2.0.0 Release Notes

Major New Features

Intermediate Progress for Queries

Researchers can observe their queries making progress from their local node to the hub, out to SHRINE's adapters and into the CRC. Researchers no longer need to take any action to retrieve QUEUED query results from the CRC; SHRINE's adapter now polls the CRC and proactively sends results back to the researcher's local node.

Major Security Enhancements

SHRINE no longer includes shrine-proxy.war which enabled several serious exploits.

SHRINE no longer requires a firewall hole to allow downstream nodes to act as a server for the hub. Downstream nodes no longer need to trust a client they do not control.

SHRINE no longer records the unobfuscated count result received from the CRC.

Network Health Web Page

SHRINE includes a new feature that periodically tests and supplies a simple report of network health.

Simpler Downstream Node Maintenance and Upgrades

Downstream node admins may now stop SHRINE for brief periods of time without missing queries from remote researchers. Those queries accumulate at the hub; the downstream node will be asked to run those queries when the node lets the hub know it is ready.

SHRINE's new internal data structures are versioned so that future upgrades may possibly be backward-compatible even if the contents of those structures changes. Future upgrades of SHRINE may not have to be coordinated across a whole network.

New Back-End Architecture for SHRINE

Most of these changes are enabled by a new back-end architecture for SHRINE based on Message-Oriented Middleware. This opens up many possible future enhancements for SHRINE.

Minor Changes

Removed shrine-metadata.war, steward.war, shrine-proxy.war, shrine-dashboard.war, and shrine-webclient.zip contents

Functionality in these .war files have been moved to shrine-api.war . (If your site does not support the ACT Aim 3 plugins, you may also remove shrine.war .) The javascript code now draws its configuration from shrine.conf instead of from several locations in javascript files. All javascript code now resides within the .war files that serve the application.

shouldQuerySelf Feature Removed

Find all the references to shouldQuerySelf in shrine.conf files and remove them.

If you really have an Adapter on the same node as the hub, add its adapter to the hub via a curl command or initial configuration - just like any other. (You will very likely have done this for the QEP at the hub already.)

Removed the "includeAggregateResults" setting

Remove the formerly required "includeAggregateResults = false" from shrine.conf .

No More Human-Readable node name property

The nodeKey property is used by SHRINE internally instead, and the node structure (stored at the hub) holds a real human-readable name with a good assortment of punctuation and spaces.

Remove the downstreamNodes section from the hub's shrine.conf

Remove the downstreamNodes section from the hub's shrine.conf. It is not used in SHRINE 2.0.0 and causes minor problems in the dashboard.

Database Changes

No need to specify a slick driver

Now the system can select the slick driver based on the shrineDatabaseType property. If the selected slick driver causes problems it is OK to use the old properties - which will override the one chosen via shrineDatabaseType.

Count-based Lockout Feature Removed

Remove adapterLockoutAttemptsThreshold from all shrine.conf files

Remove PRIVILEGED_USER tables and associated constraints and sequences from your database.

Drop the column of actual patient counts from the adapter's COUNT_RESULT table

SHRINE no longer needs to store the actual (and mildly sensitive) actual count of patients from the CRC in its COUNT_RESULT table. In mysql remove it from the shrine_query_history database with

```
ALTER TABLE COUNT_RESULT DROP COLUMN ORIGINAL_COUNT;
```

Drop the column of actual patient counts from the adapter's BREAKDOWN_RESULT table

SHRINE no longer needs to store the actual (and mildly sensitive) actual count of patients from the CRC in its BREAKDOWN_RESULT table. In mysql remove it from the shrine_query_history database with

```
ALTER TABLE BREAKDOWN_RESULT DROP COLUMN ORIGINAL_VALUE;
```

Add a status column for queries at the QEP

Add a status column to qepAuditDB's previousQueries table to support updates in query status with :

MySQL:

```
use qepAuditDB;
alter table `previousQueries` add column `status` VARCHAR(255) NOT NULL DEFAULT 'Before V26';
```

MS SQL:

```
use qepAuditDB;
alter table "previousQueries" add "status" VARCHAR(MAX) NOT NULL DEFAULT 'Before V26';
```

ORACLE:

```
use qepAuditDB;
alter table "previousQueries" add  "status" VARCHAR2(256) default 'Before V26' NOT NULL;
```

Added table QUERY_PROBLEM_DIGESTS

MySQL:

```
use qepAuditDB;
create table `QUERY_PROBLEM_DIGESTS` (`NETWORKQUERYID` BIGINT NOT NULL,`CODEC` TEXT NOT NULL,`STAMP` TEXT
NOT NULL,`SUMMARY` TEXT NOT NULL,`DESCRIPTION` TEXT NOT NULL,`DETAILS` TEXT NOT NULL,`CHANGEDATE` BIGINT NOT
NULL);
create index queryProblemsNetworkIdIndex on QUERY_PROBLEM_DIGESTS(NETWORKQUERYID);
```

MS SQL:

```
use qepAuditDB;
create table "QUERY_PROBLEM_DIGESTS" ("NETWORKQUERYID" BIGINT NOT NULL,"CODEC" VARCHAR(MAX) NOT NULL,"STAMP"
VARCHAR(MAX) NOT NULL,"SUMMARY" VARCHAR(MAX) NOT NULL,"DESCRIPTION" VARCHAR(MAX) NOT NULL,"DETAILS" VARCHAR
(MAX) NOT NULL,"CHANGEDATE" BIGINT NOT NULL);
create index "queryProblemsNetworkIdIndex" on "QUERY_PROBLEM_DIGESTS" ("NETWORKQUERYID");
```

ORACLE:

```
use qepAuditDB;
create table "QUERY_PROBLEM_DIGESTS" ("NETWORKQUERYID" NUMBER(19) NOT NULL,"CODEC" VARCHAR(256) NOT NULL,"
STAMP" VARCHAR(256) NOT NULL,"SUMMARY" CLOB NOT NULL,"DESCRIPTION" CLOB NOT NULL,"DETAILS" CLOB NOT NULL,"
CHANGEDATE" NUMBER(19) NOT NULL);
create index "queryProblemsNetworkIdIndex" on "QUERY_PROBLEM_DIGESTS" ("NETWORKQUERYID");
```

Added table RESULTS_OBSERVED

MySQL:

```
use qepAuditDB;
create table `RESULTS_OBSERVED` (`NETWORKQUERYID` BIGINT NOT NULL,`CHECKSUM` BIGINT NOT NULL,`OBSERVEDTIME`
BIGINT NOT NULL);
create index resultsObservedQueryIdIndex on RESULTS_OBSERVED(NETWORKQUERYID);
create index resultsObservedChecksumIndex on RESULTS_OBSERVED(CHECKSUM);
```

MS SQL:

```
use qepAuditDB;
create table `RESULTS_OBSERVED` (`NETWORKQUERYID` BIGINT NOT NULL,`CHECKSUM` BIGINT NOT NULL,`OBSERVEDTIME`
BIGINT NOT NULL);
create index "resultsObservedQueryIdIndex" on "RESULTS_OBSERVED" ("NETWORKQUERYID");
create index "resultsObservedChecksumIndex" on "RESULTS_OBSERVED" ("CHECKSUM");
```

ORACLE:

```
use qepAuditDB;
create table "RESULTS_OBSERVED" ("NETWORKQUERYID" NUMBER(19) NOT NULL,"CHECKSUM" NUMBER(19) NOT NULL,"
OBSERVEDTIME" NUMBER(19) NOT NULL);
create index "resultsObservedQueryIdIndex" on "RESULTS_OBSERVED" ("NETWORKQUERYID");
create index "resultsObservedChecksumIndex" on "RESULTS_OBSERVED" ("CHECKSUM");
```

Added table CONNECTIVITY_TEST_RESULTS to the hub's qepAuditDB

MySQL:

```
use qepAuditDB;create table `CONNECTIVITY_TEST_RESULTS` (
`NETWORKQUERYID` BIGINT NOT NULL,
`CHANGE_DATE` BIGINT NOT NULL
);

create index CONNECTIVITY_TEST_RESULTS_NETWORKQUERYID_INDEX on CONNECTIVITY_TEST_RESULTS (NETWORKQUERYID);
create index CONNECTIVITY_TEST_RESULTS_CHANGE_DATE_INDEX on CONNECTIVITY_TEST_RESULTS (CHANGE_DATE);
```

MS SQL:

```
use qepAuditDB;
create table "CONNECTIVITY_TEST_RESULTS" (
"NETWORKQUERYID" BIGINT NOT NULL,
"CHANGE_DATE" BIGINT NOT NULL
);

create index "CONNECTIVITY_TEST_RESULTS_NETWORKQUERYID_INDEX" on CONNECTIVITY_TEST_RESULTS
("NETWORKQUERYID");
create index "CONNECTIVITY_TEST_RESULTS_CHANGE_DATE_INDEX" on CONNECTIVITY_TEST_RESULTS ("CHANGE_DATE");
```

ORACLE:

```
use qepAuditDB;
create table "CONNECTIVITY_TEST_RESULTS" (
"NETWORKQUERYID"  NUMBER(19) NOT NULL,
"CHANGE_DATE"  NUMBER(19) NOT NULL
);

create index "CONNECTIVITY_TEST_RESULTS_NETWORKQUERYID_INDEX" on CONNECTIVITY_TEST_RESULTS
("NETWORKQUERYID");
create index "CONNECTIVITY_TEST_RESULTS_CHANGE_DATE_INDEX" on CONNECTIVITY_TEST_RESULTS ("CHANGE_DATE");
```

Remove the qep user from the PM cell, and corresponding bits from shrine.conf

SHRINE now accesses the DSA's database directly from inside shrine-api.war. It no longer needs a special qep user. Delete that user from the PM cell, and remove this (previously required) section from shrine.conf:

```
    //shrine-steward config
    shrineSteward {
      qepUserName = "qep"
      qepPassword = "changeit"
      stewardBaseUrl = "https://shrine-node1:6443"
    }
```

New Features / Enhancements

New Database Schema for Hub

SHRINE is storing all the data needed for network health at the hub, using a new schema, with a focus on json and on version compatibility. The new schema is in shrine-setup.zip, shrine-setup/hub/sql , with a .ddl for each database brand we test.

Install the shrine-api.war servlet

We moved a lot of functionality into this new servlet. Delete steward.war, shrine-dashboard.war, shrine-metadata.war, and shrine-proxy.war. Also delete the unpacked shrine-webclient (which used to be unzipped). If your site is not running the ACT Aim 3 plugins then your site no longer needs shrine.war.

Update the shrine.conf URL for messaging to use the shrine-api.war servlet

```
  messagequeue {
    blockingq {
      serverUrl = "https://shrine-dev1.catalyst:6443/shrine-api/mom"
    }
  }
```

Set properties needed on the Hub for running a connectivity test and sending emails when there are failures

```
networkhealth {
   email {
      networkName = "ACT Network"
      networkSignature = "ACT Network Operations Team"
      grantDescription = "Funded by the NIH National Center for Advancing Translational Sciences through its
Clinical and Translational Science Awards Program, grant number UL1 TR001857."
      helpContactName = "ACT JIRA Help Desk"
      helpContactEmail = "actnetwork@pitt.edu",
   }
   webpage {
      networkName = "ACT Network"
      webpageNote = "This is a note" //Optional alert message to display at the bottom of the network health
webpage
      alertMessage = "This is an alert message" //Optional alert message to display at the top of the
network health webpage
   }
   connectivityTest {
      queryId = "123456789" //The query Id of a previously run query that will be used to create a new query
for the connectivity test
      interval = "24 hours" //How often to run the connectivity test
      timeLimit = "10 minutes" //How long to wait for the query to finish running. If a node does not return
results, it will be marked as having an error.
      delay = "5 seconds" //The initial delay for starting the connectivity test
   }
}
```

The URL for the network health status webpage is available on the hub at https://shrine-dev1.catalyst:6443/shrine-api/networkhealth/webpage/index.html

Adapter controls CRC time limit to run a query

The shrine.adapter.crcRunQueryTimeLimit property in shrine.conf (default of 30 seconds) controls the time limit given to the CRC to either complete or queue the query.

Added a data service at the hub to gather and serve SHRINE's shared data structures

The hub now supports a shared data service. To reach the service (and get the config data needed to receive query results and queries) downstream nodes need to add the hub's base URL to their shrine.conf

```
shrine {
  hub {
    client {
      serverUrl = "https://shrine-hub:6443"
    }
  }
}
```

Part of the new data service allows a node to look up information about itself

Each node can ask the hub about itself and the hub. To ask about itself it needs a shared key with the hub. Add that to shrine.conf

```
shrine {
  nodeKey = "urlPathFriendlyNodeName"
}
```

The hub needs some initial information about the network, the hub node, and (optionally) other nodes

To supply that data service at start, the hub's shrine.conf needs some initial filling for those data structures. This information is only used if the hub does not have an initial configuration in its database.

```
shrine {
  hub {

  //This configuration is only used if no network is found in the hub's database
  //It is loaded into the database when the hub starts if no network is found.
  //To update the network or nodes, use the appropriate curl commands, not this config section.
  //In ifNoNetwork - nodes are added only if the node key does not already exist in the hub's database.
    ifNoNetwork {
      network {
        name = "Shrine Dev Test Network" //Name of your network
        hubQueueName = "hub" //queue used to send messages to the hub, different from the queue used to send
messages to a QEP and an adapter colocated with the hub
        adminEmail = "yourname@example.com" //email address for the hub admin to send network health test
failures
      }
      //Nodes in this network
      //In ifNoNetwork - nodes are added only if the node key does not already exist in the hub's database.
      nodes = [
        {
          name = "Shrine Dev1" //human-readable name for this node
          key = "shrineDev1" //machine-friendly key used to identify this node. Never change this.
          userDomainName = "shrine-dev1.catalyst" //domain name for users from this node.
          queueName = "shrineDev1" //queue used to send messages to the qep and adapter at this node. This
field is optional, defautls to the key if not specified
         sendQueries = "false" //true to send queries to an adapter at this node. An optional field, true by
default.
         adminEmail = "yourname@example.com" //email address for the node admin to send network health test
failures
        },
        {
          name = "shrine-dev2"
          key = "shrineDev2"
          userDomainName = "shrine-dev2.catalyst"
                  adminEmail = "yourname@example.com"
        }
      ]
    }
  }
}
```

Add and reconfigure nodes and some aspects of the network via web api commands to the hub

Use a curl command add a node to the hub:

```
curl -u username:password -w "\n %{response_code}\n" -k -X PUT "https://shrine-dev1.catalyst:6443/shrine-api
/hub/createNode/samHospital?nodeName=Mr%2E%20Sam%20Tsoi%27s%20Hospital&userDomain=tsoi%
27net&adminEmail=email@example.com"
```

Use a curl command to modify a node (and stop sending it queries):

```
curl -u username:password -w "\n %{response_code}" -k -X PUT https://shrine-dev1.catalyst:6443/shrine-api/hub
/modifyNode/shrine-dev2?sendQueries=false

curl -u username:password -w "\n %{response_code}" -k -X PUT "https://shrine-dev1.catalyst:6443/shrine-api
/hub/modifyNode/shrine-dev2?nodeName=Mr%2E%20Sam%20Tsoi%27s%20Hospital&sendQueries=true"
```

Use a curl command to modify the network:

```
curl -u username:password -w "\n %{response_code}" -k -X PUT "https://shrine-dev1.catalyst:6443/shrine-api
/hub/modifyNetwork?networkName=SHRINE%20Dev%20Network"
```

Configure the webclient in shrine.conf

```
 webclient {
    domain = "i2b2demo"
        name  = "SHRINE"
        siteAdminEmail = "email@foo"
    usernameLabel = "User Name"
    passwordLabel = "User Password"
        defaultNumberOfOntologyChildren = 10000  // the number of children to show when an ontology folder
is expanded.
    queryFlaggingInstructions = "Enter instructions for flagging queries here"
        flaggingPlaceholderText = "Enter placeholder text for the query flagging text input field"
        flaggingIconInstructions = "Enter text for when user mouses over the flagging information icon in
the header of the Query History here"
  }
```

The webclient has hard coded default values for the usernameLabel, passwordLabel and defaultNumberOfOntologyChildren fields if they are not configured, usernameLabel defaults to "SHRINE User", passwordLabel defaults to "SHRINE Password" and defaultNumberOfOntologyChildren defaults to 10000.

Redirect the old SHRINE webclient URL

The URL to the SHRINE webclient has changed from https://shrine-node-url:6443/shrine-webclient/ to https://shrine-node-url:6443/shrine-api/shrine-webclient/

To redirect the old URL and preserve existing bookmarks and history in browsers, specific a URL rewrite in Tomcat.

1. Add rewrite valve to conf/server.xml inside the Host configuration section:
<Host name="localhost" appBase="webapps"
        unpackWARs="true" autoDeploy="true">
<Valve className="org.apache.catalina.valves.rewrite.RewriteValve" />

2. Create a file name ***rewrite.config*** in conf/Catalina/localhost with the following contents:
RewriteRule ^/shrine-webclient /shrine-api/shrine-webclient

Move externalStewardBaseUrl to ${shrineHubBaseUrl}/shrine-api/shrine-steward

```
shrine {
...
  steward {
...
    emailDataSteward {
...
      externalStewardBaseUrl = ${shrineHubBaseUrl}/shrine-api/shrine-steward
...
```

Other Notes

Be sure to replace all references to the i2b2demo with your own user's domain

SHRINE identifies users via username:domain . If you use the default i2b2demo domain then SHRINE cannot identify where your researcher's queries are coming from. If you are running queries as a default user:domain then you will very likely see queries that you did not run, and others will be able to see your queries, too.

Configuration in i2b2_config_data.js

If the REST API is located on a different domain, then use the shrineUrl setting in i2b2_config_data.js.

i2b2_config_data.js

```
  {
         shrineUrl: "http://someotherdomain:6443/shrine-api/"
  }
```

In most instances the REST API will be on the same domain as the webclient and no configuration in i2b2_config_data.js is needed. However for backwards compatibility, the i2b2_config_data.js file must still exist as an empty object

i2b2_config_data.js

```
  {
  }
```

Supported Browsers

The Webclient supports Firefox, Chrome and Safari. For optimal user experience, we recommend Firefox version 66 or greater, Chrome version 74 or greater and Safari 12 or greater. We no longer support Internet Explorer or Microsoft Edge. If a user attempts to reach the Webclient with Explorer or Edge they will be presented with instructions and links to use a different browser.