SHRINE 4.2.0 Chapter 8.2 - Configuring a Hub

> **Hub Admins Only**
>
> This section is intended for hub administrators only.

Here is a sample shrine.conf file for a system running SHRINE 4.2.0 , for a node supporting researchers and distributing queries.

shrine.conf

```
shrine {

  shrineHubBaseUrl = "https://localhost:6443" //The shrine hub's URL as observed from this tomcat server
  i2b2BaseUrl = "http://i2b2.example.com:9090" //The local i2b2's URL as observed from this tomcat server
  i2b2Domain = "exampleDomain"
  i2b2ShrineProjectName = "SHRINE"

  nodeKey = "somethingHub" //node key to get information from the hub about itself as a node.

  //shrineDatabaseType = "mysql" // "mysql" by default. It can be "sqlserver" "mysql" or "oracle"

  webclient {
    siteAdminEmail = "shrine-admin@example.com"
  }

  hiveCredentials {
    username = "demo"
    crcProjectId = "Demo"
  }//hiveCredentials

  hub {
    create = true

    messagequeue {
      blockingqWebApi {
        enabled = true  //run shrine's MoM system at the hub.
      }
    }//messagequeue
  }//hub

  adapter {
    create = false
  }//adapter

  keystore {
    privateKeyAlias = "shrine-hub"
    caCertAliases = ["shrine-ca"]
  }//keystore

  steward {
    emailDataSteward {
      //provide the email address of the shrine node system admin, to handle bounces and invalid addresses
      from = "shrine-admin@example.com"
      //provide the email address of the shrine node system admin, to handle bounces and invalid addresses
      to = "shrine-steward@example.com"
      //provide the externally-reachable URL for the data steward
      externalStewardBaseUrl = ${shrine.shrineHubBaseUrl}/shrine-api/shrine-steward
    }
  }//steward
}//shrine
```

It is rare but possible to have a set of patient data at the hub. Simply include the adapter section of qep-and-adapter-shrine.conf in your shrine.conf , tailored to your system as explained earlier in this chapter.

Set the shrine i2b2 user password in the password.conf file in /opt/shrine/tomcat/lib .

password.conf

```
shrine.hiveCredentials.password = "changeit"
```

Next, configure the initial network structures and queues for the hub.

Download the shrine network lifecycle tool from https://repo.open.catalyst.harvard.edu/nexus/content/groups/public/net/shrine/shrine-network-lifecycle-tool/4.2.0/shrine-network-lifecycle-tool-4.2.0-dist.zip into /opt/shrine:

```
cd /opt/shrine


wget https://repo.open.catalyst.harvard.edu/nexus/content/groups/public/net/shrine/shrine-network-lifecycle-tool/4.2.0/shrine-network-lifecycle-tool-4.2.0-dist.zip -O shrine-network-lifecycle.zip unzip shrine-network-lifecycle.zip

cd shrine-network-lifecycle
```

Inside the conf directory, edit the override.conf file to use your database username and password:

override.conf

```
shrine {
  queryEntryPoint{
    audit {
      database {
        dataSourceFrom = "testDataSource" //Can be JNDI or testDataSource . Use testDataSource for tests and command line tools, JNDI everywhere else
        testDataSource {
          driverClassName = "com.mysql.cj.jdbc.Driver" //JDBC driver class name
          url = "jdbc:mysql://localhost:3306/qepAuditDB?serverTimezone=UTC" //URL to your database
          credentials {
            username = "yourUserName"
            password = "yourDatabasePassword"
          }
       }
     }
  }
}
```

Next, create a file named network.conf to meet your needs. At a minimum include the network section and a section for the hub's QEP:

network.conf

```
shrine {
  network {
    network {
      name = "Network Name"
      hubQueueName = "hub"
      adminEmail = "yourEmail@yourhospital.edu"
      momId = "HubQueue"
    }
    nodes = [
      {
        name = "Hub's QEP"
        key = "hub-qep"
        userDomainName = "network-hub"
        queueName = "shrinehub"
        sendQueries = "false"
        adminEmail = "yourEmail@yourhospital.edu"
        momId = "HubQepQueue"
      }
    ]
  }
}
```

This will use in-tomcat messaging. See below to use AWS SQS or Kafka message-oriented middleware.

In-Tomcat or External Messaging Systems

SHRINE uses a message-oriented-middleware system to process queries and results asynchronously. Researchers can watch progress of their queries at different nodes, and see results as soon as any results are available.

The hub admin can select from SHRINE's own in-tomcat message system, AWS SQS, or Kafka. It is possible to switch from one MOM system to another in shrine 4.2, but the process is not gentle: Stop all nodes on the network. Configure all nodes for the new messaging system. Restart all nodes in the network.

In-Tomcat

For most shrine networks we recommend using shrine's in-tomcat messaging system: Networks that want to minimize administration effort, small networks, isolated networks, networks set up for demonstrations, and networks with few researchers. The main drawback to shrine's in-tomcat messaging is that messages in-flight and in-process are stored in memory; if a message is in transit when the hub admin stops tomcat it will be lost. (The 45-node ACT network has used this messaging system since SHRINE 1.25 with few incidents.)

There is no additional configuration or set-up to use this messaging system.

AWS SQS

For large networks that desire telecom-like reliability we use AWS SQS. AWS SQS stores messages that are in-flight or in-process, not the hub's tomcat. Messages stored in AWS SQS remain available if the hub stops. The hub processes those messages after it recovers. SHRINE 4.2 supports exactly one hub tomcat server; a shrine network can be no more reliable than the hub's tomcat. We hope to make the hub replicable in the future. Moving to AWS SQS now will make that transition more gentle.

To configure for AWS SQS see SHRINE 4.2.0 Chapter 8.2.1 - Configuring a Hub for AWS SQS .

Kafka

Shrine can also be configured to use Kafka. Using Kafka requires setting up, securing, and maintaining a cluster of Kafka servers - more admin expertise, attention, and expense than AWS SQS without the benefits of AWS SQS' reliability.

To configure for Kafka see SHRINE 4.2.0 Chapter 8.2.2 - Configuring a Hub for Kafka .

Create the Network Record and Queues

Finally use the shrineLifecycle tool to set up the network:

```
./shrineLifecycle createNetwork network.conf
```