QUANTITATIVE TRADING
FOR PROFESSIONALS

Upgrade Instructions

AlgoTrader 6.2

27. January 2021

INSTITUTIONAL
ALGORITHMIC TRADING SOFTWARE
FOR **TRADITIONAL SECURITIES** AND **CRYPTO ASSETS**

ALGOTrader

WWW.ALGOTRADER.COM

Upgrade: AlgoTrader 6.2

Below are the instructions on how to upgrade your AlgoTrader 6.1 installation to the 6.2 release.

Adapter Refactoring - Multi-Account Support

We have refactored the connectivity to brokers/exchanges and created new connectors. It is now possible to use several accounts with the same broker/exchange, which requires changes in the data model and configurations.

New Entity Structure

The entity structure now looks like this:

Account * - 1 Provider * - * Exchange 1 - * Security

There is a new Provider entity (e.g., Interactive Brokers or Binance), on which you can trade 1-* exchanges. The link is configured via the DB table provider_exchange.
Accounts are linked to a Provider (you can configure more than one account per provider).

AdapterType and AccountServiceType were decomissioned:

- The adapterType (enum) and account_service_type fields were removed from the account table. You can still identify which provider the account is for with the new connector_descriptor (varchar) field
- The adapter_type (enum) field was removed from the subscription table. This is because market data subscriptions are now tracked on a more granular, account level. Due to that, all subscriptions will be removed from DB and you need to re-subscribe after the first startup after upgrade

InfluxDB Data Structure

Accordingly, to change in the entities AdapterType was decomissioned also for historical data and replaced with connector_descriptor column.

Multi-Account Backwards Compatibility

To manage how multiple accounts of a provider are used and to ensure backwards compatibility, we now have the following account-level settings:

- There is a new primary_for_trading (boolean) field. With this setting it is still possible to use the system the same was as before without much changes to strategy code. If multiple accounts are used, the Strategy should be either:
    - Reworked to use specific account IDs in trading and market data requests
    - Or it can be used as before, but primaryForMarketData and primaryForTrading toggles should be switched to ON on the account the strategy should use
- The new primary_for_market_data (boolean) field specifies the main account for market data streaming in multi-account environment. It takes no effect in single-account setup
- The new primary_for_reference_data (boolean) field - all accounts will have this set to true by default, except for CoinAPI. When set to false, this account will be used for secondary reference data loading only, meaning it will only fill-out null fields of the already present Securities and will not create new Securities by itself

Account Data Migration

After the upgrade to AlgoTrader 6.2, your account data will be automatically migrated by flyway.

AlgoTrader will discover and migrate initial data for the new setup for

- Providers
- Exchanges
- Default Accounts

Previously used accounts will be migrated to be compatible with the new AlgoTrader version. This means that old data will be preserved, but some changes to it are possible.

Initial account deactivation:

- The account.active flag is set to false by default and will be reset to this value after migration to 6.2. This means that accounts will have to be re-activated manually, either via Config UI or by resetting this flag to true in DB and restarting the system.

Exchange-Specific Changes

- Kraken-based Securities require a KRAKEN_WEBSOCKET_TICKER field to be filled out, otherwise will be not able to stream market data. To fill it out, a reference data load should be done for Kraken
- TradingView - it is now possible to explicitly state trading_view_id on the Exchange entity

Deprecations/Method Changes

The Slicing order was removed. Please use another execution algo instead

The singleDataSource spring profile was removed. Please use the pooledDataSource instead

The ReferenceDataStarters and Spring Profiles Were Removed

AlgoTrader is now capable of running multiple reference data services at once and the service will be available for all active accounts, hence dedicated ReferenceDataStarters are now redundant. There are multiple ways to load reference data into AlgoTrader without using the starter, namely:

- Using the Reference Data Manager
- Using a configuration that loads reference data on start-up: dataSource.loadReferenceDataOnStartup property (ONCE/ALWAYS/NEVER)
- Using the StrategyService: getReferenceDataService()
- Via REST:
    - /rest/referenceData/retrieve/{accountId}/{securityFamilyId}
    - /rest/referenceData/retrieveStocks/{accountId}/{securityFamilyId}/{securityId}
    - /rest/referenceData/retrieveAll/{accountId}

API Changes in Services Available to Strategies (AccountService)

The following methods were modified:

- String getDepositAddress(AccountServiceType accountServiceType, String currency, String walletType)
  is now
  String getDepositAddress(long accountId, String currency, String walletType)

The following methods were removed:

- void subscribeAccountEvent(long accountId) --> account events are now automatically subscribed
- void unSubscribeAccountEvent(long accountId) --> account events are now automatically subscribed
- ExternalAccountService getExternalAccountService(long accountId) --> all methods are now available through the generic AccountService
- long persist(Account account) --> use the RDM or ConfigUI to add or update accounts

The following methods were added:

- default List<String> getExtAccounts(long accountId, boolean managed)
  → retrieves supported external accounts (vendor-specific)
- List<SecurityPositionVO> getAccountPositions(long accountId)
  → retrieves account positions (vendor-specific)

- Allocations getAllocations(long accountId, String profileName) → allocation management (vendor-specific)
- void updateAllocations(long accountId, Allocations allocations) → allocation management (vendor-specific)

API Changes in Services Available to Strategies (LookupService)

The following methods were added:

- List<Account> findAccountsWithExchangeAccess(final Exchange exchange)
  → finds all accounts that can trade on a given Exchange

API Changes in Services Available to Strategies (MarketDataService)

The following methods were modified:

- void subscribe(String strategyName, long securityId, AdapterType adapterType);
  is now
  void subscribe(String strategyName, long securityId, long accountId);
- void unsubscribe(String strategyName, long securityId, AdapterType adapterType);
  is now
  void unsubscribe(String strategyName, long securityId, long accountId);

The following methods were removed:

- void initSubscriptions(); → non functional, use subscribe() methods to add and init new subscriptions
- void initSubscriptions(AdapterType adapterType); → non functional, use subscribe() methods to add and init new subscriptions
- void requestCurrentTicks(String strategyName); → non functional, use subscribe() methods to add and init new subscriptions
- void onDisconnected(final AdapterType adapterType);
  → use HealthService and DiagnosticEvents to find out about disconnections
- void removeNonPositionSubscriptions(String strategyName);
- void removeNonPositionSubscriptionsByType(String strategyName, Class<? extends Security> type)
- void reconnect(AdapterType adapterType);
  → not needed anymore, reconnections are automated
- boolean isTickValid(TickVO tick);
- boolean hasVol(long securityId);
- TickVO normalizeTick(TickVO tick);

API Changes in Services Available to Strategies (OrderLookupService)

The following methods were modified:

- String lookupIntIdByExtId(AdapterType adapterType, String extId);
  is now
  String lookupIntIdByExtId(ConnectorDescriptor connectorDescriptor, String extId);
- Order getActiveOrderByExtId(AdapterType adapterType, String extId);
  is now
  SimpleOrder getActiveOrderByExtId(long accountId, String extId);
- List<Order> getAllActiveOrdersByAdapterType(final AdapterType adapterType);
  is now
  List<Order> getAllActiveOrdersByConnectorDescriptor(final ConnectorDescriptor connectorDescriptor);

The following methods were added:

- OrderStatus getCurrentOrderStatus(long accountId, String extId, String intId);
- Collection<SimpleOrder> getAllActiveOrdersByAccountId(long accountId);

API Changes in Services Available to Strategies (PropertyService)

It is now possible to attach `Serializable` OrderProperties to Orders, not only Strings:

- void addOrderProperty(final long orderId, final String name, final String value,
  final OrderPropertyType type);
  is now
  void addOrderProperty(final long orderId, final String name, final Serializable value,
  final OrderPropertyType type);

API Changes in Services Available to Strategies (RateLimitService)

Users should refer to rate limitations via the account ID now instead of the adapterType.

API Changes in Services Available to Strategies (ReferenceDataService)

It is now possible to have multiple ReferenceDataServices available.
Users should refer to ReferenceData services via the account ID from now on, e.g.:

- void retrieve(long securityFamilyId)
  is now
  void retrieve(long accountId, long securityFamilyId);
- void retrieveStocks(long securityFamilyId, long securityId);
  is now
  void retrieve(long accountId, long securityFamilyId);

- void retrieveAll();
  is now
  void retrieveAll(long accountId);

Non-functional methods were removed from the API:

- String getPerpetualSwapDescription(String baseCurrency, String quoteCurrency, String exchangeName);
- String getForexDescription(String baseCurrency, String quoteCurrency, String exchangeName);
- String getSecurityFamilyName(String baseCurrency, String quoteCurrency, String exchangeName, SecurityFamilyType type);
- String getIIndexFamilyName(String symbol, String exchangeName);
- String getFutureYearMonth(ZonedDateTime expiryDate);
- String getFutureYearMonth(Instant expiryDate);
- String getFutureSymbol(String symbolRoot, ZonedDateTime expiryDate);
- String getFutureSymbol(String baseCurrency, String quoteCurrency, Instant expiryDate);
- String getOptionSymbol(SecurityFamily family, OptionType optionType, BigDecimal strike, LocalDate expiryDate, String exchangeName, String optionSymbolPattern);
- String getOptionSymbol(final String familyName, final String familySymbolRoot, final double familyContractSize, final String familyCurrency, final OptionType optionType, final BigDecimal strike, final LocalDate expiryDate, final String exchangeName, final String optionSymbolPattern);
- String getForexSymbol(String baseCurrency, String quoteCurrency);
- String getPerpetualSwapSymbol(String baseCurrency, String quoteCurrency);
- String getIndexSymbol(String quoteCurrency);
- boolean isInverseContract(String baseCurrency, String contractCurrency);
- boolean waitForMarketDataConnection();
- String getDefaultAccountName();

API Changes in Services Available to Strategies (SubscriptionService)

Users should refer to Subscriptions via the account ID now instead of the adapterType.

The following methods were modified:

- void subscribeMarketDataEvent(String strategyName, long securityId, AdapterType adapterType, boolean subscribeToConversionCurrenciesIfNecessary);
  is now
  void subscribeMarketDataEvent(String strategyName, long securityId, final long accountId, boolean subscribeToConversionCurrenciesIfNecessary)
- void subscribeMarketDataEvent(String strategyName, long securityId, AdapterType adapterType);

is now
void subscribeMarketDataEvent(String strategyName, long securityId, final long accountId);

- void unsubscribeMarketDataEvent(String strategyName,
  long securityId, AdapterType adapterType);
  is now
  void unsubscribeMarketDataEvent(String strategyName, long securityId, long accountId);
- void unsubscribeFromOrderBook(String strategyName,
  long securityId, AdapterType adapterType);
  is now
  void unsubscribeFromOrderBook(String strategyName, long securityId, long accountId);
- void subscribeToOrderBook(String strategyName, long securityId, AdapterType adapterType);
  is now
  void subscribeToOrderBook(String strategyName, long securityId, long accountId);

Changes in the REST interface

The following endpoints were renamed:

- /rest/account/{name} → /rest/account/byName/{name}
- /rest/execution/order/daily → /rest/execution/order/all/daily
- /rest/execution/order/{intId:.+} → /rest/execution/order/byIntId/{intId:.+}
- /rest/execution/order/daily → /rest/execution/order/all/daily
- /rest/execution/order/allActive" → /rest/execution/order/all/active
- /rest/execution/order/allOrderPreferences → /rest/execution/order/all/orderPreferences
- /rest/exchange/{id} → /rest/exchangeById/{id}
- /rest/transaction/{id} → /rest/transactionById/{id}
- /rest/marketData/currentMdEvents/{securityId} → /rest/marketData/currentMarketDataEvent /{securityId}
- /rest/marketData/currentMDEvent/{securityId}/{fullClassName:.+} → /rest/marketData /currentMarketDataEvent/{securityId}/{fullClassName:.+}
- /rest/marketData/currentMDEvent/{securityId:.+} → /rest/marketData /currentMarketDataEvent/{securityId:.+}
- /currentMdEvents/{fullClassName:.+} → currentMarketDataEvents/{fullClassName:.+}

Removed endpoints (note: some PathParam need to be replaced with queryParam):

- /rest/account/depositAddress/{exchange}/{currency}/{walletType}
- /rest/account/subscribe
- /rest/account/unsubscribe
- /rest/combination/resetComponentWindow
- /rest/execution/order/details/completedByIntId/{intId:.+}
- /rest/execution/order/details/completedDetails
- /rest/execution/order/details/dailyOrdersByStrategy/{strategyName:.+}
- /rest/execution/order/details/openOrderDetailsByInt/Id/{intId:.+}

- /rest/execution/order/lookupIntIdByExtId/{connectorDescriptor}/{extId:.+}
- /rest/execution/reconciliation
- /rest/historicaldata/bidAsksByMaxDate
- /rest/lookup/dailyOrders → use this one instead: /rest/execution/order/all/daily
- /rest/lookup/dailyOrdersByStrategy/{strategyName} → use this one instead: /rest/execution/order/details/dailyOrdersByStrategy/{strategyName:.+}
- /rest/lookup/securityById/{securityId} → use this one instead: /rest/security/{id}
- /rest/lookup/openPositions → use this one instead: /rest/portfolio/openPositions
- /rest/lookup/getAccountByName/{accountName} → use this one instead: /rest/account/byName/{name}
- /rest/lookup/scribedCombinationsByStrategyAndUnderlying/{strategyName}/{underlyingId}
- /rest/lookup/subscribedCombinationsByStrategyAndSecurityClass/{strategyName}/{className:.+}
- /rest/lookup/subscribedComponentsByStrategy/{strategyName}
- /rest/lookup/subscribedComponentsBySecurity/{securityId}
- /rest/lookup/subscribedComponentsByStrategyAndSecurity/{strategyName}/{securityId}
- /rest/lookup/subscribedCombinationsByStrategyAndSecurityId/{strategyName}/{securityId}
- /rest/lookup/subscribedCombinationsByStrategyName/{strategyName}
- /rest/lookup/subscribedOptions
- /rest/lookup/subscribedFutures
- /rest/portfolio/savePortfolioValueWithTransaction
- /rest/portfolio/savePortfolioValues
- /rest/referenceData/getDefaultAccountName → use this one instead: /rest/defaultAccountName
- /rest/subscription/subscribeToOrderBook/{strategyName}/{securityId} → use this one instead: /subscription/marketdata/orderBook
- /rest/subscription/unsubscribeFromOrderBook/{strategyName}/{securityId}/{adapterType} → use this one instead: /subscription/marketdata/orderBook
- /rest/subscription/subscribeToOrderBook/{strategyName}/{securityId}/{adapterType} → use this one instead: /subscription/marketdata/orderBook (using appropriate request params)
- /rest/subscription/subscribeToAggregatedOrderBook/{strategyName}/{symbol}/{securityClass:.+} → use this one instead: /subscription/marketdata/aggregatedOrderBook
- /rest/subscription/unsubscribeFromAggregatedOrderBook/{strategyName}/{symbol}/{securityClass:.+} → use this one instead: /subscription/marketdata/aggregatedOrderBook

Order book related REST changes:

Removed

- /rest/orderBookFilter/global
- /rest/orderBookFilter/byAccount

Added

- /rest/orderBookFilter/minDepth
- /rest/orderBookFilter/maxPriceLevelDeviation
- /rest/orderBookFilter/maxBookDepth
- /rest/orderBookFilter/isEnabled

About AlgoTrader

Available on-premise or in the cloud, AlgoTrader is an institutional algorithmic trading software solution for conducting quantitative research, trading strategy development, strategy back-testing and automated trading for both traditional securities and crypto assets. AlgoTrader provides everything a typical quantitative trading firm requires to run its research and trading operations. It is the very first and most advanced algorithmic trading software product to allow automated trading of Bitcoin and other crypto assets. Based in Zurich, New York, and Singapore, AlgoTrader operates globally.

For more information, please visit https://www.algotrader.com.

Contact

E-Mail info@algotrader.com

T +41 44 291 14 85

Follow AlgoTrader on Twitter and LinkedIn.

For inquiries, please contact

Support Team

T +41 44 291 14 85

support@algotrader.com