Systematically guide to run this plugin

1. Feature description
2. Plugin Development
3. Plugin Deployment
4. Plugin Configuration
5. Validate APIs using sample payload

# Feature description

Requirement is to use keycloak for OTP validation as primary login. Many portal want either user can login with username/password or can do login by entering registered mobile number and enter received OTP on portal to login. Valid token needs to be generated by calling keycloak API. This plugin serve the purpose of both user can login with either password or OTP. OOTB keycloak does not provide this feature, instead it provide 2F OTP authenticator with well-known OTP application.

# Plugin Development

Need to develop plugin for direct grant authenticator, which validate token api request and generate valid token if parameters are valid. Source code can be downloaded from below link.

<https://github.com/9228883664/keycloak-otp-password-authenticator.git>

1. pom.xml file

*<?xml version="1.0"?>  
<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd" xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">  
 <modelVersion>4.0.0</modelVersion>  
 <groupId>ppp.plugin.keycloak</groupId>  
 <artifactId>keycloak-otp-password-authenticator</artifactId>  
 <version>v1.0</version>  
 <packaging>jar</packaging>  
 <properties>  
 <keycloak.version>16.1.1</keycloak.version>  
 <jboss-jaxrs-api\_2.1\_spec>2.0.2.Final</jboss-jaxrs-api\_2.1\_spec>  
 <maven.compiler.source>8</maven.compiler.source>  
 <maven.compiler.target>8</maven.compiler.target>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <apache.httpcomponents.version>4.5.13</apache.httpcomponents.version>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>org.keycloak</groupId>  
 <artifactId>keycloak-server-spi</artifactId>  
 <version>${keycloak.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.keycloak</groupId>  
 <artifactId>keycloak-server-spi-private</artifactId>  
 <version>${keycloak.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.keycloak</groupId>  
 <artifactId>keycloak-core</artifactId>  
 <version>${keycloak.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.keycloak</groupId>  
 <artifactId>keycloak-services</artifactId>  
 <version>${keycloak.version}</version>  
 <scope>provided</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.jboss.spec.javax.ws.rs</groupId>  
 <artifactId>jboss-jaxrs-api\_2.1\_spec</artifactId>  
 <version>${jboss-jaxrs-api\_2.1\_spec}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.apache.httpcomponents</groupId>  
 <artifactId>httpclient</artifactId>  
 <version>${apache.httpcomponents.version}</version>  
 <scope>provided</scope>  
 </dependency>  
 </dependencies>  
 <build>  
 <finalName>${project.artifactId}\_${version}</finalName>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-jar-plugin</artifactId>  
 <configuration>  
 <archive>  
 <!-- This is required since we need to add the jboss module references  
 to the resulting jar -->  
 <manifestEntries>  
 <!-- Adding explicit dependencies to avoid class-loading issues at runtime -->  
 <Dependencies>  
 <![CDATA[org.keycloak.keycloak-core,org.keycloak.keycloak-services]]></Dependencies>  
 </manifestEntries>  
 </archive>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
</project>*

1. OtpPasswordAuthenticator extends AbstractDirectGrantAuthenticator

Create one class that implements Abstract Direct Grant Authenticator class

*@Override  
public void authenticate(AuthenticationFlowContext context) {  
 String loginWithOtp = retrieveLoginWithOtp(context);  
 // Normal login with password code  
 if(loginWithOtp==null || !"true".equalsIgnoreCase(loginWithOtp)){  
 String password = retrievePassword(context);  
 boolean valid = context.getSession().userCredentialManager()*

*.isValid(context.getRealm(), context.getUser(), UserCredentialModel.password(password));  
 if (!valid) {  
 context.getEvent().user(context.getUser());  
 context.getEvent().error(Errors.INVALID\_USER\_CREDENTIALS);  
 Response challengeResponse =*

*errorResponse(Response.Status.UNAUTHORIZED.getStatusCode(),*

*"invalid\_grant", "Invalid user credentials");  
 context.failure(AuthenticationFlowError.INVALID\_USER, challengeResponse);  
 return;  
 }  
 context.success();  
 }  
 // Login with OTP flow  
 else{  
 int valid = validateOTP(context);  
 if(valid==1){  
 context.success();  
 }  
 else if(valid<1){  
 context.getEvent().user(context.getUser());  
 context.getEvent().error(error);  
 Response challengeResponse = errorResponse(Response.Status.INTERNAL\_SERVER\_ERROR.getStatusCode(),*

*"invalid\_grant", "Invalid user credentials");  
 context.failure(AuthenticationFlowError.INTERNAL\_ERROR, challengeResponse);  
 }  
 else{  
 context.getEvent().user(context.getUser());  
 context.getEvent().error(Errors.INVALID\_USER\_CREDENTIALS);  
 Response challengeResponse = errorResponse(Response.Status.UNAUTHORIZED.getStatusCode(),*

*"invalid\_grant", "Invalid user credentials");  
 context.failure(AuthenticationFlowError.INVALID\_USER, challengeResponse);  
 }  
 }  
}*

1. OtpPasswordAuthenticatorFactory implements AuthenticatorFactory

Implement one Factory class to provide configuration

*@Override  
public List<ProviderConfigProperty> getConfigProperties() {  
 ProviderConfigProperty servicename = new ProviderConfigProperty();  
 servicename.setType(STRING\_TYPE);  
 servicename.setName(OTP\_VALIDATION\_EXTERNAL\_SERVICE\_URL);  
 servicename.setLabel("OTP validation service url");  
 servicename.setDefaultValue("http://localhost:8080/api/validateOtp");  
 servicename.setHelpText("Valid url for OTP validation e.g. http://localhost:8080/api/validateOtp");  
  
 List<ProviderConfigProperty> listOfConfigs = Arrays.asList(servicename);  
 return listOfConfigs;  
}*

1. org.keycloak.authentication.AuthenticatorFactory

Set configuration factory in meta info

ppp.plugin.keycloak.login.OtpPasswordAuthenticatorFactory

# Plugin Deployment

Download keycloak 16.1.1 or any latest version with any server. For JBoss we can do below steps.

Generate spring boot jar file using mvn clean install. Copy jar file and paste in jboss standalone folder.

*Copy From: keycloak-otp-password-authenticator/target/keycloak-otp-password-authenticator\_v1.0.jar*

*Copy To: keycloak-16.1.1/standalone/deployments/keycloak-otp-password-authenticator\_v1.0.jar*

Start Jboss after this using standalone command

Check log with below string for successful deployment:

*Deployed "keycloak-otp-password-authenticator\_v1.0.jar"*

Check <http://localhost:8080/auth> for UI

For reference check below full logs:

*INFO [org.jboss.as.server.deployment] (MSC service thread 1-1) WFLYSRV0027: Starting deployment of "keycloak-otp-password-authenticator\_v1.0.jar" (runtime-name: "keycloak-otp-password-authenticator\_v1.0.jar")*

*INFO [org.jboss.as.server.deployment] (MSC service thread 1-1) WFLYSRV0027: Starting deployment of "keycloak-otp-password-authenticator\_v1.0.jar" (runtime-name: "keycloak-otp-password-authenticator\_v1.0.jar")*

*WARN [org.jboss.as.dependency.private] (MSC service thread 1-1) WFLYSRV0018: Deployment "deployment.keycloak-otp-password-authenticator\_v1.0.jar" is using a private module ("org.keycloak.keycloak-services") which may be changed or removed in future versions without notice.*

*INFO [org.keycloak.subsystem.server.extension.KeycloakProviderDeploymentProcessor] (MSC service thread 1-7) Deploying Keycloak provider: keycloak-otp-password-authenticator\_v1.0.jar*

*WARN [org.keycloak.services] (ServerService Thread Pool -- 52) KC-SERVICES0047: otppasswordauthenticator (ppp.plugin.keycloak.login.OtpPasswordAuthenticatorFactory) is implementing the internal SPI authenticator. This SPI is internal and may change without notice*

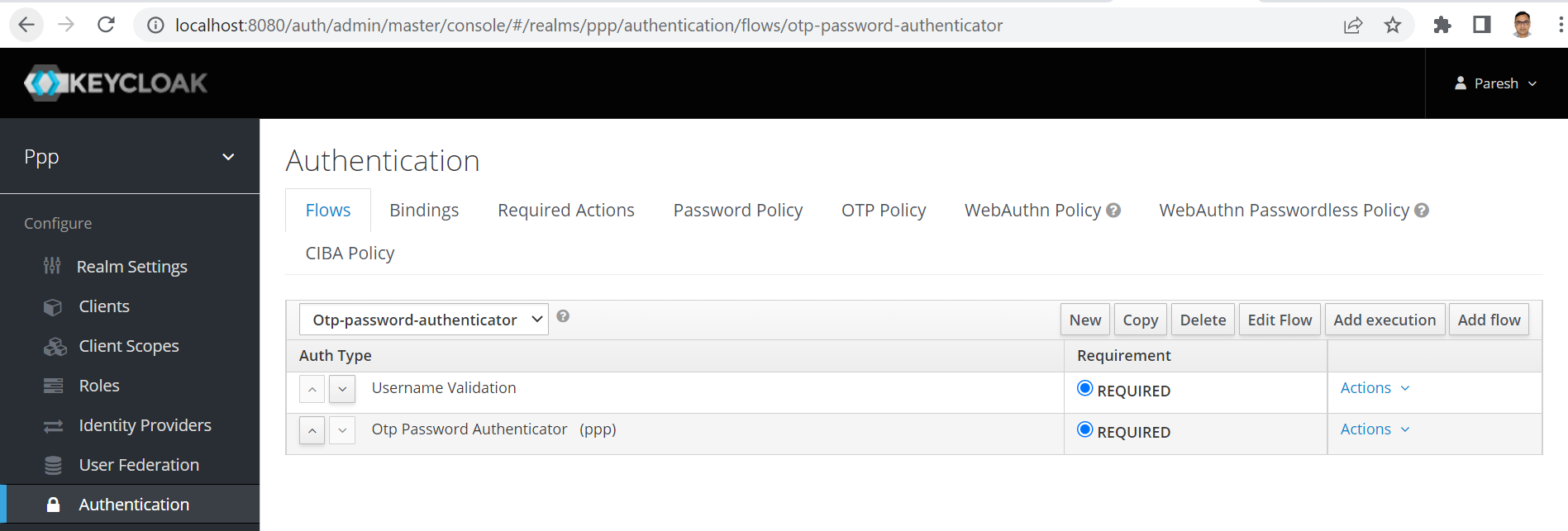
*WARN [org.keycloak.services] (ServerService Thread Pool -- 52) KC-SERVICES0047: otppasswordauthenticator (ppp.plugin.keycloak.login.OtpPasswordAuthenticatorFactory) is implementing the internal SPI authenticator. This SPI is internal and may change without notice*

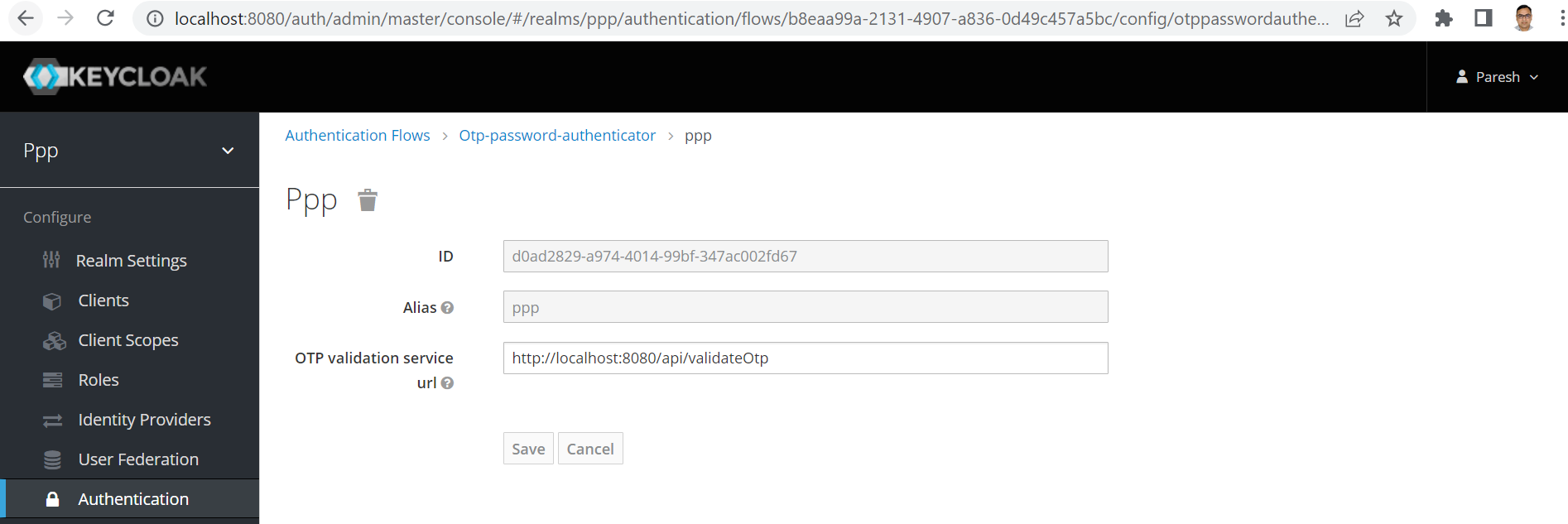
*INFO [org.jboss.as.server] (ServerService Thread Pool -- 31) WFLYSRV0010: Deployed "keycloak-otp-password-authenticator\_v1.0.jar" (runtime-name : "keycloak-otp-password-authenticator\_v1.0.jar")*

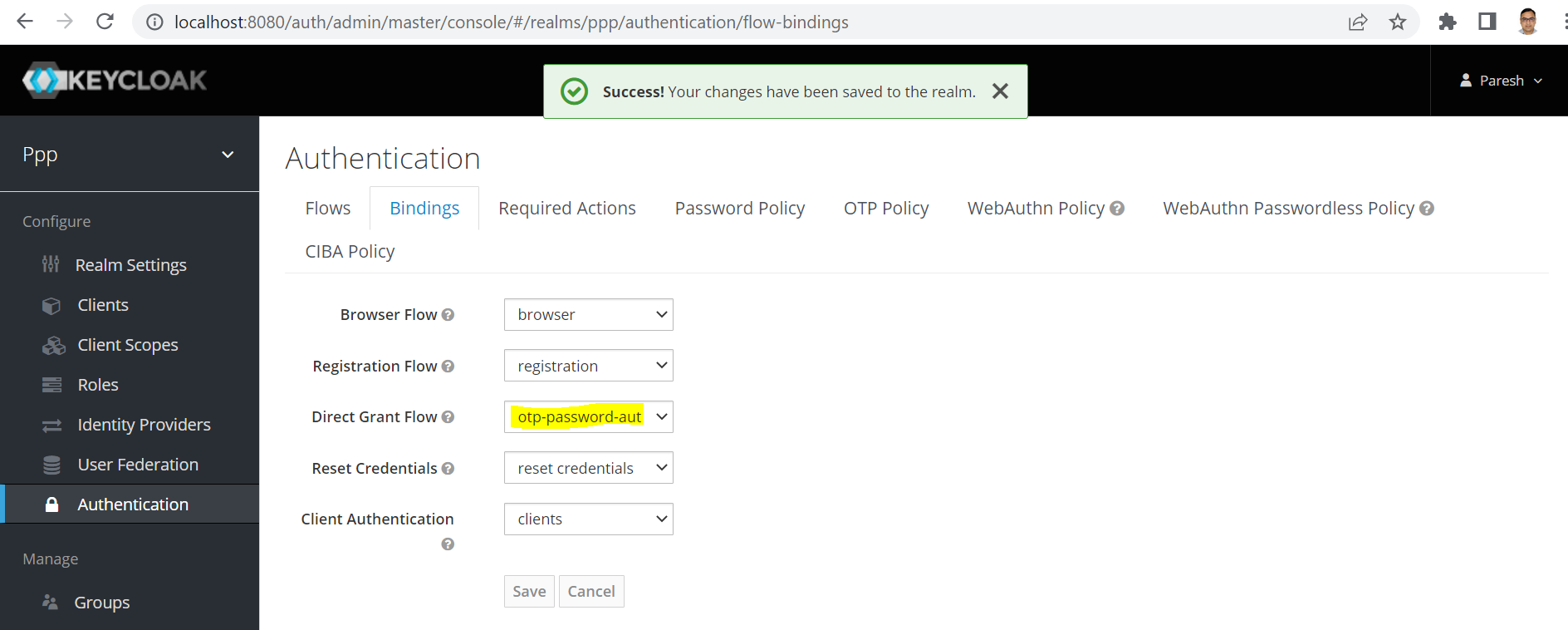
*INFO [org.jboss.as.server] (ServerService Thread Pool -- 31) WFLYSRV0010: Deployed "keycloak-otp-password-authenticator\_v1.0.jar" (runtime-name : "keycloak-otp-password-authenticator\_v1.0.jar")*

# Plugin Configuration

Configure plugin on keycloak ui.







1. Create new or copy from direct grant flow in ppp realm Authentication Flow.

Here new flow name given otp-password-authenticator.

Clean all action and keep Username Validation execution flow.

Now click on add execution and add “otp password authenticator” execution in flow.

Now configure new execution with values by clicking action->config.

1. Change direct grant auth flow

Change Binding of Direct Grant Flow to “otp-password-authenticator” flow, which is created in above step, and save configuration

# Validate APIs using sample payload

Call token API as per below payload where new parameters added for OTP

|  |
| --- |
| Below is the regular password call |
| *curl --location --request POST '{{keycloak-url}}/realms/dep7/protocol/openid-connect/token' \*  *--header 'Content-Type: application/x-www-form-urlencoded' \*  *--data-urlencode 'client\_id={{client-id}}' \*  *--data-urlencode 'client\_secret={{secret}}' \*  *--data-urlencode 'grant\_type=password' \*  *--data-urlencode 'username={{username}}' \*  *--data-urlencode 'password={{password}}'* |
| Below is the otp call |
| *curl --location --request POST '{{keycloak-url}}/realms/dep7/protocol/openid-connect/token' \*  *--header 'Content-Type: application/x-www-form-urlencoded' \*  *--data-urlencode 'client\_id={{client-id}}' \*  *--data-urlencode 'client\_secret={{secret}}' \*  *--data-urlencode 'grant\_type=password' \*  *--data-urlencode 'username={{username}}' \*  ***--data-urlencode 'totp={{otp}}' \***  ***--data-urlencode 'login-with-otp=true'*** |