User Interface - Feature #2308

Feature # 1811 (Closed): implement the AJAX client driver

remote client launch

05/21/2014 10:14 AM - Greg Shah

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Marius Gligor	**% Done:**	100%
Category:		**Estimated time:**	40.00 hours
Target version:	GUI Support for a Complex ADM2 App		
billable:	No	**vendor_id:**	GCD

Description

Related issues:

Related to User Interface - Feature #2342: fix cross-domain cookie for web cl...	**Closed**	**07/22/2014**
Related to User Interface - Feature #2343: implements a native solution to ge...	**New**	**07/22/2014**
Related to Documentation - Feature #2346: remote client launch documentation	**Closed**	**07/28/2014**

History

#1 - 05/21/2014 10:19 AM - Greg Shah

Create a special P2J client that is a dedicated client process launcher. This client must be able to run on any arbitrary system, not just on the server on which the P2J server executes. The idea is that this is a remote launcher that allows web clients, batch processes and appserver agent instances to be spawned on a system that is different from the P2J server's system.

This approach will trigger at least one known problem with cross-origin cookies used for authentication token processing. See #1811 note 599 for details.

#2 - 05/21/2014 10:38 AM - Greg Shah

To be clear, this special launcher client would be started on the remove system and it would be configured to connect to the P2J server and authenticate using certificates. Then it would wait for the P2J server to tell it to spawn a process. The same capabilities must be present as if the P2J server was spawning processing directly. This means there is some bidirectional data exchange that will have to occur as part of the spawning process.

The portions of the spawning process that use a login to the P2J server should translate to this approach naturally.

#3 - 05/22/2014 02:57 AM - Constantin Asofiei

The P2J Server should have no configuration/restriction on how many P2J clients act as remote launchers and the machines on which they are running. Instead, the P2J Clients acting as remote spawners will register themselves with the P2J Server and the P2J Server will know only about the ones which are registered/connected. But the question here is: will we have some load balancing/routing mechanism? Because if more than one machine is used then, at the time of the web client login (when the OS credentials are used):

- the P2J Server can choose a P2J Client spawner randomly or using some load balancing policy (and this means all the machines are configured in the same way)
- each P2J Client spawner has a list of known OS usernames, and the P2J Server will choose the P2J client spawner to which the OS username belongs; problems may appear if the same OS username is registered with more than one machine.

#4 - 05/22/2014 08:25 AM - Greg Shah

Doesn't the 4GL appserver implementation have some features in this regard? We should at least document their basic approach here as part of this task.

Since we don't have any specific customer requirements right now, I'm inclined to start with a very simple load balancing approach (evenly distribute clients between remote launchers. We can possibly add some min/max range limits for each one. That is about it, unless the 4GL compatible approach is really simple to implement.

#5 - 06/25/2014 12:48 PM - Greg Shah

- Assignee set to Marius Gligor

#6 - 07/04/2014 03:54 AM - Marius Gligor

- Status changed from New to WIP

#7 - 07/04/2014 04:42 AM - Constantin Asofiei

For remote client launching, first component we need is a broker; this will be a special P2J client which authenticates (via SSL) and registers with a remote server. The remote server will keep a registry of registered brokers and will use these brokers for remote launching P2J clients. Both web and appserver agent clients need to be covered by a broker. I think 4GL does some load balancing via the NameServer Load Balancer associated with an appserver - please check how this is done.

What the broker needs to do:

- When registering with the remote server, the P2J server will determine the broker's configuration (from the directory), which includes:
  - details about the appserver(s) which can be started on by this broker
  - details about the web clients (the P2J usernames?) which can be started by this broker
- when starting a remote client, if more than one broker is registered for that client (web or appserver agent), the P2J server should choose the broker depending on the load balancing policy in use:
  - for web clients, I think it should be OK to choose the broker with the lesser load
  - for appserver agents, we should use the 4GL's load balancing policy, if this can be easily identified (from the documentation or something else). This will mean the broker will need to expose some APIs related to how many agents were started by that broker (and are still running), to determine its load. Alternatively, I think this is easier to be managed on the P2J Server side, if the client (server-side part) knows which broker started it.
- the same broker client can be started multiple times; this will allow live attachement of brokers to a running P2J server.

Implementation approach:

- the broker clients will need special configuration in the directory, to identify themselves as brokers. The StandardServer.standardEntry will check if the client starts as a broker, similar to how the appserver agents are checked, via a if (BrokerManager.startBroker) { return false; }.
- when the P2J client acting as a broker is terminated, we need to choose if we want to either terminate all P2J clients started by this broker or let them run.
- the P2J broker client will need to export a special interface as a network server: this interface will be used by the P2J server to send a command to the remote P2J broker.

Code changes:

- create a new object-class in the dir_schema.xml file, named broker.
- the process object-class will need a new attribute, broker, with the name of the broker being started.
- I think we should keep the existing approach which allows startup of appserver agents/web clients on the same machine as the P2J server. If no brokers are configured for that web client/appserver agent, the existing infrastructure will be used. If brokers are configured, then the broker will be used.
- please check how the BrokerManager can be integrated with the existing ClientBuilder and ClientSpawner classes and document it here.
- ClientCore.start I think will need to be aware that this is a P2J broker client and bypass the normal startup (i.e. ThinClient, etc).

Please let me know if you have any comments.

#8 - 07/04/2014 06:06 AM - Marius Gligor

For the beginning I have some comments and questions please.

1. P2J server and AppServer are distinct terms? If yes:
- StandardServer which implements MainEntry acts like an AppServer?
I saw that is used by the P2J clients to execute the converted applications because the clients are thin and acts like a presentation layer.
The business code is always executed on the server side (AppServer)
- ServerDriver used to start server acts like a P2J server?

2. The broker run on a remote machine A and the P2J server on another machine B.
The broker should connect and register as broker on a P2J server on a secured manner (SSL).
When a remote client is spawned by this broker on machine A the AppServer is running on machine A where the broker runs or on machine B where P2J runs?

3. The start of the remote broker is similar with the start of a P2J client like for example the Swing driver via a shell script?
What I have to ask here in fact is if the launching of brokers are not done by P2J server.

4. I didn't found a BrokerManager class inside P2J project.

#9 - 07/04/2014 06:38 AM - Constantin Asofiei

Marius Gligor wrote:

> For the beginning I have some comments and questions please.
>
> 1. P2J server and AppServer are distinct terms? If yes:

Yes, they are distinct. A P2J Server can have one or more appserver exposed via P2J processes.

> - StandardServer which implements MainEntry acts like an AppServer?

StandardServer.standardEntry is the entry point which is reached by all started P2J Clients, when their server-side part is initiated. The ClientCore.start:264 line (running = main.standardEntry(params); will execute the server-side entry point of a P2J client. On a side note, I think ClientCore should be left alone. ClientDriver.start should choose between ClientCore and BrokerCore (a new class), depending on some bootstrap configuration.

> I saw that is used by the P2J clients to execute the converted applications because the clients are thin and acts like a presentation layer.
> The business code is always executed on the server side (AppServer)

Correct.

> - ServerDriver used to start server acts like a P2J server?

Yes. ServerDriver relies on StandardServer.bootstrap to start a P2J Server.

> 2. The broker run on a remote machine A and the P2J server on another machine B.

Yes.

> The broker should connect and register as broker on a P2J server on a secured manner (SSL).

Yes. It will be configured and authenticate as a P2J process account.

> When a remote client is spawned by this broker on machine A the AppServer is running on machine A where the broker runs or on machine B where P2J runs?

Well, the appserver (i.e. the AppServerManager.startAppServer) will always execute on the P2J Server side (machine B). Machine A will hold only the P2J clients which act as the appserver agents. Currently, these agents are started via the ProcessClientSpawner class.

> 3. The start of the remote broker is similar with the start of a P2J client like for example the Swing driver via a shell script?

The P2J server has no knowledge about where the broker clients reside. These special P2J clients (brokers) need to be started manually or other way.

> What I have to ask here in fact is if the lunching of brokers are not done by P2J server.

Correct, the P2J server is not involved in starting them.

> 4. I didn't found a BrokerManager class inside P2J project.

This is a new class which we need to add. It will provide broker management on P2J server side (i.e. which brokers are started, choosing a broker, etc). The launch command will be computed on P2J server side (what the ClientBuilder classes do), the broker will just delegate the command launching to its associated P2J client.

#10 - 07/04/2014 10:16 AM - Marius Gligor

- File asadm.pdf added

Here is what I found so far related to OpenEdge brokers and load balancing.

In order to achieve load balancing with the AppServer, a "NameServer Load Balancing" license is required.

It is important to understand the way Progress brokers registers with the NameServer.

There are three main things to understand:
Broker name
Broker uuid
Broker supported AppService names

1. Broker name
This is the name that is used to identify the broker as defined in the ubroker.properties e.g. "asbroker1".
The only important thing about this name is that it needs to be unique within the ubroker.properties, that is within the AdminServer instance.

2. Broker uuid
The uuid (Unique Universal IDentifier) helps maintain uniqueness accross multiple brokers with the same name.
Using the Progress utility named 'genuuid' (generator for uuid) you are guaranteed to obtain the unique ID valid at the time and space within a single computer network.
The algorithm used is taking into account things such as the hostname, tcpip address and other things that are explained in detail in Knowledge Center article 21175.
When using the Progress Explorer or OpenEdge Explorer to create new broker instances, generation of the UUID is done automatically by the AdminServer.
In the ubroker.properties the 'uuid' will be defined and visible for every broker instance.

3. Broker supported AppService names
These are the names that the broker publishes when it registers with the NameServer.
All applications requesting a certain AppService from the NameServer will obtain the information about host and port where the broker supporting this AppService is running.
That means that within the 4GL application code the AppService name should be specified, not the broker name.
If no distinct AppService name is specified then the default AppService name will match the AppServer broker name.

Example:
If the AppServer broker is named 'asbroker1', the default AppService name will also be 'asbroker1'.
A new broker could be created such as 'MyBroker' if the administrator chose to have multiple brokers servicing the application then 'MyBroker' could be configured with the AppService name of 'asbroker1'.

Finally, on the machine where the NameServer LoadBalancing license is installed, any brokers which register with that NameServer must have different uuid regardless of their broker name or AppService names.
The uuid used to register with the NameServer is what counts.
Therefore, one can have 5 servers each running AppServer broker named 'asbroker1' and each supporting AppService 'asbroker1' as long as they have different uuid values.

How to achieve load balancing with the AppServer?
Load Balancing for AppServer

How to Implement Fail-over Configuration for Unified Broker?
What is Progress NameServer Load Balancing?
What is OpenEdge NameServer Load Balancing?

The Progress load balancing feature is used to implement a fail-over configuration for a unified broker.
Load balancing is a NameServer option that comes installed with some products (for example, the WebSpeed Enterprise Transaction Server), or that you must install as an option with other products (for example, the AppServer or for other unified brokers).
For the OpenEdge Release 10.x, you need to install the NameServer load balancing as a separate product.

In order to configure load balancing with the NameServer, you need to specify fail-over weight factor (priorityWeight) in the ubroker.property file either based in percentage or based in "arbitrary sum".
If you specify priorityWeight to zero (0) for a Unified Broker instance, the NameServer load balancing will not be in effect.
The value of zero will route all the request to one available broker.
The priorityWeight value can be set using the Progress Explorer under the Advance property page of the broker's properties.

Also, make sure that the brokers are registered on the same load balancing capable NameServer using the same Application Service Name.
There can be more than one broker registered to the one NameServer, but in order to get the load balancing capability, you need to set the same Application Service Name for different brokers.
You can do that setting under the Broker section, AppService Name List in the broker's properties.

#11 - 07/07/2014 02:59 AM - Marius Gligor

The remote launcher (broker) connect to the P2J server and authenticate using certificates.
Is any example in P2J on how to implements authentication based on certificates only?
So far all that I found require also an User ID and a Password.

#12 - 07/07/2014 03:48 AM - Constantin Asofiei

Marius Gligor wrote:

> The remote launcher (broker) connect to the P2J server and authenticate using certificates.
> Is any example in P2J on how to implements authentication based on certificates only?
> So far all that I found require also an User ID and a Password.

See the p2j.main.NativeSecureConnection class: this class is used by the spawner to authenticate with the specified P2J server and retrieve the command associated with the given UUID. By the way, I think the broker still needs to rely on the spawner to launch the P2J client; thus the P2J server will send the broker a UUID for the command + plus the P2J server details (in case when a process is launched) or the entire command, in case a web-client is launched.

#13 - 07/07/2014 04:34 AM - Marius Gligor

I'm using your code from NativeSecureConnection class but we still need an account to connect to P2J server.
You are using the temporary account by getting the environment variables:

```
cfg.setConfigItem("access", "subject",  "id",   TemporaryAccount.fromHex(subject));
cfg.setConfigItem("access", "password", "user", TemporaryAccount.fromHex(password));
```

If these credentials are missing from configuration the user will ask to enter a subject id and a password from stdin.
I tried both connectDirect and connectVirtual and both require subject id and password.
The environment variables cannot be used because the broker is usually on another machine.
Even when we start the broker on the same machine the environment variables for temporary account are not available because the broker is not spawned by server.

#14 - 07/07/2014 04:36 AM - Constantin Asofiei

Marius Gligor wrote:

> I'm using your code from NativeSecureConnection class but we still need an account to connect to P2J server.
> You are using the temporary account by getting the environment variables:
> [...]
> If these credentials are missing from configuration the user will ask to enter a subject id and a password from stdin.
> I tried both connectDirect and connectVirtual and both require subject id and password.

Yes, the broker needs a specific process account defined in the P2J directory. This process account will have a broker attribute, to identify itself as a broker.

#15 - 07/07/2014 04:45 AM - Constantin Asofiei

Sorry, forgot to mention: the following needs to be set in the bootstrap config:

- keystore, via security:keystore:filename, access:password:keystore, access:password:keyentry
- keystore alias, via security:keystore:processalias - this will identify the process cert alias (which needs to be the same as the P2J process Id).

Originally we thought that SSLCertGenUtil will not need to save the process private key in a separate keystore file, but it looks like this is no longer valid; I'll modify SSLCertGenUtil to allow saving of a process key store to a separate file.

#16 - 07/07/2014 06:00 AM - Constantin Asofiei

- File ca_upd20140707a.zip added

See attached for the update. Execute the SSLCertGenUtil class using as parameter the directory name, and it will reconfigure the certificates/private keys in the directory. At some point, it will ask you to save the private keys in external stores. Also, the trust-store will need to be saved.

This will allow you to build a client.xml to start the broker:

```
<node type="client">
   <security>
      <certificate validate="true" />
      <truststore filename="[trust-store-file]" />
      <truststore alias="[server-alias]" />
      <keystore filename="[process-key-store-file]" />
      <keystore processalias="[process-alias]" />
   </security>

   <access>
      <password truststore="[trust-store-password]" />
      <password keystore="[key-store-password]" />
      <password keyentry="[key-entry-password]"/>
   </access>
</node>
```

Note that the passwords may contain characters which will need to be XML-encoded, like ">" and "<" chars. Once you have the broker process account in the directory, use a client.xml to connect to the P2J server as a process.

#17 - 07/07/2014 08:35 AM - Marius Gligor

Using SSLCertGenUtil I reconfigured the certificates/private keys in the server directory and I created a client.xml file for client strong authentication.
Now it's works like I expected and no user credentials are needed for authentication. The authentication is done based on certificates only.

#18 - 07/07/2014 12:49 PM - Marius Gligor

I've implemented a P2J client (Broker) which connect to P2J server and authenticate using certificates.
Both the client and the sever exports some some services for register, unregister, etc.
It works fine when both server and client are started within Eclipse IDE.
When I tried to start the client and server from separate batch files the SSL authentication fails.
I exported the p2j-root alias as a PEM certificate and than I imported the certificate into the Java truststore.
Now when I tried to connect to the server I've got an "Authentication failed"
Do you have any idea how to fix this issue?

#19 - 07/07/2014 01:08 PM - Marius Gligor

I've fixed the issue by rebuilding the P2J project. I deleted the p2j-root certificate from Java keystore.

#20 - 07/07/2014 05:13 PM - Greg Shah

> Both web and appserver agent clients need to be covered by a broker.

Don't forget that we need to allow batch clients to be started too.

#21 - 07/08/2014 10:01 AM - Marius Gligor

- File mag_upd20140708a.zip added

This is an intermediate work, an initial implementation of a remote launcher (broker).

1. The remote launcher is a special P2J client used to spawn P2J batch and web clients remotely. The main implementation is in class BrokerCore.java
When the Broker is started on a remote machine he try to establish a secure session with a P2J server and authenticate based on a digital certificate.
The connection parameters are read from a configuration file client.xml which looks like:

```
<node type="client">
  <net>
      <server host="192.168.1.11" />
      <server secure_port="3334" />
      <connection secure="true" />
   </net>

   <security>
      <certificate validate="true" />
      <truststore filename="broker-trust.store" />
      <truststore alias="standard" />
      <keystore filename="broker-key.store" />
      <keystore processalias="p2j_proc_alias" />
      <authentication type="program"/>
   </security>
```

```
   <access>
      redacted
   </access>

   <remote>
      <accounts filename="accounts.xml" />
      <retry count="10" />
      <retry minutes="1" />
      <spawner location="./spawn" />
   </remote>
</node>
```

When the broker is started the command line should looks like:

```
java … com.goldencode.p2j.main.ClientDriver client.xml remote:broker:name=gcd
```

Where client.xml argument is used to specify the configuration file and the argument
remote:broker:name=gcd is mandatory and is used to identify the client as a broker.

2. On successful connection the Broker call a server exported method in order to register.
On register a structure of data is presented to server which contains the broker name, the target application server, the target server address and a list of user accounts that could be used to remote spawn clients on machine on which the broker runs. The spawn tools is used by broker to spawn clients. Because the server and broker could runs on different OS the spawner name is also specified in configuration file and send to server. If P2J server runs on Linux OS and broker on a Windows OS the used spawner is spawn.exe located on machine on which the broker is running, usually on the same folder. The user accounts are read from an XML file defined in configuration file remote:accounts:filename having the following structure:

```
<users>
   <name>mag</name>
   <name>foo</name>
   <name>zoo</name>
</users>
```

When a broker is registered the server will assign an unique UUID on each broker which allow to register multiple brokers having the same name.

3. The brokers are automatically unregistered from server whenever they close the session by attaching a session listener to session on server side. A session listener is also attached on the client side which allow to detect when the connection with server is lost. In this case the broker does not simply exit. Instead a retry flaw has been designed. The broker will try from time to time for a number of tried to reconnect to the server. The broker will exit only if he cannot (re)connect to the P2J server after a number of retry. The retry parameters are available in configuration file as a "remote" category.

LE: GES redacted the keystore passwords.

#22 - 07/08/2014 02:18 PM - Greg Shah

> a list of user accounts that could be used to remote spawn clients on machine on which the broker runs

Where does the list come from?

I don't want the customer to have to maintain this on every system where there is a broker. It would be a source of lots of maintenance effort and support problems. Can we find an alternative approach that is dynamic? For example, can we ask a broker to try to login for a given user and fall back to a different broker if that did not work?

#23 - 07/08/2014 02:21 PM - Constantin Asofiei

Greg, I was thinking for all the broker configuration to reside in the directory, similar to how the appserver is configured: the process has a broker attribute, and this attribute points to its configuration, in some <server-id>/brokers node. When the broker initializes on server-side, it reads its configuration from the directory, which includes info about which i.e. system usernames and/or P2J accounts and/or appserver names it can manage.

#24 - 07/08/2014 02:46 PM - Marius Gligor

Yes I think that is better to keep broker configurations on server directory. In this case when a broker connect and register to the server it should send only the broker name.
Brokers having the same name acts like a load-balancing system since they have the same configuration. Is this true?

#25 - 07/08/2014 02:48 PM - Constantin Asofiei

Marius Gligor wrote:

> Yes I think that is better to keep broker configurations on server directory. In this case when a broker connect and register to the server it should send only the broker name.

You don't need to send the broker name, this can be determined from the associated P2J process account - the broker attribute.

> Brokers having the same name acts like a load-balancing system since they have the same configuration. Is this true?

Yes, but not only this. The idea was that if a P2J account (be it appserver, web client or process) can be handled by multiple brokers, than a load-balancing policy applies.

#26 - 07/08/2014 02:55 PM - Greg Shah

Constantin Asofiei wrote:

> Greg, I was thinking for all the broker configuration to reside in the directory, similar to how the appserver is configured: the process has a broker attribute, and this attribute points to its configuration, in some <server-id>/brokers node. When the broker initializes on server-side, it reads its configuration from the directory, which includes info about which i.e. system usernames and/or P2J accounts and/or appserver names it can manage.

That sounds reasonable, with one caveat: no configuration should be needed in the default setup. In such a case, if there is only 1 broker then that will be tried. And if there are more than one broker (but no user lists), then it will try them all in a round robin until one works.

#27 - 07/08/2014 03:04 PM - Constantin Asofiei

Greg Shah wrote:

> Constantin Asofiei wrote:
>
> > Greg, I was thinking for all the broker configuration to reside in the directory, similar to how the appserver is configured: the process has a broker attribute, and this attribute points to its configuration, in some <server-id>/brokers node. When the broker initializes on server-side, it reads its configuration from the directory, which includes info about which i.e. system usernames and/or P2J accounts and/or appserver names it can manage.
>
> That sounds reasonable, with one caveat: no configuration should be needed in the default setup. In such a case, if there is only 1 broker then that will be tried. And if there are more than one broker (but no user lists), then it will try them all in a round robin until one works.

One more note: do you want to remove the current setup, which relies on the clientConfig nodes to launch a appserver/web client on the same machine as the P2J server? To me, this looks like the easiest way to setup a P2J server with a "built-in spawner".

#28 - 07/09/2014 09:43 AM - Greg Shah

> To me, this looks like the easiest way to setup a P2J server with a "built-in spawner".

We should retain the ability to have a P2J server that has the built-in spawner.

#29 - 07/10/2014 03:35 AM - Marius Gligor

- File mag_upd20140710a.zip added

Here is a simplified skeleton for remote brokers.

1. As I understood when a broker connect and register to P2J server no information is send to server. The broker configuration is keep in server directory.

- What kind of information should contains a directory entry for a broker?
- How broker configuration is used by clients and processes?
- How registered brokers are associated with the directory entries if more than one broker is registered?

Could you please offer me a short description regarding how brokers works?

2. When a process is spawned remotely some launch parameters are specific to machines on which the brokers runs:

- classpath.
- native libraries path.
- spawner location.
- the remote spawned client should use the IP address of P2J server instead localhost in order to make a connection with P2J server.

Where should be kept all this information?

#30 - 07/10/2014 11:18 AM - Marius Gligor

Today I've continued to work on remote launcher client (broker).
I tried to spawn a web client locally via a remote launcher client running on the same machine where the P2J server runs.
I did the appropriate changes to prepare the spawn but I found an issue and I need some help to fix them. Let me to explain.
Both the server and the broker exports a lot of services.
When the broker connect and register he call a server exported method. Within this method I'm able to get the broker exported interface from server and do service calls on the interface.
Later when I tried to call broker services within another session I cannot do. The call remains blocked undefined.
It seems that the broker exported interface is available only within the broker session or perhaps special rights are necessary.
How could I do to access the broker exported services on server from other sessions?

I implemented a flaw on which the broker call a server exported method each second and check if a spawn task is scheduled for him, than he get the spawn parameters and do the spawn. This flaw woks fine but is not so elegant like calling broker interface directly.

#31 - 07/10/2014 11:24 AM - Greg Shah

I suspect this is an ACL problem. Each API that is exported must be configured with the appropriate ACLs to allow the actual access to occur at runtime.

#32 - 07/10/2014 12:41 PM - Marius Gligor

I added the following node to server directory.xml but still not working.

```
      <node class="container" name="net">
        <node class="container" name="000100">
          <node class="strings" name="subjects">
            <node-attribute name="values" value="all_others"/>
          </node>
          <node class="netRights" name="rights">
            <node-attribute name="permissions" value="'00111111'B"/>
          </node>
          <node class="resource" name="resource-instance">
            <node-attribute name="reference" value="com.goldencode.p2j.main.BrokerClientServices"/>
            <node-attribute name="reftype" value="TRUE"/>
          </node>
        </node>
      </node>
```

The interface BrokerClientServices is exported from client not from server.
How directory.xml should be configured on this case?

#33 - 07/10/2014 02:28 PM - Marius Gligor

I fixed the access to BrokerClientServices. The node that I inserted is OK. After a build all of P2J now it works.

#34 - 07/10/2014 02:43 PM - Marius Gligor

Unfortunately I was wrong. I tested some old code on which the client connect to server and this works.
So I have to make more diggings.

#35 - 07/10/2014 04:54 PM - Greg Shah

I wasn't thinking correctly. On the client side there are no ACLs in play since the client's security manager is "empty". It approves all access decisions without any checking.

I don't know what is wrong in this case.

#36 - 07/11/2014 05:10 AM - Constantin Asofiei

About your update:

1. ClientDriver.start. This code is not reliable:

```
    if (BROKER_LOG.equals(logName))
        BrokerCore.start(config, args);
     else
        ClientCore.start(config, true, args);
```

   You should have a bootstrap config, something like "client:mode:broker=true": if this is set, start the broker, else the client.
   Also, in setClientLog you are suggesting that the broker's name can be set only via command line arguments. I don't like forcing the user to always use a command line argument, the broker's name should be a generic bootstrap config.
2. about the ACLs problem. Check the resource-plugins node in the directory, for the kind of registered resources. Depending on these resources, certain ACLs need to be added. Check #2124-38 for what was needed for MAJIC, when the web clients were added. Also, are you sure the thread from which you are accessing the network service has the proper context set?
   Also, you can debug into the Dispatcher.getRoutingKey and processInbound (for the NetResource usage) - and see where the access is denied.

#37 - 07/11/2014 05:28 AM - Constantin Asofiei

Marius Gligor wrote:

> Here is a simplified skeleton for remote brokers.
>
> 1. As I understood when a broker connect and register to P2J server no information is send to server. The broker configuration is keep in server directory.
>
> - What kind of information should contains a directory entry for a broker?

If think is enough to keep a list of P2J account names, which are managed by this broker. And as Greg noted, "if there are more than one broker (but no user lists), then it will try them all in a round robin until one works."

> - How broker configuration is used by clients and processes?

When the P2J server decides that a new remote client needs to be started, it will either choose a broker based on the account name (if brokers are configured), or it will fallback to the embedded spawner. Also, the P2J processes associated with the broker need to have a broker attribute set, with the broker name.

> - How registered brokers are associated with the directory entries if more than one broker is registered?

Not sure what you mean here. If more than one broker (with the same name) is registered, then it will be treated as a distinct broker instance.

> Could you please offer me a short description regarding how brokers works?

See note 7. Also, look at how the AppServerManager treats the Agents: each Agent thread has a listening loop, where it listens for commands. When a command is received, it executes it. You can use a similar approach:
- each broker has its own thread, with a loop listening for commands
- after the proper broker is picked, the BrokerManager sends a command to the Broker thread
- the Broker thread picks the command and sends it to its remote side (the P2J broker client).

If you think is best that each broker command is to be executed in a different thread, then be careful about the thread's context: if the broker's context is not set, then you will not be able to access the network servers.

Btw, what is your current approach? I wonder if it is not enough to use the broker from the thread where the authentication is performed... and do some context switch/restore to access the broker.

2. When a process is spawned remotely some launch parameters are specific to machines on which the brokers runs:

- classpath.
- native libraries path.

I think these can be determined from the runtime of the P2J client acting as a broker

- spawner location

I think a client config should be used for this - it will be a folder name where the spawner is installed, not the location of the spawn tool.

- the remote spawned client should use the IP address of P2J server instead localhost in order to make a connection with P2J server.

Agreed; but doesn't the P2J broker already know the IP address of the P2J server to which it is connected? Each broker instance will be dedicated to only one P2J Server, and all clients started by it will always connect to that P2J server.

Where should be kept all this information?

At this time, the only config which needs to be set at the client.xml for the broker client is the spawner folder; use a bootstrap config to determine it. Also, if it is missing, you can assume that there is a spawner folder in the same folder as where the P2J Client was started.

#38 - 07/11/2014 05:48 AM - Constantin Asofiei

ca_upd20140707a.zip was committed to bzr rev 10567.

#39 - 07/11/2014 06:41 AM - Marius Gligor

I did some debugs and I found:

1. When I call a client exported method from server, the request is arrive to client side.
In the Conversation.request line 222 the request message is stored and a waiting thread is notified.
Unfortunately nothing happen and the client goes to reader Protocol.Reader.run line 337 waiting to read new objects from network.
No reply message is send back to server and server remains in a waiting state forever Conversation.block line 298.

2. On the server side when a new broker is connected (register) I stored the session and the remote proxy objects in a per thread structure. More interesting using the broker stored session I'm able to obtain the remote network interface and do calls using proxy objects from other contexts.

```
params.remote = (BrokerClientServices) RemoteObject.obtainNetworkInstance(
                                       BrokerClientServices.class,
                                       params.session);
params.remote.ping(params.uuid);
```

The code above woks when is executed at the broker registration time.
When I tried to execute the same code from another context it does not works.

My question is: Where is the problem on server side, client side or both?

#40 - 07/11/2014 06:45 AM - Marius Gligor

- File mag_upd20140711a.zip added

Here are the changes that I did so far.

#41 - 07/11/2014 06:46 AM - Marius Gligor

- File deleted (mag_upd20140711a.zip)

#42 - 07/11/2014 06:49 AM - Marius Gligor

- File mag_upd20140711a.zip added

Here are the latest changes.

#43 - 07/11/2014 07:37 AM - Constantin Asofiei

I think the changes are OK, and it's just a matter to decide what the server-side Conversation thread (for the broker) needs to do. What you are experiencing (server remains in a waiting state forever in Conversation.block line 298) is correct, as it's waiting for the other side to instruct it what to

do. What you need to do is either:

1. if you are calling BrokerClientServices APIs via BrokerParameters.remote from a different thread, you need to switch contexts before doing so. The problem here is that you will need to save and expose the broker's context in i.e. BrokerParameters.context. If you choose this path, you need to secure the access to the broker's context very tight - but looking at RouterSessionManager.setContext, I don't think this is a good choice (security wise). Instead, see next point.
2. implement a server-side command listening loop, by calling a new API, i.e. BrokerServerServices.start, which will start the listening loop and wait for work. When work arrives, the work can be done in a separate thread, created via p2j.security.AssociatedThread - this new thread will inherit the security context from the creator.

#44 - 07/11/2014 07:57 AM - Marius Gligor

I still have something that is not clear for me.

Let say we have 2 brokers in server directory broker1 and broker2. We have a process having broker attribute broker1.
At one moment we have 3 p2j clients acts as brokers which runs on 3 different machines connected to p2j server (registered).
We have no association between brokers name and the connected brokers.
Which broker is selected from registered brokers when we need broker1 or broker2?
Are all the 3 registered brokers equals from the selection point of view?

#45 - 07/11/2014 09:26 AM - Constantin Asofiei

OK, think about it like this:

- we have broker names broker1 and broker2
- broker1 has 3 connected instances, broker2 has 5 connected instances. broker1 has authenticated using P2J process pbroker1 and broker2 has authenticated using P2J process pbroker2. How do you know which broker is associated with a certain context? Each P2J process will have an attribute with the broker's name, in its directory configuration:
  - see how the appserver attribute is set at the P2J process (in dir_schema.xml) and in the directory
  - see the SecurityManager.getAppserver, which determines the appserver name for the current context (from the ProcessAccount.appserver field).
  - see the SecurityManager.getAppserverForProcess, which determines the appserver name for the specified account (from the ProcessAccount.appserver field).
  Similar changes are needed for brokers - you need to mark a P2J process account as belonging to a certain broker, and you need to determine which broker is associated with the current context. Thus:
  - process pbroker1 will have a "broker=broker1" attribute
  - process pbroker2 will have a "broker=broker2" attribute
  Also, each broker name will have these nodes, under the security/server/{default|<server-name>}/ node:

```
<node class="container" name="brokers">
   <node class="broker" name="broker1">
      <node-attribute name="account" value="p2j_account_1"/>
      <node-attribute name="account" value="p2j_account_2"/>
      ...
   </node>
</node>
```

  You need to define a broker class in dir_schema.xml. When reading the broker configuration, use the p2j.util.Utils.findDirectoryNodePath API with the scope parameter set to false.
- lets say we want to launch a web client for P2J user "foo":
  - first step is to determine which broker(s) can handle this P2J user
  - second step is to apply a load-balancing policy over the broker(s) determined on previous step, and choose one, which will launch it.
  - a tricky problem is when the brokers have an empty user list, when load balancing will not be that easy to implement, as the broker's don't know in advance which users can manage.

To answer your questions:

> We have no association between brokers name and the connected brokers.

See above how to make the associations. The broker name should not be at the client-side, it's something server specific; the client-side doesn't even care what name has the broker, all it cares is that the client is a broker. Thus, the remote:broker:name is not needed: to identify the client as a broker, you need to look into the BootstrapConfig instance and check for a client:mode:broker=true setting.

> Which broker is selected from registered brokers when we need broker1 or broker2?

The broker names are not relevant in choosing a certain broker; use UUID's if you want to uniquely identify each connected instance. The broker instance knows its name (as it is something context-specific), which will be used to access the broker configuration from the directory. These configurations can be loaded once upon BrokerManager initialization, to not re-read it from the directory each time.

> Are all the 3 registered brokers equals from the selection point of view?

Yes, only if a broker is configured to handle a specific P2J user, than all its instances can handle it. If the broker doesn't have a account list, than each broker instance needs to check if it can handle the specified account.

#46 - 07/14/2014 02:12 PM - Marius Gligor

- File mag_upd20140714a.zip added

1. I tried many ways to call broker exported services within server processes unfortuinately with no success.
Finally I found a method which works fine, please take a look over the attached code and please let me to offer more details:

First I tried to spawn a task on server side using an AssociatedThread instance when a broker call the register method on server. I checked and the spawned thread runs on the broker context. Doing calls on broker exported interface from the spawned thread works as long as the register method is executed on server. After return from this function the spawned thread is no longer able to make calls on broker exported methods.

Based on this observation I created a special method on server (start). After the broker is register he get an UUID from server which make him unique on the server. Immediately after he call the register API the broker call the special start method on server with the UUID as a parameter. The server exported start method is special because he never return to broker, instead the thread is put on a waiting state. As a result both the broker and the server are waiting and the conversation channel is always open.

What is interesting is that while the broker is waiting for a response on start method call I'm able to call any broker exported method from any server accounts.
If either the server or the broker close the connection (session) they are leave the waiting state. The broker will try automatically to reconnect to server.

2. I did some directory configurations for brokers. Here I have a short question please:
Inside P2J server directory I found two entries for application servers:

```
default
```

and

`standard`

When call

`Utils.findDirectoryNodePath`

with

`scope=false`

(server) will start to lookup on standard server than if path is not found it looks on default server.
In other words we could not have brokers configuration on both servers in the same time.
Is this true? Should we have two or more application servers all having a configuration node for brokers?

#47 - 07/15/2014 04:07 AM - Constantin Asofiei

Marius Gligor wrote:

2. I did some directory configurations for brokers. Here I have a short question please:

The idea behind Utils.findDirectoryNodePath (and similar APIs) is to first look for a per-server configuration, and if it does not exist, use a global (default) configuration. Thus, i.e. if you have a server/standard/brokers node, that node will be used. If this node is missing, it will search for a server/default/brokers node (the global configuration).

About the update: do you think is still needed to associate the P2J process with a specific broker, in the directory? Your current approach is to configure the P2J client upfront as a broker.

#48 - 07/15/2014 04:28 AM - Marius Gligor

The association between defined brokers inside directory and registered brokers (instances) is still no clear for me.
When the broker is connected to server he is authenticated using a P2J account in my case p2j_proc
On directory brokers configuration I have no informations regarding the account used by brokers to authenticate.
The only information we have is the account used by broker extracted from broker context when he ask server to register him as broker.
I think that the broker account should be the link between defined brokers and registered brokers.
What is in fact the rule?

#49 - 07/15/2014 04:43 AM - Constantin Asofiei

Marius Gligor wrote:

> The association between defined brokers inside directory and registered brokers (instances) is still no clear for me.
> When the broker is connected to server he is authenticated using a P2J account in my case p2j_proc

p2j_proc is a process account, right?

> On directory brokers configuration I have no informations regarding the account used by brokers to authenticate.

The brokers node does not need to know which P2J process accounts are configured to start a specific broker. To be able to retrieve the broker name associated with a P2J context, you need to add a new attribute at the P2J process account, named broker; this way you will be able to retrieve the broker's name, from the P2J context - see note 45. You will need to add some new SecurityManager APIs to retrieve the broker name, from the context.

> The only information we have is the account used by broker extracted from broker context when he ask server to register him as broker.

Correct; this is why you need mark the P2J process account with the broker's name.

BTW, something I had wrong: at the moment the user wants to start a i.e. web process, the P2J process doesn't know which P2J account the user will connect as; all it knows is the OS username. Thus, the broker needs to know a list of OS users, not P2J account names. More, when BrokerManager.selectBroker is called in ClientBuilder.remoteStart, you need to pass the OS username as parameter, not the P2J broker account.

Some other notes: you need to add some checks so that a P2J client which is marked as a broker (via bootstrapconfig) will allow startup only if the P2J account is a process account which is marked as a broker.

#50 - 07/15/2014 04:55 AM - Marius Gligor

OK tahnks. What I understood is that the brokers are always authenticated using a P2J process account not a user account.
The process account has an attribute "broker" which is used to link the broker instance to defined brokers.
The user accounts has also a "broker" attribute specifying which broker is used when a remote spawn is made for this account.
Web clients are indeed specials because they are use OS accounts instead P2J accounts.

#51 - 07/15/2014 04:58 AM - Marius Gligor

Or more general. Both process and user account has the broker attribute.
On broker register the attribute is used to link the instance with defined brokers and on remote spawn to select the broker.

#52 - 07/15/2014 05:06 AM - Constantin Asofiei

> What I understood is that the brokers are always authenticated using a P2J process account not a user account.

Correct.

> The process account has an attribute "broker" which is used to link the broker instance to defined brokers.

Correct.

> The user accounts has also a "broker" attribute specifying which broker is used when a remote spawn is made for this account.

I don't understand this one. The idea is that the broker configuration (in the brokers node, the per-server/default configuration) knows a list of OS usernames. The P2J user accounts don't know which broker will be used to launch their P2J client, this is determined at runtime (depending on the load balancing policy, which brokers are registered, etc).

Marius Gligor wrote:

> Or more general. Both process and user account has the broker attribute.

No, only the P2J process can have a broker attribute. We don't allow P2J users to authenticate as brokers (that's why I noted that we will allow startup only if the P2J account is a process account which is marked as a broker.).

#53 - 07/15/2014 08:58 AM - Greg Shah

> Web clients are indeed specials because they are use OS accounts instead P2J accounts.

Batch processes need OS logins too.

#54 - 07/15/2014 01:49 PM - Marius Gligor

I added the broker attribute to process account used by broker to authenticate:

```
<node class="container" name="processes">
  <node class="process" name="p2j_proc">
    <node-attribute name="enabled" value="TRUE"/>
    <node-attribute name="alias" value="p2j_proc_alias"/>
    <node-attribute name="description" value="a P2J batch process"/>
    <node-attribute name="server" value="FALSE"/>
    <node-attribute name="master" value="FALSE"/>
    <node-attribute name="broker" value="broker1"/>
  </node>
</node>
```

I did changes in SecurityManger, SecurityCahche and ProcessAccount classes in order to get broker attribute.
Special rights are mandatory in order to get broker attribute, so I added the follwing node to directory ACL.

```
<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>
```

I'm able to read the broker name attribute in process account and I did the appropriate changes in my code.
Now the link between defined brokers in directory and broker instances is clear.

#55 - 07/17/2014 11:19 AM - Marius Gligor

- File mag_upd20140717a.zip added

Here is an alpha release for remote launcher.

1. When a P2J web or batch client is spawned a list of brokers for this client is create.
First the list is build using the account user name used by client. If no brokers for account user name are found than a list of brokers having no account list defined is create.
Finally if no brokers are found the client is spawn local otherwise remote.

2. The broker scheduler is a simple algorithm which select brokers one by one from the list of broker candidates. When all brokers from list are scheduled the algorithm is reset and the brokers are select again. All brokers have the same priority.
More complex schedulers could be designed by taking in account the system loading where broker runs. Java management package expose a method OperatingSystemMXBean.getSystemLoadAverage() which works on Linux but is not available on Windows. A native code should be used if we want to use such system parameters. I found a native solution http://www.hyperic.com/products/sigar licensed under the terms of the Apache 2.0 license but I didn't do any tests with this library so far.
Using our spawner we cannot manage the spawned P2J clients. When a P2J client is spawn the spawn returns but P2J client is running. We don't know how many P2J clients are still running at one moment.

2. I did the first tests using the current implementation and I found:

a). Web clients can be spawned local or remote using a broker. The cross domain issue cookie is true and occur even on the same machine.
I tested using localhost on P2J server and the machine IP to connect from broker.
- First I used: http://localhost:7443/chui to connect and authenticate to P2j server
- When the P2J is remote spawned using a broker I substituted local host to real machine IP and when the redirection is made http://192.168.1.10:19443/ I've got an HTTP 403 error.

b). Batch clients are working when are local spawned as is already implemented but are not works with brokers.
According to my tests the command is spawned by broker.

```
INFO: {main} Command line arguments: [f:\p2j\spawner\spawn.exe, 0, 3334, localhost, standard, acdffaf4-5249-49
76-b01b-092334280575, -O, client_%pid%_%appserver%_p2j_proc_1405605182688.log]
Iul 17, 2014 4:53:03 PM BrokerCore.spawn()
INFO: {main} Spawn exitValue=0
```

The spawner should create and start a JVM instance than call NativeSecureConnection.command but this not happens. The srv-certs.store file is in the same directory where spawner is located.
I have to do a deep debug inside spawner code to find out why it's not working.

The client.xml file on Windows looks like:

```
node type="client">
  <client>
      <mode broker="true" />
  </client>

  <net>
      <server host="localhost" />
      <server port="3333" />
      <server secure_port="3334" />
      <connection secure="true" />
   </net>

   <security>
      <certificate validate="true" />
      <truststore filename="broker-trust.store" />
      <truststore alias="standard" />
      <keystore filename="broker-key.store" />
      <keystore processalias="p2j_proc_alias" />
      <authentication type="program"/>
   </security>

   <access>
      redacted
   </access>

   <remote>
      <retry count="10" />
      <retry seconds="30" />
      <spawner file="f:\p2j\spawner\spawn.exe" />
```

```
   </remote>
</node>
```

On Linux only the spawner file should be different like:

```
      <spawner file="/mag/p2j/spawner/spawn" />
```

LE: GES redacted the keystore passwords.

#56 - 07/18/2014 05:26 AM - Constantin Asofiei

About 0717a.zip:

1. In cases when the IF/FOR/ELSE block has a single statement, always enclose it in curly braces. i.e. Broker.java:82

   ```
               if (params != null && params.session != null)
                  params.session.terminate();
   ```

   should have been:

   ```
               if (params != null && params.session != null)
               {
                  params.session.terminate();
               }
   ```

   There are other files with the same problem (i.e BrokerCore.spawn, BrokerMAnager.initialize, BrokerParameters.equals/close, BrokerManager.findBroker, BrokerManager.getBrokers, etc).
2. Make sure the methods/fields are ordered properly, in the class (by their access modifiers).
3. Do not use System.err.println to log stuff. Instead, define a static variable like this:

   ```
   /** Logger. */
   private static final Logger LOG = LogHelper.getLogger(<class-name>.class.getName());
   ```

   and use that with a logging level.
4. the header in the new files, needs to be 98 lines long, like this:

   ```
   ** -#- -I- --Date-- --------------------------------Description---------------------------------
   ```

5. Please rename Broker.haveAccounts and Broker.haveBrokers to hasAccounts and hasBrokers.
6. The while on BrokerCore.connect:216 needs to be on its own line. Same for the else if clause in BrokerParameters.equals:95.
7. BrokerParameters, BrokerSpawnParameters, BrokerSpawnResult - please make the fields private and access them via getters/setters.
8. BrokerParameters.commands - is this field used?
9. BrokerSpawnResult.exception - I think you should pass this to the server side, to log it in the server's log.
10. ClientDriver:115 - remove this empty line.
11. BrokerManager.spawn - the synchronization I think is too restrictive. You are allowing only one broker at a time to do its work:

    ```
         synchronized (activeBrokers)
         {
            // select a broker from list.
    ```

```
         BrokerParameters broker = schedule(brokers);
         // spawn P2J client
         BrokerSpawnResult result = broker.remote.spawn(args);

         // on error mark as available for schedule
         broker.scheduled = (result.exitCode == 0);

         return result;
      }
```

I think it should be split this way:

```
      synchronized (activeBrokers)
      {
         // select a broker from list.
         BrokerParameters broker = schedule(brokers);
      }

      // spawn P2J client - the broker is already deactivated, so no one else can use it.
      BrokerSpawnResult result = broker.remote.spawn(args);

      synchronized (activeBrokers)
      {
         // on error mark as available for schedule
         broker.scheduled = (result.exitCode == 0);

         return result;
      }
```

12. BrokerManager.schedule - if no broker is available, why are you resetting them all? Doesn't this break the current approach (which requires that a broker can handle only one request at a time)?

```
      // search for an available broker
      for (BrokerParameters broker : brokers)
      {
         if (!broker.scheduled)
         {
            broker.scheduled = true;
            return broker;
         }
      }

      // all brokers was scheduled reset
      for (BrokerParameters broker : brokers)
         broker.scheduled = false;

      // pick the first broker from list mark as scheduled and return
      BrokerParameters broker = brokers.get(0);
      broker.scheduled = true;
      return broker;
```

    The key question here I think is: why can a broker handle only one request at a time? Is it possible for the broker client to execute the request in a separate thread, and start listening immediately after? Look into how async requests can be sent by the P2J net protocol - at least, the thread needs to be registered with the SessionManager as async.
13. I think you've missed something: if no broker is registered with a certain user, and there are running brokers, then P2J needs to try them all in a round robin until one works.

Also, please make sure to test:

- concurrency issues: launch two or more remote clients (using the same broker) at the same time, to check for concurrency issues.
- edge cases: what happens when all brokers are busy, if the launch failed, if the broker gets terminated during a launch, etc.

  Using our spawner we cannot manage the spawned P2J clients. When a P2J client is spawn the spawn returns but P2J client is running. We don't know how many P2J clients are still running at one moment.

I think the P2J server can track this, if the remote P2J client knows which broker started it (the broker's UUID). When the P2J client starts, it can send to the P2J server this broker UUID, and the BrokerManager can keep track of live P2J clients (via session listeners). But this will only limit us to the

number of P2J clients still active, not the actual system load. And in a machine-load point of view, is best to know the actual machine load, not how many P2J clients are running. So I think we should go with the native approach, but this can be done after we make sure the remote launching works properly.

a). Web clients can be spawned local or remote using a broker. The cross domain issue cookie is true and occur even on the same machine.

Yes, the cross-domain cookie is a issue we need to solve, but can be deferred for another task (as is related to the web client protocol, not necessarily to remote spawners).

Batch clients are working when are local spawned as is already implemented but are not works with brokers.

How do you start them? Via ServerDriver's -b argument?

The client.xml file on Windows looks like:

Please post a minimal and maximal broker configuration for both Linux and Windows (if some nodes are only for windows and some only for linux).

#57 - 07/18/2014 08:15 AM - Marius Gligor

I found why the batch clients does not work with brokers on my current implementation:
The batch client command is stored on the server side in ClientBuilder.pendingCmds having an UUID as a key.
Than a special client is spawned which create a JVM instance , connect to server, get the batch command by UUID and execute the command. All this happen inside the spawner in C code. This special client works local or remote when is spawned by broker. No problems here,

For remote execution I'm using placeholders for classpath, native library path and spawner file. The placeholders are replaced by broker when the client is spawn. The class path and native library are inherited from broker runtime context. The spawner file is get from config.xml file and all localhost strings are replaced with net:server:host parameter read from configuration file. see BrokerCore.spawn
This works fine for web clients but not for batch clients because web clients are always spawned by broker and placeholders are replaced before spawning.

When a batch client command is prepared for remote spawn the command arguments contains placeholders that should be replace on broker side.

Because the batch client is spawn within spawner code and not from broker palceholders remains unchanged and the Java command can not be executed due to errors in command line, arguments that are not understand by Java.

A possible solution is to replace holders on server side not on broker side.
In this case we have to get the values for placeholders from broker, or on broker registration to get this values and keep them on the broker structure.
A solution could be to implements this on spawner code by chancing the C code.
So far I'm thinking to the better solution.
What are your suggestions regarding this issue?

#58 - 07/18/2014 10:11 AM - Greg Shah

> is best to know the actual machine load, not how many P2J clients are running. So I think we should go with the native approach

I think we can defer this part for now. Let's create a separate task for this. I don't think we need it for the current project.

> > a). Web clients can be spawned local or remote using a broker. The cross domain issue cookie is true and occur even on the same machine.
>
> Yes, the cross-domain cookie is a issue we need to solve, but can be deferred for another task (as is related to the web client protocol, not necessarily to remote spawners).

Yes, this can be a separate task, but it is definitely needed for this project. Marius: work on this cross-domain task once this current task is complete.

#59 - 07/18/2014 12:19 PM - Marius Gligor

I found a solution for batch clients which works. When a broker starts he export his JVM arguments and classpath into a file classpath.broker inside spawner directory.
When NativeSecureConnection.command is called within spawner in order to get the batch client command before returning the command arguments the placeholders for JVM
arguments and CLASSPATH are replaced from the exported file classpath.broker. All these happen on broker side.
The only problem could arise if broker has no enough rights to create a file inside the spawner folder. Usually the broker and spawner should reside on the same folder.
I'm thinking also to other kind of interprocess communications instead files suitable for our purposes.

#60 - 07/18/2014 12:22 PM - Marius Gligor

A possible solution might be via an environment variable which is limited on size.

#61 - 07/19/2014 04:47 AM - Constantin Asofiei

Marius Gligor wrote:

> I found a solution for batch clients which works. When a broker starts he export his JVM arguments and classpath into a file classpath.broker inside spawner directory.

I don't think this is a good approach, as it forces each client broker to have its own spawner folder, beside the possible permissions problems. Instead, try this:

- when the broker invokes the spawn tool in server auth mode, it sends as arguments any placeholders which need to be replaced (i.e. classpath, libpath, etc)
- when the NativeSecureConnection class is invoked, you either pass these placeholders as parameters to the command method and prepare the command in the Java code, or the native code in launchP2JClient will prepare the retrieved command, to replace the placeholders. Also, do not allow launch if placeholdrs can't be resolved.

#62 - 07/21/2014 11:58 AM - Marius Gligor

- File mag_upd20140721a.zip added

Here are my latest changes:

1. For batch clients placeholders finally I implemented a solution which use environment variables like for temporary accounts credentials.
Whenever a client is spawn by the broker the P2J server host, JVM arguments and CLASSPATH values are added to the map
of environment variables which contains temporary account variables.
For batch clients the palceholders are replaced on Java code inside NativeSecureConnection class before returning command arguments.
For web clients and batch launchers the placeholders are replaced inside BrokerCore class before running the spawn tool.
It's the simple way to implements placeholders replacements on all cases without doing any changes inside the C code for spawner.
The exported environment variables for P2J server host, JVM arguments and CLASSPATH are not sensitive informations.

2. Regarding the broker scheduler we could have two kind of brokers registered:

```
a). Brokers which have a list of accounts containing user names.
   b). Brokers having no list of accounts which I called "generic"



When a process is about to spawn the following cases could occur:



a). No brokers for user account are found and no generic brokers are registered.
       In this case the process is spawn locally.



b). A list of brokers having at least one broker is registered for user account.
       In this case a broker is selected from the list having the less system loading factor at the moment of
schedule.
       The process is spawn using the scheduled broker. It returns the spawning results regarding the spawn pr
ocess is ended with success or error.



c). If no brokers are registered for user account but we have generic brokers registered than from the list of
 generic brokers
       a broker is scheduled one by one based on the less system loading factor at the moment of schedule and
the broker is used
       to spawn the process remotely. If the spawning ends with success it return the results.
       If an error occur during spawning the next broker from the list is scheduled until no brokers are avail
able.
       The result of the last broker spawn execution is returned when all brokers has been used without succes
s.
```

#63 - 07/22/2014 02:58 AM - Constantin Asofiei

Review for 0721a:

1. the header first line needs to have the Description text centered, as in note 56.
2. NativeSecureConnection:153 // add JVN arguments should be // add JVM arguments
3. why did you remove the ping service? This can be usefull, in i.e. the Administration Console, to test if a broker is alive/responsive, or for latency check. Please add it back, with a more appropriate javadoc.
4. BrokerManager:241 - the while needs to be on its own line
5. BrokerManager.spawn - the javadoc is formatted using a 80 char line instead of 98?
6. ClientBuilder.prepareToLaunch

```
         // localhost placeholder
         for (String arg : getSpawnArguments())
         {
            spawnCmd.add(arg.indexOf("localhost") > -1 ?
                         arg.replace("localhost", "${remote.host}") :
                         arg);
         }
```

   Why do you use "localhost" here? It seems too restrictive: what does this localhost mean? Is it the server's address? Also, is this part of a bootstrap config arg? If so, then search for the key, not for the value.
7. you have some referrences to application server, in Broker:26 and BrokerManager:125 and 128. Please change these to P2J server, to not confuse with the legacy appserver.
8. BrokerManager.spawn - before calling the remote proxy, please check if the broker is still alive. More, you should protect this for errors via a try/catch block - as the remote proxy call might fail. Same with BrokerManager.schedule - protect the remote proxy calls.

Beside the issues above, I don't see anything else wrong with the logic. So please make sure to test edge cases thoroughly, plus concurrency issues (launching two or more clients using the same broker, at the same time - you can place breakpoints into BrokerCore.spawn and BrokerManager.spawn and step through each call side-by-side).

#64 - 07/22/2014 03:47 AM - Marius Gligor

When client is spawned locally I sow that localhost is used as host for server address but when the client is spawn remotely I think that real IP should be used.
For example in ProcessClientBuilder line 83 localhost is used and a remark:

```
// add host; this is hard-coded to localhost because is expected that the in-process JVM
// will connect to the same P2J server which initiated the request (as this is the P2J
// server knowing the command to be executed).
```

Is this true also for remote spawn?

#65 - 07/22/2014 04:26 AM - Constantin Asofiei

Marius Gligor wrote:

> When client is spawned locally I sow that localhost is used as host for server address but when the client is spawn remotely I think that real IP should be used.
> For example in ProcessClientBuilder line 83 localhost is used and a remark:
> [...]
> Is this true also for remote spawn?

No, is no longer true for remote spawn. ProcessClientBuilder line 83 needs to be changed, to send the address on which the server is running. There are some questions here:

- do we assume that both the server and the remote broker are on the same network?
- if they are not on the same network, then we need to determine an address in the same network as the remote broker.

With this in mind, I think is best to change ProcessClientBuilder line 83 so that instead of localhost it uses a placeholder, ${server.host}. This will be changed to:

- when local spawner is used, change it to localhost
- when remote spawner is used, change it to a server address in the same network as the broker client.

#66 - 07/22/2014 05:07 AM - Marius Gligor

The broker an server could be on different networks for example if server is behind a network router.
I think that is better to let the broker to do placeholders replacements using the server address he know.
We could do changes to resolve localhost placeholders exactly where they are used. For example:

```
ProcessClientBuilder.getSpawnArguments(boolean remote)
{
...
   cmd.add(remote ? "${remote.host}" : "localhost");
}
```

#67 - 07/22/2014 05:09 AM - Constantin Asofiei

Marius Gligor wrote:

> The broker an server could be on different networks for example if server is behind a network router.
> I think that is better to let the broker to do placeholders replacements using the server address he know.

Yes, you are correct, the broker already knows the P2J server address to which it has connected. Your approach is OK, but make sure to make the secure port configurable, too - when remote, let the broker replace it with the port from its bootstrap config.

#68 - 07/22/2014 11:03 AM - Marius Gligor

- File mag_upd20140722a.zip added

- % Done changed from 0 to 80

- Status changed from WIP to Review

1. I've fixed the issues reported on previous code review.

2. I did a lot of improvements in code:
- Placeholders are defined and used as string constants.
- Placeholders are set when the parameters are prepared to spawn.
- I added a new placeholder for secure port.
- I added a state field to broker structure which help us to know when we could call functions exported by broker.

3. I did a lot tests and I found no problems so far. However in Linux p2j.jar and srv-certs.store must have root access when spawn batch clients.
If the broker use the same p2j.jar in classpath the broker must be started as root in order to works properly.
In Windows OS no such restrictions exists.

#69 - 07/23/2014 03:24 AM - Constantin Asofiei

Marius Gligor wrote:

> 1. I've fixed the issues reported on previous code review.

Please fix the header in the new files so that instead of this:

```
** -#- -I- --Date-- ----------------Description----------------------------------------------------
```

we have this:

```
** -#- -I- --Date-- --------------------------------Description---------------------------------
```

> 3. I did a lot tests and I found no problems so far. However in Linux p2j.jar and srv-certs.store must have root access when spawn batch clients.

The p2j.jar in the spawner folder (which must have 0550 permissions) is used only by NativeSecureConnection, when connecting to the remote P2J server, to retrieve the actual command. This P2J jar should not be used by the broker client.

Otherwise, the code looks OK, so good work on this one. Please run a single runtime regression and look for obvious abends during client startup/connection.

#70 - 07/23/2014 03:30 AM - Constantin Asofiei

PS: please remove the author tag from the class javadoc - we don't use that.

#71 - 07/23/2014 03:51 AM - Marius Gligor

- File mag_upd20140723a.zip added

I've fixed the header description line and I started the regression test on devsrv01.

#72 - 07/23/2014 03:58 AM - Marius Gligor

- File deleted (mag_upd20140723a.zip)

#73 - 07/23/2014 03:59 AM - Marius Gligor

- File mag_upd20140723a.zip added

Removed @author from javadoc.

#74 - 07/23/2014 10:49 AM - Marius Gligor

- File regression_test.txt added

Here are the results of my regression test. CTRL-C test took 1 hour and 8 minutes and the errors are false negative.
Regarding the MAIN part I'm not sure if the test is completely done.

A severe error occur at one moment:

```
** Shared variable curFac has not yet been created. (392)
```

Comparing results with other results from previous regression tests seems to be completed.
Looking inside the server log I found:

```
07/23/2014 07:05:58 EDT] (com.goldencode.p2j.persist.trigger.DatabaseTriggerManager:INFO) DatabaseTriggerManag
er registered with TransactionManager.
[07/23/2014 07:05:59 EDT] (com.goldencode.p2j.persist.lock.InMemoryLockManager:WARNING) [0000018F:00000506:sym
an] --> local/majic/primary:  cleaning up 2 leaked record lock(s) for exiting context ({purchase_order_item:T=
SHARE [syman:0000018F], temp_sum:T=SHARE [syman:0000018F syman:0000018E deweyw:0000019A deweyw:00000198 syman:
00000189 syman:00000186 syman:00000199]})
[07/23/2014 07:06:00 EDT] (com.goldencode.p2j.persist.lock.InMemoryLockManager:WARNING) [00000198:000004FF:dew
eyw] --> local/majic/primary:  cleaning up 1 leaked record lock(s) for exiting context ({temp_sum:T=SHARE [sym
an:0000018E deweyw:0000019A deweyw:00000198 syman:00000189 syman:00000186 syman:00000199]})
[07/23/2014 07:06:02 EDT] (ErrorManager:SEVERE) {0000019A:00000505:deweyw} ** Shared variable curFac has not y
et been created. (392)
[07/23/2014 07:06:02 EDT] (com.goldencode.p2j.util.ControlFlowOps$ExternalProgramResolver:WARNING) Unable to r
esolve external program.
com.goldencode.p2j.util.ErrorConditionException: ** Shared variable curFac has not yet been created. (392)
        at com.goldencode.p2j.util.ErrorManager.recordOrThrowError(ErrorManager.java:1041)
        at com.goldencode.p2j.util.ErrorManager.recordOrThrowError(ErrorManager.java:934)
        at com.goldencode.p2j.util.ErrorManager.recordOrThrowError(ErrorManager.java:913)
        at com.goldencode.p2j.util.SharedVariableManager.errorHelper(SharedVariableManager.java:766)
        at com.goldencode.p2j.util.SharedVariableManager.lookupWorker(SharedVariableManager.java:742)
        at com.goldencode.p2j.util.SharedVariableManager.lookupVariable(SharedVariableManager.java:337)
        at aero.timco.majic.po.PocoT.<init>(PocoT.java:121)
        at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
        at sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:57)
        at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:45
)
        at java.lang.reflect.Constructor.newInstance(Constructor.java:526)
        at java.lang.Class.newInstance(Class.java:374)
```

I started the tests from the beginning with a Majic conversion.
Could be a conversion issue? Should we restart the MAIN part?
Please let me know.

#75 - 07/23/2014 12:15 PM - Constantin Asofiei

Marius Gligor wrote:

> I started the tests from the beginning with a Majic conversion.
> Could be a conversion issue? Should we restart the MAIN part?

Yes, it might be. Please reconvert and run again.

#76 - 07/23/2014 12:43 PM - Marius Gligor

OK. But first of all I have to fix a bug related to web client remote spawn.
The parameter client:web:host=<web server address> should be replaced with the IP of machine on which the web client is running not with the P2J server.
This is basically the same address on which broker runs.

Another issue that I found. The user working directory

```
<node class="string" name="workingDir">
  <node-attribute name="value" value="./"/>
</node>
```

depends on platform Windows or Linux.
When the server run on Windows and broker on Linux the working directory could be ".\" but this should be "./" on Linux
This is a simple case but could be more complicated like "c\users\mag" in Windows and "/home/mag" on Linux.
I think that on remote spawn should be always "./" in Linux and ".\" on Windows.
Do you have any idea here?

#77 - 07/23/2014 12:55 PM - Constantin Asofiei

Marius Gligor wrote:

> Another issue that I found. The user working directory
> [...]
> depends on platform Windows or Linux.
> When the server run on Windows and broker on Linux the working directory could be ".\" but this should be "./" on Linux

Hm... the working dir is specific to the P2J account being launched remotely, right? Although this looks like is valid only for remote processes, not web clients - see ProcessBuilderOptions.getNode, which reads the configs from the user first and server second. I think this should be made standard, ClientBuilderOptions.getNode APIs should search user first and server second.

For the main issue, we have two choices:

1. make this a limitation, and allow a remote P2J client to be spawned only on windows or only on linux brokers.
2. allow two configurations, workingDir for windows and linux. And use the correct one, depending on which OS the broker is running.

#78 - 07/23/2014 12:58 PM - Marius Gligor

When command is prepared on P2J server doesn't know which broker will be selected and on which platform is running.

#79 - 07/23/2014 01:16 PM - Marius Gligor

I've fixed web server address for remote spawn. The solution is to not send this client:web:host=<web server address> parameter on remote spawn.
When client:web:host=<web server address> is missing the web server will choose the IP addresses and the browser will be redirected correct.
The browser and the web client must be on the same network.

#80 - 07/23/2014 01:24 PM - Greg Shah

> make this a limitation, and allow a remote P2J client to be spawned only on windows or only on linux brokers.

For now, this will be a limitation. Make sure it is included in our documentation updates.

#81 - 07/23/2014 02:04 PM - Marius Gligor

- File mag_upd20140723b.zip added

I fixed the problem regarding web server address by making a single change in WebClientBuilder.addClientOptions line 93
I did tests for both web clients and batch clients with P2J server running on Windows and broker running on Linux using and fount to works.

#82 - 07/24/2014 02:55 AM - Marius Gligor

- File test-results.zip added

- File mag_upd20140724a.zip added

I started regression tests during the night with a full Majic conversion. This time CTRL-C part passes the test without errors.
The MAIN part fails but with false negative errors. However the same errors are found in log files at the end of regression test.
Looking on the log files seems to be a runtime error rather than a conversion error. My changes should not produce such kind of errors.
Looking on my changes I found and fixed a small bug in ClientDriver.setClientLog(String[]) related to stderr redirection for web clients.
But this should not be the cause of the test failure. Nevertheless I restarted the tests without Majic conversion using this new package.

#83 - 07/24/2014 03:15 AM - Constantin Asofiei

Marius, I think the only issue here is that a log is created for the error. I checked my logs and the message related to the "curFac has not yet been created" exists, but without a stacktrace.

#84 - 07/24/2014 03:27 AM - Constantin Asofiei

OK, the stacktrace comes from a code in ControlFlowOps recently added... so you can stop your testing, is not your problem.

I'm good with the 0724a.zip changes. If your OK with the testing you've done, you can release them.

#85 - 07/24/2014 03:57 AM - Marius Gligor

OK. I'm going to do some final tests on my network and than I will commit the changes and distribute to team members.

#86 - 07/24/2014 06:59 AM - Marius Gligor

- % Done changed from 80 to 100

Passed regression tests. Committed revision 10580.

TODO:
1. Implements a JNI solution to determine the system load at a moment in time. The value is queried from each broker and used in scheduled algorithm. The broker which report a less system loading is scheduled. See #2343.

Known limitations at this time:

1. Working directory parameter read from server directory is platform specific. When a process is spawn the working directory is set by default to user home directory. Than a change directory is made using working directory parameter. If working directory parameter is platform specific than server and brokers must run on the same platform type Linux or Windows. However using "." as working directory works on both Windows and Linux and the server and brokers could runs on mixed platforms.

```
<node class="string" name="workingDir">
   <node-attribute name="value" value="." />
</node>
```

2. For web clients the IP of the embedded web server is the IP of machine where the client was spawned. Because the browser is redirected to this address the client and browser must be on the same network.

3. Batch clients are using an OS account to spawn batch processes. The directory settings are OS specific. On Windows we need systemUser and systemPassword while on Linux only systemUser is used. This OS account should exists on all machines on which we want to remote launch batch clients via brokers.

```
<node class="container" name="clientConfig">
  <node class="string" name="systemUser">
       <node-attribute name="value" value="foo"/>
   </node>
    <node class="string" name="systemPassword">
        <node-attribute name="value" value="foopsw"/>
     </node>
 </node>
```

#87 - 07/24/2014 09:15 AM - Greg Shah

Before we close this task, it is important that the "P2J Runtime Installation, Configuration and Administration" book be updated with the full details of how to:

- install a remote launcher (e.g. what files go where)
- the OS dependencies that the remote launcher needs to have satisfied in order to properly run (e.g. security/permissions, environment...)
- the additional OS dependencies that the launched clients require in a remote launch scenario (only those dependencies, if any, that are different than we would already need for a standard client system)
- the P2J configuration of the remote launcher/broker
- the P2J configuration of the server (e.g. the directory changes) for remote launching
- limitations and issues

I'm not sure if I have forgotten something. The objective is to make the documentation complete in regards to the proper implementation of remote launching.

Marius: please make the changes to the book (it is all in bzr). Constantin will review it and will guide any questions.

To start, please note that when you edit an existing file, you should enclose your changes inside brackets like <mag>this is some added text</mag>. This lets the reviewer quickly find your changes and the reviewer can edit/cleanup the text and will remove the enclosing tags as part of the review. If you believe that some or all content is best put into a new chapter, that is fine too. Any new chapters must be added to the master document. Make sure you review the overall book first to know where your changes need to go.

#88 - 07/24/2014 10:06 AM - Marius Gligor

OK. I did the checkout from bzr. I have a first question. If I want to put all in a new chapter this means to create a new odt file and add a link of the document to master.odm file?

#89 - 07/24/2014 10:23 AM - Greg Shah

> If I want to put all in a new chapter this means to create a new odt file and add a link of the document to master.odm file?

Yes, exactly right. Make sure to use the books/templates/book_chapter.ott as the template for that chapter.

#90 - 07/29/2014 09:13 AM - Greg Shah

- Target version set to Milestone 12

- Status changed from Review to Closed

#91 - 11/16/2016 12:12 PM - Greg Shah

- Target version changed from Milestone 12 to GUI Support for a Complex ADM2 App

#92 - 11/30/2016 08:36 AM - Greg Shah

- File deleted (test-results.zip)

#93 - 11/30/2016 08:51 AM - Greg Shah

- File deleted (asadm.pdf)

Files

ca_upd20140707a.zip	8.6 KB	07/07/2014	Constantin Asofiei
mag_upd20140708a.zip	27.8 KB	07/08/2014	Marius Gligor
mag_upd20140710a.zip	26 KB	07/10/2014	Marius Gligor
mag_upd20140711a.zip	32.7 KB	07/11/2014	Marius Gligor
mag_upd20140714a.zip	35.4 KB	07/14/2014	Marius Gligor
mag_upd20140717a.zip	112 KB	07/17/2014	Marius Gligor
mag_upd20140721a.zip	117 KB	07/21/2014	Marius Gligor
mag_upd20140722a.zip	121 KB	07/22/2014	Marius Gligor
mag_upd20140723a.zip	121 KB	07/23/2014	Marius Gligor
regression_test.txt	6.28 KB	07/23/2014	Marius Gligor
mag_upd20140723b.zip	121 KB	07/23/2014	Marius Gligor
mag_upd20140724a.zip	121 KB	07/24/2014	Marius Gligor