Ember Quick Start

How to get started with Execution Server 5.X (Ember)

# Installation

## Prerequisite

Execution Server needs the latest version of Java 8 or Java 11 (NOTE: JDK version rather than JRE).

### Quick Start

Extract Ember distribution package (deltix-ember-<version>.zip) into destination directory. For example: /home/deltix/ember.

Various launch scripts described below can be found under /bin subdirectory.

1. Set environment variable that will define ember home directory:

Linux:  
export EMBER\_HOME=/home/deltix/emberhome

Windows:  
set EMBER\_HOME=D:\Deltix\QuantServer\emberhome

2. Create empty configuration file ember.conf under EMBER\_HOME

3. Start ember using script /bin/ember inside your EMBER\_HOME:

29 Sep 12:49:30.123041 INFO [main]

\_\_\_\_\_\_ \_\_

/ \_\_\_\_/\_\_\_ \_\_\_ / /\_ \_\_\_ \_\_\_\_\_ Version: 0.2-SNAPSHOT  
 / \_\_/ / \_\_ `\_\_ \/ \_\_ \/ \_ \/ \_\_\_/ Timebase: 5.2.6x.44760  
 / /\_\_\_/ / / / / / /\_/ / \_\_/ / Home: D:\Deltix\QuantServer\ember  
/\_\_\_\_\_/\_/ /\_/ /\_/\_.\_\_\_/\\_\_\_/\_/ Copyright (c) 2017 Deltix, Inc.

29 Sep 12:49:30.358422 INFO [trade-engine] Log is replayed, messages read 0

29 Sep 12:49:30.361264 INFO [trade-engine] Startup complete. This server is ready to process new requests.

Congratulations, Ember is running!

## Further steps

### Separate output directory

In addition to environment variable EMBER\_HOME user can define EMBER\_WORK to define work directory. Work directory is used for all output files: transaction journal, log files, counters, etc.

Linux:  
export EMBER\_WORK=/mnt/nvme/ember

Windows:  
set EMBER\_WORK=M:/ember

We recommend keeping work directory separate from home directory. For higher performance work directory can be placed on fast storage (SSD, NVMe, etc.).

### Making Configuration changes

Ember configuration files use HOCON/JSON format described [here](https://github.com/typesafehub/config). Previously you created empty ember.conf file in your EMBER\_HOME directory. This configuration file contains user-specific custom settings. Default settings are contained in ember-default.conf file. Depending on distribution this file can be either included in you EMBER\_HOME directory or placed into one of Ember JAR files.

### Connect to TimeBase

Add the following block to ember.conf:

timebase.setting {  
 url = "dxtick://localhost:8011"   
}

Correct host, port, and logon credentials to match your TimeBase server and restart Ember. Also, it may be a good idea to tell pricing service what TimeBase stream(s) market data comes from:

pricing.settings {  
 liveSubscription {  
 streams = ["sine"]  
 }  
}

### Configure a trade connector

Trade connectors route orders to execution venues and bring back trading events.

For example, let's assume we have connection to CME iLink MSGW gateway. The following configuration fragment enables CME connector and defines connection parameters:

connectors {  
 **CME : ${CMEMSGW} {  
 settings {  
 senderCompId = ROGUE1N  
 targetCompId = CME  
 host=69.50.112.141  
 port=61529  
 }  
 }**}

The above connector is registered under the name "cme" (case sensitive). This definition uses predefined connector type "cmemsgw" that implements Ember to CME MSGW iLink order routing. Execution Server is integrated with 100+ different execution venues like CME.

There is a special trading connector that doesn't really connector to actual market but instead simulates order execution. The following fragment deploys trading simulator with destination id "sim":

connectors {  
 ...  
 **SIM : ${SIM} { }**  
}

You can read more on design of algorithms and deployment parameters in Trading Connector Developer's Guide.

### Deploy an algorithm

Trading Algorithms are building blocks of Execution Server. Let's deploy "ICEBERG" algorithm that takes a large order and slices it into smaller chunks visible on execution venue.

Add the following block into ember.conf to deploy ICEBERG algorithm implemented by deltix.ember.service.algorithm.samples.iceberg package:

algorithms {  
 ICEBERG : ${template.algorithm.default} {  
 factory = deltix.ember.service.algorithm.samples.iceberg.IcebergSampleAlgorithmFactory  
 queueCapacity = 4M  
 settings {  
 orderCacheCapacity = 4096  
 maxInactiveOrdersCacheSize = 1024  
 initialActiveOrdersCacheSize = 1024  
 initialClientsCapacity = 16  
 }  
 }  
}

You can read more on design of algorithms and deployment parameters in Algorithm Developer's Guide.

### Monitor trade flow

Ember includes standalone component that allows monitoring system vitals. This is a web-based application that connects to ember transaction log and message bus and displays up-to-date information in a web-based GUI.

Monitor back-end[[1]](#footnote-1) can be started using command /bin/ember-monitor:

### Send a test order

There are several ways to send test orders:

* Adapt and run TickToOrder sample algorithm that sends order on every tick
* Modify and run our SubmitOrderSample
* Enable FIX Server and receive order via FIX protocol, for example from Deltix Strategy Server or Trading Console

## Further reading

The following documentation is available:

* Read **Execution Server User's Guide** for more details about Ember design and application options.
* Trading API
* Market Data API
* Trading Connector Developers Guide
* Algorithm Developers Guide
* Deltix FIX API
* If you have a problem, check **FAQ** document and **Troubleshooting Guide**

1. Ember Monitor finds Ember transaction log and message bus by examining ember configuration. It assumes that configuration files can be found in the directory defined by environment variables EMBER\_HOME and EMBER\_WORK. [↑](#footnote-ref-1)