User Interface - Feature #4853

implement support for reporting a device-id which is meant to identify a unique client system/browser accessing the FWD server

08/13/2020 12:48 PM - Greg Shah

Status:	Test	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Galya B	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**vendor_id:**	GCD

Description

Related issues:

Related to Base Language - Feature #4855: add a 4GL extension SESSION:SESSION...	**Closed**
Related to User Interface - Feature #4854: origin affinity	**Closed**

History

#2 - 08/13/2020 05:07 PM - Greg Shah

The initial discussion for this feature took place in #4522.

We want to create a device-id which would be a kind of device-unique hash, similar to a UUID. This value must be available in all FWD client types.

- Web Client
  - The device-id is meant to identify a unique installation of a browser for a specific operating system user on a given system. All sessions that come from that browser should appear to be the same device-id.
  - We will create a tuple of:
    - A fingerprint (using the MIT licensed https://github.com/fingerprintjs/fingerprintjs2) that is calculated at the login page (for either virtual desktop mode OR embedded mode). This would never be stored in the browser itself, always calculated.
    - A UUID which if not present in local storage would be generated in Java, sent down to the browser and then would be stored in local storage in the browser. This would be unique across all browser instances that connect to the same FWD server. The local storage used should be for the FWD server URL so that it is available at subsequent logins.
  - This should be sent to the server such that it can be passed as a -D property when the FWD client is launched.
  - One tricky part is that we really would want to have the same UUID stored for both the virtual desktop and for embedded mode. This is important because otherwise the same browser will appear as 2 different devices. #4854 will depend on this.
  - The UUID is meant to help make up for possible collisions in fingerprints, especially in the case where highly standardized client environments are being used. The calculated fingerprint is useful to avoid cases where a malicious user copies the UUID to duplicate a device.
  - It is not perfect, but I think there is no perfect solution at this time. It should handle the vast majority of cases with a minimum of fuss.
- Non-Web Client
  - All other client types would need to implement this from direct Java code, the same implementation should work for all cases.
  - At a minimum, we can generate a UUID and store it in offline storage (see #4286).
  - We could consider if a MAC address, host name or other values should be added. I see some value there but overall, I'm not sure they are needed. A downside of using such values is that they aren't necessarily unique and they can be duplicated independently. So even if we include these we still need the UUID to avoid collisions (malicious or unintentional).
  - The result of a UUID approach would uniquely identify a Java installation for a specific operating system user on a given system.

This value should be present early in the lifetime of the FWD client and it should be provided to the server in the ClientParameters.

We will also add a 4GL extension SESSION:DEVICE-ID which will be a read-only value that returns this to the 4GL code so that customers can use it for their own licensing/session/device management.

#3 - 08/14/2020 11:55 AM - Greg Shah

- Related to Feature #4855: add a 4GL extension SESSION:SESSION-ID attribute which returns a UUID which uniquely identifies a FWD session added

#4 - 08/14/2020 12:51 PM - Greg Shah

- Related to Feature #4854: origin affinity added

#5 - 06/12/2023 01:01 PM - Galya B

- Assignee set to Galya B

- Status changed from New to WIP

#6 - 06/13/2023 05:31 AM - Galya B

> This would be unique across all browser instances that connect to the same FWD server.

Does it mean a browser instance at this point should be distinguishable? All the related tasks seem to create infinite recursion, I feel.

On login form submitted the server can have the knowledge of the OS user, end-user and the fingerprint from the lib (70% accuracy). The UUID it needs to create will duplicate the fingerprint purpose if it's created for a browser instance, because the knowledge about the instance should come from the fingerprint itself. If the UUID is unique for each session, then the same browser instance can have multiple UUIDs. Also saving in the local storage means it won't be accessible from another session in the same browser because of the different ports, but even if we implement persistent ports (mapped to end-users) virtual terminals won't be sharing with embedded. Also persistent ports won't support multiple sessions in the same browser. Am I missing something?

I think we need a more grand global solution to accommodate for all cases...

#7 - 06/13/2023 08:06 AM - Greg Shah

> Does it mean a browser instance at this point should be distinguishable?

Yes.

> If the UUID is unique for each session, then the same browser instance can have multiple UUIDs.

It isn't unique per session. It is created once for the browser instance and would need to be saved in offline storage. Or if offline storage is cleared, then it would be recreated and stored again.

> Also saving in the local storage means it won't be accessible from another session in the same browser because of the different ports,

That is why we are implementing port affinity.

> but even if we implement persistent ports (mapped to end-users) virtual terminals won't be sharing with embedded.

This might require that we store the same device id in 2 different offline storage instances of the same browser so long as the only differences where the path portion of the URL.

> Also persistent ports won't support multiple sessions in the same browser. Am I missing something?

It will if we implement #4856.

#8 - 06/28/2023 06:04 AM - Galya B

Client preferences will not be related to this id, but to the FWD user and will be applied after login / web client spawn.

That's why the sole purpose of the id is to provide info for licensing by identifying different hosts (and different browsers) at one specific point in time. The id doesn't need to be the same after browser update or hardware change, that is it can change in time. It is important only to identify uniquely the host (+browser) at one point in time (for all live sessions).

With java apps running on the client host (standalone gui/chui/batch clients) the host id can be easily obtained and it can uniquely identify the host:

```
  public static void main(String[] agrs)
  throws Exception
  {
    Process process = Runtime.getRuntime()
                              //.exec("wmic csproduct get UUID"); // win
                              .exec("head /etc/machine-id"); // linux
    printResults(process);
  }
```

```
  public static void printResults(Process process)
  throws IOException
  {
     BufferedReader reader = new BufferedReader(new InputStreamReader(process.getInputStream()));
     String line = "";
     while ((line = reader.readLine()) != null)
     {
        System.out.println(line);
     }
  }
```

For web clients fingerprintjs2 is the best we can get in terms of accessing the browser properties that can help identifying a unique system. But there are several major issues with depending solely on the lib:

- a lot of the accessed properties are no longer supported by any modern browser, mainly those related to the hardware on the host and although the lib tries to fetch them, they don't come to play in the fingerprint generation
- (as mentioned in previous notes) highly standardized workstations found in many enterprise orgs (that are likely to be target end-users) will easily produce the same fingerprints

The only identifier of the host we can access through the browser (from the http request) is the external IP of the requester, but in networks using proxies or vpns it's not reliable either.

I've read that many apps deploy the web protocol WebRTC used for media/file sharing to access the IP of the requester. They use STUN servers to get back the external IP and apply licensing logic inside the JS app in the browser. The paid version of fingerprintjs2 is most definitely using STUN server to provide the 99.5% accuracy of visitor identity through calls to their STUN server.

But we don't need the external IP in a js app. RTCPeerConnection provides a **local IP** as well and this is what we can access without any server:

```
    const rtc = new RTCPeerConnection();
    rtc.createDataChannel('');
    rtc.createOffer().then(offer => rtc.setLocalDescription(offer))
    rtc.onicecandidate = (ice) => {
        if (!ice || !ice.candidate || !ice.candidate.candidate)
        {
          rtc.close();
          return;
        }
        const segments = ice.candidate.candidate.split(" ");
        if (segments[7] === "host")
        {
          console.log(`Local IP : ${segments[4]}`);
        }
    };
```

This code is supposed to return a local IP and I imagined a standard IPv4 or IPv6 address in the local network, but it's something else: 58697958-ea5e-4a32-a4cd-7c9364153d85.local (modified, not mine). It's unique in every tab and I guess its format and uniqueness are specified in the protocol. This local address is probably used to identify where the response of the WebRTC server should go exactly.

Pros and cons of using WebRTC browser API:

- it's official web standard https://www.w3.org/TR/webrtc/
- RTCPeerConnection icecandidate event is available in all major browsers https://developer.mozilla.org/en-US/docs/Web/API/RTCPeerConnection
- considered introducing vulnerability in privacy and can be disabled, although I don't think a common user/admin would do it

The combination of external IP, browser fingerprint and RTC local address, app username and session id should be enough for our customers to implement licensing policy in web.

If WebRTC is for some weird reason disabled, we can have a directory configuration for admins to decide if receiving the WebRTC local IP will be a requirement for login and show a message in the browser to guide the user on how to enable it.

#9 - 06/28/2023 06:14 AM - Galya B

Without WebRTC local IP we can't differentiate between web users behind a proxy using standard machines with the same browser version and display.

#10 - 06/28/2023 06:23 AM - Galya B

- Status changed from WIP to Review

#11 - 06/28/2023 08:20 AM - Greg Shah

> Without WebRTC local IP we can't differentiate between web users behind a proxy using standard machines with the same browser version and display.

2 tabs will have different "local IP" as will tabs in 2 different browser instances, correct?

#12 - 06/28/2023 08:21 AM - Greg Shah

- Status changed from Review to WIP

#13 - 06/28/2023 08:21 AM - Galya B

Greg Shah wrote:

> 2 tabs will have different "local IP" as will tabs in 2 different browser instances, correct?

Yes, all tabs everywhere should have different identifiers.

#14 - 06/28/2023 08:39 AM - Galya B

I was originally expecting the local IP to be the host IP and it probably is, when the protocol is used outside the browser (which is a valid case), but now that it represents a tab (at least according to my tests), it may still not be enough for proper licensing. Actually the purpose of the WebRTC related protocols (ICE (Interactive Connectivity Establishment) consisting of STUN and TURN) are to find a direct route to the machine, so maybe I'm just missing how to find the local network IP as well.

#15 - 06/28/2023 08:40 AM - Greg Shah

The idea of a device-id is that it is stable and associated with the system on which the browser runs. The "local IP" seems useful for differentiating tabs but not for establishing a stable device-id.

What is possible to achieve here seems to be less than effective. I'm not sure we will move ahead with this, unless I am misunderstanding things.

#16 - 06/28/2023 08:46 AM - Galya B

The problem is that I don't know enough about networks and network related code, so I'm not completely sure how the protocols are deployed. At the end of the day STUN server might be needed. Actually the purpose of the WebRTC related protocols (ICE (Interactive Connectivity Establishment) consisting of STUN and TURN) are to find a direct route to the machine, so they definitely will know about the host ip in the local network as well. In every case that's what's used by the pro version of the fingerprint library.

Here I've found a simple open-source Java implementation of the STUN server: https://jstun.javawi.de/ , but it will take me quite some time to completely understand it and find what I'm looking for.

#17 - 06/28/2023 08:49 AM - Galya B

WebRTC is the only way to find more network details in the browser. If we don't rely on network details (host / ip), then I don't see any other way to solve the problem, because browsers don't provide other tools for identification. Even if you attach uuid in cookie / localStorage, it needs to know where it is, basically you should be able to first identify the uniqueness of the host, before attaching it.

#18 - 06/28/2023 09:03 AM - Greg Shah

> so they definitely will know about the host ip in the local network as well

Actually, I'm not so sure that is the case. Anything behind a NAT firewall will not expose the local IP at the TCP/IP level. The NAT firewill is the thing that has the knowledge to connect packets on the internal interface to the packets sent/rec'vd via the external interface. It is possible that WebRTC shares the local IP address, but I don't see why it would be needed and it seems like a security risk to share that kind of information externally.

Even the host itself can be a shared IP address for multiple users in the case of a terminal server or other similar environment.

Pause work on the device id for now.

#19 - 06/28/2023 09:12 AM - Galya B

Greg Shah wrote:

> Anything behind a NAT firewall will not expose the local IP at the TCP/IP level.

NAT is exactly what this ICE protocol is working around.

Here is a summary https://temasys.io/guides/developers/webrtc-ice-sorcery/ .

Anyways, it's complicated. Most companies rely on using up and running servers. Microsoft even offer a guide how to set a such server in Azure. But I don't think it's good to expose end-users details via 3rd party server and it will also require budget allocation.

All in all I just wanted to explain what is possible (as far as I understand it).

#20 - 06/28/2023 09:30 AM - Greg Shah

I appreciate it. Please pause work on this specific task.

#21 - 06/28/2023 09:31 AM - Galya B

I would love to indicate I'm pausing it. Unfortunately this is not an option in the Status field.

#22 - 06/28/2023 09:34 AM - Greg Shah

- Status changed from WIP to Hold

#23 - 06/29/2023 11:09 AM - Galya B

It's on hold.. Just to mention what was wrong with my idea: WebRTC changed the specification and doesn't expose the local IP any more (since a year ago). So it somehow bypasses NAT and knows about the local IP, it's just that it's not any more easy to obtain it.

Not working on it.

#24 - 10/16/2023 01:59 AM - Galya B

https://github.com/fingerprintjs/fingerprintjs changed their license and their current version is under commercial license.

#25 - 11/07/2023 08:03 AM - Greg Shah

Since there is no good solution for fingerprinting or even for detecting the network configuration, I think we just ignore those parts. I'm not even sure how much they would help if we could implement them.

What we can easily do is to implement a stored UUID that is created whenever a session starts and there is no existing stored id. This ID would be visible to the 4GL code as SESSION:DEVICE-ID and would be different than the SESSION:SESSION-ID that was implemented in #4855.

A key question to answer, where to store it?

- A simple approach would be to honor the current location used for offline storage (browser local storage or Java preferences).
- A (probably) more useful approach would be to store it "closest to the user":
  - For web clients, this would always be in local storage, regardless of the setting for offline storage.
  - For non-web clients, this would be Java preferences.

We should also consider "security" since this is meant to be a FWD-level feature and not really accessible for edits from the application. On the other hand, there is no easy way to disallow editing the browser's local storage or for anyone with the right OS account, the Java prefs.   We could add some level of cryptographic signature processing to detect integrity, which would probably be a good idea since customers might rely upon this for licensing purposes (see #4522). Knowing that the value has been tampered with would be pretty useful.

#26 - 11/07/2023 08:09 AM - Galya B

I have one blurred idea without clear form yet. It's doable to create a mechanism confirming which web sessions are in the same browser with Broadcast Channel communication between tabs. It could be before or after login. I'm just not sure when and how to present the information to make it useful.

#27 - 11/07/2023 08:11 AM - Galya B

Galya B wrote:

> I have one blurred idea without clear form yet. It's doable to create a mechanism confirming which web sessions are in the same browser with Broadcast Channel communication between tabs. It could be before or after login. I'm just not sure when and how to present the information to make it useful.

This used by itself doesn't violate GDPR and similar policies in contrast to device id, because it's not related to user identifiable data.

#28 - 11/07/2023 08:19 AM - Galya B

Also obviously it doesn't depend on id that can be altered, but real time communication between the live sessions.

#29 - 11/07/2023 09:01 AM - Greg Shah

> It's doable to create a mechanism confirming which web sessions are in the same browser with Broadcast Channel communication between tabs.

This makes sense. I wonder if we should track and report the number of active sessions for each active device id. That seems like a very useful bit of information that might make it easier for licensing decisions.

> It could be before or after login. I'm just not sure when and how to present the information to make it useful.

My first thought is that if it exists already, we probably should report it to the SsoAuthenticator. We would only want to report it if it was properly signed. If it doesn't exist, we would define a new id and pass it so that the SsoAuthenticator could record/track it and we would only store it if the authentication succeeded.

Outside of SSO, I think we would want to pass it to the server if it exists already (for logging purposes). If it doesn't exist, then we set it after a successful login.

#30 - 11/07/2023 09:18 AM - Galya B

Greg Shah wrote:

> > It's doable to create a mechanism confirming which web sessions are in the same browser with Broadcast Channel communication between tabs.
>
> This makes sense. I wonder if we should track and report the number of active sessions for each active device id. That seems like a very useful bit of information that might make it easier for licensing decisions.

My problem is the device id part, if the id has to be unique and persisted and vulnerable to alteration. I'm thinking of a more dynamic approach. We may still need an id to identify the "group" of live session in the same browser. Let's say a dynamic browser id is generated by the first session and spread to the new sessions. It's a js variable, but not a global one, so only the FWD clients know the value and it can't be altered.

Opening the login screen says "hi" in the channel: 1. receives back no answer -> generates the browser id; 2. receives answers -> populates the id.

Clicking the Sign In button says "who's there" in the channel and waits a certain very short time for answers. Answers contain session ids. They get attached to the login request by fwd_sdk.js.

SsoAuthenticator keeps track of live sessions server-side by registering the start / terminate session hooks. It gets the info that a new session has to be started in a browser together with these certain sessions.

And here is where I'm at. What now.

#31 - 11/07/2023 09:22 AM - Galya B

The browser id will be changed the moment all sessions are closed. We can have server-side initiated checks for who lives in the browser of a particular session, if we can manage to give access to the ClientExports of that session.

#32 - 11/07/2023 09:36 AM - Galya B

A few aspects to be taken into consideration are: simultaneous logins; the fact that sessions in the same browser can be under different users (can be tracked in the authenticator) or different apps.

But the most challenging aspect for me is to imagine how this dynamic info can be used. Let's say the server hasn't terminated 5 sessions of that user, but 2 are already closed client-side and one is in a different browser. A new login in the browser with the 2 active sessions will only declare them. There is one more active session in a different browser. If there is no 'single browser' policy, it may get tricky.

#33 - 11/07/2023 10:20 AM - Greg Shah

There should be no association (at the FWD level) between accounts and devices. I don't want to mix the two concepts and I don't think there is an advantage to doing so.

The device id needs to be persistent across browser restarts and past the point where all FWD sessions in a given browser have been closed. That is why I'm focused on a stored UUID that is signed. Signing allows us to refuse forgeries.

The core problem with this idea is that we want to disallow the copying of this UUID into a different browser. That is I would expect the broadcast channel can be used to help check for this case. In other words, using this channel we can know which sessions are on the same browser. If we have an active session on the system, from a given device id, then any other session that attempts to start with that device id must be able to see the existing sessions via the channel. If not, then we know someone is copying the device id and we reject it.

Where this "breaks down" is if the copying happens when all sessions are closed. In that case, we can't detect it, but who cares? So long as the device id can't be used by more than one browser at a time, it doesn't matter if it has been copied. This is no different from copying a hard disk image and only being able to use one of the copies at a time. It is sufficient for our purposes.

#34 - 11/08/2023 12:39 AM - Galya B

Greg Shah wrote:

> There should be no association (at the FWD level) between accounts and devices. I don't want to mix the two concepts and I don't think there is an advantage to doing so.

I was thinking the filtering can be done by the customer.

> The device id needs to be persistent across browser restarts and past the point where all FWD sessions in a given browser have been closed. That is why I'm focused on a stored UUID that is signed. Signing allows us to refuse forgeries.
>
> The core problem with this idea is that we want to disallow the copying of this UUID into a different browser. That is I would expect the broadcast channel can be used to help check for this case. In other words, using this channel we can know which sessions are on the same browser. If we have an active session on the system, from a given device id, then any other session that attempts to start with that device id must be able to see the existing sessions via the channel. If not, then we know someone is copying the device id and we reject it.
>
> Where this "breaks down" is if the copying happens when all sessions are closed. In that case, we can't detect it, but who cares? So long as the device id can't be used by more than one browser at a time, it doesn't matter if it has been copied. This is no different from copying a hard disk image and only being able to use one of the copies at a time. It is sufficient for our purposes.

It's not so simple. On the server you can have two registered sessions in one browser, but then the user closes the tabs and attempts a new login. The connections haven't still timed out, the server hasn't terminated the two sessions. We can't reject the new login. At best we can terminate the missing sessions, ~~but to do so we (or the customer's code) need to know they are under the same user~~. Actually the whole "detect" thing should be part of the customer's code. Otherwise we need to keep track of sessions in FWD and this is stepping over into the licensing policies part.

#35 - 11/08/2023 12:56 AM - Galya B

Greg Shah wrote:

> The device id needs to be persistent across browser restarts and past the point where all FWD sessions in a given browser have been closed. That is why I'm focused on a stored UUID that is signed. Signing allows us to refuse forgeries.

Also this part is problematic. It can't be 'persistent across browser restarts' if the browser has been configured to clear data on exit or the customer likes to delete history from time to time. In Firefox it's easy to setup clean on exit, but since it's not used and it's more difficult in Chrome, it's probably not going to be a common issue.

#36 - 11/22/2023 09:27 AM - Galya B

Can I implement a simple fingerprint myself? It will take a while to figure out how to test the canvas and the audio, but most of the properties are straight forward copied from the browser properties and there are plenty of documents online explaining how fingerprinting should work. The problem with fingerprinting libs is that they get outdated and we should constantly update them, because browser props change, and eventually any lib can change its license. At the same time we have the IP of the user (including behind proxy) server-side and this is a huge advantage for licensing, so we may not need something too refined.

I really don't see a place for the unique uuid, saved in localStorage, because it can be forged easily, also cleaned up easily. If the IP and the fingerprint are the same, the uuid should not change the outcome.

#37 - 11/22/2023 09:32 AM - Galya B

Also I'm always thinking of a flow where the IP and fingerprint don't matter, but it works only when 'one browser' policy is enforced and I'm not sure if we can put such constraints to the feature.

#38 - 11/22/2023 09:39 AM - Greg Shah

Don't work on this right now. I am discussing the requirement with the customer and they may be dropping it on their side.

Even if we do implement this feature, I would hesitate to take on our own fingerprinting approach because of the constant churn of maintenance. We aren't in the business of writing a fingerprint library and as you note, even those who do that "for a living" often disappear over time.

I'm not opposed to exposing some useful state out to the 4GL code (like the web client IP address, FWD client IP address...) which might make it possible to track and manage sessions from the application level. But for now, let me discuss this with the customer that made the original request.

#39 - 11/22/2023 09:42 AM - Galya B

Greg Shah wrote:

> I'm not opposed to exposing some useful state out to the 4GL code (like the web client IP address, FWD client IP address...) which might make it possible to track and manage sessions from the application level. But for now, let me discuss this with the customer that made the original request.

Actually I already did it with 3931a. In SsoAuthenticator implementation in the authenticate methods they have access to the IP (internal with proxy and external) the request came from.

#40 - 02/07/2024 02:30 PM - Greg Shah

- Status changed from Hold to WIP

I spoke with the customer today. I explained the limitations of this concept:

- Any stored device-id can be cleared out from local storage by the user at any time (or automatically when the browser is restarted, if configured as such).
- It can be copied and used in other browser instances, so it is only safe:
  - If we put some checks in place to refuse new sessions (or force exit) that have the same stored device-id but are running on a different browser instance (a.k.a. the "one browser" limit).
  - While a given browser instance is running. The "one browser" limit cannot work when the user exits that browser and then opens a different browser with the same device-id. We won't detect the difference in that case.

The customer accepts all of these limitations. It will work for their purposes.

The customer also asked if JSON Web Token could be helpful. It think it could replace the UUID and has the advantage of being cryptographically confirmable that it is a valid ID. In other words, we don't have to keep a registry of the UUIDs. But I think this is still vulnerable to copying so it would still need to be safeguarded by the "one browser" check.

I understand there are some complexities in implementing the one browser limit (e.g. #4853-34). Let's work through the list and come up with solutions or accept limitations where needed.

#41 - 02/08/2024 04:20 AM - Galya B

- Is this One Browser Policy feature supposed to work for non-SSO?
- Is there supposed to be a limitation to the number of clients from the same browser or any number is acceptable? If there is a limitation, do we allow different users to login at the same time from the same browser?

#42 - 02/08/2024 09:35 AM - Greg Shah

> Is this One Browser Policy feature supposed to work for non-SSO?

Yes

> Is there supposed to be a limitation to the number of clients from the same browser or any number is acceptable?

We have no limits at this time.

> If there is a limitation, do we allow different users to login at the same time from the same browser?

This should be independent of the user. The idea is it is a device level identifier and more than one user might use the same device.

#43 - 02/08/2024 10:25 AM - Galya B

There is one major issue having no browser fingerprint and no unique identifier for the user who is logging in: the device id can be missing with every request and the requests will be allowed. We need a unique identifier for the user to be able to keep track of the relevant sessions. Without SSO, this can be the OS user, but it needs to be unique for each end-user. With SSO, it can be the FWD user, but it needs to be unique for each end-user OR SsoAuthenticator needs to provide a unique id for each end-user.

Greg, do you think customers interested in this feature, would comply to these requirements?

#44 - 02/08/2024 10:57 AM - Greg Shah

> There is one major issue having no browser fingerprint and no unique identifier for the user who is logging in: the device id can be missing with every request and the requests will be allowed.

That is OK. I don't see why this is a problem. We would generate a UUID and store it after authentication.

> We need a unique identifier for the user to be able to keep track of the relevant sessions.

The device-id has nothing to do with the user. I don't want to mix the two concepts.

> Greg, do you think customers interested in this feature, would comply to these requirements?

No. The FWD account of OS account should not be involved.

#45 - 02/08/2024 11:01 AM - Galya B

Greg Shah wrote:

> > There is one major issue having no browser fingerprint and no unique identifier for the user who is logging in: the device id can be missing with every request and the requests will be allowed.
>
> That is OK. I don't see why this is a problem. We would generate a UUID and store it after authentication.

UUID associated with what? Server-side it's not related to any user, client-side it's not related to any browser. If a user logs in from 10 different browsers where no uuid is stored, what stops this?

#46 - 02/08/2024 11:05 AM - Galya B

One browser policy can be implemented without a flaw if sessions are associated with unique users.

#47 - 02/08/2024 11:18 AM - Greg Shah

> > > There is one major issue having no browser fingerprint and no unique identifier for the user who is logging in: the device id can be missing with every request and the requests will be allowed.
> >
> > That is OK. I don't see why this is a problem. We would generate a UUID and store it after authentication.
>
> UUID associated with what?

A single browser instance.

> Server-side it's not related to any user,

Good

> client-side it's not related to any browser.

Why not?

> If a user logs in from 10 different browsers where no uuid is stored, what stops this?

Nothing. Stopping that is not our objective here. What we care about is that the device-id that is found for the session will be different for each browser instance.

#48 - 02/08/2024 11:24 AM - Galya B

What is the goal the customer wants to achieve with this feature? Allow all ten colleagues working from the same office to have each one validated uuid on their browsers, when logging in with the same credentials?

#49 - 02/08/2024 11:41 AM - Greg Shah

Galya B wrote:

> What is the goal the customer wants to achieve with this feature? Allow all ten colleagues working from the same office to have each one validated uuid on their browsers, when logging in with the same credentials?

This isn't about users at all. It is about identifying a unique "terminal" or device in the system, which in our terms correlates to a browser instance.

By being able to report such a thing, they can use it to associate any state or processing with a given device. This will be used to store (device not user) settings in the database, for tracking the number of devices (not users) simultaneously in use (licensing) and helping to achieve the device-specific UUID for #8258.

It is understood that this can be copied, so long as we disallow/exit sessions that have the same UUID on a different browser instance. It is accepted that this can only be detected if both browser instances are running at the same time.

It is understood that this can be cleared, possibly at every browser restart or even by a savvy user explicitly clearing state. All we can do is create and store the UUID when we don't find one already there. That limitation is accepted.

#51 - 02/09/2024 06:54 AM - Galya B

Greg Shah wrote:

> - It can be copied and used in other browser instances, so it is only safe:
>   - If we put some checks in place to refuse new sessions (or force exit) that have the same stored device-id but are running on a different browser instance (a.k.a. the "one browser" limit).
>   - While a given browser instance is running. The "one browser" limit cannot work when the user exits that browser and then opens a different browser with the same device-id. We won't detect the difference in that case.
>
> I understand there are some complexities in implementing the one browser limit (e.g. #4853-34). Let's work through the list and come up with solutions or accept limitations where needed.

Implementation steps:

- On login attempt all cookies received with the request are checked for a cookie with a device UUID. If no device UUID cookie is present, UUID is generated and attached to the response.
- A new config in directory enables the One Browser Policy feature.

The following is applicable only with the new config enabled:

- The server starts recording all active web client sessions in a disposable / in-memory DB table and registers listeners for the start / terminate session events. Session's data is simply the session id and device id.
- If a device UUID is found in the request cookies:
  - and no other sessions are live in the records table, no additional actions required;
  - and there are recorded sessions with the same device id in the DB table, then the client receives a list of all live sessions on spawn and checks if all of the live sessions are responding in the same browser. The sessions not responding to ping in the broadcast channel are immediately terminated server-side. The last spawned client continues execution as normal.

#52 - 02/09/2024 07:12 AM - Galya B

For #8258 and for the sake of differentiating between devices the device id should be GUID. The UUID is generated and encrypted with a public key, then sent to the browser, received on login attempt and decrypted with the private key. Both operations are on the server and we don't need JWT, but a simple cookie with the encrypted value. If the id is not to be stored in a persistent way, then we can check if the format of the decrypted value is the one of the generated UUID, but I'm not 100% sure this is enough.

This will require the generation of public-private key pair and their registration in directory for the sole purpose of generating device ids. But this also doesn't sound right, having both the public and private key in one place.

#53 - 02/09/2024 08:02 AM - Greg Shah

I think the design described in #4853-51 is close, but I don't think the "last spawned client" should "win" (stay alive). The existing sessions should stay alive and the new session should exit. We should post a message (with a uniquely generated random payload) to the existing sessions **on the server** (perhaps using cross-session pub-sub), they should each post that message across the broadcast channel to the new session. I realize it is more complicated this way, but from the user/admin's perspective it is more reasonable. The existing browser instance is the valid one and any new browser instance is rejected.

> If the id is not to be stored in a persistent way, then we can check if the format of the decrypted value is the one of the generated UUID, but I'm not 100% sure this is enough.

It is enough to verify the signature of the GUID. At that point we know it was generated by our server's private key.

> This will require the generation of public-private key pair and their registration in directory for the sole purpose of generating device ids.

The server already has a key pair generated and registered in the server. We should just use that.

> But this also doesn't sound right, having both the public and private key in one place.

The public key is sometimes shared as a certificate that can be used by clients to verify connections with the server. When this is done, there is a separate truststore used. I don't see an issue with the server having access to both its public and private keys. That is pretty normal. Those are stored in the directory, which must be very carefully secured (for many reasons).

#54 - 02/09/2024 08:10 AM - Greg Shah

What about the non-web client cases? We obviously don't need to implement the one browser approach. How do we implement a reasonable eqivalent of the device-id?

#55 - 02/09/2024 09:23 AM - Galya B

Greg Shah wrote:

> I think the design described in #4853-51 is close, but I don't think the "last spawned client" should "win" (stay alive). The existing sessions should stay alive and the new session should exit.

I've spoken before about the timeout and the fact that it's pretty common to have live sessions on the server, that are closed in the browser. Not allowing a new client to spawn if the other tabs are closed, will be interesting. Cookie hijacking is the only issue in the reverse scenario, but we can open a warning message 'Your cookie has been stolen by another browser' (with a gif of browser running after a cookie) in the other browsers, before closing the old sessions.

> We should post a message (with a uniquely generated random payload) to the existing sessions **on the server** (perhaps using cross-session pub-sub), they should each post that message across the broadcast channel to the new session.

Broadcast channels are safe, because they can be used only by the same origin, no need of fancy payload. Also what is then the validation **client-side**? The actual validation is to check if one tab associated with session id is in the same browser with other tabs associated with session ids. In your case what and when are the clients returning? What do we do with inconsistent results? What do we do with no results? What latency are we adding to the client spawning process with such approach?

> > If the id is not to be stored in a persistent way, then we can check if the format of the decrypted value is the one of the generated UUID, but I'm not 100% sure this is enough.
>
> It is enough to verify the signature of the GUID. At that point we know it was generated by our server's private key.

The device id can be encrypted, what's the point of introducing another value that is encrypted? How do we validate it's generated by that key if not by decrypting it? So why not decrypting the device id? Also in standard JWT the signature can be reproduced based on data and this is how it's validated without being decrypted, but we don't have unique data to associate with each device to be able to validate it. In standard JWT the signature is user data that doesn't change often (user id, name, privs) and it also expires and get replaced when the payload used for the signature changes. This data is not decrypted, but it's reproduced on the server with the same data to be validated. So signature - not our case.

> The public key is sometimes shared as a certificate that can be used by clients to verify connections with the server.

The public key is used to encrypt the ids. If such is the case I'm not sure if this is a good idea to reuse the same key for securing the new feature, where no party besides the server should know about how this id is generated.

#56 - 02/09/2024 09:59 AM - Galya B

Greg Shah wrote:

> What about the non-web client cases? We obviously don't need to implement the one browser approach. How do we implement a reasonable eqivalent of the device-id?

Linux and Windows both have a unique id for the device and it can be read cmd line. But now thinking about it again, I'm not sure if admin privs will be needed. I need to review it I guess.

#57 - 02/09/2024 10:26 AM - Greg Shah

> > I think the design described in #4853-51 is close, but I don't think the "last spawned client" should "win" (stay alive). The existing sessions should stay alive and the new session should exit.
>
> I've spoken before about the timeout and the fact that it's pretty common to have live sessions on the server, that are closed in the browser.

If a session does not respond to messages, it will naturally exit.

> Not allowing a new client to spawn if the other tabs are closed, will be interesting.

To do otherwise is to allow a stolen device-id to be used to kill a valid user's sessions. We must presume that the existing sessions are the valid sessions.

> Cookie hijacking is the only issue in the reverse scenario, but we can open a warning message 'Your cookie has been stolen by another browser' (with a gif of browser running after a cookie) in the other browsers, before closing the old sessions.

Yes, and that hijacking can cause pretty bad results. Considering this is a web system, that is not OK.

> > We should post a message (with a uniquely generated random payload) to the existing sessions **on the server** (perhaps using cross-session pub-sub), they should each post that message across the broadcast channel to the new session.
>
> Broadcast channels are safe, because they can be used only by the same origin, no need of fancy payload.

The advantage is that these are harder to spoof. And the checking should be done on the server side, so that the client neither generates the "secret" nor checks the "secret".

> Also what is then the validation **client-side**? The actual validation is to check if one tab associated with session id is in the same browser with other tabs associated with session ids.

I don't want a client-side check.

In your case what and when are the clients returning?

I'm not clear what you mean by "returning". I would think that the message that is broadcast would be sent up to the server to checking.

If the id is not to be stored in a persistent way, then we can check if the format of the decrypted value is the one of the generated UUID, but I'm not 100% sure this is enough.

It is enough to verify the signature of the GUID. At that point we know it was generated by our server's private key.

The device id can be encrypted, what's the point of introducing another value that is encrypted?

I don't want encryption, I want the UUID to be digitally signed. That signiture can then be checked for validity. Only an entity with the private key could have created such a value.

We don't have unique data to associate with each device to be able to validate it. In standard JWT the signature is user data that doesn't change often (user id, name, privs) and it also expires and get replaced when the payload used for the signature changes. This data is not decrypted, but it's reproduced on the server with the same data to be validated. So signature - not our case.

It is enough to sign the UUID. That means that the UUID could only have been generated by that specific server (or at least by an entity that has that private key).

We could add more data like a timestamp or whatever but I don't think we need it.

#58 - 02/09/2024 10:27 AM - Greg Shah

Galya B wrote:

> Greg Shah wrote:
>
> > What about the non-web client cases? We obviously don't need to implement the one browser approach. How do we implement a reasonable eqivalent of the device-id?
>
> Linux and Windows both have a unique id for the device and it can be read cmd line. But now thinking about it again, I'm not sure if admin privs will be needed. I need to review it I guess.

Sounds reasonable. I prefer a native call to a shell command.

#60 - 02/15/2024 09:40 AM - Galya B

I'm having troubles to find the correct method to terminate a session (not the current one) from a non-related context. Is it actually doable?

#61 - 02/15/2024 10:19 AM - Galya B

Galya B wrote:

> I'm having troubles to find the correct method to terminate a session (not the current one) from a non-related context. Is it actually doable?

I've found SecurityManager.SessionSecurityManager.killSession used by Admin Console and it works as needed. I'll make it public with caller checks.

#62 - 02/15/2024 10:36 AM - Galya B

- % Done changed from 0 to 70

#63 - 02/16/2024 09:12 AM - Galya B

I need to determine if a session is actually still alive, when not found in the same browser as a newly spawned client. It has to do with WebSocket timeouts server-side. Currently the defaults are:

WebClientConstants:

```
   /** The default maximal idle time of the configured websocket */
   int MAX_WEB_SOCKET_IDLE_TIMEOUT = 90000;
   /** The default watch dog timeout */
   int WATCH_DOG_TIMEOUT = 120000;
```

GuiWebDriver:

```
         timeout   = config.getInt("client", "web", "socketTimeout", MAX_WEB_SOCKET_IDLE_TIMEOUT);
         wdtimeout = config.getInt("client", "web", "watchdogTimeout", WATCH_DOG_TIMEOUT);
```

ChuiWebSimulator:

```
         timeout   = config.getInt("client", "web", "socketTimeout", -1);
         wdtimeout = config.getInt("client", "web", "watchdogTimeout", -1);
```

Jetty javadoc says Session.setIdleTimeout:

> Set the number of milliseconds before this conversation will be closed by the container if it is inactive, ie no messages are either sent or received.

When a negative value is set, Jetty's default is used, that is 30 sec idle timeout.

So the socketTimeout counts from the last message sent or received. After the connection times out, the watchdog waits watchdogTimeout amount of time before closing the process or being canceled by a reconnect.

I'm not sure what is considered 'message sent' that resets the timeout. So I guess the only way to check the actual state of a connected websocket server-side is to send a new message and count a short timeout real-time. Since the check needs to be done by another server session, I'll need to save a reference to the network interface ClientExport of each web client, which is not ideal.

#64 - 02/18/2024 12:30 PM - Greg Shah

> I'm not sure what is considered 'message sent' that resets the timeout.

Sergey: Can you answer this?

#65 - 02/19/2024 05:00 AM - Galya B

- Status changed from WIP to Review

- % Done changed from 70 to 100

4853a r14999 based on trunk r14989 ready for review. What has been done:

- SESSION:DEVICE-ID added:
    - device ID in the browser is an UUID generated on the server and attached to GET / POST login responses. The cookie is digitally signed on the server. The signature and the device id are available in the cookie for verification server-side on login attempts. The cookie expiration is the max supported by modern browsers or 400 days and is refreshed with each GET / POST login response (that's the closest to permanent). When the cookie is deleted, a new device ID is generated.
    - device ID in non-web clients is the client process OS device ID (generated by Linux/Windows). It's sent to the server with ClientParameters.
    - the device ID is made available to SsoAuthenticator on authentication.
- A new boolean config webClient/oneBrowserPolicy supported to enable One Browser Policy feature.
    - When One Browser Policy is enabled, all live web sessions are recorded in-memory in WebDriverHandler and a InitTermListener is registered to keep the records correct. Changes to StandardServer make multiple session listeners supported.
    - On a new web client spawned the following method chain is called server-side: StandardServer.standardEntry -> StandardServer.setupClient -> StandardServer.enforceOneBrowserPolicy. Find enforceOneBrowserPolicy method javadoc for detailed explanation how the feature works, basically verifying that all web sessions with the same device id are in the same browser and terminating the offending sessions, if any.
    - BroadcastChannel messages and the corresponding methods reworked to standardize the search for sessions in the browser. All clients register BroadcastChannel with the name of the app legacy-system/pkgroot. Then each instance (in each tab) keeps a reference to the session UUID, basically all clients are communicating on the same channel, but knowing their own identities.

I'll be doing more testing next.

#66 - 02/20/2024 05:04 AM - Galya B

Manual testing plan to be executed:

- Run a procedure with content MESSAGE SESSION:DEVICE-ID VIEW-AS ALERT-BOX and confirm the value for the device id is:
    - (in web) the same as in the cookie FWD-Device-ID (before the divider for the digital signature).
    - (Linux) the same as in head /etc/machine-id.
    - (Windows) the same as in wmic csproduct get UUID.

- Confirm the device id stays the same between sessions.
- Delete the cookie FWD-Device-ID, open a new web session and confirm SESSION:DEVICE-ID has a new value. Test with OS login, SSO login, auto-login and forked session (the pop-up should be blocked to test the last case).
- Enable SSO, add a debug breakpoint in SsoAuthenticator.authenticate to confirm that the param LicensingData has browserDeviceId set to the expected value.
- Verify Terminate session works in Admin Console.
- Enable webClient/oneBrowserPolicy in directory, open Admin Console -> Sessions for reference what sessions are alive and do the following tests:
  - Start a web session, then start a second one. Make sure none of them are terminated and continue work. Can repeat with a few more sessions simultaneously open in the same browser without exiting any of them.
  - Start a web session, close the tab, confirm its presence and id in Admin Console, then open a new session. The expected behavior is the second session to show a blank screen with the text "Verifying Browser Eligibility..." for several seconds (webClient/broadcastChannelPingTimeout or 10 by default) and then the first session to be terminated and to disappear from the Admin Console and the second session to continue work.
  - Start two web sessions, close one of the tabs, open a new session. Confirm the last session starts without waiting a timeout and without displaying "Verifying Browser Eligibility...".

#67 - 02/20/2024 10:31 AM - Galya B

Test plan executed. Fixes in r15000.

Ready for review.

#68 - 02/21/2024 06:04 AM - Greg Shah

Code Review Task Branch 4853a Revisions 14989 through 15000

These are very good changes.

1. I assume you have already tested the simple "one browser" case of opening a session in browser A (Chrome), leaving that session open and then trying to open another session in browser B (e.g. Firefox), with the B session rejected. I didn't see that explicitly in the test plan, which is why I mention it.

2. I would prefer to not expose SessionUtils.setDeviceId() but the separation of the creation of the device-id makes that harder. Do you have ideas on how we could make it read-only to the code while allowing the existing architecture to continue working as designed?

#69 - 02/21/2024 06:04 AM - Greg Shah

Sergey: Please review.

#70 - 02/21/2024 07:36 AM - Galya B

Greg Shah wrote:

> 1. I assume you have already tested the simple "one browser" case of opening a session in browser A (Chrome), leaving that session open and then trying to open another session in browser B (e.g. Firefox), with the B session rejected. I didn't see that explicitly in the test plan, which is why I mention it.

I can't bypass the SEC_ERROR_UNKNOWN_ISSUER self-signed certificate issue in Firefox. Some suggest that it needs be re-issued without the constraints. I tried to open a session from a VM, but it also doesn't work, because of the client url being localhost. I need to either install a less known browser supported in Linux, or do more gymnastics to configure the server on a new host address accessible for the VM.

#71 - 02/21/2024 07:52 AM - Greg Shah

What about incognito mode? Doesn't that act as a separate browser?

#72 - 02/21/2024 07:59 AM - Galya B

Greg Shah wrote:

> What about incognito mode? Doesn't that act as a separate browser?

This works. BroadcastChannel is not shared with incognito. The new session in the other mode gets terminated as expected.

#73 - 02/28/2024 03:00 AM - Galya B

Greg Shah wrote:

> Sergey: Please review.

Sergey, would you please review. This task is blocking the progress of #8258.

#74 - 02/28/2024 05:23 AM - Sergey Ivanovskiy

Does it make sense to read process.getErrorStream() too when we run an external program that can write results to the output and error steams?

```
=== modified file 'src/com/goldencode/p2j/main/ClientCore.java'
--- src/com/goldencode/p2j/main/ClientCore.java    2023-11-27 13:44:13 +0000
+++ src/com/goldencode/p2j/main/ClientCore.java    2024-02-21 12:56:41 +0000
@@ -2,7 +2,7 @@
 ** Module   : ClientCore.java
 ** Abstract : core client loop which handles init and connects to server
 **
-** Copyright (c) 2011-2023, Golden Code Development Corporation.
+** Copyright (c) 2011-2024, Golden Code Development Corporation.
 **
 ** -#- -I- --Date-- ----------------------------------Description----------------------------------
 ** 001 GES 20110928 Moved core logic here from ClientDriver so that the same
@@ -90,6 +90,7 @@
 ** 049 TT  20230915 Use PostInitErrorManager to properly show errors after initialization.
 ** 050 TT  20231129 Added the propath client parameter.
 ** 051 GBB 20231130 ClientParameters.driverClass replaced by driverName.
```

```
+** 052 GBB 20240214 Add OS device id to client params.
 */
.....................................................................................................
+
+   /**
+    * Returns the OS specific device ID.
+    *
+    * @return   The found device ID.
+    */
+   private static String getOsDeviceId()
+   {
+      String deviceId = null;
+
+      boolean isWin = System.getProperty("os.name").contains("win");
+      try
+      {
+         if (!isWin)
+         {
+            // in the format 9f5c9c65b34d4055a746f69bd86a4b8b
+            Process process = Runtime.getRuntime().exec("head /etc/machine-id");
+            Scanner scanner = new Scanner(new InputStreamReader(process.getInputStream()));
+            deviceId = scanner.hasNext() ? scanner.nextLine() : null;
+         }
+         else
+         {
+            // in the format 18B1FBB5-2728-701A-A89A-D8BBC19F009C
+            Process process = Runtime.getRuntime().exec("wmic csproduct get UUID");
+            BufferedReader reader = new BufferedReader(new InputStreamReader(process.getInputStream()));
+            deviceId = reader.lines()
+                             .map(line -> line.replaceAll("\\s", "").replace("UUID", ""))
+                             .reduce("", String::concat);
+         }
+      }
+      catch (Exception e)
+      {
+         LOG.warning("Unable to find device ID.", e);
+      }
+
+      return deviceId;
+   }
 }
```

Searching System.getProperty("os.name") through the java FWD project gives

```
SwingGuiDriver.java
1,238: private static String OS = System.getProperty("os.name");
EnvironmentOps.java
2,881: fallback = System.getProperty("os.name");
OsPropertiesDaemon.java
116: return System.getProperty("os.name").toLowerCase();
NormalizePath.java
213: if (System.getProperty("os.name").startsWith("Windows"))
PlatformHelper.java
79: System.getProperty("os.name").toLowerCase().indexOf("win") >= 0;
```

so I propose to use PlatformHelper to get this check isWin = System.getProperty("os.name").contains("win");

#75 - 02/28/2024 06:06 AM - Galya B

Sergey Ivanovskiy wrote:

> Does it make sense to read process.getErrorStream() too when we run an external program that can write results to the output and error steams?

From functional point of view things won't change, but we can add a log record for the error.

> Searching System.getProperty("os.name") through the java FWD project gives
> [...]
> so I propose to use PlatformHelper to get this check isWin = System.getProperty("os.name").contains("win");

Fixed.

Both changes in r15001.

#76 - 02/28/2024 07:50 AM - Greg Shah

> I would prefer to not expose SessionUtils.setDeviceId() but the separation of the creation of the device-id makes that harder. Do you have ideas on how we could make it read-only to the code while allowing the existing architecture to continue working as designed?

Any ideas on this?

#77 - 02/28/2024 07:54 AM - Galya B

Greg Shah wrote:

> > I would prefer to not expose SessionUtils.setDeviceId() but the separation of the creation of the device-id makes that harder. Do you have ideas on how we could make it read-only to the code while allowing the existing architecture to continue working as designed?
>
> Any ideas on this?

Ah :) I forgot it. I'll get to it.

In the meantime I'm not sure if Sergey has more to add to the review.

#78 - 02/29/2024 04:39 AM - Galya B

4853a r15002 setDeviceId to accept only one (the initial) value for the device id in each session.

Can this be merged now?

#79 - 02/29/2024 06:24 AM - Greg Shah

- Status changed from Review to Merge Pending

That solves the problem nicely. You can merge to trunk now.

#80 - 02/29/2024 06:46 AM - Galya B

- Status changed from Merge Pending to Test

4853a was merged to trunk as rev. 15017 and archived.

#81 - 03/13/2024 08:41 AM - Robin Harris

Whatcha,

Glorious news that this is in "test"! what 4GL code changes do we need to do to make use of this functionality?

#82 - 03/13/2024 08:52 AM - Galya B

You can find the device id with SESSION:DEVICE-ID and the code needs to be converted with FWD r15017 or later.

#83 - 03/13/2024 08:58 AM - Robin Harris

Galya B wrote:

> You can find the device id with SESSION:DEVICE-ID and the code needs to be converted with FWD r15017 or later.

Thank you :D