The trading process instance must reference the

Multi-Time-Frame

process definition of the Jason trading bot in the Masters data mine.

Backtesting Session

Foundations->Node->Backtesting Session->Definition

A backtesting session is a trading mode by which the trading bot instance reads historic market data in a user-defined datetime range, applies the rules defined in the associated trading system, and generates a trading simulation.

Foundations->Node->Backtesting Session->Content

A backtesting session node must reference a trading system to gain access to the trading logic to be applied during the session. Other considerations framing the session come from the set of parameters attached to it.

Paper Trading Session

Foundations->Node->Paper Trading Session->Definition

A paper trading session is a trading mode by which the trading bot instance reads a live market data feed, applies the rules defined in the associated trading system, and generates a trading simulation.

Foundations->Node->Paper Trading Session->Content

A paper trading session node must reference a trading system to gain access to the trading logic to be applied during the session. Other considerations framing the session come from the set of parameters attached to it.

Forward Testing Session

Foundations->Node->Forward Testing Session->Definition

A forward testing session is a trading mode by which the trading bot instance performs live trading with a user-defined fraction of the available capital.

Foundations->Node->Forward Testing Session->Content

A forward testing session node must reference a trading system to gain access to the trading logic to be applied during the session. Other considerations framing the session come from the set of parameters attached to it.

Foundations->Concept->Reusable Snippets->Important for Live Sessions

Important: Running a live session requires the setup of a key reference at the

Task

. It also requires a live data feed, meaning that the corresponding

Sensor Bot Instance

, along with all indicators used by the referenced

Trading System

, must be up and running. Finally, a live session also requires at least 48 hours of historic market data.

Bear in mind that the

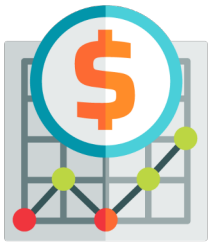
Trading System

may require even more historic market data to properly analyze the market depending on the

indicators used.

Live Trading Session

Foundations->Node->Live Trading Session->Definition



A live trading session is a trading mode by which the trading bot instance reads a live market data feed, applies the rules as defined in the associated trading system, places the corresponding orders at the associated exchange, and stores the defined data products.

Foundations->Node->Live Trading Session->Content

A live trading session node must reference a trading system to gain access to the trading logic to be applied during the session. Other considerations framing the session come from the set of parameters attached to it.

Foundations->Concept->Reusable Snippets->Important for Live Sessions

Important: Running a live session requires the setup of a key reference at the

Task

. It also requires a live data feed, meaning that the corresponding

Sensor Bot Instance

, along with all indicators used by the referenced

Trading System

, must be up and running. Finally, a live session also requires at least 48 hours of historic market data.

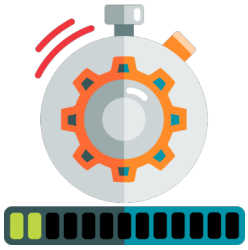
Bear in mind that the

Trading System

may require even more historic market data to properly analyze the market depending on the indicators used.

Execution Started Event

Foundations->Node->Execution Started Event->Definition



The execution started event is the event that triggers the execution of a process. It usually references the execution finished event of another process on which the process depends on.

Foundations->Node->Execution Started Event->Content

These references determine when a process is due for another run. By listening to the execution finished event of the process it depends on, it may wake up just in time to process the new batch of data the dependency has just delivered.

Bots form a sort of multi-branched execution sequence with an indeterminate number of dependencies. Every time the bot further down the tree of dependencies finishes a cycle, it triggers the execution of multiple bots listening to its execution finished event.

In the context of a trading process instance running a trading session on the network hierarchy, the execution started event may be used to force the trading process to run only after the last indicator bot dependency finishes its job. This guarantees that all dependencies are up to date and that the trading bot will evaluate the information corresponding to the same candles for all indicators used by the trading system.

Not setting up this event on a trading session may result in eventual data inconsistencies, as—in theory—the trading bot may run with some indicators up to date and some slightly delayed.

Trading Session References

Summary: A

Trading Session

must reference the

Trading System

whose rules are to be implemented during the session, and the

Trading Engine

hierarchy that shall keep track of the runtime data produced by the

Trading Bot

.

Trading System Reference

Foundations->Node->Trading System Reference->Definition

A trading engine reference determines which trading system shall be evaluated by the trading bot to run the trading session.

Foundations->Node->Trading System Reference->Content

In other words, the trading system reference lets the process instance know which trading rules to process for the given session.

Trading Engine Reference

Foundations->Node->Trading Engine Reference->Definition

A trading engine reference determines which trading engine the trading bot shall use to structure the data it processes while running the trading session.

Foundations->Node->Trading Engine Reference->Content

In other words, the trading engine reference lets the process instance know which data structure to use to store information related to the trading session. This means that a workspace may have more than one trading engine.

Trading Session Parameters

Summary: Parameters control the behavior of trading sessions and improve the quality of simulations.

Foundations->Node->Trading Parameters->Definition

Parameters are properties of trading sessions, defined by users, to determine their behavior and improve the quality of simulations.

Foundations->Node->Trading Parameters->Content

The behavior of parameters may vary depending on the type of session.

Each testing session has its own set of parameters. This allows you to configure different trading sessions with different parameters, and go back and forth between them as required. For instance, you may have different backtesting sessions with different date ranges, different exchange fees or different slippage settings to account for different possible scenarios.

Foundations->Concept->Reusable Snippets->Note for Hierarchy Tables

Note: This page discusses the top-level information about the section of the hierarchy represented in the below table. Click on any of the nodes to get the details, including particulars on their configuration, when applicable.

Trading Parameters

Session Base Asset

Session Quoted Asset

Time Range

Time Frame

Slippage

Fee Structure

Snapshots

Heartbeats

User Defined Parameters

Session Base Asset

Foundations->Node->Session Base Asset->Definition

The base asset is the asset whose price is determined by the market. It is usually the first asset in the pair, as listed by the exchange.

Foundations->Node->Session Base Asset->Content

Among other things, the parameter allows defining an initial balance of the corresponding asset, which may be used for trading with the corresponding trading system and trading session. Please see the configuration.

Session Quoted Asset

Foundations->Node->Session Quoted Asset->Definition

The quoted asset is the asset on which the price of the base asset is denominated in the market. It is usually the second asset in the pair, as listed by the exchange.

Foundations->Node->Session Quoted Asset->Content

The parameter allows defining an initial balance for the corresponding asset, which may be used for trading with the corresponding trading system and trading session.

> **Note:** Notice that the initial balance may be defined on either or both assets in the market.

Time Range

Foundations->Node->Time Range->Definition

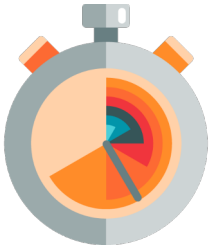
The time range parameter determines the period of time on which the trading session is run.

Foundations->Node->Time Range->Content

The parameter offers precise control over the duration, starting and ending points of the session. Several options are available, and there are differences in how backtesting and the rest of the types of trading sessions function in this regard.

Time Frame

Foundations->Node->Time Frame->Definition



The time frame determines the collection of candles to be analyzed during a backtesting session, and the frequency with which the trading bot runs on paper trading, forward testing, and live trading sessions.

Foundations->Node->Time Frame->Content

In Live Sessions

In the context of live sessions, that is, paper trading, forward testing, and live trading, you may want to run the session on the 01-min time frame so that the trading bot reacts fast when the price tags the take profit or stop loss targets.

> **Important:** Remember that Superalgos does not place stop or take profit orders at the time of entering the position. Instead, the trading bot checks the current price upon each execution Cycle

and determines whether

Managed Stop Loss

or

Managed Take Profit

targets have been hit or not. If targets are hit, the

Close Stage

opens and orders are placed on the next cycle, which happens to be on the next candle.

If for whatever reason you don't need to minimize the potential for slippage when hitting stop or take profit targets, you may choose whatever time frame you like, taking into account the explanations below.

In Backtesting Sessions

In the context of backtesting sessions, what time frame you decide to run the session depends on the trading system being tested. If the trading system makes decisions based on the 1-hour candle and above, then 01-hs may be the best choice. However, if decisions are influenced by sub-hour candles then you should match the time frame accordingly.

In other words, in backtesting sessions, you should match the time frame to the smallest period on which the trading system makes decisions.

Important: The above, however, is only partially correct. Setting the time frame to match the

decision-making criteria of the

Trading System

is desirable to save time in backtests. You save time because the smallest reasonable collection of

candles is evaluated. However, the time-saving may come at the cost of precision!

Remember what was said about stop loss and take profit orders earlier: they do not exist in Superalgos. When a target is hit, orders are placed in the following

Cycle

, which happens to be on the next candle!

Let's take a minute to reflect on the implications of the above! If targets are hit on one candle

---

and orders are placed on the next candle, there is the potential for huge

---

Slippage

---

if the session is run on higher time frames! This is the cost in precision mentioned earlier.

The above explanation seems to point in the following direction: if you want backtesting simulations to be comparable to live sessions run on the 01-min time frame, then you should run the backtesting session on the 01-min time frame as well. There is, however, a workaround.

Workaround

You may use the

Simulated Exchange Events

structure of nodes to arbitrarily define the exit rate of the position.

For example, you may use the following formula in the

Simulated Actual Rate

node as well as in the

Target Rate

of the

Close Stage

:

```
targetRate()

function targetRate() {
    switch (tradingEngine.tradingCurrent.position.exitType.value) {
```

```
            case 'No Exit': {
                return tradingEngine.tradingCurrent.tradingEpisode.candle.close.value
                break
            }
            case 'Take Profit': {
                return
tradingEngine.tradingCurrent.position.takeProfit.finalValue.value
                break
            }
            case 'Stop Loss': {
                return tradingEngine.tradingCurrent.position.stopLoss.finalValue.value
                break
            }
        }
}
```

The formula sets the

Simulated Actual Rate

to be the last known takeProfit rate if the exit was triggered by hitting of the

Managed Take Profit

, or the last known stopLoss in case it was the

Managed Stop Loss

target.

Why the Time Frame Matters

Running trading sessions of any given trading system on different time frames may produce different

results. This is because the behavior of a trading session may vary depending on how well the time frame

matches the logic of the strategy.

The trading bot evaluates closed candles only. At any given point in time, the current candle in each time frame is the candle that closed last.

For example, let's say it's 11 hours, 39 minutes and 30 seconds of June 11th, 2020. This is how the system determines which is the current candle:

- **1-minute candle:** it is the one that closed at 11:38:59.999.
- **5-minute candle:** it is the one that closed at 11:34:59.999.
- **30-minute candle:** it is the one that closed at 11:29:59.999.
- **1-hour candle:** it is the one that closed at 10:59:59.999.
- **2-hour candle:** it is the one that closed at 09:59:59.999.
- **6-hour candle:** it is the one that closed at 05:59:59.999.
- **24-hour candle:** it is the one that closed at 23:59:59.999 of June 10th!

Let's say the trading system implements conditions that evaluate 30-minute and 1-hour candles.

If a session is run at the 30-minutes time frame, all 30-minutes candles are evaluated. Also, all 1-hour candles are evaluated twice.

However, if the session is run at the 1-hour time frame, only one out of two 30-minute candles are evaluated.

And if the session is run at the 2-hour time frame, only one out of four 30-minute candles and one out of two 1-hour candles are evaluated.

This means that running the session (for this particular trading system) at the 30-minute time frame has higher probabilities of conditions evaluating 30-minute candles to be `true` during the session.

In other words, when running the session on time frames higher than the time frame on which decisions

---

are made, chances are the bot will eventually skip candles on which conditions would have evaluated true,

---

potentially skipping trading opportunities.

The above is `true` for all types of trading sessions.

Slippage

Foundations->Node->Slippage->Definition

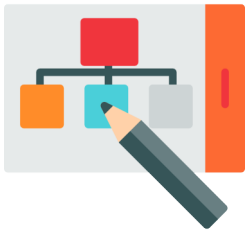
The slippage is an assumption on the difference between the simulated rate and the actual fill rate of a market order, most relevant in the context of backtesting and paper-trading sessions. The parameter is a tool to make simulations more realistic.

Foundations->Node->Slippage->Content

In the context of forward testing and live trading sessions, slippage does not affect the actual transactions. However, the parameter is taken into account when creating the live trading simulation until the actual order fill values are obtained from the exchange.

Fee Structure

Foundations->Node->Fee Structure->Definition



The fee structure is a parameter fundamental to the calculation of fees, both in testing and live trading sessions.

Foundations->Node->Fee Structure->Content

Exchange fees are a crucial part of trading. A strategy may work like a charm when you leave fees out of the equation but would lead you to bankruptcy in a live trading situation.

Exchanges don't usually report the amount charged in fees on each transaction, thus, the system calculates fees and subtract them from balances. Learn more about the handling of fees.

> **Important:** The accuracy of the internal account-keeping depends on this parameter. Make sure you obtain the correct fee structure from the exchange corresponding to the tier of your account.

Snapshots

Foundations->Node->Snapshots->Definition

Snapshots

are CSV files output by the trading bot listing every trade in a backtesting session. The snapshots parameter determines whether snapshots shall be produced, and how.

Foundations->Node->Snapshots->Content

Learn more about snapshots and check the configuration file to control if and how snapshots are produced.

Heartbeats

Foundations->Node->Heartbeats->Definition

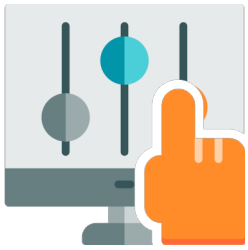
During a trading session, the backend communicates with the frontend via heartbeats to inform the frontend about the status of the session. The parameter controls what information is made available to the user through the frontend.

Foundations->Node->Heartbeats->Content

This parameter affects how you see the progress of the trading session below the trading bot instance node on the design space.

User Defined Parameters

Foundations->Node->User Defined Parameters->Definition



Users may define parameters to be used within the trading system during the trading session.

Foundations->Node->User Defined Parameters->Content

Parameters defined in the configuration file of these node become available to trading systems using the following path: `sessionParameters.userDefinedParameters.config.parameterName`

These parameters may be useful, for example, when you wish to use a constant multiple times across several nodes in the definition of a strategy. Instead of feeding a constant value to all formulas involved, you may feed the parameter instead. This gives you the added benefit of being able to change the value of such a constant in a single point.

Data Storage Section