Runtime Infrastructure - Feature #2124

improve/simplify the startup of server-side batch processes and appserver "agent" processes

04/16/2013 10:38 AM - Greg Shah

Status:	Closed	**Start date:**	05/27/2013
Priority:	Normal	**Due date:**	06/07/2013
Assignee:	Constantin Asofiei	**% Done:**	100%
Category:		**Estimated time:**	80.00 hours
Target version:	Cleanup and Stablization for Server Features		
billable:	No	**vendor_id:**	GCD

Description

Related issues:

Related to Base Language - Feature #1608: implement full appserver support (f...	**Closed**	**02/08/2013**	**05/24/2013**
Related to User Interface - Feature #1787: implement batch mode support	**Closed**	**06/10/2013**	**06/21/2013**
Blocks Runtime Infrastructure - Feature #2243: integrate spawn.c changes into...	**Closed**		

History

#1 - 04/16/2013 10:44 AM - Greg Shah

The core idea here is that the current setup and execution of batch programs is more cumbersome than needed. The requirement that all batch processes have certificates is a burden than is not needed. We won't remove the option of having a well secured environment, but rather we will make it an option instead of mandatory (as it is today). It should be possible to launch a batch client process with no special configuration. That will require the client driver to be updated to set batch mode, set the account context appropriately and to set the entry point.

Likewise, the appserver "agent" processes need to be as simple as possible to start. This should be very similar to the batch client startup, except that it probably needs to call into a pre-defined P2J entry point for appserver processing. The logic will then stay completely on the server except when a client-specific feature is used (e.g. a file is accessed, a process launched, a DLL entry point is called etc.).

#2 - 01/06/2014 01:18 PM - Greg Shah

- Target version changed from Milestone 7 to Milestone 11

#3 - 01/10/2014 09:35 AM - Greg Shah

- Estimated time changed from 60.00 to 80.00

- Assignee changed from Greg Shah to Constantin Asofiei

I have long been thinking on the details of this work. I will document a range of ideas here.

The ongoing effort to setup a functioning appserver test environment has raised my sense of the importance of this task. This work has also highlighted several areas which need attention, which I had not previously been considering.

1. Batch Mode for appservers

Now that we have good support for batch mode (#1787), all appserver agents should always be started with batch mode enabled.

2. Batch Processes Auto-Launching

It should be easy to configure any number of batch clients to be auto-launched from the server. This configuration should be in the directory, similar to the approach with appservers.

We don't want to remove the ability to manually launch a batch client. But we do want to easily setup a "real" production batch process environment completely from the server via the directory. Please consider that the customer Server project has some batch client requirements AND so does

MAJIC. It should be possible to setup both of these environments completely from the directory. No local file system configuration should be needed at all (e.g. client.xml, scripts, keystores).

3. Process Launching Approach

The work in #2201 to spawn full P2J ChUI client sessions is instructive. I am not sure that we need to use the full spawner approach, but there are probably big advantages to doing so in that the terminal setup/process environment and process parent hierarchy will be much more correct. So I would lean toward using the spawner. It will also make the code more consistent.

I wonder if some of the session cleanup problems we are seeing with the current appserver test environment would be eliminated in this case.

If used, this will be used for both the batch and appserver sessions.

4. Eliminate Remaining Local Configuration

The current test setup showed that some local files still had to be present (e.g. client.xml) for appservers to be launched. These need to be eliminated (hopefully this will be handled by the common code with batch process startup).

5. Good Default Configuration

Where feasible, I want the directory configuration to "default well". The idea here is to minimize the directory configuration needed for the common case. This would apply to both batch and appserver clients.

6. Security

We should be using the secure port by default for all batch and appserver sessions. I am OK with leaving a way for the insecure port to be configured for use, but I don't want it to be the default.

The work in #2201 to spawn full P2J ChUI client sessions is instructive. A technique was created there to automatically establish a P2J session in a reasonably secure manner. It is important that we do not open security holes here. Let's make sure that every batch or appserver process we launch is well authenticated.

I would also like to consider a scenario where we auto-generate the necessary certificates/keystores at server startup if such are not manually configured. The certificate management is one of the worst parts of the setup and maintenance of the P2J server and it will be a rare case where the customer will want to have proper certificates (instead of self-signed). With the addition of auto-authenticated and auto-started batch/appserver processes, it seems like there is no remaining need to force all this cert/keystore stuff to be done statically/manually. The bottom line: starting a P2J server in secure mode should be easy and one should not have to even think about it, unless you have special requirements.

7. Scheduler

To fully replace most batch environments, we will have to have a scheduler or a way to schedule the auto-launch of a batch process. For example, in MAJIC there are several batch processes that are run only at a specific time/day using cron.

My inclination would be to create a generic scheduler that is built-in to the P2J server and configured via the directory. Then make sure that the batch processes can easily be launched by such a mechanism.

For now, I am only concerned with recurring scheduled events but I imagine in the future that it would be extended to support one-time events too.

Providing a cron-like ability to define recurring events should work fine. This means minute, hour, day of month, month and day of week options.

#4 - 01/20/2014 09:19 AM - Constantin Asofiei

- Status changed from New to WIP

#5 - 02/03/2014 03:53 PM - Constantin Asofiei

Greg, a question about launching a process using the spawner. Currently, the spawn application requires as input valid user/pass for the system where the server is ran. This will require the system credentials (user/password) to be saved somewhere in the directory, if we are to use the spawn application for launching a process. No matter how they are saved in the directory (plain text, hex-coded, encrypted), at some point they will need to be decoded, at least in server's memory. Is this an acceptable approach? Else, we need to choose from:

- change spawn to bypass the system credentials check and just start a process using the same account as the server.
- if possible (maybe by giving the spawner app higher privileges), start the process under the given username, without requiring a password (something like executing a command using sudo su -c "command" user).
- something else?

#6 - 02/03/2014 05:50 PM - Greg Shah

Yes, I understand the problem.

> if possible (maybe by giving the spawner app higher privileges), start the process under the given username, without requiring a password (something like executing a command using sudo su -c "command" user)

This is probably the right approach. As far as I know, this is already something that can be done because spawn already has root privileges. We don't want to disable the ability for spawn to do the full login processing, because that is an important part of securing access to the web client. But we could make this an optional feature that can be invoked by the server when configured to do so.

Questions:

1. How do we differentiate between these 2 modes at runtime?

2. I am worried about the security implications. Right now we chown root:root and chmod 4755 which means that all users on the system can launch spawn. Can we figure out a way to protect this special mode such that only the P2J server can invoke it?

#7 - 02/04/2014 02:20 AM - Marius Gligor

1. The credentials used by spawner are user/password of the user account, that is, the user must have an account.
The password is never decrypted (see the code) instead the provided password is encoded and compared.
Using PAM the authentication is done by the PAM plug-in.

2. The spawner is used only by server to lunch the client so we could "hide" the spawner form the user perspective.
Another improvement that we can do is to keep the \n when we read the password.
Doing so the spawner cannot be used from command line because when the password is provided via terminal input the \n is also part of the string entered by user but is not part of the real password and the authentication fails.

#8 - 02/04/2014 02:25 AM - Marius Gligor

Marius Gligor wrote:

> 1. The credentials used by spawner are user/password of the user account, that is, the user must have an account.
> The password is never decrypted (see the code) instead the provided password is encoded and compared.
> Using PAM the authentication is done by the PAM plug-in.

The authentication has nothings to do with server directory. The credentials are never stored in server directory.

> 2. The spawner is used only by server to lunch the client so we could "hide" the spawner form the user perspective.
> Another improvement that we can do is to keep the \n when we read the password.
> Doing so the spawner cannot be used from command line because when the password is provided via terminal
> input the \n is also part of the string entered by user but is not part of the real password and the authentication fails.

#9 - 02/04/2014 04:00 AM - Constantin Asofiei

Marius Gligor wrote:

> 1. The credentials used by spawner are user/password of the user account, that is, the user must have an account.
> The password is never decrypted (see the code) instead the provided password is encoded and compared.
> Using PAM the authentication is done by the PAM plug-in.

I'm aware of this. But there is another usage for the spawner: we need to let the server automatically start P2J processes (these are special P2J clients which always authenticate using a certificate). In this case, there is no user interaction, thus we need another way of launching a system process under the desired user, without knowing the user's password.

Do you have any suggestions how we can do this in spawn.c? As Greg noted, we need a secure way of launching a process under a user without access to the user's password.

#10 - 02/04/2014 07:41 AM - Greg Shah

> Another improvement that we can do is to keep the \n when we read the password.
> Doing so the spawner cannot be used from command line because when the password is provided via terminal input the \n is also part of the string entered by user but is not part of the real password and the authentication fails.

I don't think we can use this trick. A user could redirect STDIN or write a program to execute spawn just like we do.

#11 - 02/04/2014 08:43 AM - Greg Shah

> we need a secure way of launching a process under a user without access to the user's password

How about this idea:

JNI is a bidirectional interface. It is most often thought of as a way for Java code to call native code. But in fact, it is easily possible for native code to call Java.

It is also possible for native code to use the JNI interfaces to create a new JVM instance (inside the native program's process). Once it has loaded the JVM in-process, it can then call arbitrary Java code that is hosted in that JVM.

It seems to me that instead of driving spawn using STDIO, we can invert the processing:

0. As part of the installation/setup of spawn, we would have a keystore in the file system. That keystore would have the certifcate(s) necessary to authenticate an SSL session with one or more P2J servers. Of utmost importance is that this keystore must NOT be "world-readable" (or writable). In other words, it would be chmod 0440 and chown root:root. We will call this the "private spawn keystore".

1. The P2J server (who has the private key that matches at least one of the certificates in the private spawn keystore) would launch spawn. Instead of providing the local OS authentication credentials (userid/password), it would provide the minimum details needed to establish a secure connection BACK to the P2J server.

2. spawn uses JNI to create a new JVM in-process. That JVM would have the p2j.jar in its classpath.

3. spawn would to execute code in the P2J jar. This would be a Java class (an in-process P2J client) that can create a temporary secure session using the "callback" details provided via STDIO. Those details would have to include the P2J authentication info and also the details about how to contact the P2J server, so that the proper bootstrap config could be constructed.

4. As part of this connection it is ESSENTIAL that client-side SSL validation be enabled. This would utilize the cert in the private spawn keystore to validate that the server it is connecting to is in fact in possession of the private key that created the certificate. Any failure here would cause spawn to exit.

5. If it gets this far, then the request is deemed legitimate. The in-process P2J client would use RemoteObject to call the P2J server and obtain instructions for what to do next. Once it obtains the instructions, the in-process P2J client would disconnect from the P2J server and allow spawn to continue.

6. Based on the response, spawn can drive an su path (no authentication) or a login path (a "remote" interactive authentication like we do today).

I believe that to breach this approach, one would have to somehow breach the security of the keystore, which means that there must be some other kind of root access (outside of spawn itself). This is as secure as anything we already do.

#12 - 02/04/2014 09:45 AM - Constantin Asofiei

Greg, another issue: to automatically launch processes via the scheduler, I want to be able to read the various configurations from server/{ <server-id> | default }/runtime/{<account-id> | default}/<id> node and fallback to server/{ <server-id> | default }/<id> node if not defined per-user. This will allow us to have global values (defined at the server) and also be able to have account-specific values (like dedicated working dir, dedicated key store, etc)

Currently, Utils.getDirectoryNode APIs do not allow for this kind of search: this is either per-server or per-account. I just want to know your opinion before starting making changes into this code.

#13 - 02/04/2014 09:47 AM - Greg Shah

> Currently, Utils.getDirectoryNode APIs do not allow for this kind of search: this is either per-server or per-account. I just want to know your opinion before starting making changes into this code.

It's a good idea. Go with it.

#14 - 02/06/2014 11:11 AM - Constantin Asofiei

Greg, about the security configuration (keystores, truststores, etc):

1. do we need to provide a way to read into the directory an existing configuration? I.e. for MAJIC, it will read the client.xml and .store files for each process and load them into the directory. Also for server configuration: it will read the server.xml files and load them in the directory (plus the trust-store).
2. do we need to provide a way to persist the process configuration (key store, passwords, etc) and trust store from the directory to client.xml and .store files, for each process?

Also, let me know if this policy is OK:

1. on server startup, if the server.xml file has security configuration (i.e. truststore, passwords, etc), these will have precedence over any server configuration from directory.
2. on process startup, if the client.xml has security configuration (i.e. keystore, truststore, alias, etc), these will have precedence over any configuration incoming from the server.

#15 - 02/06/2014 11:14 AM - Greg Shah

> do we need to provide a way to read into the directory an existing configuration? I.e. for MAJIC, it will read the client.xml and .store files for each process and load them into the directory. Also for server configuration: it will read the server.xml files and load them in the directory (plus the trust-store).

No.

do we need to provide a way to persist the process configuration (key store, passwords, etc) and trust store from the directory to client.xml and .store files, for each process?

No.

on server startup, if the server.xml file has security configuration (i.e. truststore, passwords, etc), these will have precedence over any server configuration from directory.

Yes.

on process startup, if the client.xml has security configuration (i.e. keystore, truststore, alias, etc), these will have precedence over any configuration incoming from the server.

Yes.

#16 - 02/06/2014 01:16 PM - Constantin Asofiei

- File ca_upd20140206b.zip added

Attached update contains the scheduler support, used to launch the appserver and batch processes. The spawner infrastructure was refactored to allow batch processes to be started too:

- core classes for configuring the spawned client are ClientBuilder, ClientBuilderOptions, ClientBuilderParameters, ClientSpawner
- core classes to spawn the client are: Spawner, SpawnerImpl, SpawnerListener, TemporaryClient, TemporaryAccount*.
- implementation to spawn a chui web client: WebClientBuilder, WebClientBuilderOptions, WebClientBuilderParameters, WebClientSpawner. An important note is that Jetty threads have no P2J context: thus it can not call code which relies on P2J context (i.e. reading directory configurations or something else). For this reason, the web socket timeout is read by the server and sent via the client:web:socketTimeout argument for the P2J client.
- implementation to spawn a P2J process: ProcessBuilderOptions, ProcessClientBuilder, ProcessClientSpawner. The important note here is that, when a Job starts a P2J process, it must use the process subject ID to search the directory. As the job is being executed in a thread using the server's context, ProcessClientBuilder will compute and pass to the Utils.getDirectoryNode APIs an explicit list of subject IDs for the target process.

The scheduler infrastructure is composed of the following classes:

- Job, JobMode, JobType - job configuration
- Scheduler - the scheduler implementation. Its javadoc and Job's javadoc provide details about how these are configured and how they work. The most important feature is that although the Java timer is used for scheduling, each task must end up being executed in a thead with the P2J server's context. For this reason, once a job is fired and caught by the timer, it will be queued to the Scheduler$JobProcessor, which will execute it using the server's context.

Starting a process: ClientCore was changed so that the custom work needed to be done (using the temporary credentials) is hidden. Thus, spawner.getTemporaryClient is used to retrieve a TemporaryClient implementation from the server and do the custom work. For process case, the ProcessClientSpawner$TemporaryClientTask class implements this work. When launching a P2J process, currently I'm stuck with passing the credentials for a machine user: see the spawner.spawn("bogus", "bogus") calls in Scheduler$ProcessLaunchTask and AppServerLauncher.launcherWorker. Once we change the native spawn code to receive only a machine user name, these will be changed to pass the process name as the user name.

Starting an appserver: the policy was changed, so that only P2J processes can be associated with an appserver. Info related to starting the P2J client was moved to clientConfig nodes.

Starting a web chui client: the custom work done in ClientCore was moved to the WebClientSpawner.TemporaryClientTask class.

Client configuration on server-side: there are clientConfig nodes which can be defined at the server or per P2J process. The analog information from the chuiWeb node was removed. clientConfig nodes look like:

```
        <node class="container" name="clientConfig">
          <node class="boolean" name="secure">
            <node-attribute name="value" value="TRUE"/>
          </node>
          <node class="string" name="spawner">
            <node-attribute name="value" value="p2j/build/native/spawn"/>
          </node>
          <node class="string" name="jvmArgs">
            <node-attribute name="value" value="-Xmx512m -XX:MaxPermSize=64m -Djava.awt.headless=true -Xdebug
-Xnoagent -Djava.compiler=NONE"/>
          </node>
          <node class="string" name="classpath">
            <node-attribute name="value" value="<classpath"/>
          </node>
          <node class="string" name="libPath">
            <node-attribute name="value" value="p2j/build/lib"/>
          </node>
          <node class="string" name="workingDir">
            <node-attribute name="value" value="testcases/simple/client"/>
          </node>
          <node class="string" name="configFile">
            <node-attribute name="value" value="client_tty.xml"/>
          </node>
        </node>
```

The configuration is inherited from the per-server down to the per-account nodes. A config set per-account will have priority over the per-server config (see the ProcessBuilderOptions javadoc).

The configuration for the temporary account pool is common for both chui web clients and P2J processes. Thus, this can not be set neither at the clientConfig or chuiWeb nodes. Instead, they are read from the tempClient/poolSize and tempClient/timeout nodes, searching by server's ID.

Marius: please take a look and let me know if I've missed any chui web code, during refactoring.

Greg: please review. The code for directory auto-configuration will be posted tomorrow. Next on track will be changing the native spawner code as you sugested on note ~~3~~ 11.

#17 - 02/06/2014 07:47 PM - Constantin Asofiei

I've been trying not to add more external dependencies to the project and use what J2SE provides, but I couldn't find a way without linking to sun proprietary APIs. The solution in the end was to use Bouncy Castle. The license is here: http://www.bouncycastle.org/licence.html and the needed jars are bcprov-jdk15on-150.jar and bcpkix-jdk15on-150.jar from this page: https://www.bouncycastle.org/latest_releases.html

#18 - 02/07/2014 04:08 AM - Marius Gligor

I saw that is a major code refactoring. As I understood the chuiWeb node will be replaced by a global node and per client nodes. However I attached a complete example of a chuiWeb node. The values are read from directory when server start-up and are stored inside the ClientBuilderOptions class.

```
<node class="container" name="chuiWeb">
  <node class="boolean" name="enabled">
    <node-attribute name="value" value="TRUE"/>
  </node>
  <node class="boolean" name="secure">
    <node-attribute name="value" value="TRUE"/>
  </node>
  <node class="string" name="spawner">
    <node-attribute name="value" value="/home/mag/p2j/build/lib/spawn"/>
  </node>
  <node class="string" name="command">
    <node-attribute name="value" value="java"/>
  </node>
  <node class="integer" name="poolSize">
    <node-attribute name="value" value="8"/>
  </node>
  <node class="integer" name="timeout">
    <node-attribute name="value" value="30000"/>
  </node>
  <node class="integer" name="WebSocketTimeout">
    <node-attribute name="value" value="-1"/>
  </node>
  <node class="integer" name="watchdogTimeout">
    <node-attribute name="value" value="300000"/>
  </node>
  <node class="string" name="jvmArgs">
    <node-attribute name="value" value="-Xmx512m -XX:MaxPermSize=64m -Djava.awt.headless=true"/>
  </node>
  <node class="string" name="classpath">
    <node-attribute name="value" value="/home/mag/p2j/build/lib/p2j.jar"/>
  </node>
  <node class="string" name="libPath">
    <node-attribute name="value" value="/home/mag/p2j/build/lib"/>
  </node>
  <node class="string" name="workingDir">
    <node-attribute name="value" value="./"/>
  </node>
  <node class="string" name="configFile">
    <node-attribute name="value" value="config.xml"/>
  </node>
  <node class="integer" name="rows">
    <node-attribute name="value" value="24"/>
  </node>
  <node class="integer" name="columns">
    <node-attribute name="value" value="80"/>
  </node>
  <node class="string" name="background">
    <node-attribute name="value" value="0x000000"/>
  </node>
  <node class="string" name="foreground">
    <node-attribute name="value" value="0xFFA500"/>
  </node>
  <node class="string" name="selection">
    <node-attribute name="value" value="0x0000FF"/>
  </node>
  <node class="string" name="fontname">
    <node-attribute name="value" value="monospaced"/>
  </node>
  <node class="integer" name="fontsize">
    <node-attribute name="value" value="18"/>
  </node>
```

```
</node>
```

#19 - 02/07/2014 07:58 AM - Greg Shah

The use of BouncyCastle is OK.

#20 - 02/07/2014 04:44 PM - Constantin Asofiei

Greg Shah wrote:

> The use of BouncyCastle is OK.

OK, Bouncy Castle is limited to only one class and is hidden behind an interface - thus if at some point we need to move to another library, we can do so easily.

Certificates and private keys are being generated nicely, P2J process and web chui clients can connect without having access to the external key store, but I have some questions/notes:

- all private keys are saved under the /security/certificates/private-keys/<alias> nodes.
- keys are hard coded to 1024-bytes - should we make this configurable?
- another cipher parameter - the exponent - was hard coded to 65537 (this is what openssl uses). Bouncy Castle allows BigInteger exponents, and it was a PITA to find why firefox would not accept the certificate sent by the server.
- the directory reconfiguration can be done either via a server startup argument ("-c") or via an utilitity.
- in the directory each private key is encrypted using its own random password (which is saved as a "bytes" node). Do we need to enhance this, i.e. use an external password for all of them?
- the root CA private key is saved too in the directory, but this one can be deleted/removed safely: instead of signing a new certificate by hand using the root CA, is best to just reconfigure all of them. I've added a warning at the utility which generates this.
- processes can be started via the scheduler (which, after it will be integrated with the admin console, an admin will be able to start them by hand). But, at this time, there is no way of manually starting a process which has its private key in the directory (unless we move the private key into an external key store). Thus, I'm thinking to add a ServerDriver parameter (i.e. "-b <process>") to start a process.

I will post the update after I merge with 10459 and finish some javadoc.

#21 - 02/07/2014 06:52 PM - Constantin Asofiei

- File ca_upd20140207e.zip added

This adds the SSL certificate generation into the directory and the support to use this data to start processes / web chui clients. Is merged with 10459.

Marius: please review, and make sure I got the logic in 10459 right (this was not actually a merge, but more a "put the logic in the right place").

#22 - 02/10/2014 07:48 AM - Constantin Asofiei

Greg Shah wrote:

> 0. As part of the installation/setup of spawn, we would have a keystore in the file system. That keystore would have the certifcate(s) necessary to authenticate an SSL session with one or more P2J servers. Of utmost importance is that this keystore must NOT be "world-readable" (or writable). In other words, it would be chmod 0440 and chown root:root. We will call this the "private spawn keystore".

I don't think we should keep 4755 permissions for the current spawn tool. spawn should be limited to 0750 and we should have another 4755 tool (spawn_jvm ?) which will be used to create the new JVM and connect to the server. After this P2J client receives the command from the P2J server, it will invoke spawn to perform the authentication and process launching; as the P2J client is already with root privileges, there is no need to allow setuid for spawn tool.

Also, about the p2j.jar and @srv-certs.store files: these two should reside in the current folder as the spawn_jvm and spawn tools, both with 0550 permissions. Should the srv-certs.store file (the key store with the server certificates) be encrypted? If yes, I guess we could have a srv-certs.properties file which contains the key store password.

#23 - 02/10/2014 12:23 PM - Constantin Asofiei

Greg, a question: when using JNI_CreateJavaVM to create a new JVM instance, the compiled code needs to link with the libjvm.so library. If this is not in the LIBRARY_PATH, we need to hard-code the location of this library in the makefile script, to be able to build the source. Also, to be able to run the app, libjvm.so needs to be in the LD_LIBRARY_PATH. Is there a better way to link to libjvm.so?

#24 - 02/10/2014 12:42 PM - Greg Shah

We should not hard code the paths into the makefile. It is best to require that to be defined as a parameter on the ant command line. If it is not present, we can bypass that part of the build. I don't want to pull in new dependencies to every P2J build, but it should still be easy to include this.

#25 - 02/11/2014 11:40 AM - Greg Shah

> keys are hard coded to 1024-bytes - should we make this configurable?

Yes. Default to 1024 but allow other sizes to be configured.

> another cipher parameter - the exponent - was hard coded to 65537 (this is what openssl uses). Bouncy Castle allows BigInteger exponents, and it was a PITA to find why firefox would not accept the certificate sent by the server.

Nasty. I assume this is configurable now too?

> in the directory each private key is encrypted using its own random password (which is saved as a "bytes" node). Do we need to enhance this, i.e. use an external password for all of them?

Yes, we should at least provide an option to enable this. The default can be the current approach. But it does make sense to allow an external password to be stored separately (bootstrap config) or passed by command line parameter. We already support encrypted bootstrap config files, so that approach would work for some use cases.

> Thus, I'm thinking to add a ServerDriver parameter (i.e. "-b <process>") to start a process.

I like this idea. For most use cases, it is much better than having configuration in the file system.

> I don't think we should keep 4755 permissions for the current spawn tool. spawn should be limited to 0750 and we should have another 4755 tool (spawn_jvm ?) which will be used to create the new JVM and connect to the server. After this P2J client receives the command from the P2J server, it will invoke spawn to perform the authentication and process launching; as the P2J client is already with root privileges, there is no need to allow setuid for spawn tool.

Is this approach only being suggested for appserver/batch clients OR do you propose this as suitable for web clients too?

> Also, about the p2j.jar and @srv-certs.store files: these two should reside in the current folder as the spawn_jvm and spawn tools, both with 0550 permissions.

Yes, this makes sense.

> Should the srv-certs.store file (the key store with the server certificates) be encrypted? If yes, I guess we could have a srv-certs.properties file which contains the key store password.

I don't think this needs to be encrypted. This is just a public key, not anything that is sensitive. The trick here is that while anybody might be able to read it, only those than can change it can control the authentication/validation process.

#26 - 02/11/2014 12:08 PM - Constantin Asofiei

Greg Shah wrote:

> keys are hard coded to 1024-bytes - should we make this configurable?

Yes. Default to 1024 but allow other sizes to be configured.

OK

> another cipher parameter - the exponent - was hard coded to 65537 (this is what openssl uses). Bouncy Castle allows BigInteger exponents, and it was a PITA to find why firefox would not accept the certificate sent by the server.

Nasty. I assume this is configurable now too?

I can make it configurable, but wrong value will lead to weird errors. Originally I had the exponent set to something random and big (as BC suggests, an odd number with ~100 digits or so), but firefox did not want to accept the connection; it was very confusing because P2J secure connections worked with no problems, using the same keys. I guess being big numbers and all, there might be limitations in 3rd party software, related to big number arithmetic.

> in the directory each private key is encrypted using its own random password (which is saved as a "bytes" node). Do we need to enhance this, i.e. use an external password for all of them?

Yes, we should at least provide an option to enable this. The default can be the current approach. But it does make sense to allow an external password to be stored separately (bootstrap config) or passed by command line parameter. We already support encrypted bootstrap config files, so that approach would work for some use cases.

OK

> Thus, I'm thinking to add a ServerDriver parameter (i.e. "-b <process>") to start a process.

I like this idea. For most use cases, it is much better than having configuration in the file system.

OK

> I don't think we should keep 4755 permissions for the current spawn tool. spawn should be limited to 0750 and we should have another 4755 tool (spawn_jvm ?) which will be used to create the new JVM and connect to the server. After this P2J client receives the command from the P2J server, it will invoke spawn to perform the authentication and process launching; as the P2J client is already with root privileges, there is no need to allow setuid for spawn tool.

Is this approach only being suggested for appserver/batch clients OR do you propose this as suitable for web clients too?

I want to have a central place to spawn P2J clients, regardless of their type, as this will allow us to restrict the spawn tool only to root. So, this will be common for all type of P2J clients started by the server.

> Also, about the p2j.jar and @srv-certs.store files: these two should reside in the current folder as the spawn_jvm and spawn tools, both with 0550 permissions.

> Yes, this makes sense.

OK

> Should the srv-certs.store file (the key store with the server certificates) be encrypted? If yes, I guess we could have a srv-certs.properties file which contains the key store password.

> I don't think this needs to be encrypted. This is just a public key, not anything that is sensitive. The trick here is that while anybody might be able to read it, only those than can change it can control the authentication/validation process.

Actually, I think only the private keys in a store will be encrypted with the store's password. The public certificates from it can be read without a password. But when saving the store to disc, a password needs to be provided, even if it is not needed (i.e. the store has no private keys).

Finally, something about the in-process P2J client authentication. The steps currently are:

- the server invokes spawn_jvm with these parameters:
  - secure-port - the secure port number
  - alias - the server's alias, an alias in the srv-certs.store file
  - uuid - the UUID to identify this request.
  - the host is not required, is hardcoded to localhost
- spawn_jvm invokes a NativeSecureConnection.connect. This in turn establishes a secure P2J connection to the P2J server.
- this P2J client still needs to authenticate. I'm inclined to use P2J process-like authentication, but this requires the private key to be known by spawn_jvm. We can use:
  - its own private key stored on disk to authenticate as a P2J process
  - we can expose the server's private key to spawn_jvm in a srv-keys.store, so we can authenticate as the server which invoked it.
  - we can authenticate using temporary credentials
- after authentication, it resolves a RemoteSpawner proxy and will invoke RemoteSpawner.useCommand(uuid) to resolve a SpawnerCommand POJO with the command details to be passed to spawn:
  - the environment variables (i.e. temporary credentials) needed by spawn
  - the arguments to be passed to the spawn tool
  - the user's password on this machine, if we are in a password-base authentication (i.e. web chui client)
  - note that for P2J processes, currently I assume that there is an account on the machine with the username the same as the process' name. Should we make this configurable? I'm thinking about something like: all P2J process have the same working dir/credentials on the machine.

#27 - 02/11/2014 04:14 PM - Greg Shah

Code Review 0207e

Sorry, it took me a while to digest this large update. It is quite good. It seems to me that the refactoring has resulted in a cleaner and more extensible design for both the web clients and for appserver launching. Some thoughts:

1. In RandomWordGenerator, please set GEN_LENGTH to 36, unless this makes the performance unusable.

2. BootstrapConfig.copyConfigItems() probably should have some safety logic to protect against null elements in items[] and more importantly, for missing or malformed strings that result in names[] having length != 3 or having empty string as an element.

3. In WebChuiSimulator.onConnect(), why was the this added on line 116? I don't see the conflict. And if it is needed, why not add it to line 118 too?

4. Use asterisk for new imports in SecurityManager, ServerKeyStore.

5. SecurityManager.transSec can be null, but in getEncryptedTrustStore() it is used without any safety code.

6. For SSLCertGenUtil, we should probably allow the user to provide a root CA password. If not provided, we generate it and write it to the console instead of storing it in the directory unencrypted. That way, the admin has the password and can record it in case they want to modify things later, but it is never stored unencrypted in our files.

7. The SSLCertGenUtil.main() syntax help refers to LoadCert.

8. Do I understand correctly that our BCCertFactory uses SHA1 hashing? Security best practice at this time is to eliminate use of SHA1 and use nothing less than SHA2 (e.g. SHA-256) or better.

9. How can we ensure that the AES key-length meets some minimum standard (256 bit if possible)?

10. ProcessClientBuilder module name is incorrect in the header. The javadoc is not quite right either.

11. It is worth making a new package for the scheduler-related classes (com/goldencode/p2j/scheduler/). Please also move the inner classes of Scheduler into separate classes in this package.

12. I think the scheduler only supports fixed time values. Some of the normal cron constructs can be encoded. cron allows asterisk for (all possible values), a comma-separated list of values and a hyphen for a range of values. These allow a single job definition to be used much more flexibly. For example, by using 15,45 for minutes, you can have the job executed twice per hour at the quarter before and quarter after OR if you use asterisk for the minute or hour or day, you can execute a job once every minute, hour or day. I prefer to allow these use cases now otherwise we may have customers that still need to use an external cron.

13. It isn't clear to me what we do to implement auto-starting of appserver clients now. Isn't that a feature in the 4GL (that has now been removed)?

#28 - 02/11/2014 04:37 PM - Greg Shah

> I want to have a central place to spawn P2J clients, regardless of their type, as this will allow us to restrict the spawn tool only to root. So, this will be common for all type of P2J clients started by the server.

I agree with your intention. My only concern is that the extra connection to the server for web clients will add overhead (it may hurt scalability) and it will also slow the establishment of each web client. Can we use the same approach but still bypass the extra connection when we have OS user/pw credentials to process? In that case we can allow the OS security to work as it does today.

> the host is not required, is hardcoded to localhost

In the future, we will enable this on remote systems as well. Just as in the 4GL, appserver agents can be on a different system than the broker and database.

If it can be enabled without much effort it is best to do it now.

> this P2J client still needs to authenticate
> ...
> we can authenticate using temporary credentials

I prefer the temporary credentials approach.

> I assume that there is an account on the machine with the username the same as the process' name. Should we make this configurable? I'm thinking about something like: all P2J process have the same working dir/credentials on the machine.

Yes, make it configurable. Using the same OS name as P2J acct name is a good default. But an alternate approach is definitely important.

#29 - 02/12/2014 03:40 AM - Constantin Asofiei

Greg Shah wrote:

> I want to have a central place to spawn P2J clients, regardless of their type, as this will allow us to restrict the spawn tool only to root. So, this will be common for all type of P2J clients started by the server.

I agree with your intention. My only concern is that the extra connection to the server for web clients will add overhead (it may hurt scalability) and it will also slow the establishment of each web client. Can we use the same approach but still bypass the extra connection when we have OS user/pw credentials to process? In that case we can allow the OS security to work as it does today.

OK, I keep forgetting that spawn starts the process using the specific user, not root. So its safe to allow anyone to use the tool, as the executed command will be ran using the user's credentials. I'm moving into merging spawn and spawn_jvm into a single spawn tool and I'll make sure to bypass the extra connection, if explicit password is provided.

#30 - 02/12/2014 01:34 PM - Constantin Asofiei

- File ca_upd20140212e.zip added

Greg, about your review notes:
A. from note 26:
1. the private key strength: 1024 bytes or better is enforced and is configurable in the /security/certificates/company/rsaKeyStrength.

2. the public exponent: is configurable, defaults to 65537 (a Fermat prime), and is configurable in the /security/certificates/company/rsaPublicExponent node. Wrong value can make the private key vulnerable, and this value needs to be chosen very carefully.

3. private keys saved in the directory:
- added option to use single encryption password for all private keys (except root CA's). This can be set via access:password:masterkeyentry config.
- added option to configure the encryption password for each alias, via access:password:keyentry-<alias> config. This allows us to externalize the encryption passwords to external xml file.

4. batch processes: added -b process to start a batch process using the ServerDriver.

B. from note 27:

> It seems to me that the refactoring has resulted in a cleaner and more extensible design for both the web clients and for appserver launching.

Yes, this was my goal: to not restrict the spawn tool just to web chui or processes.

3. it was left behind by me, after I done some changes to that method and reverted them.

6. I don't think we should allow the user to provide the password. For AES encryption, there are constraints about the encryption key to be of 128, 192 or 256 bit length; so, I think is best to generate it randomly and write it to standard output. The encryption key will not be saved to the directory for the root CA, but it can be specified via bootstrap config.

8. Yes, that was what openssl was using by default. I've switched to SHA256WithRSA now.

9. The enforcement for all AES encryption keys is to be either 128, 192 or 256 bits: these are the only lengths accepted by Bouncy Castle AES implementation.

11. the scheduler classes are left in the same package, to ease review. I'll move them tomorrow.

12. Currently, only the minute is mandatory: the others are optional. To provide full cron-style support for job scheduling, I think we should move to string encoding of the cron expression. Also, I'll try to find an external library which gets the next execution time from a current time and a cron expression, as I don't want to reinvent the wheel.

13. auto-starting of appserver clients is done by scheduling a job with mode NOW. At server startup, all jobs marked as NOW are being executed immediately.

C. Misc about the spawn tool:

- has 1 <user> <workdir> <command> [args] syntax for password authentication (works as it used to work for web chui clients).
- has 0 <secure-port> <host> <server-alias> <uuid> syntax for P2J secure connection authentication. This uses temporary credentials to connect to the remote P2J server and get the command associated with the given UUID.
- spawn is built by default by the native task and now it needs libjvm.so to compile and run (it will not even start without libjvm.so in path). The quick solution is to symlink /usr/lib/libjvm.so to the proper lib; the alternative is to use #ifdef directives to exclude the JNI code during compilation, but in this case, the P2J secure authentication will not be possible.
- I still need to write the windows native code for the spawn tool. Marius: please review spawn.c and let me know if you see any problems.

D. Other misc:

- configclient/appArgs can be used to pass application arguments to the P2J Client and configClient/cfgOverrides can be used to override the P2J Client bootstrap configuration. Both will be added to the P2J client spawn command.
- the configClient/systemUser node can be used to provide an explicit system user name.
- the watchdog doesn't seem to do its job... the P2J client doesn't terminate if I close the browser tab in firefox. Am I missing something or there are still TODOs with the watchdog?

The attached 0212e.zip update is missing this:

- scheduler changes per note 26/11
- Windows version of the spawn.c changes

#31 - 02/12/2014 07:06 PM - Greg Shah

Code Review 0212e

I am OK with the changes.

My only concern is that the server is very tightly coupled to the command line generation for the client. This means that later, when we enable a

remote launcher, it will be more work. I don't want to change this right now, I just want to highlight the issue.

The src/native/ changes are also missing header entries.

The JNI code in spawn seems OK, although I do get nervous when there are malloc() calls without matching free() calls. I think it is OK because the parent process which does the allocations will exit quickly. If we ever keep that process around, we will want to clean things up. It may make sense to clean up anyway, so that we aren't surprised later.

> The quick solution is to symlink /usr/lib/libjvm.so to the proper lib;

OK.

#32 - 02/13/2014 01:47 AM - Constantin Asofiei

Greg Shah wrote:

> My only concern is that the server is very tightly coupled to the command line generation for the client. This means that later, when we enable a remote launcher, it will be more work. I don't want to change this right now, I just want to highlight the issue.

To make it fully generic, I can allow the 0 <secure-port> <host> <server-alias> <user> <workdir> <command> [args] syntax for no password authentication. This will use the target P2J server only for authentication. But I don't think I fully grasp the scenario you are referring to: if P2J server S1 wants to start a batch process on P2J server S2, then it will:

- call spawn with <secure-port> <host> <server-alias> targeting S1, to authenticate
- use the spawn command returned by the NativeSecureConnection.command (with a <user> <workdir> <command> [args] syntax) to target a P2J client for server S2
- note that if S1 and S2 are on distinct machines, then <user> <workdir> need to be on S1's machine.

But, if S1 wants to start a batch process on S2, why not establish a connection to S2, resolve the BatchProcess network server and call BatchProcess.launch(process) to launch it? This is cleaner and a lot easier.

#33 - 02/13/2014 07:08 AM - Constantin Asofiei

About the scheduler and cron: there is the Quartz library from Terracota released under Apache Licence V2: http://quartz-scheduler.org/overview/license-and-copyright and which can be downloaded from here: http://quartz-scheduler.org/downloads. Although Quartz looks like a fully-fledged cron scheduler, I want to use from it only the APIs which get the next execution time (from a cron expression and a current date). I don't think the added overhead from replacing java timers with Quartz is needed, using java timers to send the jobs to our dedicated thread is enough.

With this, the Job dir schema class will change to:

```
      <object-class name="job" leaf="true" immutable="false">
         <class-attribute name="type"         type="STRING"  mandatory="true"  multiple="false" immutable="fal
se" />
         <class-attribute name="target"       type="STRING"  mandatory="true"  multiple="false" immutable="fal
se" />
         <class-attribute name="enabled"      type="BOOLEAN" mandatory="true"  multiple="false" immutable="fal
se" />
         <class-attribute name="mode"         type="STRING"  mandatory="true"  multiple="false" immutable="fal
se" />
         <class-attribute name="second"       type="STRING"  mandatory="false" multiple="false" immutable="fal
se" />
         <class-attribute name="minute"       type="STRING"  mandatory="false" multiple="false" immutable="fal
se" />
         <class-attribute name="hour"         type="STRING"  mandatory="false" multiple="false" immutable="fal
se" />
         <class-attribute name="day-of-week"  type="STRING"  mandatory="false" multiple="false" immutable="fal
se" />
         <class-attribute name="day-of-month" type="STRING"  mandatory="false" multiple="false" immutable="fal
se" />
         <class-attribute name="month"        type="STRING"  mandatory="false" multiple="false" immutable="fal
se" />
         <class-attribute name="year"         type="STRING"  mandatory="false" multiple="false" immutable="fal
se" />
      </object-class>
```

or we can implode the cron expression nodes into a single node:

```
         <class-attribute name="cronExpression" type="STRING"  mandatory="false" multiple="false" immutable="f
alse" />
```

From the CronExpression javadoc:

```
Cron expressions are comprised of 6 required fields and one optional field separated by white space.
The fields respectively are described as follows:

Field Name       Allowed Values       Allowed Special Characters
-----------------------------------------------------------------
Seconds          0-59                 , - * /
Minutes          0-59                 , - * /
Hours            0-23                 , - * /
Day-of-month     1-31                 , - * ? / L W
Month            1-12 or JAN-DEC      , - * /
Day-of-Week      1-7 or SUN-SAT       , - * ? / L #
Year (Optional)  empty, 1970-2199     , - * /
```

#34 - 02/13/2014 10:08 AM - Marius Gligor

1. Looking inside the spawner.c code I understood that the spawner became an universal tool.
You add an option to get the command from a server session via a JNI call or from command line like on the original implementation.
The spawn method remain unchanged and according to my tests seems to work fine now by attaching the pseudo terminal.
I found no problems on your code.

2. Regarding WatchdogTimer today I added log messages inside the class and I did a lot of tests.
According to my tests it works properly. (see my post of ref #1811)

#35 - 02/13/2014 10:27 AM - Greg Shah

> But I don't think I fully grasp the scenario you are referring to: if P2J server S1 wants to start a batch process on P2J server S2, then it will:

This is not what I mean by "remote launching".

The idea is that the "client" system may be physically or logically separate from the "application server" system. In other words, if the clients (appservers, batch clients, web clients) run on a different system than the P2J application server, then a tight coupling will be a problem.

Today we assume that the clients will run on the same system, but I see many use cases where that would not be true.

> there is the Quartz library from Terracota

Using Quartz is fine.

> With this, the Job dir schema class will change to:

Yes, keep the separate fields in the directory.

#36 - 02/13/2014 12:41 PM - Constantin Asofiei

Greg Shah wrote:

> Today we assume that the clients will run on the same system, but I see many use cases where that would not be true.

You mean only the clients started with the spawner, right?

> The idea is that the "client" system may be physically or logically separate from the "application server" system. In other words, if the clients (appservers, batch clients, web clients) run on a different system than the P2J application server, then a tight coupling will be a problem.

OK, I think I understand what you mean: you are referring to the fact that ProcessClientBuilder and ClientBuilder hard code the P2J Server's host to localhost, right? When we will have to implement this remote launching, we will at least need a broker to run on the target machine, to which the P2J server would connect and send the spawn command (unless we choose some other way to invoke spawn remotely, via ssh or something like this). Also, if the P2J server's machine is connected to multiple networks, we will need to send in the spawn command the server's IP/hostname which is on the same network as the remote P2J client. But I don't think this will be problematic in terms of the spawner infrastructure: the ProcessClientBuilder and ClientBuilder can easily be changed to receive the real P2J server address instead of localhost.

Also, note that for no-password authentication mode, there are two host names used: one received by the spawn tool, targeting the P2J Server which invoked it and another host, in the argument list returned by the P2J server, which will actually be used by the P2J client to connect. We can always make these two different, if we need.

#37 - 02/13/2014 01:17 PM - Greg Shah

> You mean only the clients started with the spawner, right?

Right.

> we will at least need a broker to run on the target machine, to which the P2J server would connect and send the spawn command

Exactly.

> But I don't think this will be problematic in terms of the spawner infrastructure: the ProcessClientBuilder and ClientBuilder can easily be changed to receive the real P2J server address instead of localhost.

Good. I just wanted to record the ideas here for future reference.

#38 - 02/14/2014 09:27 AM - Constantin Asofiei

I'll use this node to document all the needed setup/configuration for the spawner infrastructure, used in appserver, batch process or chui web modes.

==========
1. spawn dependencies: make sure libjvm.so is visible. Use locate libjvm.so to find the location of this library. The following command was needed for devsrv01, to create the symlink:

```
sudo ln -s /usr/lib/jvm/java-7-openjdk-amd64/jre/lib/amd64/server/libjvm.so /usr/lib/libjvm.so
```

2a. build P2J: -Dhave.pam=yes -Dpost.build=yes are not mandatory, just use ant all. PAM was not installed on devsrv01. If PAM needs to be installed, use sudo apt-get install libpam0g-dev to install it.

2b. setup the spawner (sudo rights are needed); assume we have a ~/testing/spawner/ folder. We add an empty srv-certs.store file for now. The following script needs to be execued after P2J was built.

```
sudo rm -fr ~/testing/spawner
install -d ~/testing/spawner
# use an empty file
echo > ~/testing/spawner/srv-certs.store
# copy files
cp ~/testing/majic/p2j/build/native/spawn ~/testing/spawner/
cp ~/testing/majic/p2j/build/lib/p2j.jar ~/testing/spawner/
# set permissions
sudo chown root:root ~/testing/spawner/spawn ~/testing/spawner/p2j.jar ~/testing/spawner/srv-certs.store
sudo chmod 4755 ~/testing/spawner/spawn
sudo chmod 0550 ~/testing/spawner/p2j.jar ~/testing/spawner/srv-certs.store
# check if spawn can be started. There should be only the syntax help should be shown.
~/testing/spawner/spawn
```

3. configure the clientConfig node. This node can be specified either per-server or per-account: /server/{default | <server-id>}/clientConfig or /server/{default | <server-id>}/runtime/{default | <subject-id>}/clientConfig.

- the minimal node (full paths are requied):

```
        <node class="container" name="clientConfig">
          <node class="string" name="spawner">
            <node-attribute name="value" value="/home/ca/testing/spawner/spawn"/>
          </node>
          <node class="string" name="workingDir">
            <node-attribute name="value" value="/home/ca/testing/majic/run/client/"/>
          </node>
        </node>
```

- the full node (just to document all possible configurations):

```
        <node class="container" name="clientConfig">
          <node class="string" name="appArguments">
            <node-attribute name="value" value="val1 val2 val3"/>
          </node>
          <node class="string" name="cfgOverrides">
            <node-attribute name="value" value="k1=v1 k2=v2"/>
          </node>
          <node class="string" name="systemUser">
            <node-attribute name="value" value="bogus"/>
          </node>
          <node class="boolean" name="secure">
            <node-attribute name="value" value="TRUE"/>
          </node>
          <node class="string" name="spawner">
            <node-attribute name="value" value="/home/ca/testing/spawner/spawn"/>
          </node>
          <node class="string" name="jvmArgs">
            <node-attribute name="value" value="-Xmx512m -XX:MaxPermSize=64m -Djava.awt.headless=true -Xde
bug -Xnoagent -Djava.compiler=NONE -Xrunjdwp:transport=dt_socket,address=2998,server=y,suspend=n"/>
          </node>
```

```
          <node class="string" name="classpath">
            <node-attribute name="value" value="/home/ca/testing/majic/p2j/build/lib/p2j.jar"/>
          </node>
          <node class="string" name="libPath">
            <node-attribute name="value" value="/home/ca/testing/majic/p2j/build/lib"/>
          </node>
          <node class="string" name="workingDir">
            <node-attribute name="value" value="/home/ca/testing/majic/run/client/"/>
          </node>
          <node class="string" name="configFile">
            <node-attribute name="value" value="client.xml"/>
          </node>
          <node class="string" name="command">
            <node-attribute name="value" value="java"/>
          </node>
        </node>
```

4. configure the chuiWeb node. This node can be specified per-server: /server/{default | <server-id>}/chuiWeb.

- the minimal node:

```
        <node class="container" name="chuiWeb">
          <node class="boolean" name="enabled">
            <node-attribute name="value" value="TRUE"/>
          </node>
        </node>
```

- the mimimal configuration with static port/hosts:

```
        <node class="container" name="chuiWeb">
          <node class="boolean" name="enabled">
            <node-attribute name="value" value="TRUE"/>
          </node>
          <node class="integer" name="port">
            <node-attribute name="value" value="9443" />
          </node>
          <node class="string" name="host">
            <node-attribute name="value" value="localhost" />
          </node>
        </node>
```

- the full node:

```
        <node class="container" name="chuiWeb">
               <node class="boolean" name="enabled">
                  <node-attribute name="value" value="TRUE" />
               </node>
               <node class="string" name="background">
                  <node-attribute name="value" value="0x000000" />
               </node>
               <node class="string" name="foreground">
                  <node-attribute name="value" value="0xFFA500" />
               </node>
               <node class="string" name="selection">
                  <node-attribute name="value" value="0x0000FF" />
               </node>
               <node class="string" name="fontname">
                  <node-attribute name="value" value="monospaced" />
               </node>
               <node class="integer" name="fontsize">
                  <node-attribute name="value" value="12" />
               </node>
               <node class="integer" name="rows">
                  <node-attribute name="value" value="24" />
               </node>
               <node class="integer" name="columns">
                  <node-attribute name="value" value="80" />
               </node>
```

```
            <node class="integer" name="webSocketTimeout">
               <node-attribute name="value" value="-1" />
            </node>
            <node class="integer" name="watchdogTimeout">
               <node-attribute name="value" value="300000" />
            </node>
            <node class="integer" name="port">
               <node-attribute name="value" value="9443" />
            </node>
            <node class="string" name="host">
               <node-attribute name="value" value="localhost" />
            </node>
        </node>
```

5a. configure the tempClient node (optional). poolSize defaults to 8 and timeout to 30 seconds. This node can be specified per-server: /server/{default | <server-id>}/tempClient

```
        <node class="container" name="tempClient">
          <node class="integer" name="poolSize">
            <node-attribute name="value" value="10"/>
          </node>
          <node class="integer" name="timeout">
            <node-attribute name="value" value="120000"/>
          </node>
        </node>
```

5b. Give admin rights to the master server

- for MAJIC:
  - add GSO user to the admin->"/" ACL; add this node under the security/acl/admin section:

```
          <node class="container" name="001300">
            <node class="strings" name="subjects">
              <node-attribute name="values" value="admins"/>
              <node-attribute name="values" value="gso"/>
            </node>
            <node class="adminRights" name="rights">
              <node-attribute name="permissions" value="'01'B"/>
              <node-attribute name="type" value="0"/>
            </node>
            <node class="resource" name="resource-instance">
              <node-attribute name="reference" value="/"/>
              <node-attribute name="reftype" value="TRUE"/>
            </node>
          </node>
```

  - expose the Spawner interface via a net->com.goldencode.p2j.main.Spawner ACL; add this node in the security/acl/net section:

```
          <node class="container" name="002250">
            <node class="resource" name="resource-instance">
              <node-attribute name="reftype" value="TRUE"/>
              <node-attribute name="reference" value="com.goldencode.p2j.main.Spawner"/>
            </node>
            <node class="netRights" name="rights">
              <node-attribute name="permissions" value="'0101'B"/>
            </node>
            <node class="strings" name="subjects">
              <node-attribute name="values" value="all_others"/>
            </node>
          </node>
```

  - expose the Spawner interface via a net->com.goldencode.p2j.main.RemoteSpawner ACL; add this node in the security/acl/net section:

```
          <node class="container" name="002275">
            <node class="resource" name="resource-instance">
              <node-attribute name="reftype" value="TRUE"/>
```

```
            <node-attribute name="reference" value="com.goldencode.p2j.main.RemoteSpawner"/>
          </node>
          <node class="netRights" name="rights">
            <node-attribute name="permissions" value="'0101'B"/>
          </node>
          <node class="strings" name="subjects">
            <node-attribute name="values" value="all_others"/>
          </node>
        </node>
```

- for server project, security/acl/system ACLs needed to be changed/added::

```
        <node class="container" name="000100">
          <node class="resource" name="resource-instance">
            <node-attribute name="reference" value="admin"/>
            <node-attribute name="reftype" value="TRUE"/>
          </node>
          <node class="systemRights" name="rights">
            <node-attribute name="check" value="true"/>
          </node>
          <node class="strings" name="subjects">
            <node-attribute name="values" value="admins"/>
            <node-attribute name="values" value="standard"/>
          </node>
        </node>
```

```
        <node class="container" name="000500">
          <node class="strings" name="subjects">
            <node-attribute name="values" value="all_others"/>
          </node>
          <node class="systemRights" name="rights">
            <node-attribute name="check" value="true"/>
          </node>
          <node class="resource" name="resource-instance">
            <node-attribute name="reference" value="accounts"/>
            <node-attribute name="reftype" value="TRUE"/>
          </node>
        </node>
```

6. appserver configuration:

- from all user accounts, remove the appserver node
- from the appservers/<appserver> nodes, remove the auto_start, user_dir and the jvm_args nodes
- the appserver must be associated with a P2J process account, not user account. So, add an appserver node to the process associated with the appserver:

```
<node-attribute name="appserver" value="ca_srv" />
```

- add a /server/{default | <server-id>}/scheduler node to start the appserver agents at server startup; replace target with the process ID:

```
          <node class="container" name="scheduler">
             <node class="job" name="start_ca_srv">
                <node-attribute name="type" value="process" />
                <node-attribute name="target" value="ca_srv" />
                <node-attribute name="enabled" value="TRUE" />
                <node-attribute name="mode" value="now" />
             </node>
          </node>
```

- the /server/{default | <server-id>}/runtime/<process-id>/clientConfig node must exist, to set the appserver agents in background mode:

```
        <node class="container" name="clientConfig">
          <node class="string" name="cfgOverrides">
            <node-attribute name="value" value="client:driver:background=true"/>
          </node>
        </node>
```

7. hard-coded client configuration sent to the command which starts the P2J Client:

- for P2J process clients, by ProcessClientBuilder.addClientOptions:

```
      command.add("client:mode:batch=true");
```

- for P2J web chui clients, by WebClientBuilder.addClientOptions, which reads them from the chuiWeb node or uses defaults:

```
      command.add("client:driver:type=" + terminalOptions.get("type"));
      command.add("client:chui:rows=" + terminalOptions.get("rows"));
      command.add("client:chui:columns=" + terminalOptions.get("columns"));
      command.add("client:chui:background=" + terminalOptions.get("background"));
      command.add("client:chui:foreground=" + terminalOptions.get("foreground"));
      command.add("client:chui:fontname=" + terminalOptions.get("fontname"));
      command.add("client:chui:fontsize=" + terminalOptions.get("fontsize"));
      command.add("client:web:socketTimeout=" + terminalOptions.get("webSocketTimeout"));
      command.add("client:web:watchdogTimeout=" + terminalOptions.get("watchdogTimeout"));
      command.add("client:web:port=" + terminalOptions.get("port"));
      command.add("client:web:host=" + terminalOptions.get("host"));
```

```
      // Add referrer address
      command.add("web:referrer:url=" + webConfig.getReferrer());
```

#39 - 02/14/2014 09:31 AM - Constantin Asofiei

- File ca_upd20140214f.zip added

This update adds the scheduler cron support and will go into testing; contains other minor fixes too.

#40 - 02/14/2014 12:19 PM - Constantin Asofiei

- File ca_upd20140214h.zip added

This one is merged with mag_upd20140213c.zip from #1811.

#41 - 02/15/2014 09:48 AM - Greg Shah

Code Review 0214h

The code changes look good. If I understand correctly, this contains fixes for all issues found in #1811 except for the problem in note 370. Is that right?

Who has the lead in fixing that issue?

You can check in and distribute this when it passes testing.

#42 - 02/15/2014 09:53 AM - Constantin Asofiei

Greg Shah wrote:

> If I understand correctly, this contains fixes for all issues found in #1811 except for the problem in note 370. Is that right?

Correct.

> Who has the lead in fixing that issue?

That System.exit problem is related to web sockets/browser redirection/etc - so Marius should work on it, because redirecting the browser from P2J Client side is not OK (just think someone kills the P2J Client manually...). I think we need a browser-level solution to determine "know it's time to redirect back to login".

#43 - 02/15/2014 10:24 AM - Constantin Asofiei

Greg Shah wrote:

> You can check in and distribute this when it passes testing.

Eric: I don't want to commit it until I update the directory.xml you use to run the server, too. Can you please send me the directory.xml files you use? Else, you will need to follow the changes at note 38, to make it work.

#44 - 02/15/2014 12:59 PM - Greg Shah

Constantin Asofiei wrote:

> Greg Shah wrote:
>
> > If I understand correctly, this contains fixes for all issues found in #1811 except for the problem in note 370. Is that right?
>
> Correct.
>
> > Who has the lead in fixing that issue?
>
> That System.exit problem is related to web sockets/browser redirection/etc - so Marius should work on it, because redirecting the browser from P2J Client side is not OK (just think someone kills the P2J Client manually...). I think we need a browser-level solution to determine "know it's time to redirect back to login".

Marius: please fix this as your next work.

#45 - 02/15/2014 02:53 PM - Eric Faulhaber

- File directory.xml added

Constantin Asofiei wrote:

> Eric: I don't want to commit it until I update the directory.xml you use to run the server, too. Can you please send me the directory.xml files you use? Else, you will need to follow the changes at note 38, to make it work.

Sorry, didn't see this earlier. Attached.

#46 - 02/16/2014 04:26 AM - Constantin Asofiei

- File ca_upd20140216a.zip added

The attached update fixed some oversights (debug was enabled for the spawn's P2J secure connection client and a bug reading per-account config). Passed testing, committed to bzr rev 10469.

#47 - 02/20/2014 12:19 PM - Greg Shah

- Status changed from WIP to Closed

- % Done changed from 0 to 100

#48 - 02/20/2014 06:27 PM - Constantin Asofiei

- File ca_upd20140221a.zip added

Fixes a bug in the JobProcessor waiting loop: wait only if there are no posted jobs.

LE: no testing required, I'll commit it tomorrow.

#49 - 02/21/2014 03:19 AM - Constantin Asofiei

0221a.zip was committed to bzr rev 10475.

#50 - 02/25/2014 12:14 PM - Constantin Asofiei

- File ca_upd20140225b.zip added

Fixed ServerDriver.connect to work with in-directory private keys.

#51 - 02/25/2014 02:33 PM - Greg Shah

Code Review 0225b

I'm fine with the changes. Check these in when they pass testing.

#52 - 02/26/2014 03:08 AM - Constantin Asofiei

- File ca_upd20140226a.zip added

Passed runtime testing, attached version was committed to bzr rev 10480. Changes compared to 0225b:

- set the security:truststore:alias, if not already set
- NativeSecureConnection needs to write to the exception info to STDERR.

#53 - 04/04/2014 06:09 AM - Constantin Asofiei

- File ca_upd20140404a.zip added

Changed to allow postbuild.sh to install the spawn tool and associated files in a specified location. The command looks like:

```
ant jar native -Dpost.build=yes -Dspawn.install.folder=/home/ca/testing/spawner
```

or

```
ant jar native -Dpost.build=yes -Dspawn.install.folder=/home/ca/testing/spawner -Dsrv.certs=/home/ca/testcases
```

```
/simple/server_appserver/srv-certs.store
```

Is recommended to use absolute paths for spawn.install.folder and srv.certs; else, all paths are relative to the build/native folder.

sudo rights are needed when using -Dpost.build=yes.

The build on devsrv01 is not affected by these changes. Please review.

#54 - 04/08/2014 03:42 PM - Greg Shah

Code Review 0404a

I am fine with the changes.

#55 - 04/09/2014 03:31 AM - Constantin Asofiei

Greg Shah wrote:

> Code Review 0404a
>
> I am fine with the changes.

Committed to bzr rev 10508.

#56 - 09/04/2015 09:18 AM - Eric Faulhaber

Some notes on moving from Java 7 to Java 8 w.r.t. building the spawn program on Ubuntu:

I retained openjdk-7-jdk on my system and installed openjdk-8-jdk via apt-get install. After this, I could not get appserver agents to start. This was due to a problem with the spawn part of the native P2J build.

When linking, ld (via gcc) will use the ldconfig cache to search for shared objects to link. My /etc/ld.so.conf.d directory contained a configuration file named java-7-server.conf. That file contained a single line:

```
/usr/lib/jvm/java-7-openjdk-amd64/jre/lib/amd64/server
```

So, whenever ldconfig would run (presumably after an installation or perhaps reboot?), it would rebuild the ldconfig cache to contain (among many others) these entries:

```
libjvm.so (libc6,x86-64) => /usr/lib/jvm/java-7-openjdk-amd64/jre/lib/amd64/server/libjvm.so
libjvm.so (libc6,x86-64) => /usr/lib/libjvm.so
```

Since the java-7 entry came first, it would link to this version of the shared library when building spawn, and I ended up with this:

```
ecf@ecf:~/spawner$ ldd -v spawn
    linux-vdso.so.1 =>  (0x00007ffe844a4000)
    libcrypt.so.1 => /lib/x86_64-linux-gnu/libcrypt.so.1 (0x00007f6499c59000)
```

```
    libjvm.so => /usr/lib/jvm/java-7-openjdk-amd64/jre/lib/amd64/server/libjvm.so (0x00007f6498e13000)
    libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f6498a49000)
    libstdc++.so.6 => /usr/lib/x86_64-linux-gnu/libstdc++.so.6 (0x00007f649873a000)
    libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0x00007f6498432000)
    libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f649822e000)
    libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007f6498010000)
    /lib64/ld-linux-x86-64.so.2 (0x00007f6499e91000)
    libgcc_s.so.1 => /lib/x86_64-linux-gnu/libgcc_s.so.1 (0x00007f6497dfa000)
```

The solution is to remove the java-7-server.conf file or to replace it with a more generic configuration that points to the current JVM subdirectory, then run sudo ldconfig to rebuild the cache. I chose to just remove the file, rather than replacing it.

I'm not sure where this java-7-server.conf file came from. I didn't add it manually, AFAIR -- I don't think I would have known to. Maybe a previous JVM installation put it there, but openjdk-8-jdk did not remove/replace it, and update-alternatives doesn't appear to be integrated with the ldconfig cache mechanism.

#57 - 11/16/2016 12:06 PM - Greg Shah

- Target version changed from Milestone 11 to Cleanup and Stablization for Server Features

Files

ca_upd20140206b.zip	204 KB	02/06/2014	Constantin Asofiei
ca_upd20140207e.zip	2.93 MB	02/07/2014	Constantin Asofiei
ca_upd20140212e.zip	2.96 MB	02/12/2014	Constantin Asofiei
ca_upd20140214f.zip	3.55 MB	02/14/2014	Constantin Asofiei
ca_upd20140214h.zip	3.55 MB	02/14/2014	Constantin Asofiei
directory.xml	25.2 KB	02/15/2014	Eric Faulhaber
ca_upd20140216a.zip	3.55 MB	02/16/2014	Constantin Asofiei
ca_upd20140221a.zip	1.9 KB	02/20/2014	Constantin Asofiei
ca_upd20140225b.zip	59.7 KB	02/25/2014	Constantin Asofiei
ca_upd20140226a.zip	61.9 KB	02/26/2014	Constantin Asofiei
ca_upd20140404a.zip	12.4 KB	04/04/2014	Constantin Asofiei