


REMOVED_Migrating Custom Authorization Handlers

 THE CONTENT ON THIS PAGE IS A WORK IN PROGRESS.

Post authentication tasks are operations that take place upon successful completion of authentication steps. (ex: Authorization Handler and Missing Claim Handler). Authorization handler will be engaged after successful completion of authentication steps and checks whether the logged in user has proper rights to access the application.

These post authentication criteria were unified in 5.5.0 onwards. In other words, there is no difference between missing claim handler and authorization handler since they are both post-authentication handlers.

You need to migrate your existing missing claim handler and authorization handler only if you have done any customization to the existing handlers. If no customizing is done, migration is not needed because with 5.5.0, the above two handlers are implemented as Post Authentication Handlers by default.

 If you are not sure whether you have customized this or not, Just go have a lookup for Default Authorization Handler setting in `application-authentication.xml` in your old distribution which resides inside the `<IS_HOME>/repository/conf/identity` directory.

```
<AuthorizationHandler>org.wso2.carbon.identity.application.authz.xacml.handler.impl.XACMLBasedAuthorizationHandler</AuthorizationHandler>
```

If you do not find a `PostAuthenticationHandler` configured in `application-authentication.xml`, then you have not configured a post-authentication handler.

If you have customized one of them follow the below steps to migrate the code.

1. If you have it extended from `'PostAuthenticationHandler'`, keep the same parent class and change the method signature to throw a `PostAuthenticationFailedException` and change the return type to `PostAuthnFlowStatus`.
2. If you have it extended from `'AuthorizationHandler'`, change the parent class to `'PostAuthenticationHandler'` and change the method `"isAuthorized"` to `"handle"` and change the return type to `PostAuthnFlowStatus`. Also change the type of the exception which is being thrown to `PostAuthenticationFailedException`.
3. In all places where the status is returned in your code, it should return a `PostAuthnFlowStatus` as below:
 - a. `SUCCESS_COMPLETED` - If the post-authentication process is done, then this status is returned. Then, the next post handler will be invoked since the current one is completed.
 - b. `INCOMPLETE` - If the post-authentication is incomplete, but still you need to go to out and come back. An example of this is a redirection to an outside page. You can simply redirect the response and expect it to come back to your post authentication handler once the response is submitted to IS again. When the response is submitted from outside page to the post-authentication handler again which is in progress, you need to have two things in the request apart from the input data you need to get from the external page.
4. Finally, register your Post Authenticator as an OSGI service.