U1 TECHNOLOGIES

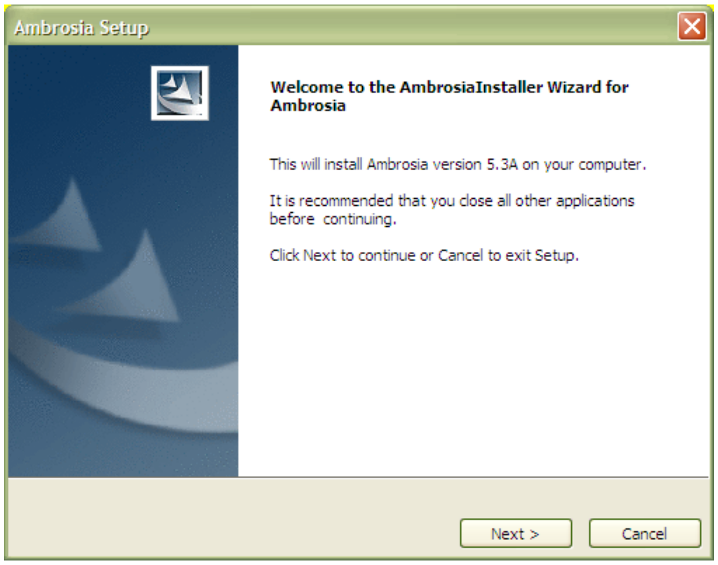
AmbrosiaMQ™ Multi-Broker Configuration

The purpose of this guide is to provide a set of instructions for quickly creating an AmbrosiaMQ multi-broker configuration. There are many other advanced configurations, including multi-regional collectives that are connected through redundant bridge brokers. The configuration of these types of deployments is beyond the scope of this document.

1 Installation

AmbrosiaMQ can be installed very easily and quickly using a four step process. Simply execute the command setup.bin (for Solaris and Linux) or setup.exe (for windows) and follow the installation instructions as illustrated below.

1



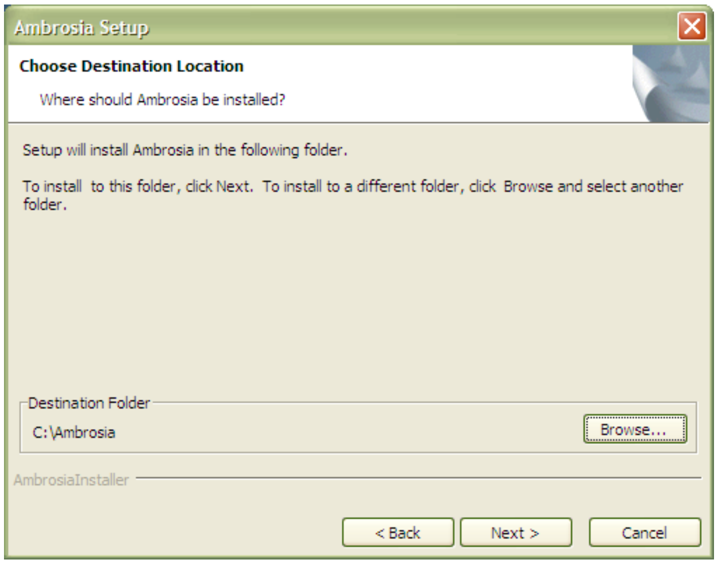
Welcome to the AmbrosiaInstaller Wizard for Ambrosia

This will install Ambrosia version 5.3A on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue or Cancel to exit Setup.

2


Ambrosia Setup

Choose Destination Location

Where should Ambrosia be installed?

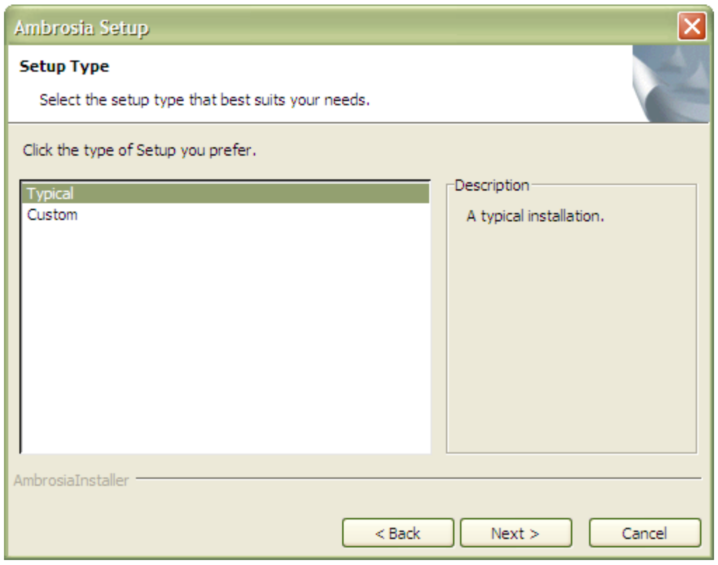
Setup will install Ambrosia in the following folder.

To install to this folder, click Next. To install to a different folder, click Browse and select another folder.

Destination Folder

C:\Ambrosia

3


Ambrosia Setup

Setup Type

Select the setup type that best suits your needs.

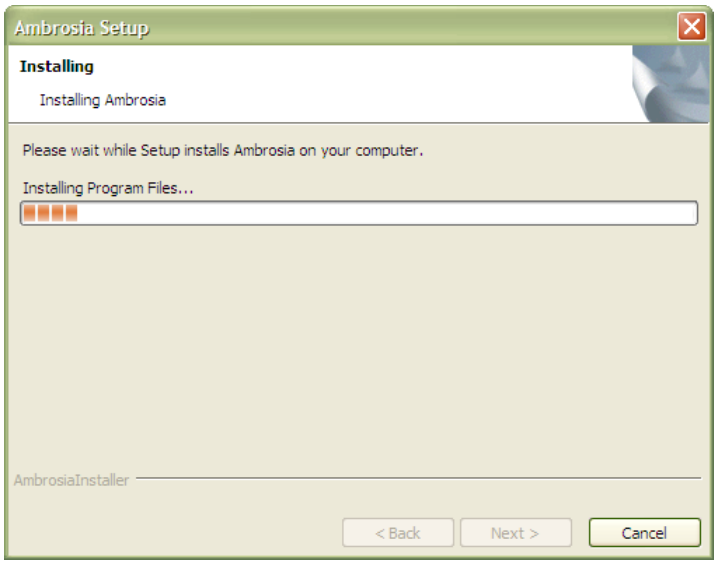
Click the type of Setup you prefer.

Typical
Custom

Description

A typical installation.

4


Ambrosia Setup

Installing

Installing Ambrosia

Please wait while Setup installs Ambrosia on your computer.

Installing Program Files...

2 Overview of Multi-Collective Configuration

AmbrosiaMQ uses the following terminology:

- Broker — a message broker that is responsible for routing messages between consumers and producers. Client applications (message producers and consumers) connect with a broker.
- Collective — a cluster of interconnected brokers that route messages between their clients. A message producer that is connected to a broker in the collective can send messages to a consumer that is connected to any other broker in the collective.
- Primary configuration server/broker — a broker that is responsible for distributing configuration information to other brokers. This includes security information, collective configuration, load-balancing pool information, and so forth. A configuration server need not be a member of the same collective as the brokers it serves. In fact, a configuration server need not be part of any collective at all. Configuration servers can also function as regular brokers, routing messages between producers and consumers. However, in practice, this kind of broker does not do so.
- Secondary configuration server/broker — a broker that is responsible for distributing configuration information to other brokers in case the primary configuration server is down or unreachable. AmbrosiaMQ supports up to 255 secondary configuration servers.
- Regular broker — A broker that is not a primary or secondary configuration server. Its function is to route messages between producers and consumers

The standard installation of AmbrosiaMQ includes several multi-broker configurations. One such configuration is called *basic-ib-sec* (Basic Interbroker with Security). We will use this configuration as the base and will build on it. The diagram below depicts the broker configuration for basic-ib-sec.

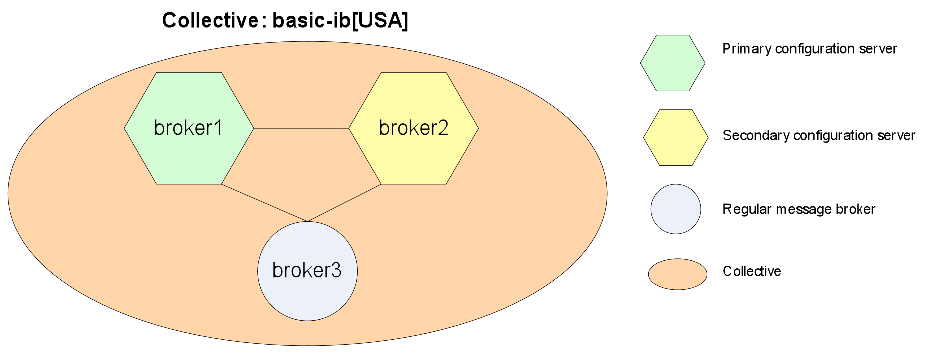

Collective: basic-ib[USA]
broker1
broker2
broker3
Primary configuration server
Secondary configuration server
Regular message broker
Collective

Figure 1 Configuration of basic-ib-sec

3 Adding a new Collective

Assume that we want to add a new collective called *east-coast* with three regular brokers in it called *NY1*, *NY2* and *NY3*. The machines that will host NY1, NY2, and NY3 are called NY1Host, NY2Host and NY3Host, respectively. The ports to which these brokers listen will be 58101, 58201, and 58301. We will use broker1 and broker2 from basic-ib-sec as the primary and secondary configuration server. The resulting configuration is depicted below.

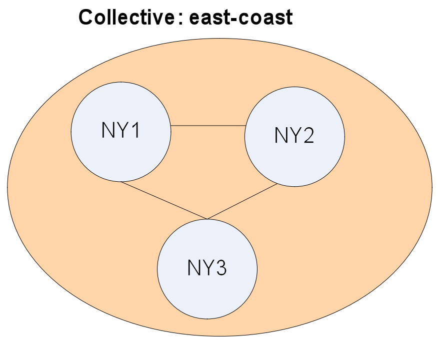


Figure 2 The collective east-coast with three brokers

Note that the collective *east-coast* and *basic-ib[USA]* are not connected to each other. This can be done using a *bridge* collective, but the configuration is beyond the scope of this document.

The steps for creating the collective and running its brokers are as follows. Replace [AMBROSIAMQ] with the AmbrosiaMQ installation directory.

1. Ensure that no broker in the basic-ib-sec collective is running. You can run the script `stop.sh` in [AMBROSIAMQ]/config/basic-ib-sec to stop the running brokers.

2. Create broker specification and collectives as follows
   a. Change directory to [AMBROSIAMQ]/config/basic-ib-sec/broker1
   b. Edit the file `interbroker.cfg`. This file consists of several sections (`.brokers`, `.lbpool`, `.collective`). In the `.brokers` section add the specification for the three NY brokers as follows (replace `NY1Host`, `NY2Host` and `NY3Host` with the host name or IP address of the machine on which you plan to run these brokers):
      ```
      NY1, NY1Host:58101
      NY2, NY2Host:58201
      NY3, NY3Host:58301
      ```
   c. At the end of this file, add a new collective section as follows:
      ```
      .collective east-coast
      NY1
      NY2
      NY3
      ```
   d. Commit your changes and quit the editor

3. Start the configuration server brokers.
   a. Change directory to [AMBROSIAMQ]/config/basic-ib-sec
   b. Run the script *run.sh*

4. Start Security Administration Utility as follows (the procedure assumes you are executing the utility on the same machine as the primary configuration server. Otherwise, replace *localhost* with the host name or IP address of the machine that runs the primary configuration server)
   a. Source the environment
      ```
      source [AMBROSIAMQ]/linux/setcp
      ```
   b. Run the tool `SecAdminConsole`
      ```
      java com.u1.tools.SecAdminConsole localhost:8001
      Administrator Administrator
      ```
   c. Add security credentials for the NY brokers
      ```
      SecAdmin> set user NY1, pass1
      SecAdmin> set user NY2, pass2
      SecAdmin> set user NY3, pass3
      ```
   d. Add NY brokers to the `brokers` group
      ```
      SecAdmin> add member Brokers, NY1
      SecAdmin> add member Brokers, NY2
      SecAdmin> add member Brokers, NY3
      ```
   e. Quit from `SecAdminConsole`
      ```
       SecAdmin> quit
      ```

5. Configure each of the three NY brokers:

   a. Under the directory [AMBROSIAMQ]/config/basic-ib-sec create three other directories called NY1, NY2 and NY3
      ```
      cd [AMBROSIAMQ]/config/basic-ib-sec
      mkdir NY1 NY2 NY3
      ```

   b. Copy the file [AMBROSIAMQ]/config/basic-ib-sec/broker3/ambroker.ini to each of the directories NY1, NY2 and NY3
      ```
      cd [AMBROSIAMQ]/config/basic-ib-sec
      cp broker3/ambroker.ini NY1
      cp broker3/ambroker.ini NY2
      cp broker3/ambroker.ini NY3
      ```

   c. For each ambroker.ini file in NY1, NY2, and NY3 create a log directory
      ```
      mkdir [AMBROSIAMQ]/config/basic-ib-sec/NY1/log
      mkdir [AMBROSIAMQ]/config/basic-ib-sec/NY2/log
      mkdir [AMBROSIAMQ]/config/basic-ib-sec/NY3/log
      ```

   d. For each ambroker.ini file in NY1, NY2, and NY3, modify the file to reflect the correct settings. The following properties should be changed. Replace [ConfigServerHost] with the host name or IP address of the machine that runs the configuration servers.

      In file NY1/ambroker.ini
      ```
      BROKER_NAME=NY1
      BROKER_PASSWORD=pass1
      ACCEPTORS = 58101, ssl://58102
      ```

```
IB_CONFIG_SERVER=[ConfigServerHost]:8001,
[ConfigServerHost]:8011
```

In file NY2/ambroker.ini

```
BROKER_NAME=NY2
BROKER_PASSWORD=pass2
ACCEPTORS = 58201, ssl://58202
IB_CONFIG_SERVER=[ConfigServerHost]:8001,
[ConfigServerHost]:8011
```

In file NY2/ambroker.ini

```
BROKER_NAME=NY3
BROKER_PASSWORD=pass3
ACCEPTORS = 58301, ssl://58302
IB_CONFIG_SERVER=[ConfigServerHost]:8001,
[ConfigServerHost]:8011
```

6. Initialize the database of each broker.

   a. For broker NY1
      ```
      cd [AMBROSIAMQ]/config/basic-ib-sec/NY1
      java com.u1.broker.InitBrokerDatabase create
      ```

   b. For broker NY2
      ```
      cd [AMBROSIAMQ]/config/basic-ib-sec/NY2
      java com.u1.broker.InitBrokerDatabase create
      ```

   c. For broker NY3
      ```
      cd [AMBROSIAMQ]/config/basic-ib-sec/NY3
      java com.u1.broker.InitBrokerDatabase create
      ```

7. Run each of the three NY brokers

   a. For broker NY1
      ```
      source [AMBROSIAMQ]/linux/setcp
      cd [AMBROSIAMQ]/config/basic-ib-sec/NY1
      java com.u1.broker.Broker
      ```

   b. For broker NY2
      ```
      source [AMBROSIAMQ]/linux/setcp
      cd [AMBROSIAMQ]/config/basic-ib-sec/NY2
      java com.u1.broker.Broker
      ```

   c. For broker NY3
      ```
      source [AMBROSIAMQ]/linux/setcp
      cd [AMBROSIAMQ]/config/basic-ib-sec/NY3
      java com.u1.broker.Broker
      ```

4 Using MySQL as the broker's databases

Every broker's ambroke.ini file has a section for database configuration parameters. By default, AmbrosiaMQ uses Derby. However, you can use other databases such as MySQL, Oracle or SyBase. To replace Derby with MySQL, perform the following steps:

1. Comment out the Derby section of ambroker.ini by placing a # at the beginning of each of the following lines (or, you can just delete these lines)
```
# DB_USER=user1
# DB_PASSWORD=user1
# DB_CONNECT=jdbc:derby:AmbrosiaMQ_db;create=true
# JDBC_DRIVER=org.apache.derby.jdbc.EmbeddedDriver
# DB_PROPERTIES=../derby.cfg
```

2. Uncomment the section that has MySQL parameters and insert the correct values. The configuration below assumes a database user called `AmbrosiaMQ` with a password `AmbrosiaMQ` and MySQL server running locally at port 7997. There must be a MySQL database instance called ambroker
```
    DB_USER=AmbrosiaMQ
    DB_PASSWORD= AmbrosiaMQ
    DB_CONNECT=jdbc:mysql://127.0.0.1:7997/ambroker
    JDBC_DRIVER=com.mysql.jdbc.Driver
    DB_PROPERTIES=[AMBROSIAMQ]/config/mysql.cfg
```

Note that if you change the database configuration of an existing broker, then you will need to re-initialize the broker's database by running step 6 in Section 3.