# Table of Contents

# Introduction

## Purpose

The purpose of this document is to provide a complete definition of all co3 YAML configuration parameters and how and when to use them. The co3 project supports three different routines and each has its own unique configuration parameter requirements.

The three routines that are supported are the following:

- generate
  Generates datasets that are saved into datasets that are can be later played back (see playback).
- Playback
  Plays back previously generated playback datasets through a suite of custom OpenAI Gym environments.
- recover_orderbooks
  This tool recovers orderbooks from the main MongoDB instance and saves the resulting orderbooks to a local instance of MongoDB. The use of the word *recover* in this context is significant in that orderbook data stored in the main MongoDB instance is compressed in the sense that only an infrequent full orderbook is saved, followed by a suite of *much smaller* deltas. Hence, the recovery consists of reconsituting such orderbooks, so that each one is complete.

## Scope

This document does not attempt to explain deep reinforcement learning in general nor the specifics of any of the agent types in particular. For those requiring details of any of the agent types, the reader is referred to the materials found in the ./docs folder.

## Definition and Acronyms

| | ***Meaning*** |
|---|---|
| DRL | 0 |
| RL | Reinforcement Learning |
| | |
| | |

## References:

- Materials found in the ./docs folder.
- *Grokking Deep Reinforcement Learning* by Miguel Morales
- OpenAI gym

## Document Status

| *Date* | *Status* |
|---|---|
| 2021-01-27 | First release. GAC-specific parameter definitions not documented. |
| 2021-02-11 | Add recover_orderbooks |
| | |
| | |

# CO3_CONFIG_PATH

This environment variable contains a list of paths where configuration files are to be found. Each such path is separated with a ":" char; in this sense, it is similar to PYTHONPATH. It is commonly referred to when loading the configuration for all co3 software.

# Agents

This section enumerates the various agent types that are available, where they are categorized as a discrete or continuous action space agents. The required agent type is selected by using the algorithm parameter (e.g. algorithm: TD3)

Discrete:

- ACTOR_CRITIC
- DQN
- QRDQN

Continuous:

- DDPG
- TD3
- SAC
- GAC

# Environments

The co3 software project contains a number of environments. Although each of the environments are different in kind and have different uses, they all adhere to the OpenAI gym standard. `playback` and `generate` both require the use of an environment and hence the selection of the correct environment during configuration setup is crucial.

It is important that the selected environment is compatible with the selected agent, otherwise a mismatched agent / environment error will occur.

Every environment is prefixed with “SentientTrading:”. In the following discussion, this prefix is assumed. Also, the version number is required (all of them are version `v0` at the time of writing). Currently, the following environments are supported.

## Playback Environments

The playback environments feed data drawn from datasets that have been previously generated. Further, these environments can be further categorized as being discrete or continuous action space environments.

### *EvaluateDataFrameEnv-v0*

This environment is suitable for a discrete action space agents.

### *BuyOrderbookDataFrame-v0*

This environment is suitable for continuous action space agents.

### *SellOrderbookDataFrame-v0*

This environment is suitable for continuous action space agents.

### *OrderbookDataFrame-v0*

This environment is suitable for continuous action space agents. Although planned, this environment is not yet available.

## Generation Environments

The generation environments are solely for the purposes of generation datasets that can later played back. The results of all such generations are stored in the ./datasets folder. Although these environments are required to return a **reward**, such reward data is not pertinent in this context.

### *EvaluateTradeHistoryEnv-v0*

This environment creates a dataset derived from the configured extract of trade history. See configuration parameter for details. The observation data recorded into the datasets consists of pairs of (th_vector, it).

### *OrderbookHistoryEnv-v0*

This environment creates a dataset derived from the configured extract of trade history. See configuration parameter for details. The observation data recorded into the datasets consists of pairs of (ob_vector, mid_price).

# Configuration Parameters

## playback

This section provides documentation for all playback configuration parameters.

- console: can be set to logging or progress_bar; if not set, it defaults to logging.
- defaults_ : used to identify a default configuration file (this file itself is discussed below). Note the underscore in the key name, which is required in order to distinguish from Hydra's use of 'defaults'.
- playbacks: A collection of one or more playbacks. This key is required.
  - <playback_name>
    - agent
      Parameters specific to the agent function. Many parameters are common to all agents and hence it is these that can be stored in the defaults_ file. The common set of parameters are discussed here. Agent-specific parameters are discussed in the section Agent-Specific Parameters.
      - algorithm
        The name of the algorithm or agent type to be used for the run.
      - alpha
        SAC only. This is a initial value for the entropy coefficient. The entropy coefficient is tuned automatically. SAC employs gradient-based optimization of alpha toward a heuristic expected entropy. The recommended target entropy is based on the shape of the action space; more specifically, the negative of the vector product of the action shape. Using the target entropy, SAC is able automatically optimize alpha.
      - action_noise

  - type
    To be selected from null | OrnsteinUhlenbeck | Gaussian
  - sigma
    Standard deviation of the noise source.

- automatic_entropy_tuning
  A boolean used to activate entropy tuning. If False, entry tuning is not applied. Only applicable to SAC. Also see the alpha parameter, which is also SAC only.
- batch_size
  Size of the batch to be sampled from experience replay buffer during training.
- buffer_size
  Size of the experiences replay buffer. batch_size <= buffer_size
- episodes
  The number of episodes that the process is to run.
- epsilon_decay
  Only applicable to discrete agents.
  - rate
    Explore decay rate
  - type
    LINEAR | EXPONENTIAL
  - end
    Final (steady-state) explore probability.
  - start
    Initial explore probability
- exploration
  At startup, number of episodes to explore by selecting random actions. This parameter is only relevant to the continuous action space agents.
- actor_lr
  Actor learning rate. All agents types have an actor optimizer and therefore refer to this parameter.
- critic_lr
  Critic learning rate. Only the continuous action space agents have a critic optimizer and therefore it is only these that make use of this parameter.

- policy_noise
  TD3 only. Unused.
- noise_clip
  TD3 only. Unused.
- tau
  Where an agent is using a target model as a means to smooth learning, this parameter defines the proportion of the model's parameters to be merged with the target models during the soft update process. The frequency of soft updates is defined by the agent's target_update_interval.
- gamma
  Bellman discount factor
- gradient_clipping
  Further details available at:
  https://pytorch.org/docs/stable/generated/torch.nn.utils.clip_grad_norm_.html
  - max_norm
  - norm_type
- purge_network
  Boolean defining whether the network is to be purged before the run begins.
- pytorch_models
  Together these models define the learning network that is being used by the agent.
  - <model1_name>
    <model1_name> is actually referred to directly in the agent code.
    - hidden_dims
      This parameter is a list of positive integers, where the n'th integer defines the number of nodes in the n'th hidden layer.
    - activations
      This parameter, which is an array, contains a list of PyTorch activation functions (examples: nn.Tanh and nn.ReLU). If the number of activation functions is exactly the same as the number of hidden layers, then the n'th activation function is applied to the outputs of the n'th hidden layer. However, if the number of activation functions provided is less than the number of hidden layers, then the last activation function (activations[-1]) is applied to hidden layers for which none has been explicitly assigned.

Example: activations: [nn.ReLU] would have nn.ReLU applied to the outputs to all hidden layers.

- <model2_name>
  - hidden_dims
  - activations

- network
  Defines a path name or a file name of a PyTorch network (extension: *.pt). If this parameter is not provided, then a new network file will be created in the location `CO3_PATH/networks/` (called the 'nominal location' below) with file name `<current datetime>.pt`. If this parameter is a path name and it already exists, then it must have content that is recognizable as a valid network file by PyTorch. If the path name does not currently exist, then a new network file will be created. On the other hand, if a file name is provided, then the software will check to see if the file already exists in the nominal location and, if so, it will expect that this file is one that is recognizable as a network file by PyTorch. If no such file is found, then a new file of that name will be created in the nominal location.

- nn_trace
  Only applicable to the QRDQN
  - active
    Boolean. When active, the first count records are recorded to the target_directory.
  - target_directory
    Must be prefixed with ‘nn_traces’
  - count
    For each trace, record count records to the trace file.
  - pattern
    This must be an array of exactly 2 integers, where the first integer is the start instance id of the child process to begin tracing and the second is the interval between subsequent traces; e.g {3, 2] means that tracing will begin at child process 3 and will do so for every other child process after that.

- quantile_resolution
  Only applicable to the QRDQN agent.

- training
  Boolean selecting whether the network is to be trained during the course of the run.

  - training_interval
    Number of steps between trainings of the network.
  - target_update_interval
    Number of trainings between updating the target

- misc:
  Parameters that are process-general and not associated with a specific function such as the agent or environment.
  - csv_path
    Path to where the rewards file is to be recorded. If null, then the path name is automatically generated. If defined and a relative path, then the pathname must begin with ‘rewards’.
  - generate_csv
    A boolean variable defining whether a rewards csv is generated or not.
  - leave_progress_bar
    If set, then completed (test) progress bars are retained; otherwise complete test progress bars are discared.
  - log_interval
    The number of episodes between routine logging messages.
  - log_level
    The logging modules log level. To be selected from [DEBUG | INFO | WARNING | ERROR | CRITICAL]. See https://docs.python.org/3/library/logging.html for further details.
  - seed
    Seed to be used for all random number generators accessed by the software (Python, numpy, OpenAI gym, and PyTorch).
  - record_state
    Record (or not) state in the rewards CSV file.

- env_config
  - datasets
    An array of datasets to be used to drive the playback.
  - is_buy
    Boolean. Only applicable to continuous action space environments.

- reward_offset
  Reward offset. Only applicable to discrete action space environments.
- randomize_dataset_reads
  If True, then at the start of each episode a random selection is made from the set of all datasets (see the datasets parameter) and then, within the selected dataset, a random start point for the episode is selected. If False, then at the start of each episode, a check is applied to ensure that the remaining records in the dataset are sufficient to support another episode. If so, the next episode begins where the last left off. If not, then the next dataset in the set is selected (wrapping back to dataset 0, if necessary) and then starting the next episode at the start of the new dataset. *Note: the above assumes a fixed length episode length. This would need to be reconsidered, if the episode length is unknown.*
- order_depths
  To be selected from ./co3/modules. Only applicable to the discrete action space environments.
- pdf
  The environments refer to PDF-like (actually a histogram) data. It is possible that the PDF is out-of-date (see the parameter age_limit), in which case the PDF is regenerated before the run can begin. During regeneration, the following parameters are referred to: bins, envId, market, start_window, start_range, end_range.
  - age_limit
    Expressed in days, can be decimal; e.g. 0.25 = 6 hrs
  - bins
    Further details can be found here:
    https://numpy.org/doc/stable/reference/generated/numpy.histogram.html
  - envId
    If the PDF needs to be regenerated, then it is to be derived from trades history drawn from this Mongo environment.
  - exchange
    If the PDF needs to be regenerated, then it is to be derived from trades history drawn from this exchange.
  - name
    The name of the PDF as found in the Mongo collection configuration.PDFs

    - market
      If the PDF needs to be regenerated, then it is to be derived from trades history drawn from this market.
    - start_window
      If the PDF needs to be regenerated and this parameter is set, then the the PDF is drawn from trades history in the timeframe start_window to the current time. The UoM of this parameter is days and can have a decimal component (e.g. 0.25 = 6 hours).
    - start_range
      If the PDF needs to be regenerated and start_window is not set, then the the PDF is drawn from trades history in the timeframe start_range to end_range.
    - end_range
      The reader is referred to start_range.
    - graph
      Boolean selecting whether to graph a newly generated PDF.

- child_process
  When a child process is created, its configuration parameters are a merge of the parents configuration parameters with the parameters belonging to the child_process key, although the merge has a number of limitations as follows:
  - A child process itself cannot have a child_process key.
  - agent.training is hardwired to be False (child process networks are never trained).
  - Only the following parameters can be set to values different from that of the parent (training) process (other parameters are ignored):
    1. misc.csv_path
    2. misc.generate_csv
    3. misc.log_interval
    4. agent.episodes
    5. agent.nn_trace
    6. env_config.datasets
    7. env_config.randomize_dataset_reads

## generate

- env
  The environment from which data is to be captured.  Refer to section Generation Environmentsfor a list of the environments that support the generate function.
- destination
  The target destination for the resulting dataset.  The first folder referred to must be ‘datasets’.
- log_interval
  During generation, a log entry is displayed on the console every log_interval records.
- env_config
  - envId
    Environment id trades history of the generation.
  - exchange
    Exchange trades history of the generation.
  - market
    Market trades history of the generation.
  - start_range
    Start range of the generation.
  - end_range
    End range of the generation.
- n
  The ob vector is to compressed to n orderbook entries, or 2n floats, if side in [buy, sell]. If side is ob, then the ob vector sizes will be 2 * 2n = 4n.  Applicable only to the continuous action space environments.
- k
  k-value compression factor.  Applicable only to the continuous action space environments.
- ql
  Quantity Limit.  Applicable only to the continuous action space environments.
- side
  To be seleced from [ buy | sell | ob ].  Applicable only to the continuous action space environments.

### recover_orderbooks

This section provides documentation for all playback configuration parameters.

- recoveries: A collection of one or more recovery configurations. This key is required.
  - <recovery_name_1>
    A valid recover_orderbook YAML file must have at least one recovery.
    - log_interval
      Interval between a log recorded being presented to the console. The log records include both the count and the timestamp of the orderbook last recovered.
    - query
      - envId
        Environment id where orderbooks are located. Usually 0.
      - exchange
        Source exchange of the orderbooks to be recovered.
      - market
        Source market of the orderbooks to be recovered.
      - start_range
        Start range of the recovery.
      - end_range
        End range of the recovery.
  - <recovery_name_2>
    etc. etc.
- recovery
  This parameter is not required, but if provided, then it references a default recovery for the YAML file. If not provided, then the user must provide one at the command line or there must only be only one recovery specified in the file.

# Reference Configuration Files

There exists a set of reference configuration YAML files which can be found in the folder ./co3-configurations/dev. These files are indicative of typical configurations. These files should be referred to as further configurations are developed and historical ones are brought up to date.

## co3.yaml

Contains a suite of typical playbacks for each of the agent types. Note too that it references, using the `defaults_` parameter, the file `playback_defaults.yaml` which is discussed next.

## playback_defaults.yaml

This file contains a suite of default settings. It represents a typical usage for how to minimize configuration effort by defining a default configuration file, which can then by referred to using the `defaults_` parameter.

## generate.yaml

Contains a suite of typical generators.

## single_playback.yaml

This file contains a single playback. It demonstrates the following:

1. Using the `defaults_` parameter.
2. Setting up a YAML file containing only a single playback, which has the advantage that the single playback is implicitly selected.

## recover_orderbooks.yaml

This file contains typical configuration in order to recover a suite of orderbooks.