Clearing, the SWProcesstrade keyword will be changed to SentForClearing.

Note: Declear is not available with CME through the MarkitWire interface after the trade has been cleared.

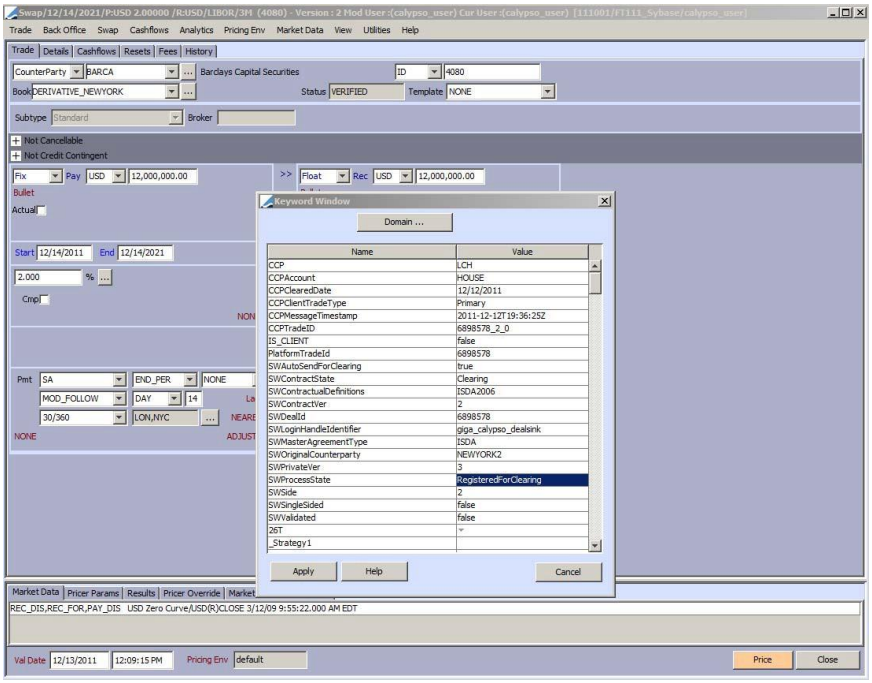
9.2 LCH Trilateral Model

When clearing as a client using the LCH Trilateral Model, a trilateral trade is being initiated by the Executing Broker and affirmed by both the client and the Clearing broker. The client initially faces the Clearing Broker.

When the trade is released, the parties then request that the trade be sent to clearing. This can be done automatically in MarkitWire.

When Sent For Clearing, the SWProcesstrade keyword is changed to SentForClearing.

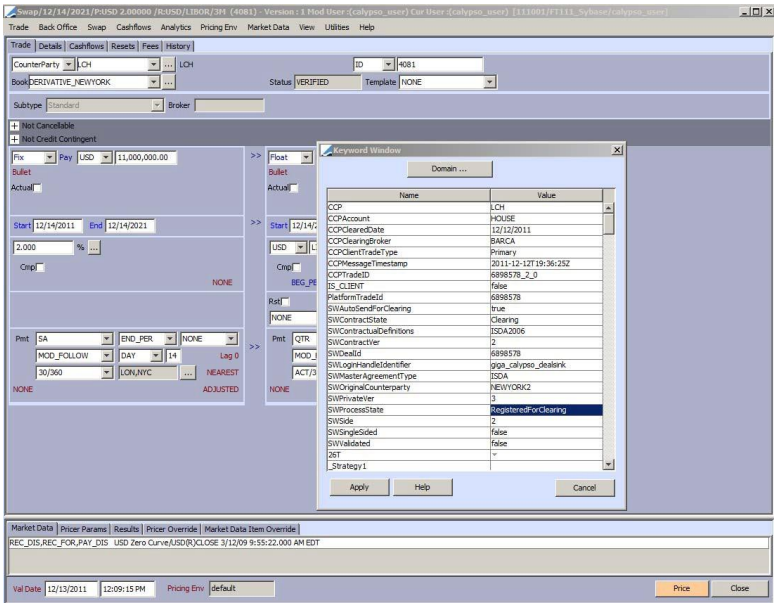
When Registered for Clearing, the SWProcesstrade keyword is changed to RegisteredForClearing in the trade that is not novated:



In case of lifecycle event after the trade has been cleared, the trade facing the Clearing Broker is amended and processed by the interface according to the changes made in MarkitWire.

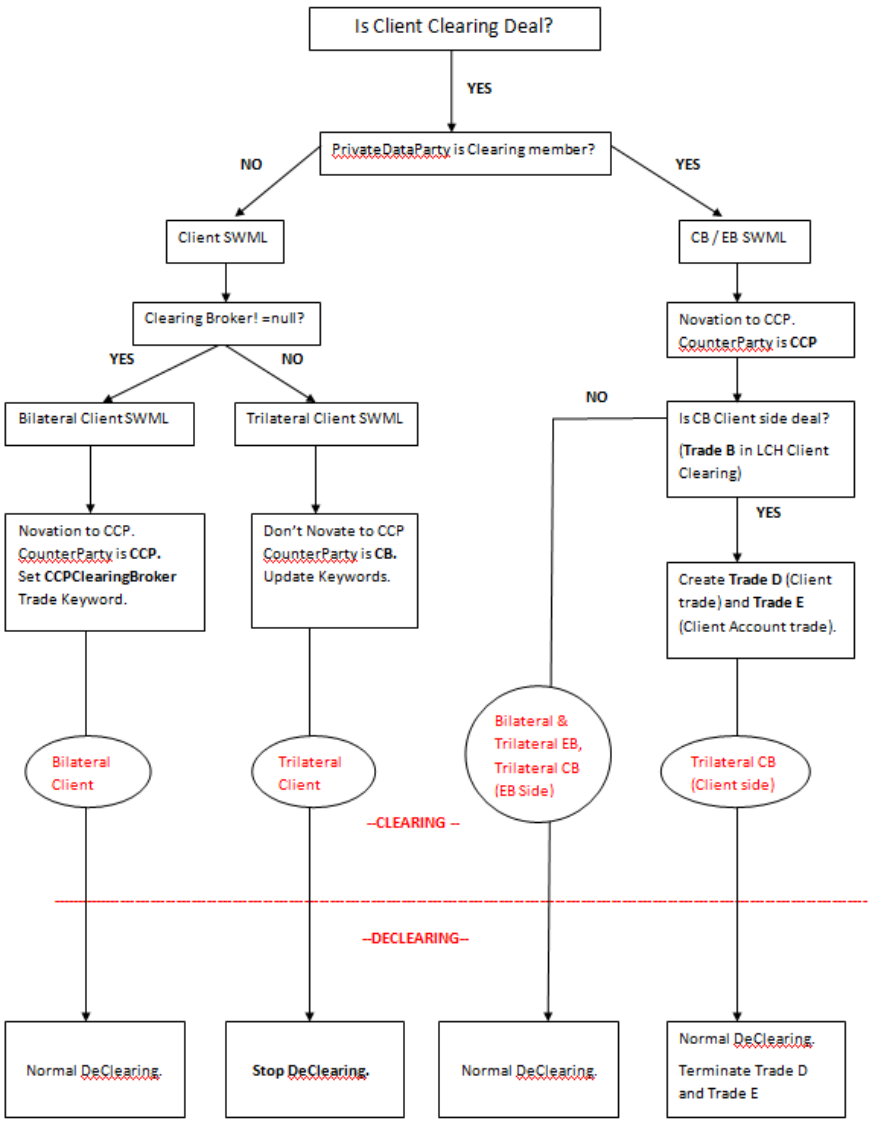
9.3 LCH FCM Bilateral Model

When clearing as client using the LCH FCM model, a bilateral trade is initiated against an Executing Broker or a client, and FCM details are entered. The FCM entity is extracted from the SWML file and stored as the **CCPClearingBroker** Trade Keyword in the cleared trade:



If a lifecycle event occurs on a cleared LCH FCM trade, it results in a declear with a novation to the original OTC counterparty, with subsequent amendment of the trade in Calypso.

9.4 Client Clearing Workflow Logic



Package Clearing Keywords Support

Packages are groups of trades that are executed as a single economic transaction.

Trades bundled together by a two part identifier called the “Package Identifier” (Issuer + Package Trade ID) are classified as a package. These trades (legs) must be intended for clearing, must be held at a state of ‘New / Pending’ until all legs are booked, and must clear ‘all or nothing’, i.e. should a particular Trade (leg) contained within a package fail to clear then the package will fail to clear.

The initial implementation of support of Package transaction submission in MarkitWire will meet the following criteria:

- Clearable OTC rates products (Single CCY IRS, Single CCY Basis Swaps, FRA,OIS)
- Packages must be initiated by a Broker (IDB) or a SEF (SEF Auto Processed, SEF Affirmation)
- The trades must be Non-allocated
- Trades can only be sent to the CME
- The same Clearing House must be used for all trades in the package.
- The same parties must be used on all trades within the package.
- The same clearing broker (if applicable) must be used for all trades in the package for a given side.

In order to process Package transactions, MW is introduced new editable fields into the Broker / SEF GUI. These fields are listed below and will be contained within a ‘Packages Trades’ Frame. The Broker / SEF may insert package trade information when selecting a CCP that is set to receive packages through MW. When a trade is submitted containing package trade elements, the Dealer / Client GUI will display the ‘Packages Trades’ frame including the package trade identifiers, however, the fields will be ‘read only’ to the receiving parties to the trade.

- Package Identifier (Issuer + Package Trade ID)
- Size of the package
- Package level Credit Acceptance Token (Credit Issuer + Credit Token) (Not supported)

10.1 Package Clearing Process

Once submitted, the legs of a package gets held in a contract state of ‘New’ and booking state of ‘Pending’ until all of the legs of the package have been processed by the Parties to the trade. Package size determines the number of trades within the package; i.e. If there are 3 trades in a package, the broker / SEF will indicate the package size as ‘3’. Once the package size has been reached the trades within the package will progress to clearing.

10.2 Package Keywords

Following are the list of keywords with sample values and SWML X-path supported in incoming dealer mode from Calypso.

sr no.	Keyword names	Xpath	Comments
1	CCPPackageIdPrefix	<SWML> <swStructuredTradeDetails> <swTradePackageHeader> <swPackageIdentifier> <swIssuer>SEF_CALYPSO_PKG_067</swIssuer>	Supported in Incoming dealer mode
2	CCPPackageIdValue	<SWML> <swStructuredTradeDetails> <swTradePackageHeader> <swPackageIdentifier> <swTradeId>TRADE_067</swTradeId>	Supported in Incoming dealer mode
3	CCPPackageSize	<SWML> <swStructuredTradeDetails> <swTradePackageHeader> <swSize>2</swSize>	Supported in Incoming dealer mode

Trade Division Support

The MarkitWire platform needs to support Trade Netting synchronization as clearing houses are now supporting advanced lifecycles like Trade netting synchronization in-order to better manage the portfolios and reduce the number of physical trades. To support Netting Synchronization, Trade division is a prerequisite for MarkitWire Platform. Hence MarkitWire Platform has been enhanced to support Trade Division functionality which involves splitting of the Alpha trade post clearing into Beta and Gamma trade between the CCP and the respective parties in Agency model and Clearing Broker and respective parties for Principal model. This will be helpful for MarkitWire platform to support the post-clearing Netting synchronization as post clearing the Beta/Gamma will be separate physical trades which can be amended individually. The Beta and Gamma trades will have a different MarkitWire deal-Id compared to the Alpha trade.

The Calypso MarkitWire module needs to keep up to the changes introduced in the MarkitWire platform and hence we support the Trade Division functionality to be compliant with MarkitWire Platform and will eventually support Netting Synchronization as and when it is supported in MarkitWire.

11.1 Scope

Support the Trade division functionality in Calypso MarkitWire interface to be compliant with the enhancements in MarkitWire platform.

- All clearable products via Calypso MarkitWire Interface to be supported.
- Subsequent post clearing lifecycles to be supported for new trade created as part of trade division.
- Support existing customers not clearing via the Trade division enabled CCPs.

The following functionality is not in scope in current support. It may be taken up for a subsequent release based on MarkitWire releases.

- Support for FCM/Clearing Broker perspective.
- Support for CCP perspective.

11.2 Assumptions

The “UpdateTermination” trade workflow rule should be added to your TERMINATE action (usually located between your Verified and Terminated status) to handle the rolling of External Reference IDs. This is already mentioned in the MarkitWire integration document in Section 2.0.

Note: Not Supported: Unilateral Amends on the Alpha trade post clearing is not supported.

11.3 Notification Handling

The following shows how the notifications from MarkitWire will be handled in Calypso:

No	MarkitWire Action	Calypso Action
1.	Alpha Trade created in MarkitWire and released.	New Alpha Trade created in Calypso.
2.	Alpha Trade Sent For Clearing	Update keywords in Calypso.
3.	Trade cleared at CCP. MarkitWire sends (Clearing,Released)	Novate the Alpha trade in Calypso to CCP. Existing Trade – Terminated. New Trade created facing CCP.
4.	Trade division happens in MarkitWire and original trade gets divided and we receive the corresponding notifications. MarkitWire sends (Cancelled,Released) notification for Alpha trade.	Update the Terminated trade in calypso with the keywords for process state, contract state etc and a new keyword – "PlatformReplacementTradeId" to have the beta trade SWDealId. We apply the action from the domain "UploadAmendAction" or AMEND action if the domain is empty to update the trade. Please make sure the action is applicable in the trade workflow.
5.	MarkitWire sends (New-Clearing,Released) for the Beta Trade.	Update the new trade facing CCP with new external reference and all new keywords including CCP keywords and a new keyword "PlatformOriginalTradeId" to have the SWDeald off the Alpha trade to indicate the clearing process is complete. We apply the action from the domain "UploadAmendAction" or AMEND action if the domain is empty to update the trade. Please make sure the action is applicable in the trade workflow.

Until Step (3) the process remains same as non-trade division enabled clearing. Hence the trades which are sent to those CCPs which do not support trade division will work as before.

After the step (5) from above table, we will have the Beta trade which is in sync with the Beta trade in MarkitWire.

We support the further lifecycle actions on the Beta cleared trade for Unilateral and Bilateral Amends as well as cancellation. The following is the new notification which is supported for the Beta cleared trades:

- Cancelled-Released

Alpha Trade Post-clearing:

Alpha trade					
Trade ID	Version	Private Version	Counterparty LE	Booking State	Contract State
15276270	3	1	Trade Division Clearing	Released	Cancelled
15276270	2	4	Trade Division Clearing	Saved	Clearing
15276270	1	4	Trade Division Clearing	Released	New

Beta Trade Post-Clearing and with an amendment performed:

Beta trade					
Trade ID	Version	Private Version	Counterparty LE	Booking State	Contract State
15137756	2	1	Trade Division Clearing	Released	Amended-Clearing
15137756	1	1	Trade Division Clearing	Released	New-Clearing

Important trade keywords on Alpha and Beta trades are as below:

Alpha trade		
	Keyword Name	Value
	PlatformReplacementTradeId	15137756
	CCP	TRADE_DIVISION_CLEARING_HOUSE
	SWContractState	Cancelled
	SWProcessState	Released
	SWContractVer	3
	SWPrivateVer	1
	SWDealId	15276270
Beta trade		
	Keyword Name	**Value**
	PlatformOriginalTradeId	15276270
	CCP	TRADE_DIVISION_CLEARING_HOUSE
	SWContractState	New-Clearing
	SWProcessState	Saved
	SWContractVer	1
	SWPrivateVer	1
	SWDealId	15137756
	New External Reference	MW_Calypso Bank_15137756
Amend on Beta trade		
	Keyword Name	**Value**
	PlatformOriginalTradeId	15276270
	CCP	TRADE_DIVISION_CLEARING_HOUSE
	SWContractState	Amended-Clearing
	SWProcessState	Released
	SWContractVer	2
	SWPrivateVer	1
	SWDealId	15137756

11.4 Do Recovery of Trade Division

At times MarkitWire query result xml has the trade division alpha and beta trades in an improper order. We have added support to re-arrange the same and process in order. This needs MarkitWire API 12.2 and higher.

11.5 Legacy Trade Migration for the Trade Division Functionality

As part of trade division MarkitWire will be dividing the legacy client trades into Beta and Gamma trades as a scheduled activity. This will enable the legacy trades to take part in netting synchronization process. Calypso MarkitWire interface supports the migration of the legacy trades to the divided trades – Alpha/Beta via the following mechanisms:

11.5.1 Updating the Legacy Trades in Calypso to Divided Trades in MarkitWire via the CSV file

MarkitServ will be performing the legacy trade migration as per scheduled time and provide a CSV file to clients which has the details of each Alpha and Beta trades that were part of migration and this CSV can be used to update the corresponding trades in Calypso.

Once updated, post clearing lifecycles will be supported on the Beta cleared trades.

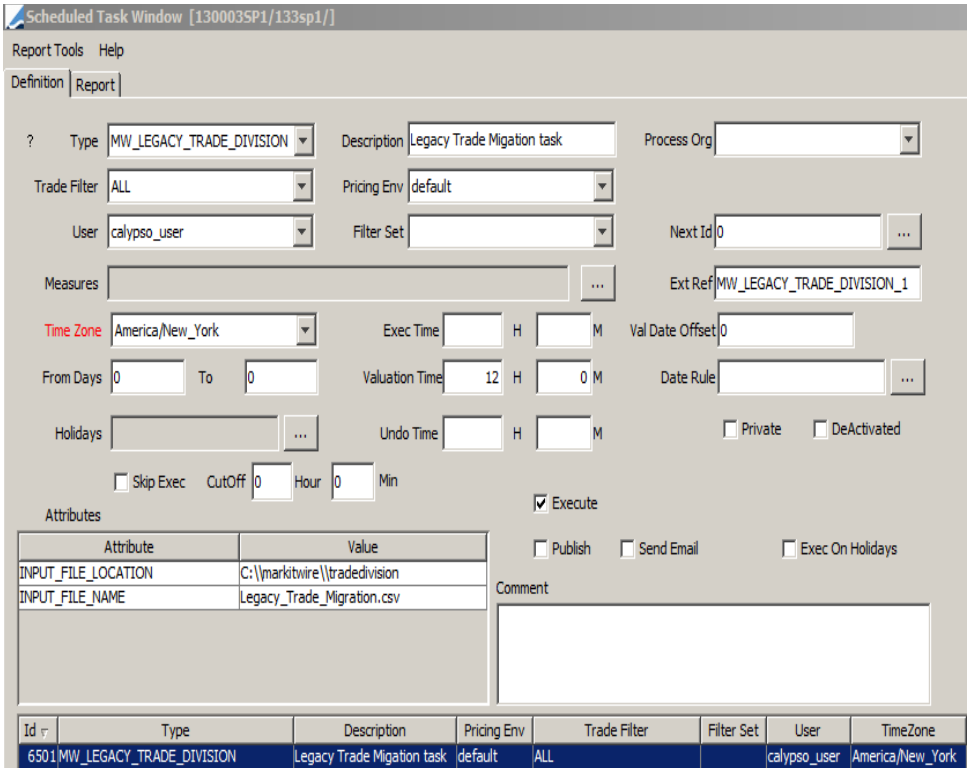
To support this we have provided a Scheduled Task in Calypso which can be run by passing the MarkitWire CSV as an input to perform the migration in calypso.

11.5.2 Scheduled Task Configuration

To configure a new schedule task go to MainEntry -> Configuration -> ScheduledTasks -> Scheduled Tasks

Select the Type as MW_ LEGACY_TRADE_DIVISION

Snapshot for MW_ LEGACY_TRADE_DIVISION in Scheduled Task Window



Enter the following mandatory attributes

- Input File Location: Directory from which the file will be picked
- Input File Name: Name of the csv file for legacy trade migration with extension

Scheduled task Assumptions:

- Trade which is getting migrated is cleared in calypso and there are two trades in Calypso, one Terminated trade and another Novated trade with the CCP as the trade counterparty.
- In MarkitWire, trade division is already performed on the corresponding trade which is being migrated and we have the Alpha and Beta trades available in MarkitWire to whom the Calypso trades will be migrated.
- The Scheduled task will update the existing trades in Calypso to Alpha and Beta trade details from CSV, so there is an action available on both these trades in Calypso to perform the update. The action can be set in the domain "UploadAmendAction" or we use the AMEND as an action to be applied on these trades.

Criteria to fetch the terminated and cleared version of Alpha trade:

- The scheduled task fetches all the trades having the keyword "SWDealId" with value same as the field "Alpha Trade Id" from the input migration CSV file. From the trades it fetches, it expects two trades to match these criteria – a Terminated trade and a Verified Trade.
- The Terminated trade is considered as the Alpha Terminated Trade if it meets the following criteria:
  - Trade keyword "TransferTo" is populated.
  - Trade keyword "TerminationReason/TransferReason" is populated with value "Clearing".

    The Alpha Terminated Trade keywords are updated with the new status.

    The Verified trade is the Cleared Alpha trade if it meets the following criteria:
  - Trade keyword "TransferFrom" is populated.
  - Trade keyword "TerminationReason/TransferReason" is populated with value "Clearing".

    This Cleared Alpha trade is modified to be in-line with Beta Cleared Trade in MW. The SWDeal ID is updated with the Beta trade id. The Trade External Reference is also updated to allow further lifecycle on this trade.
- For amending trades, the action configured in "UploadAmendAction" domain is used. If domain is not configured, then AMEND action is applied.

Screenshots of Calypso trade keywords pre and post migration:

Snapshot of trade keywords compare of Terminated trade which got updated to Alpha trade.

Terminated Trade		Alpha Trade	
Before Running Schedule Task		After Running Schedule Task	
Keyword Name	Value	Keyword Name	Value
SWContractState	Clearing	SWContractState	Cancelled
SWContractualDefinitions	ISDA2006	SWContractualDefinitions	ISDA2006
SWContractVer	2	SWContractVer	3
SWDealId	15273041	SWDealId	15273041
SWLoginHandleIdentifier	calyp_dealsink7	SWLoginHandleIdentifier	calyp_dealsink7
SWMasterAgreementType	ISDA	SWMasterAgreementType	ISDA
SWOriginalCounterparty	AAA BANK TDV	SWOriginalCounterparty	AAA BANK TDV
SWPrivateVer	3	SWPrivateVer	1
SWProcessState	RegisteredForClearing	SWProcessState	Released
SWSendForClearingTimestamp	01-09-2014 10:38	SWSendForClearingTimestamp	01-09-2014 10:38
SWSide	1	SWSide	1
SWSingleSided	FALSE	SWSingleSided	FALSE
SWValidated	FALSE	SWValidated	TRUE
TerminationDate	09-01-2014	TerminationDate	09-01-2014
TerminationFullFirstCalculationPeriod	Y	TerminationFullFirstCalculationPeriod	Y
TerminationPayIntFlow	Y	TerminationPayIntFlow	Y
TerminationReason	Clearing	TerminationReason	Clearing
TerminationTradeDate	09-01-2014 16:11	TerminationTradeDate	09-01-2014 16:11
TerminationType	Novation	TerminationType	Novation
TradeSource	MW	TradeSource	MW
TransferTo	42804	TransferTo	42804
		PlatformReplacementTradeId	15273043
		CCPTradeID	ABC00012345867
	Keywords which changed		
	Keywords which added		

Snapshot of trade keywords compare of Cleared trade which got updated to Beta trade.

New Novated Trade		Beta Trade	
Before Running Schedule Task		After Running Schedule Task	
Keyword Name	Value	Keyword Name	Value
SWContractState	Clearing	SWContractState	New-Clearing
SWContractualDefinitions	ISDA2006	SWContractualDefinitions	ISDA2006
SWContractVer	2	SWContractVer	1
SWDealId	15273041	SWDealId	15273043
SWLoginHandleIdentifier	calyp_dealsink7	SWLoginHandleIdentifier	calyp_dealsink7
SWMasterAgreementType	ISDA	SWMasterAgreementType	ISDA
SWOriginalCounterparty	AAA BANK TDV	SWOriginalCounterparty	AAA BANK TDV
SWPrivateVer	3	SWPrivateVer	1
SWProcessState	RegisteredForClearing	SWProcessState	Saved
		PlatformOriginalTradeTradeId	15273041
		NewExternalRef	MW_CALYPSO BANK_15273041
		PriorUSIPrefix	1010000236
		PriorUSIValue	MARKITWIRE00000000000000001234567
		ReportingCFTCPriorUSIPrefix	1010000236
		ReportingCFTCPriorUSIValue	MARKITWIRE00000000000000001234567
		ReportingCFTCUSIPrefix	1010000051
		ReportingCFTCUSIValue	AB100000000ABC00012348756028492 9
		USIPrefix	1010000051
		USIValue	AB100000000ABC00012348756028492 9
	Keywords which changed		
	Keywords which added		

11.5.3 Running the Do-Recovery Post Migration

We can run the Do-Recovery at the engine startup or as a scheduled task to migrate the existing legacy trades to Alpha and Beta trades post Trade Division. The do-recovery will query the deals and update the corresponding Calypso trades with Alpha and Beta trades for the divided trades. The subsequent post clearing lifecycles can be performed on these migrated Beta trades once the migration is complete via recovery process.

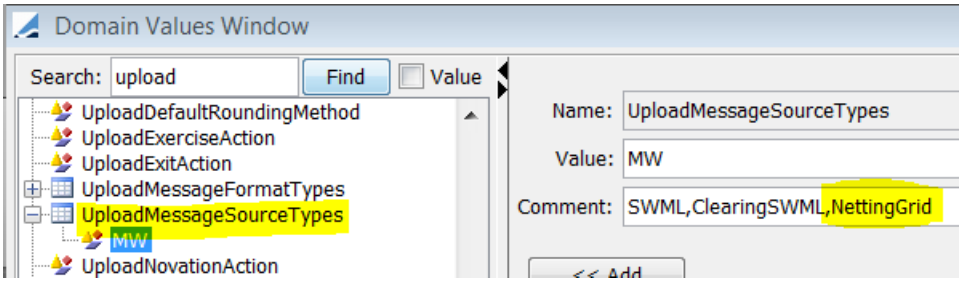
If the trades are already migrated using the scheduled task then the do-recovery will not impact the trades in Calypso.

Netting Synchronization Support

The Calypso MarkitWire interface supports the Netting and Compression functionality.

Trade division is a pre-requisite to Netting and the trades that follow the trade division process of clearing can only be part of the Netting compression cycle.

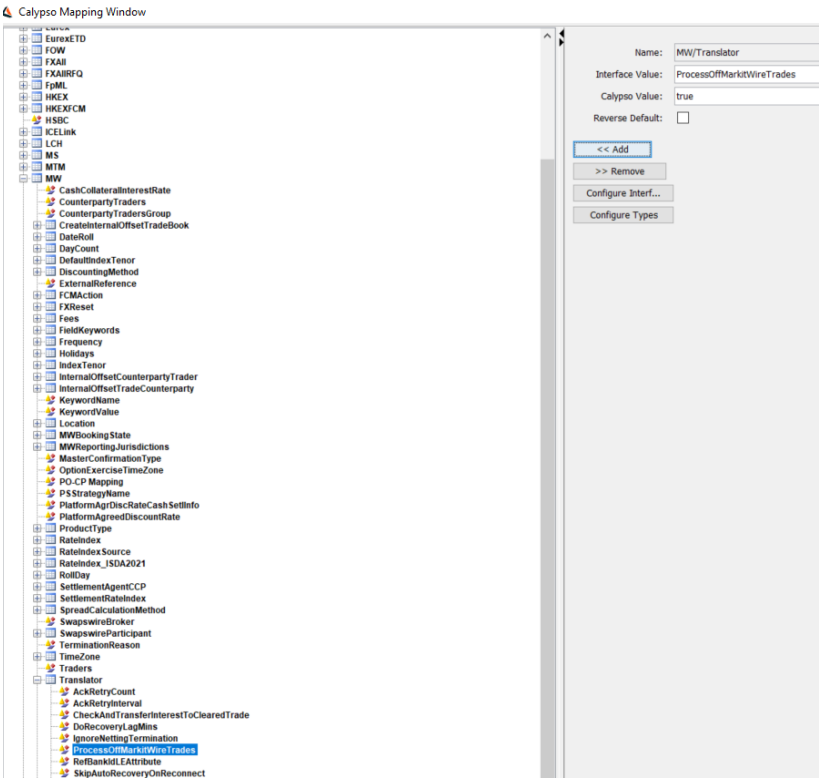
Please ensure that “NettingGrid” is present in the domain values for the domain “UploadMessageSourceTypes”. It will be added for the new installation via the Execute SQL process. For upgrader, please ensure this is added as per the below screenshot for the NettingGrid format to be supported by the interface.



CCPs perform the unilateral compression of the trades post clearing and MarkitWire will be synchronizing the corresponding trades in MW with the netting process.

To automatically terminate the off-markitwire trades in Calypso post trade compression activity. Although Off-Markitwire trades do not have the "usual" MarkitWire details (e.g. trade keywords, external reference, etc.) the netting grid message contains the CCP Trade ID identifier. On the other hand, the ‘CCPTradeId’ keyword is populated for all off-markitwire cleared swaps in Calypso and matches the ‘ClearingHouseTradeId’ included in the netting grid.

Therefore, Off-MarkitWire trades could be potentially identified in Calypso based on the ‘ClearingHouseTradeId = CCPTradeId’ argument and automatically terminated when ‘TypeOfNettingEvent = Termination’, whilst populating the usual netting keywords on the terminated trades. We have added a mapping called ‘ProcessOffMarkitWireTrades’ in the mapping window.



The Netting is of the following two types:

12.1 Full Netting

The full netting is the process where all the trades that are part of the netting run will be terminated and there would not be any residual position so no new trade will be created.

We will receive Cancelled/Released notifications for all the trades that are terminated as part of the netting run from MarkitWire.

We will perform termination of the corresponding Beta trades in Calypso and populate the netting keywords on the trade.

12.2 Partial Netting

The partial netting is a process where all the trades that are part of the netting run will be terminated, and a new trade will be created for the residual position.

We will receive the Cancelled/Released notifications for the terminated positions and New-Clearing/Released notifications for the new position as part of the netting run when there is a residual position.

The handling of Cancelled/Released notifications will be same as full netting case.

We will create a new trade in Calypso for such New-Clearing notifications with the data from the incoming SWML with the netting keywords.

We need to make the event creation lag/delay for the New-Clearing message configurable. Currently it adds delay of 2 secs prior to every event creation for the New-Clearing messages. This was done to overcome issues of out of order processing in case of bulk message processing.

The following new property added:

```
#delay for new clearing deal notify call back event creation. value in milliseconds.
#NewClearingEventCreationLag=200
```

Apart from this there is a change done to add delay only for notifications for which we would create event and process them and not for the regulatory reporting notifications which we just skip.

12.3 Netting Grid

MarkitWire platform updates the Netting Grid as and when there are updates on the trades due to netting process.

We will receive the notification for the Netting Grid same as we get a trade notification. We will process a netting grid notification only if it has Complete or Complete with error status.

We update keywords on the netted trades as part of the netting grid processing.

Calypso added support for Off-MarkitWire trades as part of Netting Grid notification from MW. It is introduced as a mapping category and if it is enabled then it would apply termination on the off-markitwire trades for which we find corresponding trade in Calypso based on the CCP trade Id.

It also includes the ids of the Off-MarkitWire trades which we ignore in Calypso and create a warning message in task station to indicate it is ignored along with the CCP id of the same.

If there is any error in MarkitWire while performing the netting synchronization we get the status as Error in the netting grid with the error reason. We save the same as trade keywords on Calypso trade – PlatformTradeNettingStatus and PlatformTradeNettingErrorReason.

12.4 Error Handling

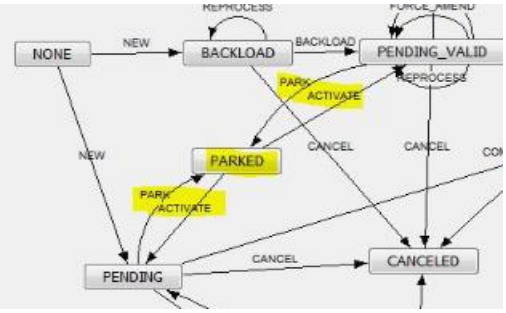
As part of the netting grid update we do keyword updates on Calypso trades. When the notifications come from MW out of order the Netting Grid update might fail in validations if the underlying trade is not terminated when applying the netting grid update.

In such cases if the Netting grid keyword update messages are the only ones pending then reprocess the messages to get the trades updated with netting keywords.

If the trade termination messages are also pending status along with the netting grid messages, then:

- We need to move the netting grid message to PARKED status by applying PARK action.
- Reprocess the termination message.

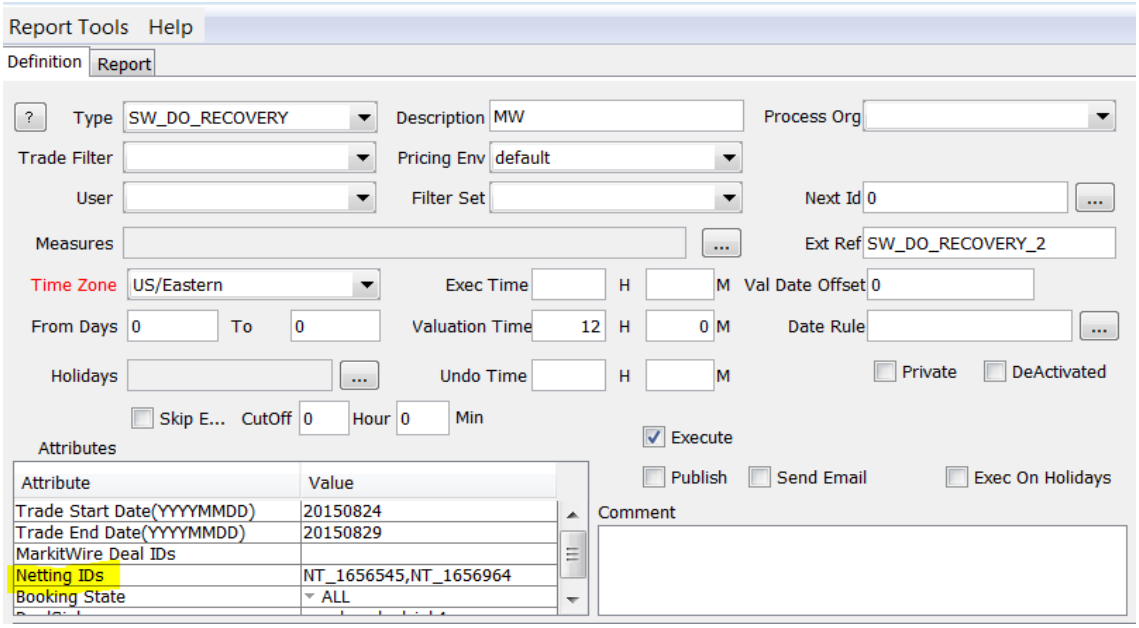
- Move the Netting message back to PENDING status by applying the ACTIVATE action.
- Reprocess the netting grid message.
- Workflow screenshot for reference:



12.5 Do Recovery of Netting Trades

We can run the do-recovery to recover the trades that were netted when the SwapswireTradeEngine was not connected to MarkitWire server. The currently supported Do Recovery process will work for the netting which is initiated as part of the engine startup. Please ensure you have the latest MarkitWire API to be able to recover the netting messages. We have enhanced the Do Recovery Scheduled task to accept the Netting batch ids in a comma separated format to allow recover the netting messages for the trades related to the provided Netting batch id. The scheduled task will not recover the original trade messages, it will only recover the netting-terminate / netting-new and netting grid messages. Please note that there can be ordering issues when running the Do Recovery of netting messages and this is a known limitation from MarkitWire platform. In case of such issues the messages will be stuck in the pending status with proper validation error messages. Please follow the Error Handling mechanisms mentioned in above section of the document.

The below is the screenshot of the Do Recovery Scheduled task:



12.6 Netting Trade Keywords

12.6.1 Common Keywords on both Terminated and new-remnant Trade

No	Keyword Name	Description
1	CCPNettingString	This is populated on all trades that are needed to be part of particular netting run. This will be updated as unilateral amends and will be sent to CCP by MarkitWire platform. This will be present on the Beta trades prior to netting.
2	CCPNettingId	A common netting Id assigned by the CCP. It will be common for all trades that are part of the netting. It will be unilateral read-only field for dealers. To be stored on all trades part of netting.
3	CCPNettingEventType	It will be assigned by CCP to indicate the type of netting event. It will have the value – “Netting” for Netting/Compression and the value “Other” for any other post clearing event such as transfers or default management. It will be a unilateral read-only field for dealers. To be stored on all trades part of netting.
4	PlatformNettingStatus	We will add a new keyword on each trade to indicate the status of the overall netting process. It can have one of the following values:

No	Keyword Name	Description
		• Complete – Netting completed successfully. • Complete with error – Netting process had errors at MW.
5	PlatformTradeNettingStatus	We will add a new keyword on each trade to indicate the status of the trade in MW netting process, As per the Netting Grid SWML, there can be errors in the netting process for a particular trade. It can have one of the following values: • Created – If the trade has been successfully created • Cancelled – If the trade has successfully been cancelled • Error – If the trade failed to be cancelled or created.
6	PlatformTradeNettingErrorReason	This keyword will have an error reason for any trade in an error status while processing the netting for a particular trade in MarkitWire.

12.6.2 Keywords on Terminated Trade

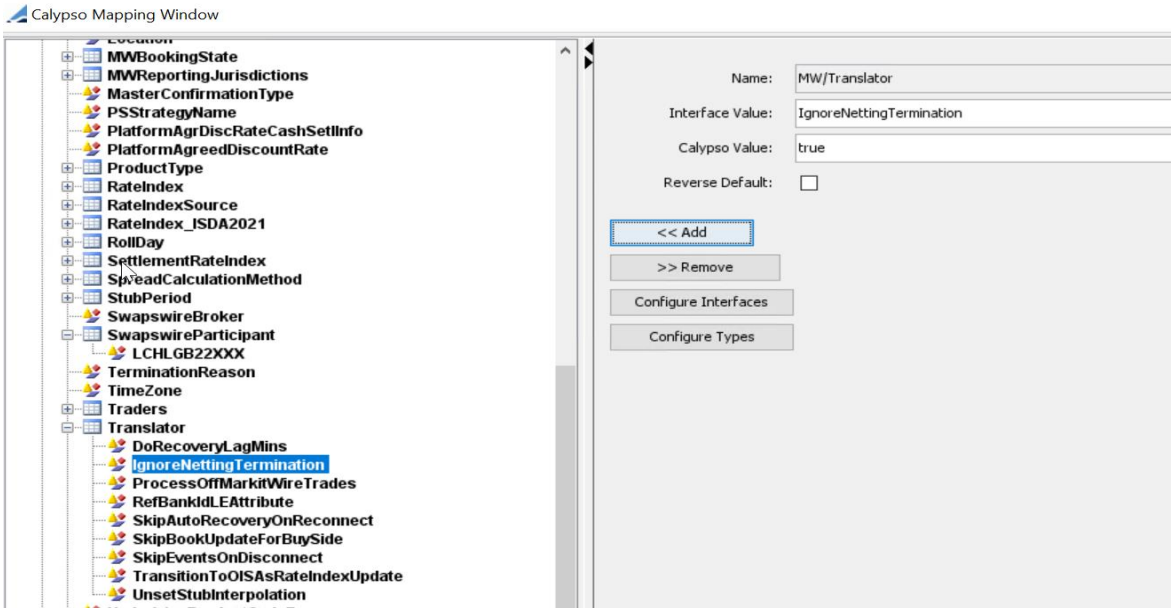
No	Keyword Name	Description
1	CCPReplacementTradeId	CCP Id of the remnant trade on all terminated trades. Will get populated only in case of partial-netting. It can have multiple values in case of coupon blending. And the values will be separated by space.
2	CCPTerminatingEvent	This is to be stored on TERMINATED trades with value – PARTIAL_NETTING or FULL_NETTING based on the netting type.
3	TerminationReason	The trades that are terminated as part of netting have the termination reason as - “Netting”.
4	PlatformReplacementTradeId	SW deal id of remnant trade on all terminated trades. Will get populated only in case of partial-netting. It can have multiple values in case of coupon blending. And the values will be separated by space.

12.6.3 Keywords on Remnant Trade

No	Keyword Name	Description
1	CCPOriginalClearedDate	It will have the earliest cleared trade's cleared date on the netting remnant trade.
2	CCPHistory	List of CCP Ids of TERMINATED trades to be stored in this keyword on the NEW trade. The CCP Ids will be separated by space. If the number of trades netted exceeds 50 then we will create another keyword – CCPHistory1, CCPHistory2 and so on. We will use the Netting grid to populate this.
4	CCPOriginatingEvent	This will be stored on the NEW trade with the value – NETTING_REMNANT.
5	PlatformOriginalTradeId	SW deal ids of the netted trades on the remnant trade. The CCP Ids will be separated by space. If the number of trades netted exceeds 50 then we will create another keyword – PlatformOriginalTradeId1, PlatformOriginalTradeId2 and so on.

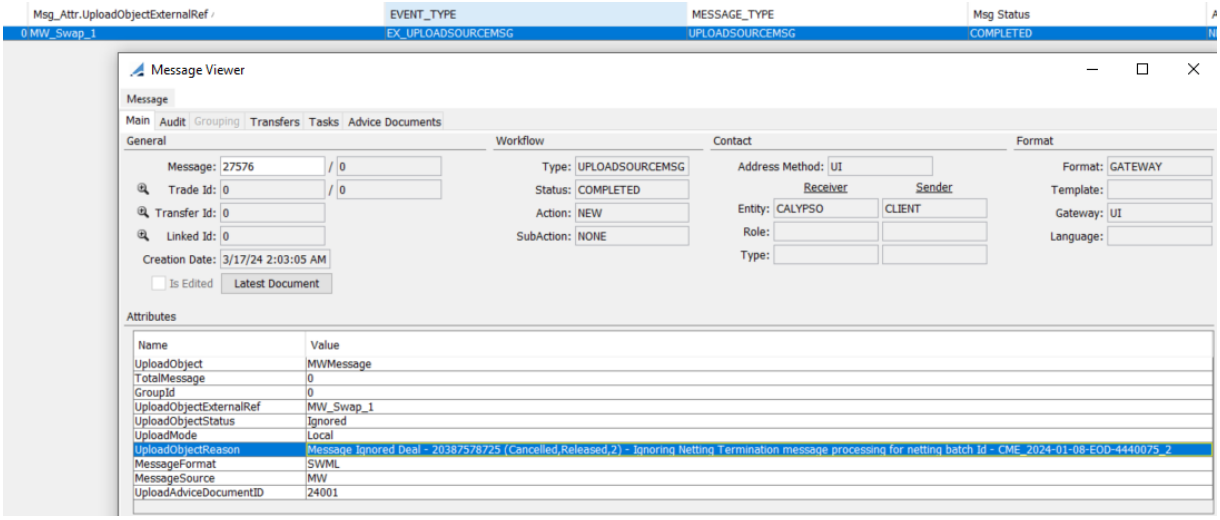
12.7 Automatic Swap Terminations

To handle automatic swap terminations, Calypso required the ability to suppress the incoming termination messages from MarkitWire. To do so, Calypso has provided the ability to skip/ignore incoming MarkitWire compression events while raising an exception to be viewed. You can also turn on/off this via setting the IgnoreNettingTermination domain value.

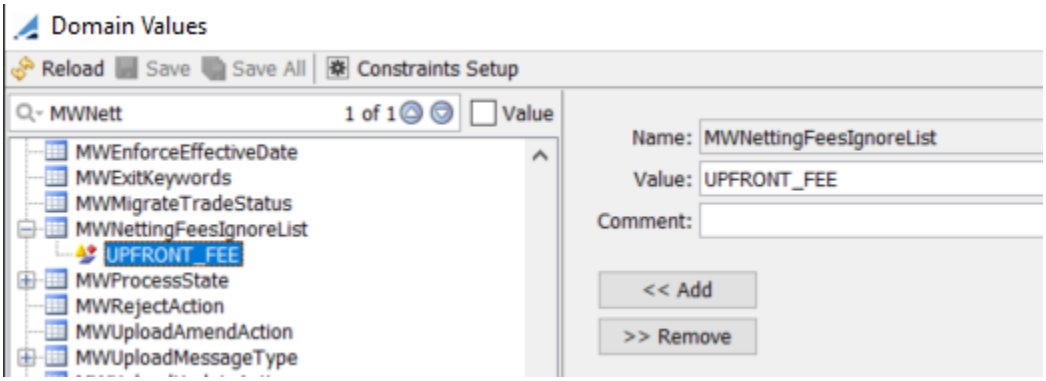


Say, if the contract state is Cancelled and the process state is Released, then Calypso check for netting details being available and presence of swNettingBatchId. If found and IgnoreNettingTermination value is set to "true".

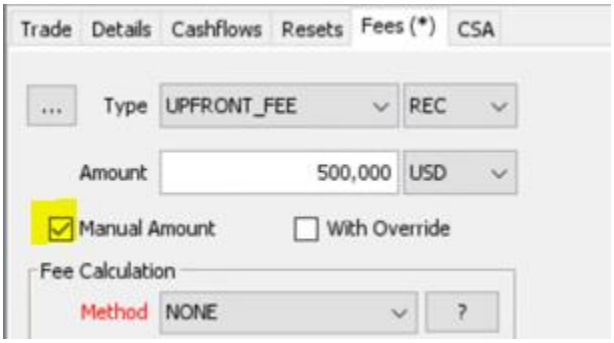
Then the message processing is ignored, and BO Message is moved to COMPLETED status. Following error message is displayed.



Calypso enhanced the UploadPreserveFee logic for cases where the existing fee type and incoming fee type are the same. Currently, when the fee type is added to the UploadPreserveFee domain, the incoming fee is booked on top of the existing one. So, Calypso has added **MWNettingFeesIgnoreList** domain value, which provides the ability to preserve the fee amount in Calypso and skip/ignore the MartkitWire incoming fee, potentially by setting a flag on the existing fee in Calypso.



Manual Amount flag on the fee added on underlying trade manually:



Let's say if the fees in the current notification are not empty then check if it is for Netting Termination. Then, Calypso looks up for the domain "MWNettingFeesIgnoreList" and check if it is not empty and the fee type in netting notification is part of this domain to ignore.

By fetching underlying Calypso trade and check if it has fees present. For each underlying Calypso trade fee, we check the following conditions:

- If the underlying trade fee type matches the fee type from netting notification.
- If the underlying trade fee type has the "Manual Amount" flag set to true.

Then ignore the fee from the Netting notification.

CCP Mode - Trade Division and Netting Support

The section provides details on the design and implementation of the support for Trade division and Netting synchronization from the CCP (Exchange Clearing) perspective.

13.1 Overview

Support for Trade division

- Incoming trade external reference to have beta id and Storing alpha id as keyword.
- Message and trade linking change as id changes.
- Acknowledgements to have alpha id.
- Clear accept/reject scenarios.
- Do-Recovery.
- Legacy trade migration.

Support for Netting Synchronization

- Support for full netting.
- Support for partial netting which requires sending new positions to MW.
- For new positions support for all cleared products, stubs, fees etc.
- Ability to reprocess the outgoing netting message in case of failures.
- Do-Recovery.
- Ability to send one trade per netting API call to MarkitWire.
- Ability to process response of one trade coming from MarkitWire.

Support for Portfolio Transfer

- Similar support as partial netting with following differences:
  - There are only two trades. One transferred which is sent as Terminated position and one new cleared trade which is sent as New position.
  - CCP Netting Id is different on the transferred and new trade.
  - Counterparty is different on the transferred and new trade.
  - We send one Netting Instruction message per trade with different Netting Instruction XML message.

Support for Default Management

Same as portfolio transfer.

13.1.1 Assumptions

The following are the assumptions:

- Calypso creates the trades with external reference having Beta/Gamma ids if it is present in the incoming Clearing XML.
- The alpha id is stored in the trade keyword – PlatformOriginalTradeId on Calypso trades.
- We skip the extra notification to update beta/gamma ids. So the new versions are not updated on the trades.
- We use the alpha id from the keyword “PlatformOriginalTradeId” while sending out the acknowledgements.
- Trade linking rules are not applied post clearing as trades need to be netted separately for each side.
- De-clears are not supported as Client can book an offsetting trades and then do netting. It is not supported in MW post clearing.
- We assume the trades are netted correctly in Calypso once the trades move to a corresponding status in trade workflow. This can be configured in domain – Clearing.Trade.NettingAction and Clearing.Trade.NettingStatus.
- New positions need to have the MW keywords pre-populated before sending the netting details to MW.
- We create the BOMessage – PlatformMsg to send the outgoing message and provide re-processing ability on the same in case of data validation errors.
- For Legacy migration, we update the Calypso trades as per the MW CSV for keywords by applying the action configured in domain “UploadAmendAction” or “UploadUpdateAction” and if not configured, use the action AMEND by default. The details are logged as we update the trades.
- We handle the message and trade linking using the keyword “PlatformOriginalTradeId” which has the alpha id as the trades are created with external reference having the beta ids.
- We send separate NettingInstructionXML messages for the transferred trade and the new cleared trade in case of Portfolio transfer and Default management as Terminated and New positions have different CCPNettingId.

13.2 Trade Division Use Case

- Trade counterparties send trade to Clearing.
- CCP receives a notification trade is Pending Clearing.
- CCP retrieves Clearing XML.
- Clearing XML contains both the reserved BetaCleared trade IDs for the counterparties, as well as the original Alpha MarkitWire ID.
- CCP creates trades with external ref having BetaCleared ids and save alpha id in keyword – “PlatformOriginalTradeId”.
- CCP accepts trade for Clearing and sends the ack having the Alpha MarkitWire ID.
- Trade division process begins on MarkitWire:
  - Trade moves to Clearing Registered (AlphaCleared trade)
  - Divided (BetaCleared) trades are created for both counterparties and AlphaCleared trade is cancelled.
- CCP elects to not receive a notification of the creation of the BetaCleared trades.

- We do not update the contract-process state to New-Clearing and we do not update the contract-private version in Calypso.

13.3 Netting Synchronization Use Case

Full Netting

- CCP updates the netting keywords on the selected trades and move them to NET_TERMINATED.
- MW engine listens to trade events when the trade moves to the Clearing.Trade.NettingAction and Clearing.Trade.NettingStatus. It fetches the netting-participant trades using the CCPNettingId and looks at the count – CCPNettingCount and if the trades fetched match the count. It starts the process of generating the netting message.
- We have a sequence policy set to CCPNettingId so that the trades for the same netting run are processed by the same Engine thread.
- MW engine receives the trade event.
- MW engine creates and saves the PLATFORM_MSG.
- MW engine calls the platform translator which generates the NettingInstructionXML.
- MW engine sends the NettingInstructionXML to MW via the MW API and receives a correlationId.
- Using CorrelationId, MW engine retrieves the NettingInstructionRespXML.
- MW engine updates the Calypso trades with PlatformNettingStatus.

Partial Netting

- CCP updates the netting keywords on the selected trades, moves them to NET_TERMINATED status, and creates the remnant trade by novating one of the participant trades (oldest in time) and moves it to CLEARED status.
- The remnant trade is expected to get all keywords from the trade from which it is novated, and CCP updates the Cleared USI Prefix and Cleared USI Value on the trade keywords.
- MW engine receives the trade event.
- MW engine creates and saves the PLATFORM_MSG having the details.
- MW engine calls the platform translator which generates the NettingInstructionXML having terminated position as well as the new position.
- MW engine sends the NettingInstructionXML to MW via the dealsink API and receives a correlationId.
- Using CorrelationId, MW engine retrieves the NettingInstructionRespXML.
- MW engine updates the Calypso trades with PlatformNettingStatus.

Partial Netting with Validation at Calypso

- CCP updates the netting keywords on the selected trades, moves them to NET_TERMINATED status, and creates the remnant trade by novating one of the participant trades (oldest in time) and moves it to CLEARED status.
- The remnant trade is expected to get all keywords from the trade from which it is novated and CCP updates the Cleared USI Prefix and Cleared USI Value on the trade keywords.
- MW engine creates and saves the PLATFORM_MSG having the details.
- MW engine calls the platform translator which generates the NettingInstructionXML, and calls the product specific translator for the new trade.
- If there are any errors in translation – like missing mapping etc., the Platform message remains in pending status, and trade keywords and logs are updated to indicate the status.
- The errors need to be rectified and trade needs to be re-saved to do the translation again. If successful, the message is sent to MW.
- MW engine sends the NettingInstructionXML to MW via the dealsink API and receives a correlationId.
- Using CorrelationId, MW engine retrieves the NettingInstructionRespXML.
- MW engine updates the Calypso trades with PlatformNettingStatus.

Partial Netting with Validation at MarkitWire

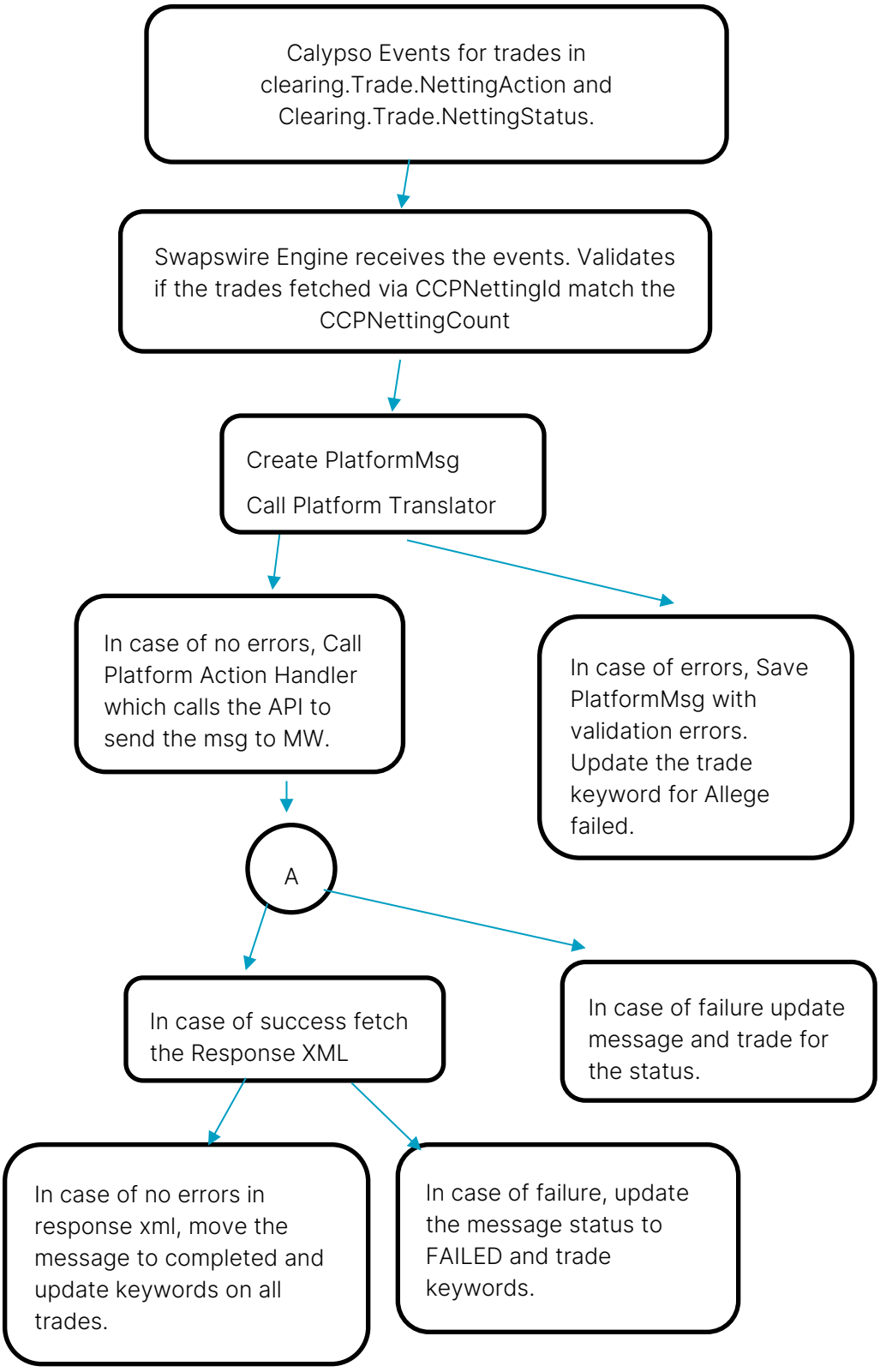
- CCP updates the netting keywords on the selected trades, moves them to NET_TERMINATED status, and creates the remnant trade by novating one of the participant trades (oldest in time) and moves it to CLEARED status.
- The remnant trade is expected to get all keywords from the trade from which it is novated and CCP updates the Cleared USI Prefix and Cleared USI Value on the trade keywords.
- MW engine creates and saves the PLATFORM_MSG having the details.
- MW engine calls the platform translator which generates the NettingInstructionXML.
- MW engine sends the NettingInstructionXML to MW via the dealsink API and receives a correlationId.
- Using the CorrelationId, MW engine retrievee the NettingInstructionRespXML.
- If there are errors in the NettingInstructionRespXML like - <ErrorText>Error: *** fixedLeg calculationPeriodDates/calculationPeriodDatesAdjustments/businessDayConvention for a ZC IRS must be set to 'NONE'. Value = MODFOLLOWING</ErrorText>, then the Platform message remains in pending status, and trade keywords and logs are updated to indicate the status.
- MW engine updates the Calypso trades with PlatformNettingStatus.
- The errors need to be rectified and trade needs to be re-saved to do the translation again. If successful, the message is sent to MW.

13.4 Portfolio Transfer Use Case

- CCP moves the original cleared trade to the status TRANSFERRED.

- CCP creates a new position trade in the CLEARED status.
- The trades are linked via CCPTransferFrom and CCPTransferTo keywords.
- MW engine listens to the CLEARED trade event and checks if it has the TransferFrom keyword. If so, it then checks if the linked trade is in TRANSFERRED status.
- MW engine also listens to trade events for status Clearing.Trade.TransferAction and Clearing.Trade.TransferStatus which are Transferred status. It checks for the new trade to be available in a cleared status.
- MW engine creates two NettingInstructionXML messages, one of them having the termination for the transferred trade and another one having the New position for the cleared trade while processing the corresponding event.
- MW engine sends the NettingInstructionXMLs to MW via the API and receives a correlationId.
- Using CorrelationId, MW engine retrieves the NettingInstructionRespXML.
- MW engine updates the Calypso trades with PlatformNettingStatus and in case of failure, it moves the corresponding message to FAILED status which can be reprocessed.

13.5 Approach



13.6 Trade keywords

Keyword Name	Description
CCPNettingCount	Count of trades involved in Netting process.
TerminationDate	Date when netting is performed. Equivalent to Trade Termination date. Only needed for terminated positions.
TerminationTradeDate	Date when netting is effective. Equivalent to Termination effective date. Only needed for terminated positions.
CCPNettingId	Netting batch id.
CCPNettingEventType	For full and partial netting – "Netting" for other lifecycles like default management and Portfolio Transfer – "Other". This needs to be sent to MW.
CCPNettingString	Populated by dealers as unilateral field.
GlobalUTI	Captures and displays the Namespace (LEI) + Value in a single field as a concatenated value
TransferTo	The id of the cleared trade to which the current trade is transferred to. This is saved on the trade in status TRANSFEERED.
TransferFrom	The id of the transferred trade for which the current trade is created. This is saved on the cleared.
TerminationReason	Trades that are terminated as part of netting have the termination reason as - "Netting".
PlatformNettingStatus	Status of the overall netting process: • Complete – Netting completed successfully. • Complete with error – Netting process had errors at MW.
PlatformTradeNettingStatus	Status of the trade in MW netting process: • Cancelled – If the trade has successfully been cancelled • Pending – If the trade is still to be cancelled or created • Error – If the trade failed to be cancelled or created. • Created – If the trade has been successfully created
PlatformTradeNettingErrorReason	Error reason for any trade in an error status while processing the netting for a particular trade in MarkitWire.
PlatformOriginalTradeId	Alpha Id to be stored on Beta trades in MW. This is used for Linking of trades as well as in sending clearing acknowledgements.
SWDealId	Beta id of MW trade. We store it in Calypso trade as SWDealId as we get this pre-allocated Beta id in the first message from MW.

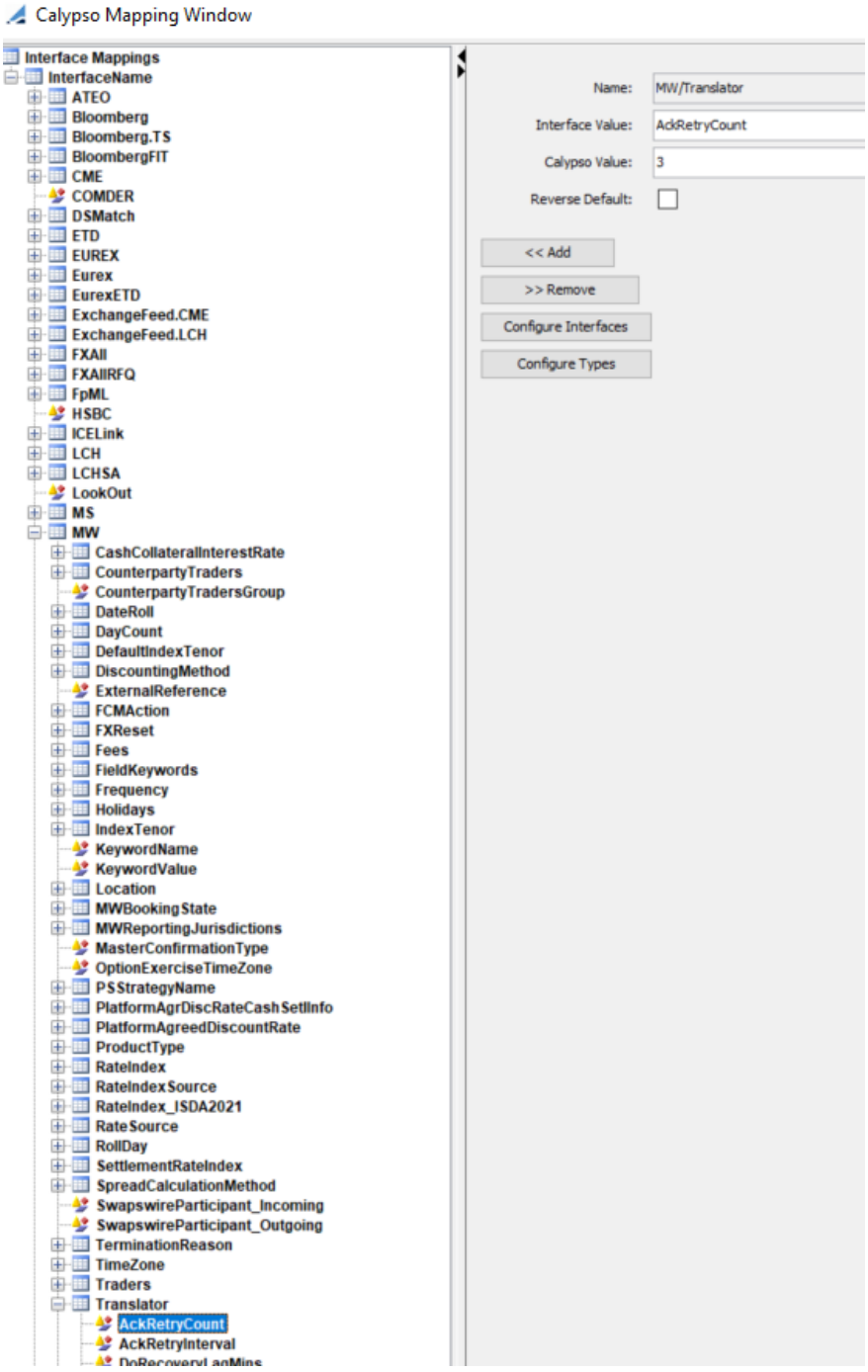
Swaps Engine should now process the net trade and create a new remnant trade in Markitwire with New GlobalUTI value under the regulatory reporting Tab of Markitwire.

For sending acknowledgements Calypso has added a new functionality where they can now add these retry count interval. We have added entries in Calypso Mapping Window in source - MW under - Translator:

AckRetryCount -> (Numeric value e.x. 3)

AckRetryInterval -> (Numeric value e.x. 5000 (in milliseconds))

If these are set, then it would do a retry for the ack sending.

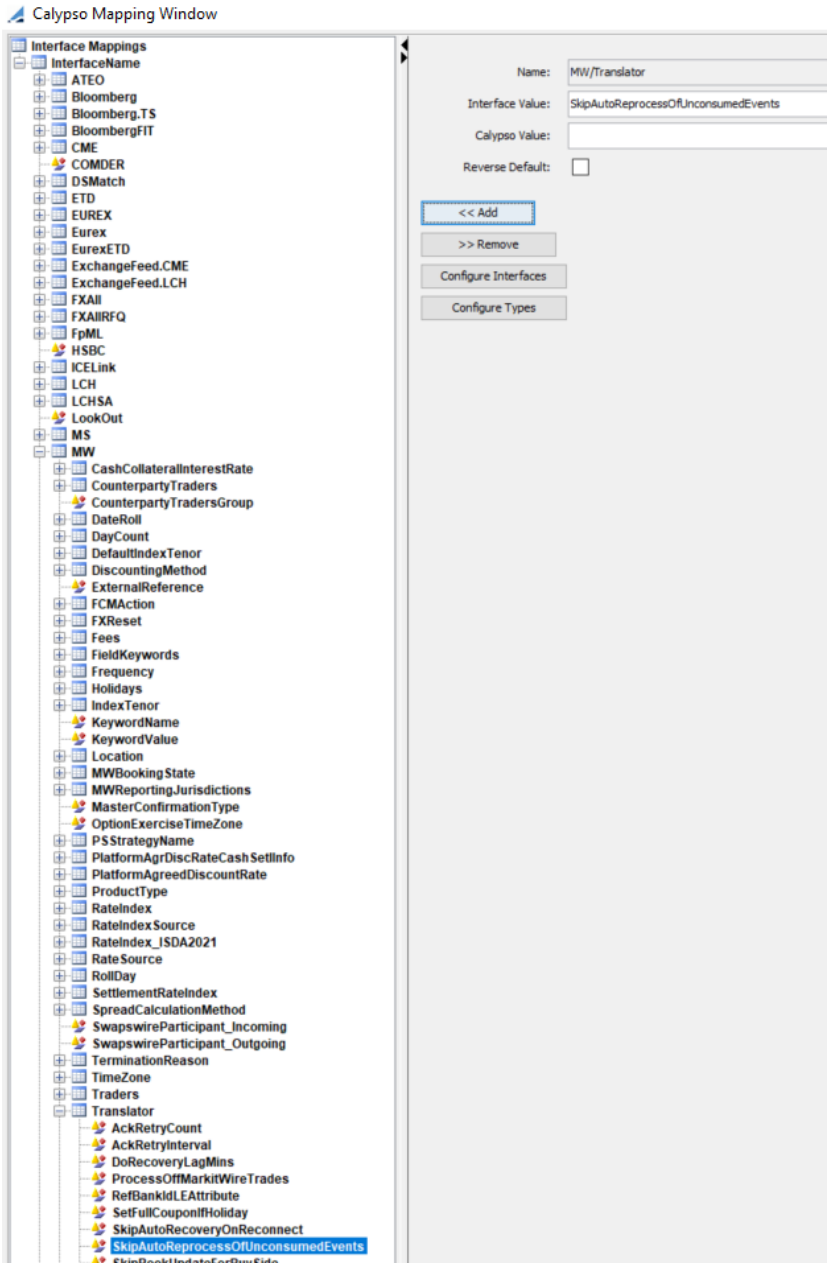


There are some cases where when MW connection goes down the engine keeps waiting for reconnection. If the trade clearing is going on during this duration and as MW connection is down the ACK could not get sent and the events get marked as unconsumed. So, when the connection comes back from MW then then engine reconnects and is able to process new trades/events. However, the existing unconsumed events is not processed automatically.

To overcome this, Calypso has workaround that when the connection is back again and engine is connected to MW. It creates and publishes event PSEventSwapswire of type - PROCESS_UNCONSUMED_EVENTS and does Dorecovery runs to fetch any new activities in MW. The SwapswireTradeEngine receives the event PSeventSwapswire.PROCESS_UNCONSUMED_EVENTS and as part of its processing queries the PSEvent table for pending events of type PSEventTrade and PSEventMessage and PSEventSwapswire (type - PROCESS_UNCONSUMED_EVENTS). Once it finds events pending to be processed, it will call the event processing which we have currently for such events. After these pending events are processed, this current event will be marked as processed.

NOTE: This feature will only be applicable if the EXCHANGE_CLEARING=true in Calypso_SW_config.properties.

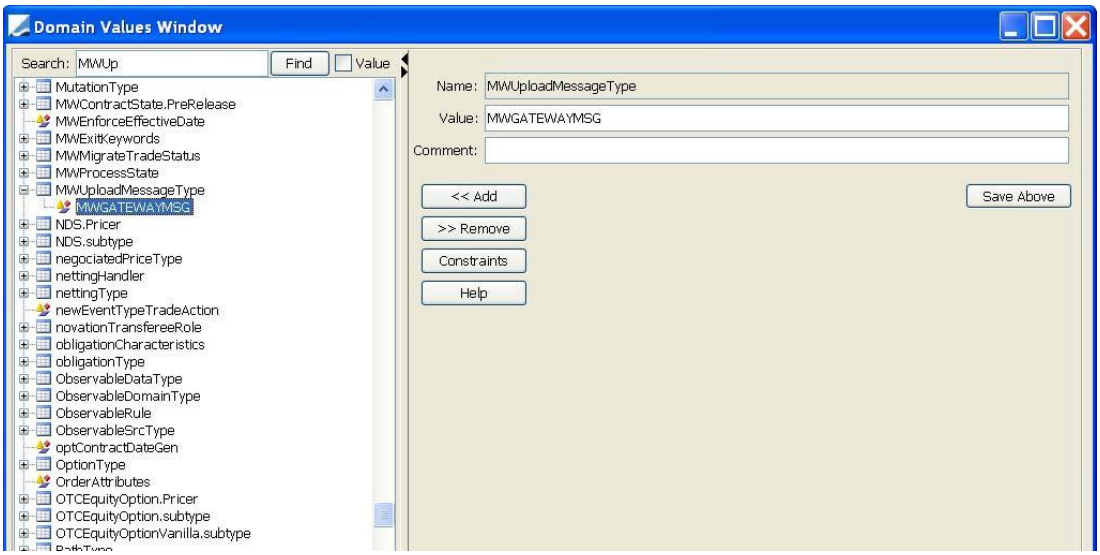
The mapping window entry "SkipAutoReprocessOfUnconsumedEvents" is not set to true:


Calypso Mapping Window
Interface Mappings
InterfaceName
ATEO
Bloomberg
Bloomberg.TS
BloombergFIT
CME
COMDER
DSMatch
ETD
EUREX
Eurex
EurexETD
ExchangeFeed.CME
ExchangeFeed.LCH
FXAll
FXAllRFQ
FpML
HSBC
ICELink
LCH
LCHSA
LookOut
MS
MW
CashCollateralInterestRate
CounterpartyTraders
CounterpartyTradersGroup
DateRoll
DayCount
DefaultIndexTenor
DiscountingMethod
ExternalReference
FCMAction
FXReset
Fees
FieldKeywords
Frequency
Holidays
IndexTenor
KeywordName
KeywordValue
Location
MWBookingState
MWReportingJurisdictions
MasterConfirmationType
OptionExerciseTimeZone
PSStrategyName
PlatformAgrDiscRateCashSetlInfo
PlatformAgreedDiscountRate
ProductType
RateIndex
RateIndexSource
RateIndex_ISDA2021
RateSource
RollDay
SettlementRateIndex
SpreadCalculationMethod
SwapswireParticipant_Incoming
SwapswireParticipant_Outgoing
TerminationReason
TimeZone
Traders
Translator
AckRetryCount
AckRetryInterval
DoRecoveryLagMins
ProcessOffMarkitWireTrades
RefBankIdLEAttribute
SetFullCouponIfHoliday
SkipAutoRecoveryOnReconnect
SkipAutoReprocessOfUnconsumedEvents
Name:
MW/Translator
Interface Value:
SkipAutoReprocessOfUnconsumedEvents
Calypso Value:
Reverse Default:
<< Add
>> Remove
Configure Interfaces
Configure Types

MarkitWire Message Workflow

MarkitWire uses the MWGATEWAYMSG message workflow. `MWGATEWAYMSG.wf` is included under `<calypso home>\client\resources`.

The message type is stored in the **MWUploadMessageTyp**e domain, which is populated when you synchronize the database with `SwapswireSchemaData.xml`. If you do not need the MWGATEWAYMSG workflow, you can simply delete MWGATEWAYMSG from the **MWUploadMessageType** domain and the application will instead use the GATEWAYMSG workflow as the default.



Gateway MarkitWire exceptions can be monitored using the EX_MWGATEWAYMSG_ERROR, EX_MWGATEWAYMSG_REJECT and EX_MWGATEWAYMSG_WARNING.

To use MWGATEWAYMSG, you must first add it to a report in the Task Station. Select all messages ending with "_MWGATEWAYMSG" and create a new tab.

Note that the message type is based on the MWUploadMessageType domain. By default, MWUploadMessageType is populated when you synchronize `SwapswireSchemaData.xml`. If you don't intend to use this workflow, you can remove the MWUploadMessageType domain and the application will instead use the default GATEWAYMSG workflow.

Error Recovery

15.1 Trade Recovery

When MarkitWire trades are booked and the Calypso Swapswire Trade Engine is not running, those trades are not automatically captured when the engine restarts.

To capture trade events from the missing period, you can use the DoRecovery process. It pulls in all activity since the date provided in the `calypso_SW_config.properties` that has not yet been applied in Calypso. If you are using the Bidirectional Mode and must also process non–picked up trades in the FCM role, then you must set SWAPSWIRE_BIDIR_MODE=true in the ENV file before running the DoRecovery Scheduled Task.

We have added the following mapping window entries under Name = MW/Translator with respective value as interface value:

SkipEventsOnDisconnect: If true, event creation is skipped when there is an empty/null SWML due to connection issue. The log messages have been improved.

SkipAutoRecoveryOnReconnect: Currently when there is a disconnect and reconnect the DoRecovery is performed automatically. If true, DoRecovery is skipped on reconnect.

DoRecoveryLagMins: Number of minutes prior to the last notification time to fetch the deals for DoRecovery.

NOTE: The mapping will be visible under FpML/Translator by default, and it would need to be manually added in MW else it will be applicable for all Interfaces which are using the FpML format like LCH/CME/Eurex.

Step 1 - Edit `calypso_SW_config.properties`. The pertinent lines are highlighted in yellow below.

Note: doRecoveryStartDate and doRecoveryEndDate are trade activity dates. Neither date is related to a specific start or end Trade Date.

Modify **doRecoveryStartDate** to the date from which you must recover trades. If you only need **Today's** trade, then you can leave **doRecoveryStartDate** empty. The date format is yyyyMMDD. For example:

```
doRecoveryStartDate=20091113
```

Modify **doRecoveryEndDate** to ending date for the range of dates you wish to recover. The date format is yyyyMMDD. For example:

```
doRecoveryEndDate=20120829
```

Alternaltively, you can set the number of minutes prior to the last notification time to fetch the deals for DoRecovery using the mapping DoRecoveryLagMins.

Name = MW/Translator

Interface Value = DoRecoveryLagMins

Calypso Value = <number of minutes>

Set performDoRecovery to true:

```
performDoRecovery=true
```

When there is a disconnect and reconnect the DoRecovery is performed automatically. If you set the mapping SkipAutoRecoveryOnReconnect to true, DoRecovery is skipped on reconnect.

Name = MW/Translator

Interface Value = SkipAutoRecoveryOnReconnect

Calypso Value = true

Also, if you set the mapping SkipEventsOnDisconnect to true, event creation is skipped when there is an empty/null SWML due to connection issue.

Name = MW/Translator

Interface Value = SkipEventsOnDisconnect

Calypso Value = true

```
# Map Markitwire products to Calypso Identifers.

# DO NOT MODIFY.

IRS=Swap FRA=FRA

Swaption=Swaption CapFloor=CapFloor Basis\u0020Swap=BasisSwap OIS=OIS

ZC\u0020Inflation\u0020Swap=ZCInflationSwap Cross\u0020Currency\u0020Basis\u0020Swap=Xccy
Cross\u0020Currency\u0020IRS=Xccy


# These are used to load the appropriate translators in Calypso

# DO NOT MODIFY

ZCInflationSwapParser=ZCInflationSwap XccyParser=Xccy

FRAParser=FRA

SwaptionParser=Swaption CapFloorParser=CapFloor SwapParser=Swap BasisSwapParser=BasisSwap
OISParser=OIS
```

```
# Indicate the version of SWML used for every product
# DO NOT MODIFY FRAVer=4.2
CapFloorVer=4.2 SwaptionVer=4.2 ZCInflationSwapVer=4.2 SwapVer=4.2 ClearingSwapVer=4.2 XccyVer=4.2
BasisSwapVer=4.2 ClearingBasisSwapVer=4.2 OISVer=4.2
ClearingOISVer=4.2

# doRecovery is used to fetch trades released from Markitwire but not yet imported in Calypso
# Only perfrom the doRecovery if the flag below is set to true
performDoRecovery=true

# Do Recovery Start Date (Must be in yyyyMMDD format for actual date or xD/xW/xM
# format for relative date). This requests trades from Markitwire from this date.
# For normal operations, it must be kept blank.
# You can specify the relative dates in below format.
# If you specify 2d deals from two business days back will be recovered
# If you specify 2w, deals from 2 week will be recovered. If it is non business
# day deals from it's prior business day will be recovered.
#doRecoveryStartDate=20091113
#doRecoveryStartDate=2D
#doRecoveryStartDate=1W

# Do Recovery End Date should be in yyyyMMDD format. If End Date is not specified it
# will consider as current date.
# End Date should be after Start Date.
#doRecoveryEndDate=20120829
#doRecoveryEndDate=1W
#doRecoveryEndDate=1D

# This flag is used to generate XML files from the SWML Messages. Two XML files
# are generated for every SWML message. One file is the SWML message itself,
# the other is Calypso Data Upload XML (which is the Calypso representation
# of the SWML. Please refer to documentation for more details.
# The files are generated in USER_HOME\Calypso\markitwire DEBUG_MW_XML=true
```

```
# Password Encryption flag
# Set this mandatorily otherwise engine will not startup AutoEncryptPassword=false

#Timeout for MW connection
session_timeout=360

#Mark EXCHANGE_CLEARING as true for Exchange Clearing solution EXCHANGE_CLEARING =false

#In exchange clearing, this flag is used for CCP as a PO model if set to true CCP_IS_PO =false

#This flag is used to start engine in Test mode
#TEST_MODE = true
#This flag is used when user needs DF Reporting in SWML. IncludeTRReportingInfo=false
```

Step 2 - Save the file and copy it to the appropriate location.

Step 3 - Launch the Swapswire Trade Engine to begin recovery. One day is the minimum period for recovery.

Note: Caution! DoRecovery imports all data and actions that have not yet been applied to the Database.

15.2 Reprocessing Failed Trade Imports

Note: For Error Handling, please refer to the *Calypso Data Uploader Integration Guide*.

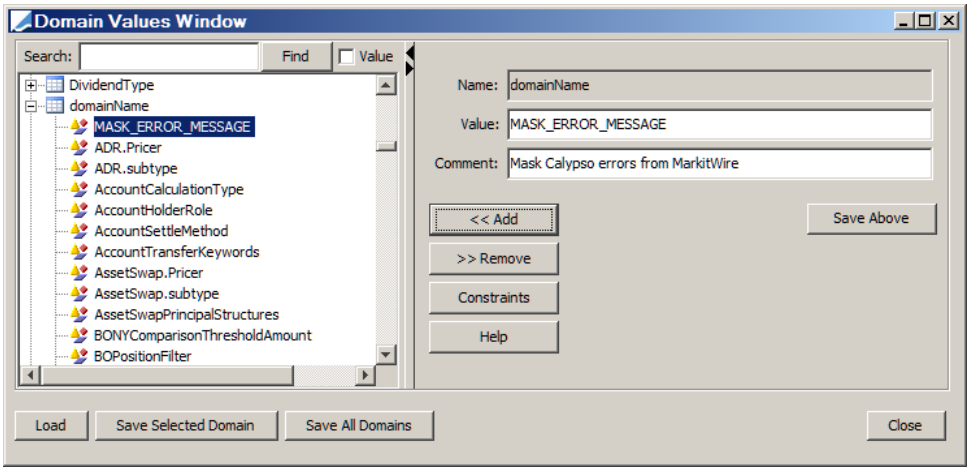
For errors related to Mapping, please correct the mapping via the Calypso Mapping Window and then apply the ReMap action in the BO Message workflow.

15.2.1 Error Masking

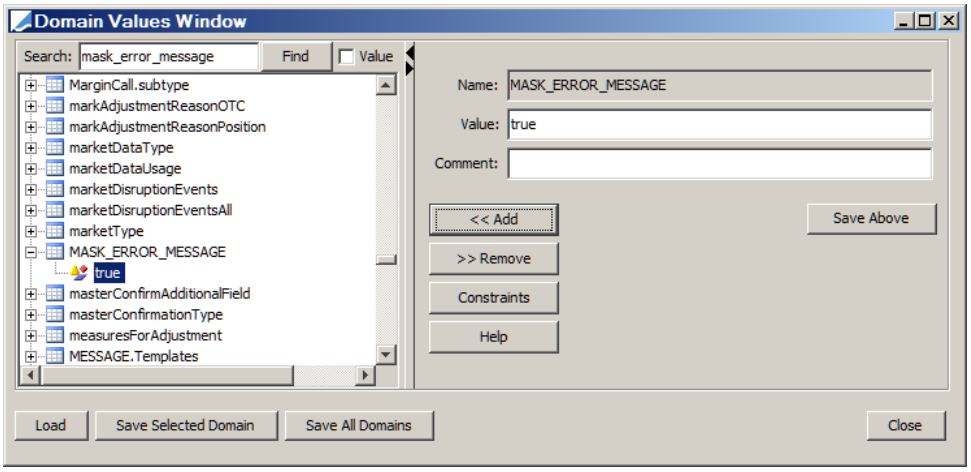
Because the actual error message created by Calypso is not relevant to MarkitWire, there is a facility to prevent such an error message from being transmitted as a Comment to MarkitWire.

To enable “error masking”:

Step 1 - Add the MASK_ERROR_MESSAGE domain using the Domain Values window.



Step 2 - Add the value true, to the MASK_ERROR_MESSAGE domain:



Step 3 - The Error Status (Swapswire Booking State) can be controlled using the Mapping Window.

After correcting the reported issue, you can reprocess the Trade Import using the SW_DO_RECOVERY Scheduled Task.

Note that this is normally not required when using the Data Uploader-based MarkitWire. The SW_DO_RECOVERY Scheduled Task is now needed only to fetch trades rejected due to BOOK Mapping. All other rejections are stored in Calypso as BOMessages and can be corrected in Calypso.

15.2.2 SW_DO_RECOVERY Scheduled Task

To use the SW_DO_RECOVERY task:

Step 1 - Set up the SW_DO_RECOVERY Scheduled Task as shown:

Task Attributes	
Trade Start Date(yyyyMMddHHmmss)	20200730100505
Trade End Date(yyyyMMddHHmmss)	20200730125800
MarkitWire Deal IDs	
Netting IDs	
Booking State	ALL
DealSink user	calyp_dealsink8

Step 2 - Ensure that the SwapswireTradeEngine is running.

Step 3 - Set **Trade Start Date**, **Trade End Date**, **Booking State**, and **DealSink** user attributes as needed.

> **Note: Trade Start Date and Trade End Date are trade activity dates and have no relation to a trade's start and end date.**

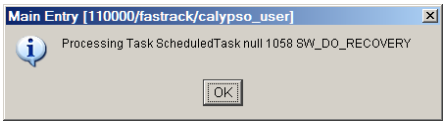
The datetimes need to be specified in the GMT timezone.

Step 4 - Add the list of Deal IDs to fetch to the **Markitwire Deal IDs** attribute, separating each ID with a comma.

Using **Markitwire Deal IDs** is not mandatory, however, if not used, then all error trades are retrieved and this may result in duplicates.

Step 5 - Set the **Exec Time**.

When the task executes, the system will display the Processing Task dialog.



When the task runs, the Swapswire Trade engine will query the MarkitWire system for trades that meet the attribute condition.

The task station will show the incoming trades as they are being retrieved from MarkitWire.

The process is identical, whether the recovery is done for a dealer/client or CCP role.

However, if you have cleared and decleared trades with a CCP role and trades have been lost in a database event, you will need to run the Recovery task to recover all the transitions and reprocess all messages sequentially.

15.2.3 Manual Trade Entry

When a trade with incorrect data is imported into Calypso, the trade is blocked (e.g., Tenor is not created in Calypso and will never be). Such a trade cannot be saved in Calypso. If the trade is subsequently amended in MarkitWire (to correct the Tenor), it flows through to Calypso as an AMEND. But since the trade has not been created in Calypso, the amend is rejected. To correct this situation, manually create the trade in Calypso with the correct Contract Version and Private Version while entering a Trade.

- ExternalReference - It should have the right format MW_POName_MarkitwireDealID, where POName is the Processing Org name and MarkitWireDealId is the ID of the MarkitWire deal that did not import correctly.
- Keyword SWPrivateVersion - This should be set to 0.
- Keyword SWContractVersion - This should be set to 1.

After manually creating the trade, Calypso can properly apply subsequent AMENDs from MarkitWire.

Note: You can perform this process while SwapswireTradeEngine is down. This allows subsequent actions from MarkitWire to flow through to Calypso.

Connecting to MarkitWire HTTPS Server

Clients have the option to connect to MarkitWire using an encrypted https connection rather than a clear-text http connection.

To make use of https:

Step 1 - Create a `sw_client_api.ini` file based on the provided sample (`calypso home/client/resources/sw_client_api.ini`), or modify the sample.

The content of ini file should be similar to the provided sample:

```
[/Transport.
host=https://uat.ia.swapswire.com
port=443
cert=UATCert.cer
Timeout=300
```

Where:

host - The complete https hostname. port - 443, the standard https port.

cert - The name of the security certificate provided by MarkitWire in the MarkitWire client package. The filename is `UATCert.cer`. If the certificate resides in a location different from your `sw_client_api.ini`, then you must also include the path to the file. For example:

```
c:\path\to\UATCert.cer.
```

Timeout - The number of seconds of idle time until the connection is terminated.

Step 2 - The SwapswireTradeEngine has two methods to locate the `sw_client_api.ini` file using the SWAPSWIRE_API_INIT_FILE environment property:

- Define the SWAPSWIRE_API_INIT_FILE environment property and leave it blank, or define it as “sw_client_api”. The SwapswireTradeEngine will then attempt to use the sw_client_api.ini in its own current working directory. Note that specifying the file extension is not necessary for this method. You can locate the current working directory of the SwapswireTradeEngine by examining its log file for the comment, “Current Working Directory:”.
- The second method is to define SWAPSWIRE_API_INIT_FILE environment property using a complete filespec (`drive:\path\to\sw_client_api.ini`). You must specify the file extension when using this method. Use this method if placing the `sw_client_api.ini` and `UATCert.cer` files in the SwapswireTradeEngine’s current working directory is not possible.

Note: If you have not defined the SWAPSWIRE_API_INIT_FILE environment property, then you must place the `sw_client_api.ini` and `UATCert.cer` files in the SwapswireTradeEngine’s current working directory.

Also note that if `UATCert.cer` file is placed directly in current working folder, you may not require .ini file.

Regression Testing

For regression testing, you can cause the Data Uploader to create XML files from SWML Messages. To enable this function, set DEBUG_MW_XML to “true” in the `calypso_SW_config.properties` file.

The application will create the XML files in `%USERPROFILE%\Calypso\markitwire` directory. Each SWML message results in two files:

- The first file is the SWML Message itself. The filename has the following format:

`SWML_<product_mapped_in_properties_file>_<sw_deal_id>_<contract_ver>.xml`

Where:

product_mapped_in_properties_file - The product as identified by mapping.

sw_deal_id - The SwapswireDealId.

contract_ver - The version of this contract. Therefore a sample filename will be similar to:

`SWML_swap42_3310303_2.xml`

- The second file is the Calypso Data Uploader XML file. This is the SWML message converted into the Upload XML. The filename has the following format:

The `MW_<ExternalKey>_action.xml`

Where:

ExternalKey - The External Key (i.e., the SwapswireDealId).

Action - The action that generated this message. Therefore a sample filename will appear similar to:

`MW_BRANCHE1_3310303_PARTIAL_NOVATE.xml`

These files should be sent to Calypso as HelpDesk attachments to simplify reproducing the issue. The files can also be used for regression testing in an offline fashion without running the Swapswire Trade engine.

To import SWML files into Calypso:

Step 1 - Clear the MWPublishers entry in `gatewayservice.properties:MWPublishers=`

> **Note: Caution! Failure to clear the MWPublishers entry will cause your implementation to send Offline Trade Notifications to MarkitWire.**

Step 2 - Copy the SWML files to a separate directory for regression testing.

Step 3 - Ensure that you have modified the `mwuploader.properties` file to the folder created above.

Step 4 - Launch the FileWatcher (refer to the Data Uploader module) using the `MWuploader.properties` file. The command will appear similar to the following:

```
java -Xmx256m -D com.calypso.apps.startup.StartFileWatcher
    -propertyFile mwuploader.properties
    -env YourEnv
    -user user_name
    -password xxxxxxx
```

Note that the above is a one-line command.

Once the SWML XML file is dropped into the regression testing directory specified by in `mwuploader.properties`, the Data Uploader reads and converts the message into a BO Message, and Calypso executes the Action.

This regression test does not test connectivity or error reporting but does speed up the lifecycle testing for all trade types supported by Calypso's MarkitWire module. Also, before reusing the SWML files, cancel (or delete) all trades to External Ref key for the regression tests.

Test Tool

18.1 Setup

The Test Tool allows you to test the event-based processing of the Swapswire Trade engine. To begin using the Test Tool, you must modify `calypso_SW_config.properties` and `mwuploader.properties`. Both files are located in `calypso home/client/resources`.

Step 1 - The TEST_MODE property controls the on/off state of the Test Tool. Add the TEST_MODE property to `calypso_SW_config.properties`:

```
TEST_MODE = true
```

Step 2 - The TestResultLocation specifies the location for Test Tool output files.

Add the TestResultLocation property to `mwuploader.properties`. Note the use of doubled slashes for the output path spec:

```
TestResultLocation =C:\\Desktop\\MWUploader_12\\Output\\
```

Step 3 - fileDir specifies the location for SWML files. fileDir is located in `mwuploader.properties`. Note the use of doubled slashes in the path spec:

```
fileDir=C:\\Desktop\\MWUploader_12\\Input\\
```

18.2 Usage

Launch the Swapswire Trade engine in “Test” mode by setting TEST_MODE to On. When the Engine is in Test mode, all the code that handles events is tested and there are no communications with MarkitWire Servers. Calls are not made to MarkitWire, nor are notifications accepted, instead, the Engine starts the File Watcher (which monitors the input folder defined in `mwuploader.properties`). When the filewatcher processes a file, it creates an event similar to that in the “live” mode so that all event handling code is tested.

At the end of event processing, the Test Tool saves an updated XML file in the output folder defined in `mwuploader.properties`.

When running the engine in Test mode, the Acknowledgment publisher should remain MW or MWClearing. Ensure that all contract states and process states are present in domains.

The filenames have the following formats:

- For Trader interface files:

  SWML_(product type)_(contract id)_(contract ver)_(private ver)_(contract state)_(login handle)_(side).xml. E.g.:

```
SWML_Swap_7643196_1_1_New_1901_1.xml
```

- For Exchange Clearing interface:

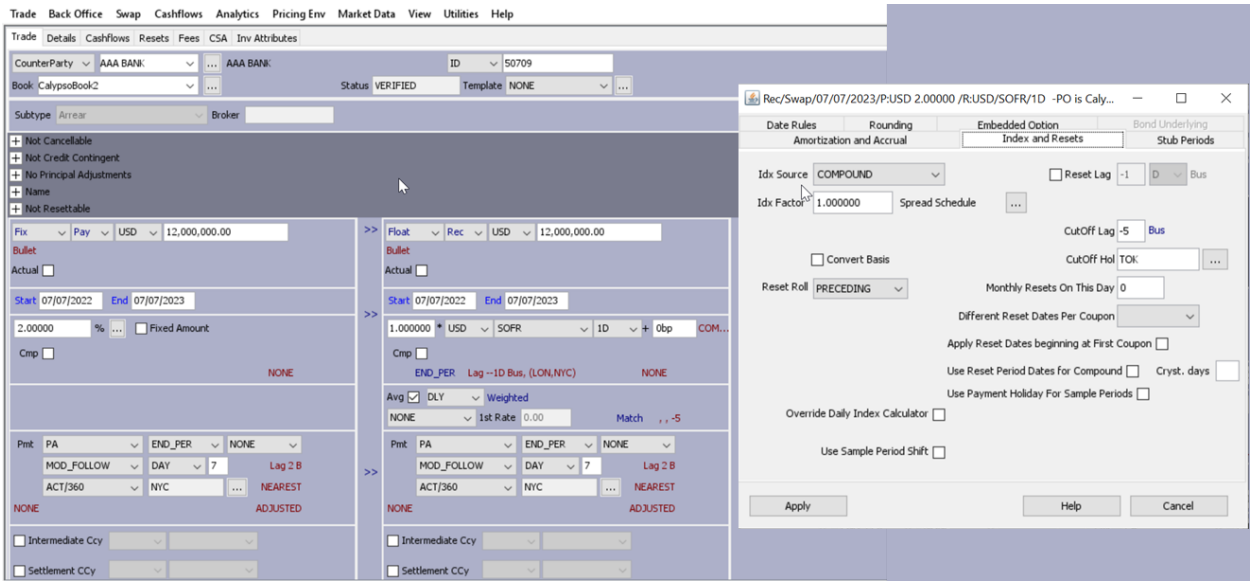
  ClearingSWML _(product type)_(contract id)_(contract ver)_(private ver)_(contract state)_(login handle)_(side).xml. E.g.:

```
ClearingSWML_Swap_7653728_2_0_Clearing_1901_4.xml
```

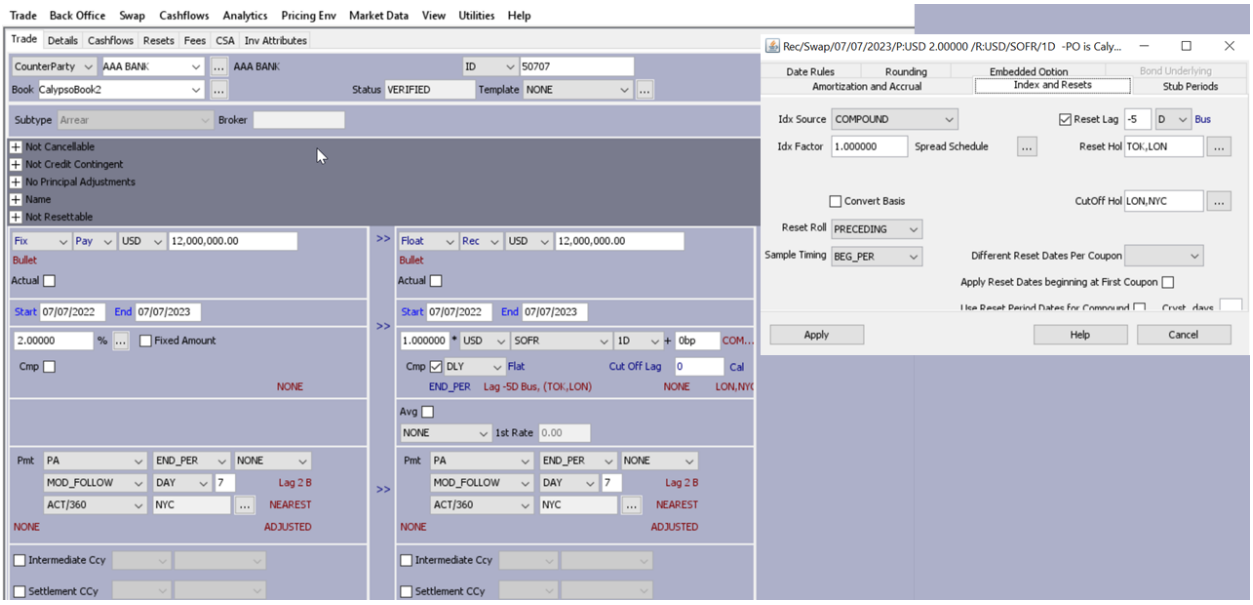
Support for Compounding and Averaging Parameters

MarkitWire will support compounding and averaging provisions on OIS and Single Currency Basis Swap as part of the MarkitWire 19.1 API release. These compounding and averaging provisions will be applicable to overnight rate FROs which ISDA has published as part of the 2021 Interest Rate Derivatives Definitions (and a subset were published in Supplement 74 to the 2006 ISDA Definitions). This support is available for Dealer and Buyside MarkitWire.

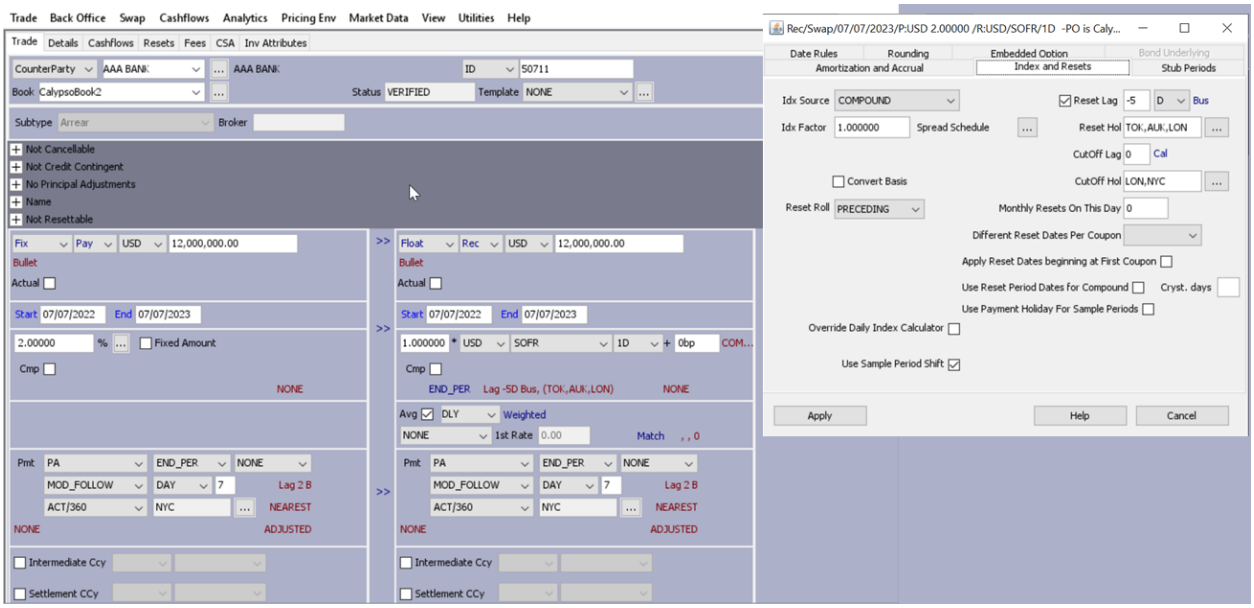
When a Trade is booked correctly in Calypso with Averaging/Compounding DLY box ticked. The CutOff Lag as is set value and Cutoff lag box with Cal/Bus label being Bus in Product Details Window. The offset holiday is set in the Product Details Window.



When a Trade is booked correctly in Calypso with Averaging/Compounding DLY box ticked in the main trade window. The reset Lag should be ticked with set value in Product Details Window. In the 'Index and Resets' tab, tick 'Reset Lag' with Cal/Bus label being Bus. The offset holiday can be set in the Product Details Window.



When a trade is booked in Calypso with Averaging/Compounding DLY box ticked in the main trade window. The reset Lag should be ticked with set value in Product Details Window. In the 'Index and Resets' tab, tick 'Reset Lag' with Cal/Bus label being Bus. Then, set the offset holiday and 'Reset Hol' box in Product Details Window. The Sample period shift flag should be ticked. In the main trade window Reset Timing Property should be set to END_PER.



MarkitWire Allocation performance Enhancement

The Allocation packages received from SEF platforms – Tradeweb and Bloomberg. These packages consist of 50+ trades wherein it includes one block trade and 50 child allocations and these allocation childs are marked for clearing so these get cleared at a CCP and we receive the clearing notifications which is translated to Novation lifecycle in Calypso for each child trade. Post clearing we receive the child trades of clearing from MarkitWire and we update the corresponding novation child trades in Calypso. For every update on the child trades in Calypso, there is an update made on the Block trade as per the Calypso Allocation design.

So, the following updates has been made:

- Reduce the delay for new event creation by setting the property – "EventCreationLag" with value in milliseconds in calypso_SW_config.properties. Default value is 1000 milliseconds.
- Reduce the delay for new clearing event (clearing child trade) by setting the property – "NewClearingEventCreationLag" with value in milliseconds in calypso_SW_config.properties. Default value is 2000 milliseconds.
- Switch to the following mode: uploadMode=Local, persistMessages=Failure in calypso_SW_config.properties.
- Set the property – DEBUG_MW_XML=false in calypso_SW_config.properties.
- Remove all other events from SwapswireTradeEngine event list from engine server web admin page other than – PSEventSwapswire.
- Remove unwanted process states from the domain "MWProcessState" this will reduce time for processing these extra notifications.
- Not having allocation performed in Calypso and saving allocation child trades as individual trades in Calypso.

The last point from the above update is explained as follows:

Handle not saving the Block trade in Calypso

We have to configure Domain values:

- BooksToIgnoreAllocationBlock: To have list of MW Book BIC codes for which we want to apply the ignore of block save. This would be taken from the tag <swTradingBookId>
- VenueTypesToIgnoreAllocationBlock: To have list of Venue types for which we should apply the ignore of block save for ex. SEF. This would be taken from tag <swExecutionVenueType>.

If above are setup, then in MarkitWire translator we would check if the contract state is either NEW or ALLOCATED and if the incoming SWML message has the "SWAllocations" element then we ignore all those message processing.

This will handle to ignore save of the Block trade.

Handling child trades

For child allocation trades we would need to apply the lifecycle NEW instead of REKEY which we currently perform and to achieve that we already support by setting the domain value - "SkipAllocatedParentExistsValidation". This domain should be set to true.

Handling Sequence policy for child trades

We handle the SwapswireTradeEngineSequencePolicy for child trades, that even if there is a keyword <swBlockDealId> we should not use that for sequencing. For this there is a code change done to identify the scenario where the Block trade is ignored which is checking the BIC code of the incoming book against the domain and also the venue name against the domain.

Dorecovery handling

As part of the dorecovery process, it should be taken care that the block is not saved and child allocations are saved as NEW trades.

Support for UPI code from MarkitWire

UPI (Unique Product Identifier) is a code that uniquely identifies the product that is the subject of the OTC derivatives transaction. A UPI will be assigned to each product, and regulators would be able to aggregate OTC derivatives transactions by product (using the UPI Code) or by individual reference data elements that comprise the product (such as the underlier). The UPI will work in conjunction with unique transaction identifiers (UTIs) and critical data elements (CDEs), which are also expected to be reportable to global regulatory authorities.

Calypso supports following keywords for:

- InstrumentUPI (Incoming – New trades)

  This keyword will come from the Alpha trade which will be received by the CCP (ASX) when MarkitWire sends the trade for Clearing.

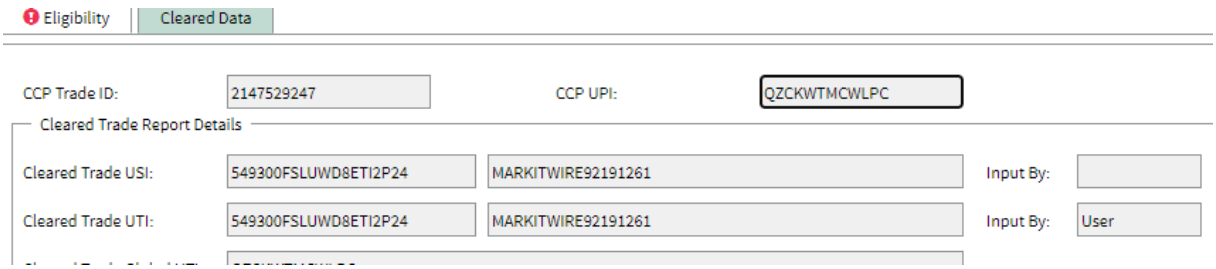
- InstrumentUPI (Outgoing - Netting New trades)

  The value available in this keyword will be sent to MW as part of the Netting Instruction XML message which is sent when Netting is alleged.

- ClearedTradeUPI (Outgoing – Clearing Accept ACKs)

  As part of the CLEAR accept the CCP would need to add the value in the above keyword. This will be sent to MarkitWire as part of the Clearing acknowledgement.

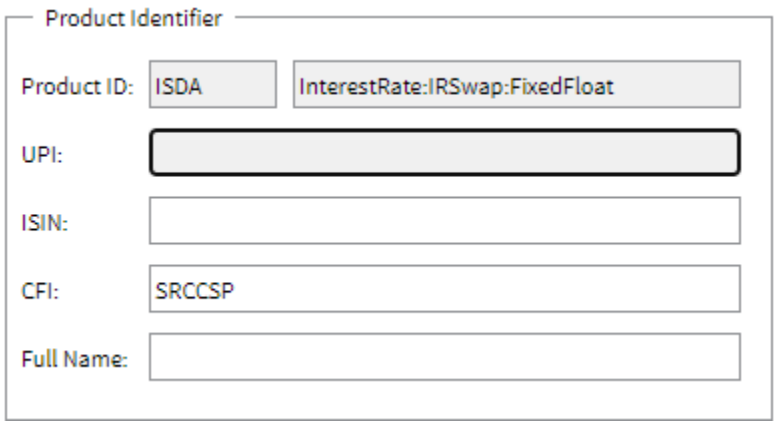
MarkitWire will display the value sent as part of the acknowledgement on their UI in Clearing tab.



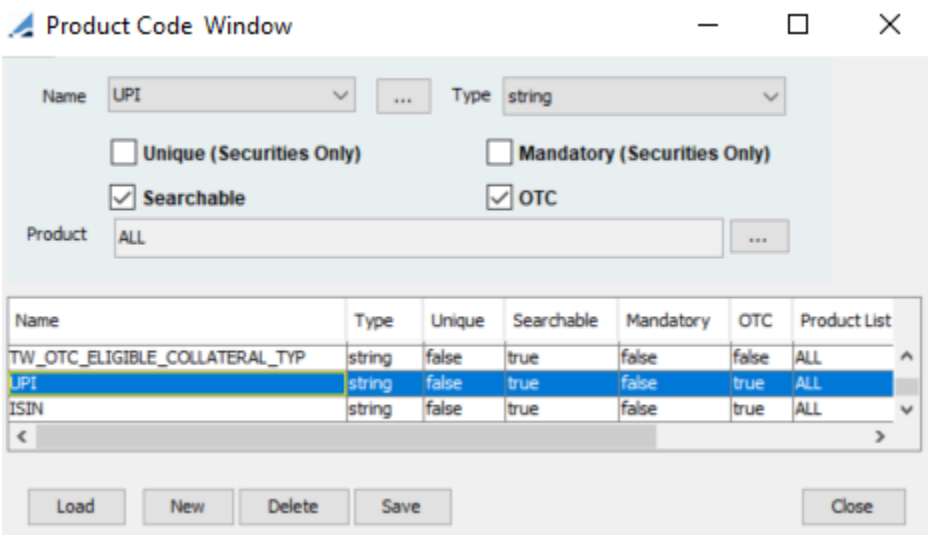
Sample ACK message:

<ClearingHouseProductID productIdScheme="http://www.fpml.org/coding-scheme/external/iso4914">QZCKWTMCWLPC</ClearingHouseProductID>

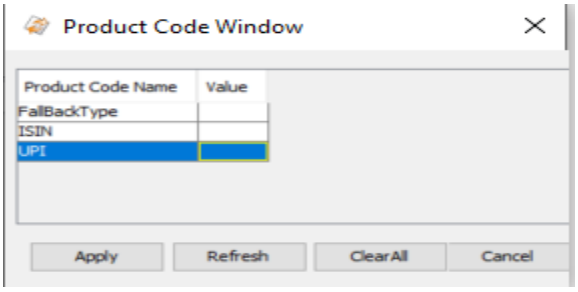
To define MarkitWire Product Code entry in the Calypso:



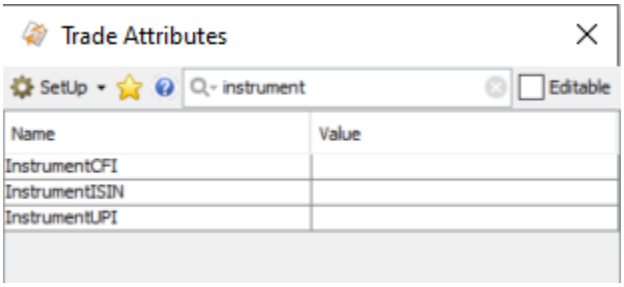
Configuring the Product code with support for OTC checkbox:



The following is the Product code window with the details tab on Trade window:



The alternate option: Trade keywords



NOTE: Calypso added support to populate product code UNDERLYINGSWAP_UPI if the keyword InstrumentUnderlyingSwapUPI is populated on the trade. Along with it, Calypso also provides support for ClearedTradeUPI in CCP mode as well as in dealer mode, where this keyword would get set post clearing. Also in FCM mode if the UPI comes in the we will store it in keyword InstrumentUPI.

Complex Trade UPIs

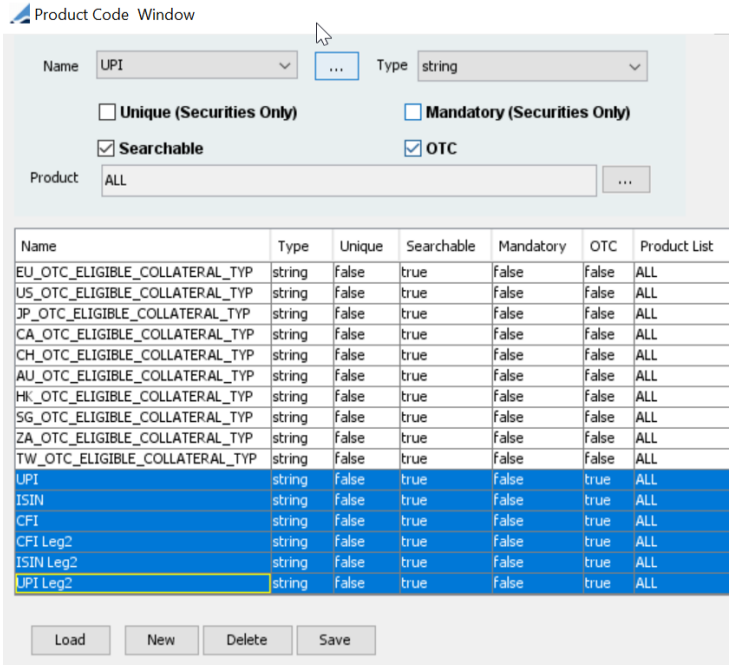
MarkitWire has added support for Leg 1 and Leg 2 specific UPI/CFI/ISIN/FullName product codes for Swaption and Cap Floor Straddle products in Incoming mode and outgoing bidirectional mode.

Product codes:

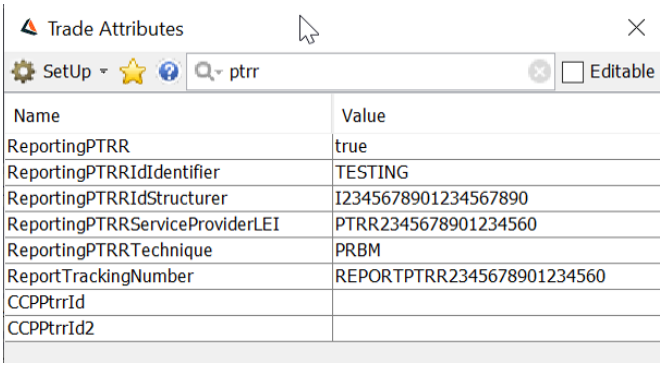
UPI / UPI Leg2

CFI / CFI Leg2

ISIN / ISIN Leg2



Calypso supports Report Tracking Number on submissions along with Additional PTRR Data on submissions including via API (Private Data), PTRR ID, PTRR, PTRR Technique, and PTRR Service Provider via Trade Attribute window.



21.1 Keywords List

No	Calypso Keyword	Datatype / Value
1	InstrumentUPI	String (Alphanumeric). E.g. QZCKWTMCWLBP
2	ClearedTradeUPI	Same as above

21.2 InstrumentUPI value requirement from MW documentation

UPI can take 12 alphanumeric characters and MarkitWire will validate the ISO 4914 format as follows:

- First two characters (prefix) must be 'QZ'.
- Nine characters after the prefix must be alphanumeric (upper case A-Z and 0-9 excluding the vowels A, E, I, O, U, and the character 'Y').
- Last character must be alphanumeric check character.