Trade Validation Engine

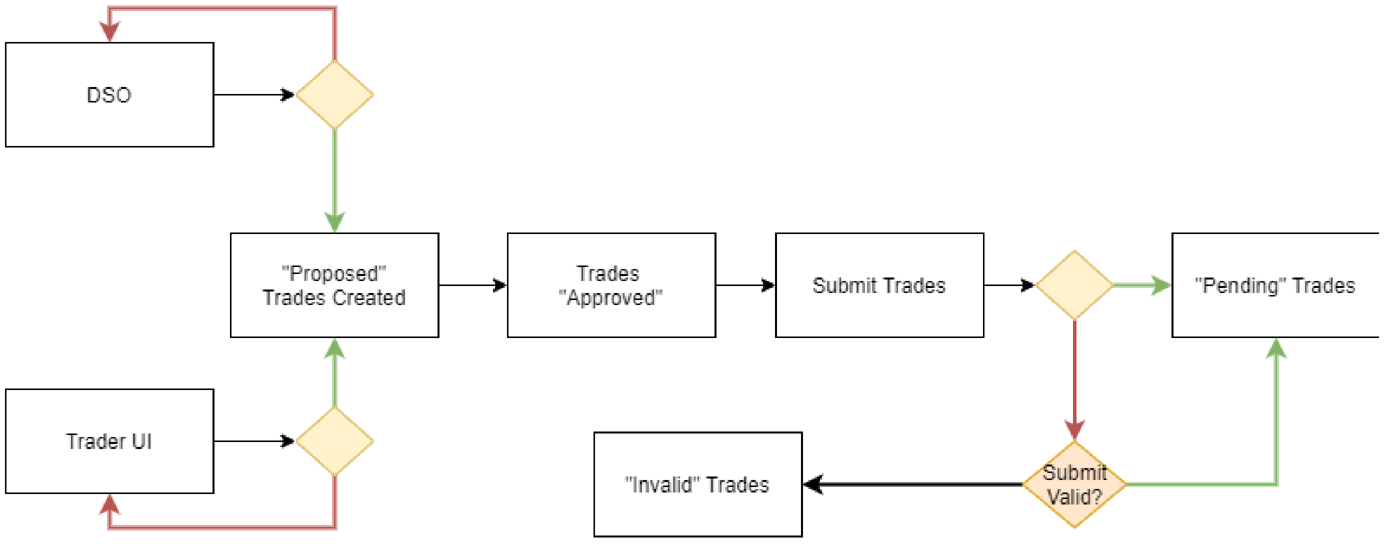
The trade validation engine aims to use data-driven methodology to conduct checks on trades that pass through PDC. This includes trades generated by DSO, as well as those that are edited and approved by traders. The purpose of a trade validation engine is to ensure that all trades that are ultimately submitted to the market meet the requirements of the market and take into account any additional guardrails such as those that are put in place by the customer organization; if trades violate the boundaries set for them they should be flagged for manual intervention.

This page aims to provide documentation on how to implement a Trade Validation Engine microservice that calls upon configuration files within the database that can be tailored by venue, market, products, and take into account customer preferences.

Note: Individual trades obviously don't exist in a vacuum; depending on the asset that is being traded, trades have interdependencies and trade checks should acknowledge them. For example, a battery asset can only discharge a certain amount of energy across a period of time before it runs out of energy and our trade validation engine should be sophisticated enough to recognize that at some point. For the time being, however, we are treating each trade as independent, and will address interdependencies in v2 of this implementation

Trade Validation Milestones

There are several points throughout the trade creation, review and submission process that may call the Trade Validation Engine:



Step	Timing	Validation Check Trigger	Behavior when trades pass validation	Behavior when trades do not pass validation
DSO-initiated trade creation	When trades are generated by DSO and written into the ne_tradetable during the DSP step OR When trades are generated by DSO and before being written into ne_trade table (in some sort of a post-processing step)	DSP publishes Kafka message informing that trades have been written into table; prompts microservice to run latest trade runs against the appropriate trade rules configurations OR As an embedded step within Kubeflow trade generation pipeline	Nothing; trades remain as-written by DSO in ne_trade table OR Trades get published via DSP into ne_trade table	❓ Error message is published in Kafka message that displays error within TraderUI and somehow notifies DSO OR Error message is flagged in Kubeflow logs associated with the post-processing step
Trader-initiated trade edit	When trades that have been generated by DSO are manually edited by traders in TraderUI and moved from status: Proposed status: Approved	Traders click on "Submit" in the trade submission pane	Trade edits are processed and trade record updated (trade status still Proposed until traders mark them as Approved)	Error message in trade submission pane that notifies traders with details of violation and prevents them from proceeding until it is corrected (similar to existing behavior)

Trader-initiated trade creation	When trades are newly created by traders in TraderUI and moved from status: Proposed status: Approved	Traders click on “Submit” in the trade submission pane	New trades are processed and created (trade status moved to Proposed until traders mark them as Approved)	Error message in trade submission pane that notifies traders with details of violation and prevents them from proceeding until it is corrected (similar to existing behavior)
Market timing-driven trade submission	When trades are submitted to the market or market conduit (i.e. PCI) before market gate closure	Scheduled batch job that initiates trade submission to the market	Trades are submitted to market conduit	If a single trade violates config rules, roll back all trades and flag the specific trade that is in question in TraderUI and allow traders to edit /resubmit OR If a single trade violates config rules, roll back only the erroneous trade and submit the rest of the trades to the market. Flag trade that is in question in TraderUI and allow traders to edit /resubmit

Add/Update Trade GQL Mutation

Core validation is to create mutation to track attributes; there are use various use cases that yields different behaviors.

Two methods:

1. Add trade and if failed don’t write it in
2. Add trade and if failed write it in and mark as invalid (pass in a flag indicating whether you want trade to be created or not)

Steps:

- Create the GQL mutation that takes every trade field and validate them against the rules
- Build a wrapper around GQL mutation - add trade or update trade

Trade Rules

Trade rules should take into account the following (non-exhaustive list). The parameters would be stored in the relevant tables, asset rules in the asset table, market rules in the market table, etc.

Area	Rule	Notes
Asset Based Rules - would link to asset table to find relevant parameters for the check. These are mainly focused on the physical capabilities of an asset.		
Asset Capacity - fixed capacity	Submission Quantity <= Max Capacity	Complications: It may need to take into account previously cleared submissions - this should be available via the COP. There may need to be different figures for different products (eg, energy v reserve limits).
Asset Capacity - fixed capacity	Submission Quantity >= Min Capacity	Complications: It may need to take into account previously cleared submissions - this should be available via the COP. There may need to be different figures for different products (eg, energy v reserve limits).

Asset Capacity - variable capacity	Submission Quantity Hour h <= Max Capacity Hour h	Complication: as above it may need to take into account previous submissions. Each hour would have a different capacity value. There may need to be different figures for different products (eg, energy v reserve limits).
Asset Ramp	Registered Ramp Rate Min <= Submission Ramp Rate <= Registered Ramp Rate Max	
Asset Price	Registered Asset Min Price <= Submission Price <= Registered Asset Min Price	
Asset Product	Asset is registered for the market/product	
Market based rules - would link to market, venue and product tables to determine the relevant parameters. These are mainly the timing, price and volume limits		
Submission timing - Market Opening	Trade DateTime >= Market Opening	Submission can't be submitted prior to opening. Could be created before opening but not submitted.
Submission timing - Market Gate Closure	Trade DateTime <= Market Opening	Submission can't be submitted after closure
Market Price Limits	Market Price Min <= Submission Price <= Market Price Max	Strike price and price pair prices must be within market product limits.
Market Price Structure Ascending	Price pairs must be ascending	
Market Price Structure - First Price	First price must be market floor price	
Market Price Structure - Last Price	Last price must be market cap	
Full capacity offered	Full registered capacity must be offered	
Smallest increment volume	Price pair volume must be at least x (eg, 0.01 MWh) increment	
Smallest price increment	Price pair volume must be at least x (eg, 0.01 $/MWh) increment	
Sum Price Pairs	Sum Price Pairs (blocks) must equal submitted volume	
Company based rules - company specific rules across the market/assets for that company. Company rules could mimic those above but with different parameters to further restrict the market limits.		
Company Price Limits	Company Price Min <= Submission Price <= Company Price Max	
Economic Price Limit	Buy price < Sell price	Complication: It needs to find if any other submissions match then check against them. If there is a buy and sell in the same interval /asset/market/product the buy price must be less than the sell price.
Cycle max	Number of cycles <= Max	

Other aspects:

- Hard stopping v warning/confirm messages
    - Flag as part of where you map rules as to whether they are stopper or not.
    - If a hard rule then cannot be submitted, if a soft rule then a warning will be issued that user can override to submit.

Config Files

Depending on which venue, market, product is being traded, specific trade rules should be applied and validated against. The following are examples of requirements that should be met:

- If the Day Ahead market is selected, bids/offers should only apply to the hours of the Day Ahead market
- If the venue selected is CAISO and the product is an ancillary service, the offer should only contain a price, not contain a quantity

The checks should be validated against what currently exists in the database (i.e. ep_asset.ne_asset_parameter_values, other market-based tables). In addition to that there’s value in creating a JSON file that holds abstractions / exceptions to what exists in the DB.

venue	market	product	trade_config
CAISO	DAY_AHEAD	REGUP	`{ {“company_constraints”: "max_capacity": 90% of rated capacity } }`
ERCOT	REAL_TIME	ENERGY	`{ {“company_constraints”: "max_capacity": 90% of rated capacity } }`