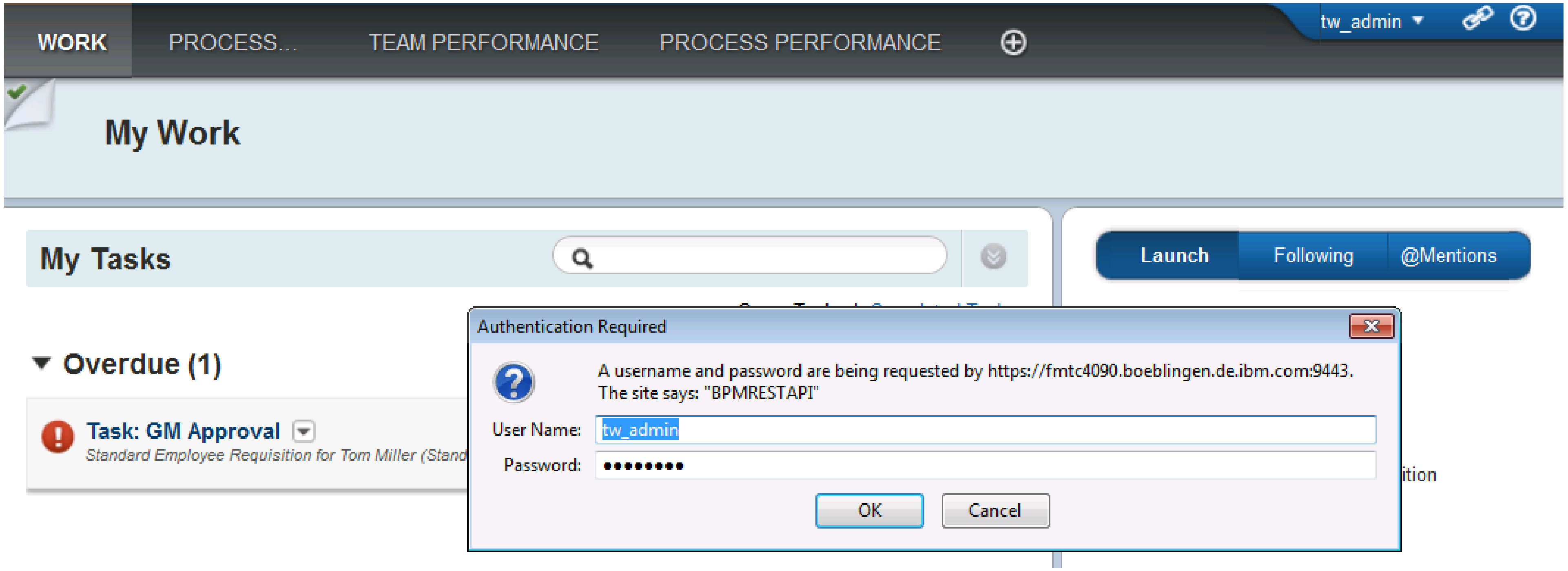
Using BPM as an end user, you have probably seen the basic authentication prompt after your authentication token (LTPA) expired:



Authentication Required

A username and password are being requested by https://fmtc4090.boeblingen.de.ibm.com:9443. The site says: "BPMRESTAPI"

User Name: tw_admin

Password: ••••••••

OK Cancel

This is very annoying for users and a particular concern of mine because of subtle security implications that are not obvious for most end users. Without getting into too much detail here, be aware that if you entered your credentials into this window once, the only way of logging off completely is to close all windows of this browser. Browsers cache userid and password of such basic authentication challenges, keyed with the hostname and realm name.

Why does this prompt appear and why doesn't the product have a fix for this yet?

This authentication prompt is the browser's reaction to a server response code HTTP 401 with an additional HTTP response header WWW-AUTHENTICATE with a value of BASIC. Basic Authentication is broadly supported by all types of clients: browsers as well as fat clients (such as eclipse or SOAP UI) or command line etc. Therefore it appeared to be a good idea to support basic authentication for the product REST APIs which can be consumed by all types of clients. Meanwhile, Process Portal is probably the most prominent client of the REST APIs and periodically polls for new tasks.

If you paste a REST API URL into your browser without being already authenticated to BPM, you see exactly this dialog, too.

Other web modules in BPM, such as Process Portal, are configured with form based authentication so that unauthenticated requests are redirected to a login page which collects your userid and password and then POSTs them to a WebSphere Application Server URL (as defined in the JEE spec) for verifcation. WebSphere will respond with an LTPA cookie that your browser needs to submit with every request. This cookie basically tracks your authentication state. If present and valid, the request is authenticated - and if the request happens to hit the REST API which is configured with basic auth - it is still authenticated and the basic auth challenge is not returned. So, during regular operations, this "just works". However, if your LTPA token expires (after 120 min by default) the next request to the server will need re-authentication and if this request happens to be one for the REST API, there is the basic auth challenge. The other frequently accessed web module from Process Portal is the comet.d notification framework - which pops up the little black boxes whenever you have a new task. It, too, is configured with basic authentication.

Being a published API that supports all types of clients, the REST API cannot simply drop support for basic authentication. As some clients don't send basic authentication credentials pre-emptively, the challenge is required. So in order not to break all types of possible custom clients, the REST API is what is today. Sorry. We are working on that.

What can you do today, before this is improved in some future release of the product?

Fortunately, WebSphere Application Server has sufficient plug points (and BPM supports you in using them) that allow customization of the authentication process. In the particular case of HTTP requests, the plug point of choice is a Trust Association Interceptor (TAI). For a full explaination, the best resource on the web is: IBM WebSphere Developer Technical Journal: Advanced authentication in WebSphere Application Server

When an unauthenticated request for a protected resource is received, WebSphere calls all configured TAIs to see if any one of them can handle authentication for this request. The first to say yes, will be given control to negotiate with the client and ultimately assert a user identity.

To work around the issue I described above, we don't actually need to authenticate someone. All we need to do is avoiding the HTTP 401 response (with the WWW-AUTHENTICATE header) and send some other signal instead. This other signal should be good enough to tell JavaScript clients in browsers that they need to handle an error situation - unless the browser does so on its own.

Find below sample code that

- declares itself the responsible TAI for unauthenticated requests to either the REST API or the comet.d framework (based on URL patterns)
  - fine tuning can be applied by looking at the user-agent header which is a good signal to recognize browsers
  - additional fine tuning is possible: by default the TAI ignores requests if an Authorization header is present (that is, some client pre-emptively sent credentials)
- redirects to a pre-configured URI, e.g. /ProcessPortal so that the regular authentication flow can be triggered

With sample TAI in place, there should be no more basic auth challenges for end users. Custom clients that do not run in a browser can be challenged. I try to reduce the impact as much as possible by looking at the user-agent header which is sometimes omitted by non-browser clients, but at least it is very characteristic.

browserfriendlytai.jar|View Details

This is both, source and compiled code. In case, this is in any way useful to you, you can

1. Place the .jar file into <INSTALL_ROOT>/lib/ext of all nodes in your BPM environment
2. Add the TAI using scripting:
   AdminTask.configureInterceptor('[-interceptor com.ibm.bpm.sample.BrowserFriendlyTAI -customProperties

["com.ibm.bpm.sample.browserfriendlytai.useragentheaderregex=.*","com.ibm.bpm.sample.browserfriendlytai.defaultRedirecturltarget=/ProcessPortal"]]')

There are 3 custom properties in that can be used to configure this sample TAI:

- com.ibm.bpm.sample.browserfriendlytai.skipIfAuthorizationHeaderIsPresent: default is true, that is, if a HTTP request header "Authorization" is present, the TAI does not act on the request
- com.ibm.bpm.sample.browserfriendlytai.useragentheaderregex: a regular expression to check the user-agent header against (if present). Note that the TAI does not act if there is no user-agent header. Browsers tend to send this header. The recommended value therefore is ".*" which matches everything
- com.ibm.bpm.sample.browserfriendlytai.defaultRedirecturltarget: the URL to which the TAI will redirect instead of responding 401. The recommended value is "/ProcessPortal" as this will fit end user expectation.

Alternatively, you can use the WAS admin console:

1. Security > Global Security > expand "Web and SIP security" > Trust Association
2. Interceptors > Add a new TAI using the "New..." button
3. Provide the class name as well as custom properties
4. Save and synchronize changes and restart the entire cell.

This is just an example and it comes with some caveats:

- In Process Center environments, Web based Process Designer relies on comet.d and its auto-save feature fails silently

- If the request that is now redirected to /ProcessPortal originates from an iframe (e.g. when displaying a Dashboard), there are some UI rendering issues, which you could avoid with custom login pages

The latter can probably be avoided by using clickjacking protection JS in the login page theme:

```
<script type="text/javascript">
   if (self === top) {
       var antiClickjack = document.getElementById("antiClickjack");
       antiClickjack.parentNode.removeChild(antiClickjack);
   } else {
       top.location = self.location;
   }
</script>
```

The code is only tested casually on a development driver of the product - but it is a start to give you an idea of what can be done.