Goal: notes about setup of Oracle ORDS running in standalone mode and Apache http server (httpd) used as a reverse proxy.

Contenu Masquer

1 Reminders:
2 Debugging
2.1 At httpd level
2.2 At ORDS level
3 Use case
4 Consequences and actions

Reminders:

SOP: Single Origin Policy (Rules to be respected in Browsers to protect users)
CORS: Cross Origin Resource Sharing (way to bypass SOP)

Httpd is used for accepting incoming requests in https.
It forwards trafic onto ORDS runing in standalone mode and in http protocol.
(we could decide to run ORDS in https, but it was not the case in my scenario)

Because the difference in protocol (https –> http) this a CORS situation which is trapped by ORDS, and therefore wich triggers an error.

In order to workaround this problem, we have to tell to httpd to forward informations about the fact that it’s a normal operation .

```
<entry key="security.forceHTTPS">true</entry>

In the virtual host section, add the following directives
RequestHeader set "X-Forwarded-Proto" expr=%{REQUEST_SCHEME}
RequestHeader set "X-Forwarded-SSL" expr=%{HTTPS}
```

Debugging

At httpd level

Activate forensic mod at httpd level.
It’s a good way for display header content.

```
<IfModule log_forensic_module>
        #ForensicLog ${APACHE_LOG_DIR}/forensic.log
        ForensicLog logs/forensic.log
</IfModule>
```

At ORDS level

Create a file like this

```
#Example 1: Log messages to a file
handlers=java.util.logging.FileHandler
.level=SEVERE

java.util.logging.FileHandler.level=ALL
oracle.dbtools.level=ALL

# full pathname for logging output file
java.util.logging.FileHandler.pattern = /tmp/ords20.log

java.util.logging.FileHandler.formatter =
```

```
java.util.logging.SimpleFormatter
```

and start ORDS in a specific mode:

```
java -
Djava.util.logging.config.file=/home/almalinux/ords20/config/ords.
properties -jar ords.war standalone
```

Use case

Using Tabbed Postman extension triggers SOP problem even if CORS has been established.
ORDS 20.4 does not appear to implement CORS. This means that a browser that strictly follows the SOP rules will react by reporting an error.

ORDS 23.2 implements CORS. This means that it sends a normal operating message back to the browser, even if the domain at the origin of the request (Origin attribute of the header) is different from the Host name.

However, a tool like Tabbed Postman (as opposed to the Postman Desktop application) is developed as an extension to Chrome. It therefore respects the Chrome security model which is very strict, like that of all browsers.
This is particularly true on a POST type request. As the call is initiated from the test tool (and not included in a page loaded from a particular site), Tabbed Postman sends in the Origin variable of the HTTP header, the following value:
`chrome-extension://coohjcphdfgbiolnekdpbcijmhambjff`

This can be observed by activating a log with the Forensic module on httpd.

Apache can be configured to implement CORS. It transmits the HTTP header as is to ORDS which then detects a difference between the Origin variable and the Host

variable containing the machine.domain name. Therefore, this is considered an SOP violation situation and ORDS returns an error to the caller. As it is a browser extension, the request is rejected

The Postman Desktop or Curl development tools do not trigger the SOP check by default. This is why we can safely test POST type APIs with it.

Consequences and actions

For unit tests, you will need to use Postman desktop and not the Chrome extension from the same publisher.
To complete the point, here are two methods to work around the issue with Tabbed Postman:

Add the application to the list of authorized origins, either via the APEX interface or by calling a plsql api:

```
begin
ords.set_module_origins_allowed(
p_module_name => 'rest-v2',
p_origins_allowed => 'chrome-
extension://coohjcphdfgbiolnekdpbcijmhambjff');
commit;
```

```
end;
```

Trusted origins can be configured through the `security.externalSessionTrustedOrigins` configuration parameter that defines a comma separated list of origins that are trusted to make CORS request. If this parameter is empty or not configured, then no CORS requests are allowed for the PL/SQL gateway and results in a 403 Unauthorized status.

Either by inserting a directive at the ssl.conf level:

```
setEnvIf Origin ^chrome-extension.*$ IS_POSTMAN=true
```

```
RequestHeader set "Origin" "https://xxx.xxx.xxx…" env=IS_POSTMAN
```