ACR®
AMERICAN COLLEGE OF RADIOLOGY
QUALITY IS OUR IMAGE

ACR Connect Authentication Service Developers Guide

Version 1.4

Revision History

Date	Version	Description
01/29/2015	1.0	Authentication using NRDR account
11/23/2015	1.1	Replaced end point url from https://secureauth01vm.acr.org/secureauth5 to https://acr-id-test.acr.org/NRDR-UserLogin Added Appendix A – Scope values
10/24/2017	1.2	Mandatory 'offline_access' scope item was added to support SecureAuth 9.0
11/29/2017	1.3	prompt=consent parameter was added to Authentication URL to keep backward compatibility with SecureAuth 8.0
12/11/2018	1.4	Added scope 'nmd_data_submission'

Background

The document describes the way ACR Connect authentication service can be used to authenticate users and get access to external secured services.

ACR Connect Authentication service implements OpenID Connect protocol. It allows Clients to verify the identity of the End-User based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the End-User in an interoperable and REST-like manner. OpenID Connect allows clients of all types, including Web-based, mobile, and JavaScript clients, to request and receive information about authenticated sessions and end-users. It is an extension on top of OAuth 2.0 authorization framework.

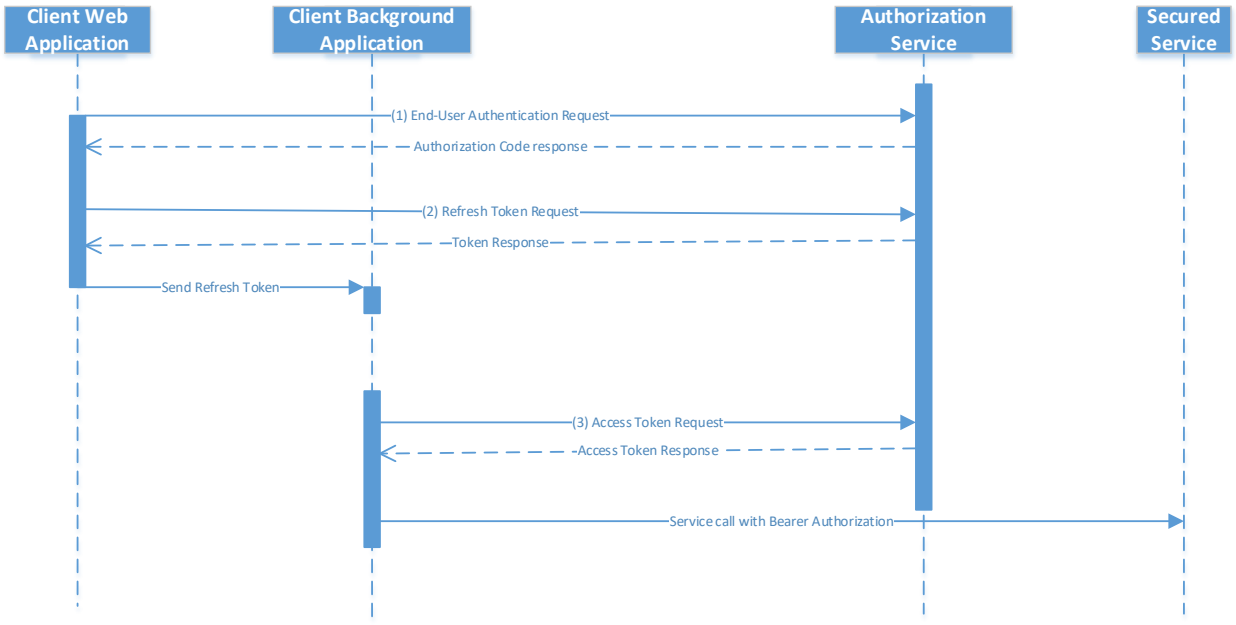
Client Application Registration

Client application must be registered at Authorization Server before it can start sending authentication requests. Client application owner must provide

1. Incoming IP address (es). We will white list your IP address (es) to allow you to come in to our test environment.
2. Application name. The name must be human readable. This name will be displayed to end-user during authentication process.
3. Redirect URL. This is redirection URI to which the response will be sent.

The client application owner will be provided with Client ID and Client Secret. These values along with Redirect URL constitute client application credentials, which are used in authentication process to validate application authenticity. Client Secret must be handled with policies similar to organization system administrator password.

NRDR Account Authorization quick guide



Authentication steps:

1. **End-user authentication.** This is done by sending the User Agent to the Authorization Server's Authorization Endpoint. Request example:

```
GET https://acr-id-test.acr.org/NRDR-UserLogin/authorized/oidcauthorize.aspx?
    client_id=1f5f39524f224df084520a2faa9a9275
```

```
&redirect_uri=https%3a%2f%2flocalhost%3a44306%2fAuthCallback
&response_type=code
&scope=openid%20offline_access%20grid_exam_submission
&state=6rrVSW20MU2rRGyoiMCceiRT
&prompt=consent
```

```
Host in test environment: acr-id-test.acr.org
```

https://acr-id-test.acr.org/NRDR-UserLogin/authorized/oidcauthorize.aspx - is authorization endpoint URL address.
client_id – Client ID value.
redirect_uri – callback url where authentication result will be send.
response_type – must be “code”.
scope – rights to be requested. “openid” and ‘offline_access’ are required. The others depend on actual rights required, see Appendix A.
state – custom client application data
prompt – consent value is mandatory, when offline_access scope is used.

Response example

```
HTTP/1.1 302 Found
Location: https://localhost:44306/AuthCallback?
      code=7B6bhNW5Ro9WgRj0mZV8xk3FMuUWZwKlBABcW3RgTOT63%252f9nPpl0JtW5r5pes8Q
      cfDc7m2OE9t46%252fknz46v8hZ64AW5PEqgF%252fUUZdnCGhWtDi5PzbFUkwTclc6STczt
      bus8w2tYrOWNHDWRp7vTDCYnIZzw5TIoXktpqrDpd3wsM5R62pDAGq9w32k2gcKUuZxnG3Zg
      OZwcZkYa7gl2bYisYp%252fkBOER9jeV%252b%252fcwSOXBYi1abXwwkArblVvzcfo%252b
      W8Usx04EyQ7jSN5W64Pszfg%253d%253d
      &state=6rrVSW20MU2rRGyoiMCceiRT
```

code – authentication code

2. **Token request.** This is done by sending HTTP request to the Authorization Server's Token Endpoint.

Request example:

```
POST https://acr-id-test.acr.org/NRDR-UserLogin/oidctoken.aspx HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: secureauth01vm.acr.org

      grant_type=authorization_code
      &code=7B6bhNW5Ro9WgRj0mZV8xk3FMuUWZwKlBABcW3RgTOT63%252f9nPpl0JtW5r5pes8
      QcfDc7m2OE9t46%252fknz46v8hZ64AW5PEqgF%252fUUZdnCGhWtDi5PzbFUkwTclc6STcz
      tbus8w2tYrOWNHDWRp7vTDCYnIZzw5TIoXktpqrDpd3wsM5R62pDAGq9w32k2gcKUuZxnG3Z
      gOZwcZkYa7gl2bYisYp%252fkBOER9jeV%252b%252fcwSOXBYi1abXwwkArblVvzcfo%252
      bW8Usx04EyQ7jSN5W64Pszfg%253d%253d
      &redirect_uri=https%3A%2F%2Flocalhost%3A44306%2FAuthCallback
      &client_id=1f5f39524f224df084520a2faa9a9275
      &client_secret=6295475514294cbeaf7a09843bf3e17b
```

https://acr-id-test.acr.org/NRDR-UserLogin/oidctoken.aspx - Token Endpoint URL address.
grant_type – must be “authorization_code”
code – authorization code received in previous call.
redirect_uri – the same as in previous call
client_id – Client ID value

client_secret – Client Secret value

Response example (the tokens are shortened for display purpose) :

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{
      "access_token":"eyJ0eXAiOiJKV1QiLCJhb...xP9qQkeiizKQ",
      "expires_in":86400,
      "id_token":"eyJ0e...s4vsZe351hyJvQ9Z0cyOa1mBfyg",
      "refresh_token":"3ahX1k7IrY...oEIWdIEa7Aqz4m7eImfHK5RdF",
      "scope":["openid","offline_access","grid_exam_submission"],
      "token_type":"Bearer"
}
```

Access token can be used to call secured services using Bearer Authentication. Refresh token can be stored in order to reissue new access tokens later.

3. **Reissue access token.** This is done by sending HTTP POST request to the Authorization Server's Token Endpoint.

   Request example:

```
POST https://acr-id-test.acr.org/NRDR-UserLogin/oidctoken.aspx HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: secureauth01vm.acr.org

   grant_type=refresh_token
   &refresh_token=3ahX1k7IrY...oEIWdIEa7Aqz4m7eImfHK5RdF
   &client_id=1f5f39524f224df084520a2faa9a9275
   &client_secret=6295475514294cbeaf7a09843bf3e17b
```

https://acr-id-test.acr.org/NRDR-UserLogin/oidctoken.aspx - is token endpoint URL address.
refresh_token – refresh token value received previously
client_id – Client ID value.
client_secret – Client Secret value.

Response example:

```
   HTTP/1.1 200 OK
   Content-Type: application/json; charset=utf-8

   {
      "access_token":"eyJ0eXAiOiJ…WQdGAyMg",
      "expires_in":172792,
      "refresh_token":"3ahX1k7IrYZAVu…%3d%3d",
      "scope":["openid","offline_access","pqrs_data_submission"],
      "token_type":"Bearer"
   }
```

The response is pretty the same as in step 2. The only difference is that it contains no ID Token. It contains new Access Token, which can be used to access secured services and renewed Refresh Token, which must be used for Access Token request next time.

Detailed steps description can be found below.

Offline authentication flow detailed description

This flow allows obtaining Authentication Token that grants access to secured resources even when end-user is not present (offline mode).

The overall authentication process includes two separate steps (the steps can be implemented by different applications).

The first step is to obtain refresh token. This step requires end-user's presence. It includes two steps:

1. End-user authentication, which results Authorization Code to be received.
2. Refresh token request

The second step is to request access token. This step does not require end-user's presence. A background application can get new valid access token and start using it for secured resources calling.

1. Authentication request

The Authorization Endpoint performs Authentication of the End-User. This is done by sending the User Agent to the Authorization Server's Authorization Endpoint for Authentication and Authorization, using request parameters defined by OAuth 2.0 and additional parameters and parameter values defined by OpenID Connect.

Authentication endpoint request can contain the following request parameters:

scope

REQUIRED. Requests MUST contain at least the "openid" scope value. Other scope values may be present. Other scopes are required, when client application will use access tokens to access secured resources. "offline_access" scope is required to enable the access tokens to be renewed with Refresh Token. The request must contain all corresponding scopes. Authentication server validates, that the client application has access to the corresponding resources. Multiple values can be provided as space-separated list. Example: "openid grid_exam_submission"

response_type

REQUIRED. Response Type value that determines the authorization processing flow to be used, including what parameters are returned from the endpoints used. When using the Authorization Code Flow, this value is "code".

client_id

REQUIRED. Client Identifier obtained during the client application registration at the Authorization Server.

redirect_uri

REQUIRED. Redirection URI to which the response will be sent. This URI MUST exactly match one of the Redirection URI values for the Client pre-registered at the Authorization Server, with the matching performed as described in Section 6.2.1 of [RFC3986] (Simple String Comparison). When using this flow, the Redirection URI MUST use the https scheme.

state

RECOMMENDED. Opaque value used to maintain state between the request and the callback. Typically, Cross-Site Request Forgery (CSRF, XSRF) mitigation is done by cryptographically binding the value of this parameter with a browser cookie.

prompt

OPTIONAL. Space delimited, case sensitive list of ASCII string values that specifies whether the Authorization Server prompts the End-User for reauthentication and consent. The defined values are:

consent

The Authorization Server SHOULD prompt the End-User for consent before returning information to the Client. If it cannot obtain consent, it MUST return an error, typically consent_required. When offline access is requested, a prompt parameter value of consent MUST be used unless other conditions for processing the request permitting offline access to the requested resources are in place.

The following is the example request that User Agent should send to the Authorization Server (with line wraps within values for display purposes only):

```
GET https://acr-id-test.acr.org/NRDR-UserLogin/authorized/oidcauthorize.aspx?
      client_id=1f5f39524f224df084520a2faa9a9275
      &redirect_uri=https%3a%2f%2flocalhost%3a44306%2fAuthCallback
      &response_type=code
      &scope=openid%20offline_access%20grid_exam_submission
      &state=6rrVSW20MU2rRGyoiMCceiRT
      &prompt=consent
   Host: secureauth01vm.acr.org
```

https://acr-id-test.acr.org/NRDR-UserLogin/authorized/oidcauthorize.aspx - is the Authorization Endpoint URL address.

response_type=code – Authorization Code flow will be used

scope=openid%20offline_access%20grid_exam_submission – "openid" – is required for OpenID Connect based authentication requests. "offline_access" – is required for Refresh Token flow to enable Access Token to be renewed. Additionally, client application is going to access secured resources, which has "grid_exam_submission" scope assigned.

client_id=1f5f39524f224df084520a2faa9a9275 – Client ID obtained during application registration.

state=6rrVSW20MU2rRGyoiMCceiRT – Client Application specific data

redirect_uri=https%3a%2f%2flocalhost%3a44306%2fAuthCallback – callback url, where response with Authorization Code will be redirected to. Must exactly match to the value provided during client application registration.

prompt=consent – is mandatory, when offline access is requested. Instructs the Authorization Server to prompt the End-User for consent before returning information to the Client.

The following is an example of successful response using this flow (with line wraps within values for display purposes only):

```
HTTP/1.1 302 Found
Location: https://localhost:44306/AuthCallback?
      code=7B6bhNW5Ro9WgRj0mZV8xk3FMuUWZwKlBABcW3RgTOT63%252f9nPpl0JtW5r5pes8Q
      cfDc7m2OE9t46%252fknz46v8hZ64AW5PEqgF%252fUUZdnCGhWtDi5PzbFUkwTclc6STczt
      bus8w2tYrOWNHDWRp7vTDCYnIZzw5TIoXktpqrDpd3wsM5R62pDAGq9w32k2gcKUuZxnG3Zg
      OZwcZkYa7gl2bYisYp%252fkBOER9jeV%252b%252fcwSOXBYi1abXwwkArblVvzcfo%252b
      W8Usx04EyQ7jSN5W64Pszfg%253d%253d
      &state=6rrVSW20MU2rRGyoiMCceiRT
```

https://localhost:44306/AuthCallback - is redirect URL provided in authentication request.

code=7B6bhNW5Ro9WgRj0m… – Authorization Code

state=6rrVSW20MU2rRGyoiMCceiRT – Data, which provided in Authentication request by client application.

2. Refresh Token Request

A Client application makes a Token Request by presenting its Authorization Grant (in the form of an Authorization Code) to the Token Endpoint using the grant_type value “authorization_code”. The client application MUST authenticate to the Token Endpoint using its credentials obtained during application registration.

The Client sends the parameters to the Token Endpoint using the HTTP POST method and the Form Serialization.

The following is an example of a Token Request (with line wraps within values for display purposes only):

```
POST https://acr-id-test.acr.org/NRDR-UserLogin/oidctoken.aspx HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: secureauth01vm.acr.org

grant_type=authorization_code
      &code=7B6bhNW5Ro9WgRj0mZV8xk3FMuUWZwKlBABcW3RgTOT63%252f9nPpl0JtW5r5pes8
      QcfDc7m2OE9t46%252fknz46v8hZ64AW5PEqgF%252fUUZdnCGhWtDi5PzbFUkwTclc6STcz
      tbus8w2tYrOWNHDWRp7vTDCYnIZzw5TIoXktpqrDpd3wsM5R62pDAGq9w32k2gcKUuZxnG3Z
      gOZwcZkYa7gl2bYisYp%252fkBOER9jeV%252b%252fcwSOXBYi1abXwwkArblVvzcfo%252
      bW8Usx04EyQ7jSN5W64Pszfg%253d%253d
      &redirect_uri=https%3A%2F%2Flocalhost%3A44306%2FAuthCallback
      &client_id=1f5f39524f224df084520a2faa9a9275
      &client_secret=6295475514294cbeaf7a09843bf3e17b
```

https://acr-id-test.acr.org/NRDR-UserLogin/oidctoken.aspx – is Token Endpoint URL address.

code=7B6bhNW5Ro9WgRj0m… – Authorization Code obtained during initial end-user authentication process using Authorization Code flow.

redirect_uri=https%3a%2f%2flocalhost%3a44306%2fAuthCallback – callback url, where response with Authorization Code will be redirected to. Must exactly match to the value provided during client

client_id=1f5f39524f224df084520a2faa9a9275 – Client ID obtained during application registration.

client_secret=6295475514294cbeaf7a09843bf3e17b – Client Secret obtained during application registration.

After receiving and validating a valid and authorized Token Request from the Client, the Authorization Server returns a successful response that includes an ID Token and an Access Token. The response uses the application/json media type.

The token_type response parameter value shall be “Bearer”.

The following is an example of successful response for token request (with the tokens values shortening and line wraps within values for display purposes only):

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
```

```
{
      "access_token":"eyJ0eXAiOiJKV1QiLCJhb...xP9qQkeiizKQ",
      "expires_in":86400,
      "id_token":"eyJ0e...s4vsZe351hyJvQ9Z0cyOa1mBfyg",
      "refresh_token":"3ahX1k7IrY...oEIWdIEa7Aqz4m7eImfHK5RdF",
      "scope":["openid","offline_access","grid_exam_submission"],
      "token_type":"Bearer"
}
```

The response body contains json object. The object includes the following tokens:

1. Access token. The client application can use it to access secured resources right after the token is received. It has short lifetime.
2. ID Token is a security token that contains Claims about the Authentication of an end-user by an Authorization Server when using a Client, and potentially other requested Claims. The ID Token is represented as a JSON Web Token (JWT). It has short lifetime as well.
3. Refresh Token is long lived token, which can be user to reissue new access token without end-user presence. This token must be handles with policies similar to ones, which are used for user name and password handling.

The scope attribute represents the operations, what the client application can do using the tokens returned.

The client application can start accessing secured resources (which corresponds to the scopes) right after the tokes were received. Additionally, it can persist refresh token and use it later to issue new access tokens.

3. Access Token Request

To refresh an Access Token, the Client MUST authenticate to the Token Endpoint using its credentials. The Client sends the parameters via HTTP POST to the Token Endpoint using Form Serialization.

The following is an example of a Refresh Request (with line wraps within values for display purposes only):

```
POST https://acr-id-test.acr.org/NRDR-UserLogin/oidctoken.aspx HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: secureauth01vm.acr.org

   grant_type=refresh_token
   &refresh_token=3ahX1k7IrY...oEIWdIEa7Aqz4m7eImfHK5RdF
   &client_id=1f5f39524f224df084520a2faa9a9275
   &client_secret=6295475514294cbeaf7a09843bf3e17b
```

https://acr-id-test.acr.org/NRDR-UserLogin/oidctoken.aspx - is token endpoint URL address.
refresh_token – refresh token value received previously
client_id=1f5f39524f224df084520a2faa9a9275 – Client ID obtained during application registration.
client_secret=6295475514294cbeaf7a09843bf3e17b – Client Secret obtained during application registration.

Upon successful validation of the Refresh Token, the response body is the same Token Response as in step 4.2 except that it contains no id_token.

The following is an example of a Refresh Response:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{
   "access_token":"eyJ0eXAiOiJ…WQdGAyMg",
   "expires_in":172792,
   "refresh_token":"3ahX1k7IrYZAVu…%3d%3d",
   "scope":["openid","offline_access","pqrs_data_submission"],
   "token_type":"Bearer"
}
```

The response body is json-object. It contains new Access Token, which can be used to access secured services and renewed Refresh Token, which must be used for Access Token request next time.

Received Access Token can be used for secured services calling using Bearer Authorization.

Appendix A – Scope values

Scope	Secure Resource
grid_exam_submission	NRDR General Radiology Improvement Database (GRID)
lcsr_data_submission	NRDR Lung Cancer Screening Registry (LCSR)
pqrs_data_submission	NRDR Merit-Based Incentive Payment System (MIPS)
nmd_data_submission	NRDR National Mammography Database (NMD)