PFRD Service Migration to OAuth 2.0

Purpose

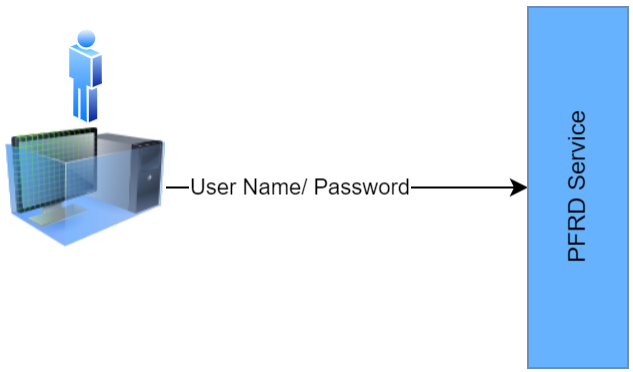
FINRA is moving all application authentication from basic authentication to OAuth 2.0 based to enhance security across its systems. In order to support this change, we are creating a new "Version 2" of PFRD service and will subsequently retire the current "Version 1" of the endpoint. This document is meant to be used by firms and vendors who are testing prior to the production implementation.

Important note: The following guidance was successful using FINRA machines running on the internal FINRA network. You may need to make adjustments based on your specific systems. If you need assistance, please contact PFRDSupport@finra.org.

Flow Diagrams

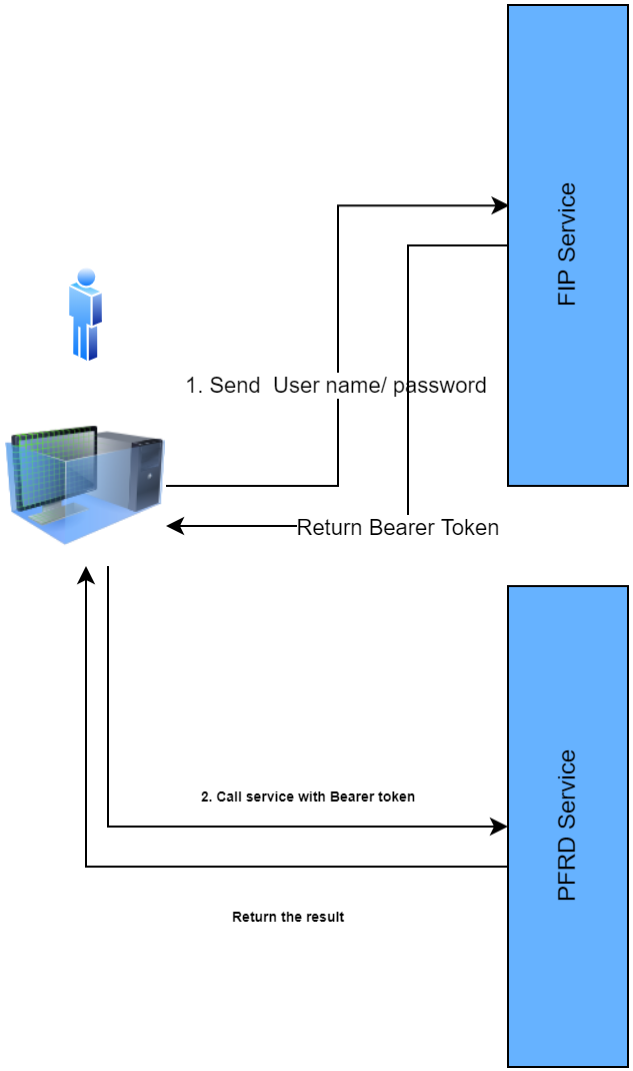
Current Flow

In current form, user credentials are directly passed to PFRD service in Soap XML which does authN/authZ and return back appropriate results.



New Flow

In this flow, FIP service is called passing user credentials in basic auth, get the bearer token and then pass that to the PFRD service.



Environments

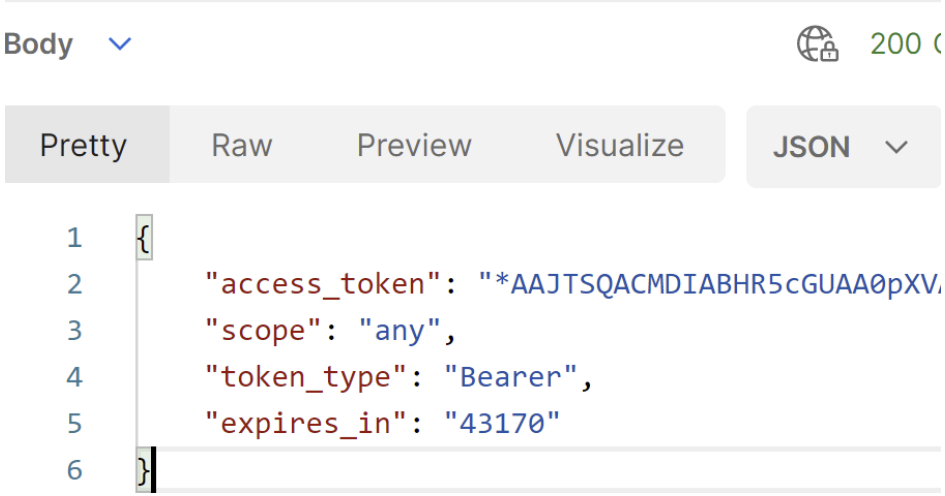
	Production	QA
FIP EWS	https://ews.fip.finra.org/fip/rest/ews/oauth2/access_token?grant_type=client_credentials	https://ews-qaint.fip.qa.finra.org/fip/rest/ews/oauth2/access_token?grant_type=client_credentials
Path to New PFRD endpoint that supports Oauth 2.0	https://crd.finra.org/v2/PFRD/webservices/PFFormSubmissionService.svc	https://webcrdtesting.crd.finra.org/v2/PFRD/webservices/PFFormSubmissionService.svc

Implementation Details

To access the OAuth endpoint

- Download the WSDL for the new endpoint.
  - Example: https://crd.finra.org/v2/PFRD/webservices/PFFormSubmissionService.svc?singleWsdl
- Obtain an OAuth token from the FIP EWS service of appropriate environment using Basic Authentication with your username/password (e.g. joint877/mypwd)
  - Example: https://ews.fip.finra.org/fip/rest/ews/oauth2/access_token?grant_type=client_credentials

Grab the ***access_token*** value from the returned json. Add the token in the header of your request. (Do not forget to add the Bearer prefix, followed by a space, followed by the token value as shown below)



Endpoint invocation

```
using (new OperationContextScope(service.InnerChannel))
{
   WebOperationContext.Current.OutgoingRequest
                .Headers.Add("Authorization", String.Format("Bearer
*AAJTSQACMDIABHR5cGUAA0pXVAACUzEAAjAx*......."));

   var result = service.Ping();
}
```

Sample binding setup for the PFRD V2 service:

Endpoint configuration

```
<bindings>
            <wsHttpBinding>
                <binding name="WSHttpBinding_PFFormSubmissionService">
                    <security mode="Transport">
                        <transport clientCredentialType="None" />
                    </security>
                </binding>
            </wsHttpBinding>
        </bindings>
```

Java Client

Sample build.gradle config task

Java Config

```
plugins {
    id 'java'
}

group 'org.example'
version '1.0-SNAPSHOT'

repositories {
    mavenCentral()
}

dependencies {
    compile "jakarta.xml.ws:jakarta.xml.ws-api:3.0.0"
    compile "com.sun.xml.ws:jaxws-rt:3.0.0"
    compile "com.sun.xml.ws:jaxws-ri:3.0.0"
    compile "com.konghq:unirest-java:3.11.09"
```

```
    testCompile group: 'junit', name: 'junit', version: '4.12'
    runtime 'com.sun.xml.ws:jaxws-tools:3.0.0'
}

task wsImport(type: JavaExec) {
    classpath sourceSets.main.runtimeClasspath
    main = "com.sun.tools.ws.WsImport"
    args "-target", "2.2",
            "-s", "src/main/java",
            "-keep",
            "-Xnocompile",
            "-encoding", "UTF-8",
            "src/main/resources/PFFormSubmissionService.xml"
}
```

Java Client

```
import jakarta.xml.ws.BindingProvider;
import jakarta.xml.ws.handler.Handler;
import jakarta.xml.ws.handler.MessageContext;
import jakarta.xml.ws.soap.AddressingFeature;
import kong.unirest.HttpResponse;
import kong.unirest.Unirest;
import org.finra.crd.pfrd.PFFormSubmissionService;
import org.finra.crd.pfrd.PFFormSubmissionServiceImpl;

import javax.xml.namespace.QName;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class Startup {

        public static class FipResponse {
        public String access_token;
        public String expires_in;
        public String token_type;
```



```
        public String scope;
        }

   //1. Production URL for FIP token service
    final static String tokenServiceUrl =
"https://ews.fip.finra.org/fip/rest/ews/oauth2/access_token?grant_type=client_credentials";
        //2. Production URL for the PFRD service
    final static String pfrdServiceUrl =
"https://crd.finra.org/v2/PFRD/WebServices/PFFormSubmissionService.svc";

    public static void main(String[] args) {

        //Instantiate the service
        var service = new PFFormSubmissionServiceImpl();


                //NOTE ----------------------------
                // The PFFormSubmissionService class is generated by the WsImport tool. You can either
generate it at the command line
                // or define a gradle task like in the build.gradle configuration file above (task wsImport)
                //
                // The endpoint name @WebEndpoint(name = "WSHttpBinding_PFFormSubmissionService") is also
generated. You can find it
                // in your generated files : PFFormSubmissionServiceImpl.java
                //---------------------------------

                //get the port name and make sure to enable WS-A addressing
                // the QName for the port name is generated by the WsImport task.
                PFFormSubmissionService port =
                 service.getPort(new QName("http://crd.finra.org/PFRD",
"WSHttpBinding_PFFormSubmissionService"),
                         PFFormSubmissionService.class,
                         new AddressingFeature(true, true)); //Important Note: this parameter enables WS_A
addressing mode

        BindingProvider bp = (BindingProvider)port;

        //1. Set up the endpoint address here
        bp.getRequestContext().put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY, pfrdServiceUrl);

        //Set up the Oauth header
```



```
        Map<String, List<String>> requestHeaders = new HashMap<>();
                //Get the OAuth token for your specific username/password
        requestHeaders.put("Authorization", Arrays.asList(getOAuthTokenFor("userName", "password",
tokenServiceUrl)));
        bp.getRequestContext().put(MessageContext.HTTP_REQUEST_HEADERS, requestHeaders);

        //call the service
        Boolean ping = port.ping();
        System.out.printf("Result: %s", ping );
    }

    private static String getOAuthTokenFor(String userName, String password, String tokenServiceUrl){

        HttpResponse<FipResponse> response = Unirest.post(tokenServiceUrl)
                .basicAuth(userName, password).asObject(FipResponse.class);

        if (response.isSuccess()) {
            FipResponse fipResponse = response.getBody();
            return String.format("Bearer %s", fipResponse.access_token);
        }
        throw new RuntimeException("Unable to get the OAuth token");
    }
}
```

.NET Client

The .NET configuration is generated automatically by adding a service reference in your .NET project. Here is an example generated for the QA environment.

For the Production environment, please replace the endpoint's address.

.NET Configuration

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
    </startup>
    <system.serviceModel>

        <bindings>
            <wsHttpBinding>
                <binding name="WSHttpBinding_PFFormSubmissionService">
                    <security mode="Transport">
                        <transport clientCredentialType="None" />
                    </security>
                </binding>
            </wsHttpBinding>
        </bindings>
        <client>
            <endpoint address="https://crd-
qaint1.qa.finra.org/v2/PFRD/WebServices/PFFormSubmissionService.svc"
                                      binding="wsHttpBinding"
                                      bindingConfiguration="WSHttpBinding_PFFormSubmissionService"
                                      contract="PFService.PFFormSubmissionService"
                                      name="WSHttpBinding_PFFormSubmissionService" />
        </client>
    </system.serviceModel>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="System.Runtime.CompilerServices.Unsafe" publicKeyToken="b03f5f7f11d50a3a"
culture="neutral" />
```

```
        <bindingRedirect oldVersion="0.0.0.0-5.0.0.0" newVersion="5.0.0.0" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```

.NET Client

Please note that we used the RestClient library to call the FIP endpoint, but it can be any REST library of your choice.

The important thing to remember is that the FIP call to get an OAuth token uses Basic Authentication and it is a POST call.

```
using RestSharp;
using RestSharp.Authenticators;
using System;
using System.IO;
using System.Net;
using System.Reflection;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text.Json.Serialization;

namespace PFClient
{
    class Program
    {
        //FIP QA Url (To be replaced with prod URL)
        public static string fipUrl = "https://ews-
qaint.fip.qa.finra.org/fip/rest/ews/oauth2/access_token?grant_type=client_credentials";
        //FIP JSON Response structure
        private class FipResponse
        {
            [JsonPropertyName("access_token")]
            public String AccessToken { get; set; }
            [JsonPropertyName("expires_in")]
            public String Expiration { get; set; }
        }
```

```
        static void Main(string[] args)
        {
            //Handle SSL cert if needed
            ServicePointManager.ServerCertificateValidationCallback += (sender, cert, chain,
sslPolicyErrors) => {
                return true;
            };

            //The settings for the service have been generated in app.config (or PFClient.exe.config)
            var service = new PFService.PFFormSubmissionServiceClient();
            var userName = "JOINT877";
            var password = "***";
            //call the EWS oAuth endpoint to obtain token
            FipResponse fipResponse = GetFipToken(userName, password);
            //Access the context scope in order to set the authorization header
            using (new OperationContextScope(service.InnerChannel))
            {
                //prepare the Auth header
                var authHeader = String.Format("Bearer {0}", fipResponse.AccessToken);
                //set the header for the outgoing request
                WebOperationContext.Current.OutgoingRequest.Headers.Add("Authorization", authHeader);

                //Test call to Ping
                var result = service.Ping();

                //Test call to UploadFilling
                using(var stream =
Assembly.GetExecutingAssembly().GetManifestResourceStream("PFClient.SampleRequest.xml"))
                using(var reader = new StreamReader(stream))
                {
                    var payload = reader.ReadToEnd();
                    string uploadResult = service.UploadFiling("TestRefId", "test@company.org", payload);
                    Console.WriteLine(uploadResult);
                }
            }
        }

        /// <summary>
        /// Call FIP endpoint to get token
        /// </summary>
        /// <param name="userName"></param>
```



```
        /// <param name="password"></param>
        /// <returns></returns>
        private static FipResponse GetFipToken(string userName, string password) {
            //We used RestClient, but it can be any REST library of your choice
            var client = new RestClient
            {
                //IMPORTANT: Use basic authentication for call to FIP
                Authenticator = new HttpBasicAuthenticator(userName, password)
            };
            var request = new RestRequest(fipUrl);
            //This needs to be a POST call
            var task = client.PostAsync<FipResponse>(request);
            //Make the call synchronous
            return task.GetAwaiter().GetResult();
        }
    }
}
```