Administering the API Gateway, 3scale 2.11

December 10, 2021

CHAPTER 12. INTEGRATING 3SCALE AND AN OPENID CONNECT IDENTITY PROVIDER

To authenticate API requests, 3scale can integrate with an identity provider that complies with the OpenID Connect specification. The identity provider can be Red Hat Single Sign-on (RH-SSO) or a third-party identity provider that implements default Keycloak client registration.

The foundation for OpenID Connect is the OAuth 2.0 authentication mechanism, which requires a JSON Web Token (JWT) in an API request to authenticate that request. When you integrate 3scale and an OpenID Connect identity provider, the process has two main parts:

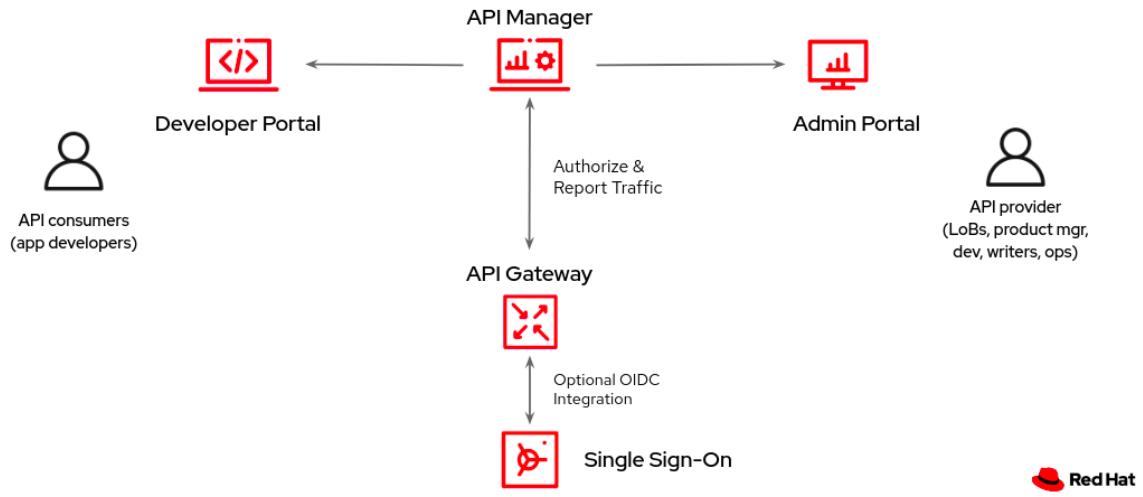
- The APIcast gateway parses and verifies the JWT in the request. If successful, this authenticates the identity of the API consumer client application as well as the particular application end-user.
- The 3scale Zync component synchronizes 3scale application details with the OpenID Connect identity provider.

3scale supports both of these integration points when Red Hat Single Sign-On (RH-SSO) is the OpenID Connect identity provider. See the supported version of RH-SSO on the Supported Configurations page. However, RH-SSO is not a requirement. You can use any identity provider that supports the OpenID Connect specification and default Keycloak client registration. 3scale APIcast integration is tested with RH-SSO and ForgeRock.

The following sections provide information and instructions for configuring the integration between 3scale and an OpenID Connect identity provider:

12.1. Overview of integrating 3scale and an OpenID Connect identity provider

The figure below shows the main 3scale components. The 3scale API gateway is where authentication happens. API providers use the Admin Portal to set up authentication flows. If a 3scale-managed API does not authenticate requests with standard API keys or with application identifier and key pairs, then API providers must also integrate 3scale with an OpenID Connect identity provider. In the figure below, the OpenID Connect identity provider is Red Hat Single Sign-on. With authentication configured and a live Developer Portal, API consumers use your Developer Portal to subscribe to an application plan that provides access to a particular 3scale API product. When OpenID Connect is integrated with 3scale, subscription triggers the OpenID Connect identity provider to send authentication credentials to the API consumer who subscribed.

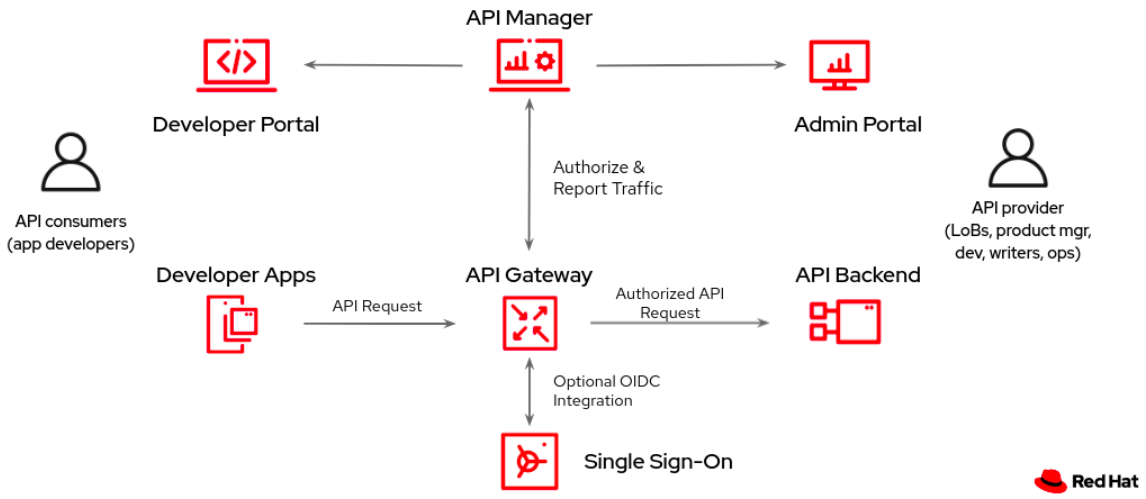


After subscribing to an application plan, an API consumer receives authentication credentials from the integrated OpenID Connect identity provider. These credentials enable authentication of requests that an API consumer application sends to an upstream API, which is the API that is provided by the 3scale product that the API consumer has access to. Credentials include a client ID and a client secret.

An application that an API consumer creates uses these credentials to obtain a JSON web token from the OpenID Connect identity provider. An API consumer must develop an application that does the following for each call to the upstream API backend:

1. Send a request that contains the client ID and client secret to the OpenID Connect identity provider.
2. Receive a JSON web token (JWT) from the identity provider upon authentication.
3. Send an API request that contains the JWT to the upstream API backend.

The 3scale API gateway receives requests from API consumers and checks the JWT in the request. If the gateway verifies the JWT, the gateway sends the request, including the JWT, to the upstream API backend. In the figure below, the OpenID Connect identity provider is Red Hat Single Sign-on but configuration with other OpenID Connect identity providers is possible.



12.2. How the 3scale APIcast gateway processes JSON web tokens

The 3scale APIcast gateway processes each request by checking the JSON web token (JWT) that the OpenID Connect identity provider returns when it authenticates a request. The request now contains the JWT in the format that was issued by the integrated OpenID Connect identity provider. The JWT must be in the `Authorization` header and it must use the `Bearer` schema. For example, the header should look like this:

```
Authorization: Bearer
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwczovL2lkcC5leGFtcGxlL
mNvbSIsInN1YiI6ImFiYzEyMyIsIm5iZiI6MTUzNzg5MjQ5NCwiZXhwIjoxNTM3ODk2MDk0LCJ
pYXQiOjE1Mzc4OTI0OTQsImp0aSI6ImlkMTIzNDU2IiwidHlwIjoiQmVhcmVyIn0.LM2PSmQ0k
8mR7eDS_Z8iRdGta-Ea-pJRrf4C6bAiKz-Nzhxpm7fF7oV3BOipFmimwkQ_-mw3kN--oOc3vU1
RE4FTCQGbzO1SAWHOZqG5ZUx5ugaASY-hUHIohy6PC7dQl0e2NlAeqqg4MuZtEwrpESJW-VnGd
ljrAS0HsXzd6nENM0Z_ofo4ZdTKvIKsk2KrdyVBOcjgVjYongtppR0cw30FwnpqfeCkuATeINN
5OKHXOibRA24pQyIF1s81nnmxLnjnVbu24SFE34aMGRXYzs4icMI8sK65eKxbvwV3PIG3mM0C4
ilZPO26doP0YrLfVwFcqEirmENUAcHXz7NuvA
```

There are a number of open source tools for securely decoding a JWT. Be careful that you do not decode a JWT in a public web tool. In a decoded JWT, you can see that the token has three parts:

- The header provides information about how the token was formed and what algorithm was used to sign the token.
- The payload identifies the API consumer that sent the request. The details can include the read and write actions that this API consumer can perform, an email address for the API consumer, and other information about the API consumer.
- The signature is a cryptographic signature that indicates that the token has not been tampered with.

The gateway checks the JWT for the following characteristics:

- **Integrity**: Is the JWT being tampered with by a malicious user? Is the signature valid? The JWT contains a signature that the token's receiver can verify to ensure that the token was signed by a known issuer and that its content has not been changed. 3scale supports RSA signatures based on public/private key pairs. The issuer signs the JWT token by using a private key. APIcast verifies the token by using a public key. APIcast uses OpenID Connect Discovery for getting the JSON Web Keys (JWK) that can be used to verify the JWT signature.

- **Timing**: Is the current time later than the time when the token becomes acceptable for processing? Has the JWT expired?

- **Issuer**: Was the JWT issued by an OpenID Connect identity provider that is known to the 3scale gateway? In other words, the gateway verifies that the issuer specified in the JWT is the same issuer that an API provider configured in the **OpenID Connect Issuer** field. Specification of the issuer is part of the procedure for integrating 3scale and an OpenID Connect identity provider.

- **Client ID**: Does the token contain a 3scale client application ID that is known to the 3scale gateway? This client ID must match a **ClientID Token Claim** that an API provider specified in the procedure for integrating 3scale with the OpenID Connect identity provider.

If any JWT validation or authorization checks fail, the APIcast gateway returns an *Authenication failed* error. Otherwise, the gateway sends the request to the 3scale upstream API backend. The `Authorization` header remains in the request, so the API backend can also use the JWT to check the user and client identity.

12.3. How 3scale `Zync` synchronizes application details with OpenID Connect identity providers

Zync is a 3scale component that reliably pushes data about 3scale applications to an OpenID Connect identity provider. In this interaction, a 3scale application corresponds to an OpenID Connect identity provider client. In other words, Zync communicates with the OpenID Connect identity provider to create, update and delete OpenID Connect clients.

Zync implements Keycloak default client registration. The use of this API means that the client representation is specific to Keycloak and RH-SSO. The identity provider returns a client ID and a client secret, which are the authentication credentials for the 3scale application.

Zync pushes the data in the form of notifications. Each time 3scale creates, updates, or deletes an application, Zync sends a notification to the OpenID Connect identity provider to update the corresponding client accordingly.

Successful synchronization requires the following settings for a given 3scale product:

- The authentication mechanism is **OpenID Connect**.
- The **OpenID Connect Issuer Type** is either:
    - **RH-SSO** when Red Hat Single Sign-on is the OpenID Connect identity provider. With this issuer type, Zync sends client registration requests according to OpenID Connect Dynamic Client Registration.
    - **REST API** for other OpenID Connect identity providers. With this issuer type, Zync sends client registration requests as shown in the Zync REST API example.
- A URL such as the following is the **OpenID Connect Issuer**:
  `http://id:secret@example.com/api_endpoint`

When deployed to an OpenShift cluster, there are two Zync processes:

- `zync` is a REST API that receives notifications from `system-sidekiq` and enqueues background jobs to `zync-que`. There are notifications for new, updated, and deleted 3scale applications.
- `zync-que` processes these background jobs, which communicate with `system-app`, with the cluster API, and with the OpenID Connect identity provider.

For example, when RH-SSO is the configured OpenID Connect identity provider, Zync creates, updates and deletes clients in the RH-SSO realm.

12.4. Integrating 3scale with Red Hat Single Sign-on as the OpenID Connect identity provider

As an API provider, you can integrate 3scale with Red Hat Single Sign-On (RH-SSO) as the identity provider for authenticating API requests. Part of this procedure is to establish an SSL connection between 3scale Zync and Red Hat Single Sign-On (RH-SSO), because Zync communicates with RH-SSO to exchange tokens. If you do not configure the SSL connection between Zync and RH-SSO, the tokens would be open for anyone listening.

3scale 2.2 and later supports custom CA certificates for RH-SSO with the `SSL_CERT_FILE` environment variable. This variable points to the local path of the certificates bundle.

Integrating 3scale with RH-SSO as the OpenID Connect identity provider consists of configuring the following elements in the following order:

- Optional: 3scale Zync to use custom Certificate Authority certificates. This is not required if RH-SSO uses a certificate issued by a trusted CA.
- Red Hat Single Sign-on to have a 3scale client
- 3scale to work with Red Hat Single Sign-on

Prerequisites

- The RH-SSO server must be available over `https` and it must be reachable by `zync-que`. To test this, you can run `curl https://rhsso-fqdn`
- OpenShift cluster administrator permissions.
- A 3scale API product for which you want to configure OpenID Connect integration with RH-SSO.

Procedure

1. Optional: Configure 3scale Zync to use custom Certificate Authority (CA) certificates. This is not required if RH-SSO uses a certificate issued by a trusted CA.

    1. Replace the placeholders as appropriate and run the following command to get a proper certificate chain:

        ```
        echo -n | openssl s_client -connect <rhsso_fqdn>:<rhsso_port>
        -servername <rhsso_fqdn> --showcerts | sed -ne '/-BEGIN
        CERTIFICATE-/,/-END CERTIFICATE-/p' > customCA.pem
        ```

        Some versions of OpenSSL accept `-showcerts` instead of `--showcerts`. If necessary, modify the command according to the version you are using. Replace the `<rhsso_fqdn>` placeholder with the fully qualified domain name (fqdn) in human-readable format, for example, `host.example.com`.

    2. Validate the new certificate with the following cURL command. The expected response is a JSON configuration of the RH-SSO realm. If validation fails your certificate may not be correct.

        ```
        curl -v https://<secure-sso-host>/auth/realms/master --cacert
        customCA.pem
        ```

3. Gather the existing content of the `/etc/pki/tls/cert.pem` file on the Zync pod by running the following command:

```
oc exec <zync-que-pod-id> cat /etc/pki/tls/cert.pem > zync.pem
```

4. Append the contents of the custom CA certificate file to `zync.pem`:

```
cat customCA.pem >> zync.pem
```

5. Attach the new file to the Zync pod as a `configmap` object:

```
oc create configmap zync-ca-bundle --from-file=./zync.pem
oc set volume dc/zync-que --add --name=zync-ca-bundle
--mount-path /etc/pki/tls/zync/zync.pem --sub-path zync.pem
--source='{"configMap":{"name":"zync-ca-bundle","items":[{"key":
"zync.pem","path":"zync.pem"}]}}'
```

   Addition of the certificate bundle to the Zync pod is complete.

6. Verify that the certificate is attached and the content is correct:

```
oc exec <zync-pod-id> cat /etc/pki/tls/zync/zync.pem
```

7. Configure the `SSL_CERT_FILE` environment variable on Zync to point to the new CA certificate bundle:

```
oc set env dc/zync-que SSL_CERT_FILE=/etc/pki/tls/zync/zync.pem
```

2. In your OpenShift RH-SSO dashboard, configure RH-SSO to have a 3scale client:

   1. Create a realm for a 3scale client or select an existing realm to contain your 3scale client.

   2. In the new or selected realm, create a client:

a. In the **Client ID** field, specify a name that helps you identify this client as the 3scale client, for example **`oidc-issuer-for-3scale`**.

b. Set the **Client Protocol** field to **`openid-connect`**.

c. Save the new client.

3. In the settings for the new client, set and save the following:

a. **Access Type** to `confidential`.
b. **Standard Flow Enabled** to `OFF`.
c. **Direct Access Grants Enabled** to `OFF`.
d. **Service Accounts Enabled** to `ON`. This setting enables this client to issue service accounts.

4. Set the service account roles for the client:

a. Navigate to the **Service Account Roles** tab of the client.
b. In the **Client Roles** dropdown list, click `realm-management`.
c. In the **Available Roles** pane, select `manage-clients` list and assign the role by clicking **Add selected >>**.

5. Note the client credentials:
   a. Make a note of the client ID (`<client_id>`).
   b. Navigate to the **Credentials** tab of the client and make a note of the **Secret** field (`<client_secret>`).

6. Add a user to the realm:
   a. On the left side of the window, expand **Users**.
   b. Click **Add user**.
   c. Enter a username, set ***Email Verified*** to `ON`, and click **Save**.
   d. On the **Credentials** tab, set the password. Enter the password in both fields, set the **Temporary** switch to `OFF` to avoid the password reset at the next login, and click **Reset Password**.
   e. When the pop-up window displays, click **Change password**.

3. In the 3scale Admin Portal, configure 3scale to work with Red Hat Single Sign-on:

    1. In the top level selector, click **Products** and select the 3scale API product for which you are enabling OpenID Connect authentication.
    2. Navigate to **[your_product_name] > Integration > Settings**.
    3. Under **Authentication** , select **OpenID Connect Use OpenID Connect for any OAuth 2.0 flow**. This displays the **OPENID CONNECT (OIDC) BASICS** section.

    4. In the **OpenID Connect Issuer Type** field, ensure that the setting is **Red Hat Single Sign-On**.
    5. In the **OpenID Connect Issuer** field, enter the URL for the configured OpenID Connect identity provider. The format for this URL looks like this:

        ```
        https://<client_id>:<client_secret>@<rhsso_host>:<rhsso_port>/auth/realms/<realm_name>
        ```

        Replace the placeholders with the noted RH-SSO client credentials, the host and port for your RH-SSO server, and the name of the realm that contains the RH-SSO client.

    6. Under **OIDC AUTHORIZATION FLOW**, select one or more of the following:
        - **Authorization Code Flow**
        - **Implicit Flow**
        - **Service Accounts Flow**
        - **Direct Access Grant Flow**

        This configures how API consumers receive JSON web tokens from the OpenID Connect identity provider. When 3scale integrates Red Hat Single Sign-on as the OpenID Connect identity provider, Zync creates RH-SSO clients that have only the **Authorization Code Flow** enabled. This flow is recommended as the most secure and suitable for most cases. Be sure to select an OAuth 2.0 flow that is supported by your OpenID Connect identity provider.

    7. Scroll down and click **Update Product** to save the configuration.

12.5. Integrating 3scale with third-party OpenID Connect identity providers

As an API provider, you can configure an HTTP integration between 3scale and a third-party OpenID Connect identity provider. That is, you can configure an OpenID Connect identity provider other than Red Hat Single Sign-on. 3scale can use this integration to authenticate requests from API consumers and to update the third-party identity provider with the latest 3scale application details.

Most of the work required to integrate 3scale with a third-party OpenID Connect identity provider involves the following two tasks:

- Meeting the 3scale Zync-related prerequisites.
- Configuring your OpenID Connect identity provider to authorize requests from 3scale applications.

After that, you just need to configure a 3scale API product to use your OpenID Connect identity provider. You do this in the 3scale Admin Portal.

Prerequisites

- 3scale Zync is installed.
- Your chosen third-party OpenID Connect identity provider:
    - Adheres to Zync's OpenAPI specification as provided by 3scale.
    - Allows registration of a client with `<client_id> & <client_secret>` declared as a parameter in the request. 3scale is always the source of client identity management in the integration between 3scale and a third-party OpenID Connect identity provider.
    - Is configured for authorizing requests from 3scale applications.

- An adapter for Zync to interact with your OpenID Connect identity provider. To create this adapter, you can modify rest_adapter.rb, which is part of the 3scale Zync REST API example.

  You can include the `rest_adapter.rb` module in the `zync` pod according to the method that best fits your requirements. For example, you could mount a `configMap` through a volume or you can build a new image for Zync. `oc build` a new image for Zync.

Procedure

1. In the 3scale Admin Portal, in the top level selector, click **Products** and select the 3scale API product for which you are enabling OpenID Connect authentication.
2. Navigate to **[your_product_name] > Integration > Settings**.
3. Under **Authentication**, select **OpenID Connect Use OpenID Connect for any OAuth 2.0 flow**.

   This displays the **OPENID CONNECT (OIDC) BASICS** section.
4. In the **OpenID Connect Issuer Type** field, ensure that the setting is **REST API**.
5. In the **OpenID Connect Issuer** field, enter the URL for your OpenID Connect identity provider. The format for this URL looks like this:

   ```
   https://<client_id>:<client_secret>@<oidc_host>:<oidc_port>/<endpoint>
   ```

   For example, in the Zync rest_adapter.rb example, the URL endpoint is hard-coded as `{endpoint}/clients`. Your endpoint might be `{endpoint}/register` or something else.
6. Under **OIDC AUTHORIZATION FLOW**, select one or more of the following:

   - **Authorization Code Flow**
   - **Implicit Flow**
   - **Service Accounts Flow**
   - **Direct Access Grant Flow**

This configures how API consumers receive JSON web tokens from the OpenID Connect identity provider. The **Authorization Code Flow** is recommended as the most secure and suitable for most cases. Be sure to select an OAuth 2.0 flow that is supported by your OpenID Connect identity provider.

7. Scroll down and click **Update Product** to save the configuration.

12.6. Testing 3scale integrations with OpenID Connect identity providers

After integrating 3scale with an OpenID Connect identity provider, you should test the integration to confirm the following:

- API consumers receive access credentials when they subscribe to a 3scale-managed API.
- The 3scale APIcast gateway can authenticate requests from API consumers.

Prerequisites

- Integration between 3scale and your OpenID Connect identity provider is in place for a particular 3scale product.
- An application plan is available for API consumers to subscribe to in your Developer Portal. This application plan provides access to a 3scale-managed API, that is, a 3scale product, for which you configured OpenID Connect authentication.
- An application that sends requests to the upstream API. The upstream API is a backend of the 3scale product that the API consumer has access to as a result of the subscription. Alternatively, you can use Postman to send requests.

Procedure

1. In the Developer Portal, subscribe to an application plan.

   This creates an application in the Developer Portal. The OpenID Connect identity provider should return a client ID and a client secret that you can see in your application's page in the Developer Portal.

2. Note the client ID and the client secret for the application.

3. Verify that your OpenID Connect identity provider now has a client with the same client ID and client secret. For example, when RH-SSO is the OpenID Connect identity provider, you should see a new client in the configured Red Hat Single Sign-on realm.

4. In the application page in the Developer Portal, in the **REDIRECT URL** field, enter the URL for the application that sends API requests to the upstream API.

5. Verify that your OpenID Connect identity provider has the correct redirect URL.

6. Discover the URL that receives authentication requests for your OpenID Connect identity provider by using this endpoint:

   `.well-known/openid-configuration`

   For example:

   `https://<RHSSO_HOST>:<RHSSO_PORT>/auth/realms/<REALM_NAME>/.well-known/openid-configuration`

   For the base URL, use the value that an API provider configured in the **OpenID Connect Issuer** field.

7. In an application that consumes the upstream API, do the following:

   a. Send an authentication request to your OpenConnect identity provider. This request must contain the 3scale application's client ID and client secret. In some cases, the end-user identity is also required.

b. Receive the identity provider's response, which contains the JWT.

c. Send an API request that contains the JWT to the upstream API backend.

If the 3scale gateway can authenticate the JSON web token in the request, your application should receive a response from the API backend.

Alternatively, in place of an API consumer application, use Postman to test that the token flow is correctly implemented.

12.7 Example of a 3scale integration with Red Hat Single Sign-on as the OpenID Connect identity provider

This example shows the flow when you integrate 3scale with Red Hat Single-Sign-on as the OpenID Connect identity provider. This example has the following characteristics:

- In the Admin Portal, an API provider defined a 3scale API product and configured that product to use Red Hat Single Sign-on as the OpenID Connect identity provider.
- This product's OpenID Connect configuration includes:
  - Public base URL: `https://api.example.com`
  - Private base URL: `https://internal-api.example.com`
  - OpenID Connect Issuer: `https://zync:41dbb98b-e4e9-4a89-84a3-91d1d19c4207@idp.example.com/auth/realms/myrealm`
  - RH-SSO realm: **myrealm**
  - 3scale `zync` client in **myrealm** has the correct Service Account roles
-

- In the 3scale Developer Portal, there is an application with the following characteristics. This application is the result of an API consumer subscribing for access to a 3scale API product provided by a particular application plan in the Developer Portal.

    - **Client ID**: `myclientid`
    - **Client Secret**: `myclientsecret`
    - **Redirect URL**: `https://myapp.example.com`

- In Red Hat Single Sign-on, in the **myrealm** realm, there is a client with these same characteristics:

    - **Client ID**; `myclientid`
    - **Client Secret**: `myclientsecret`
    - **Redirect URL**: `https://myapp.example.com`

    Authorization Code Flow, which is the standard flow, is enabled on this client.

- The **myrealm** realm has this user:

    - Username: `myuser`
    - Password: `mypassword`

The flow is as follows:

1. The application that the API consumer created sends an authorization request to RH-SSO by using this endpoint:

    `https://idp.example.com/auth/realms/myrealm/protocol/openid-connect/auth`

    In the request, the application provides these parameters:

    - `Client ID: myclientid`
    - `Client Secret: myclientsecret`
    - `Redirect URL: https://myapp.example.com`

2. RH-SSO shows the login window, where the user must provide their credentials:

    - **Username**: `myuser`
    - **Password**: `mypassword`

3. Depending on the configuration, and whether this is the first time that the user is authenticating in this specific application, the consent window might display.

4. After RH-SSO authenticates the user, the application sends a token request to RH-SSO by using this endpoint:

    `https://idp.example.com/auth/realms/myrealm/protocol/openid-connect/token`

    The request contains these parameters:

    - Client ID: `myclientid`
    - Client secret: `myclientsecret`
    - Redirect URL `https://myapp.example.com`.

5. RH-SSO returns a JSON web token with an "access_token" field such as `eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lk…xBArNhqF-A`.

6. The application that the API consumer created sends an API request to `https://api.example.com` with the header `Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lk…xBArNhqF-A`.

7. The application should receive a successful response from `https://internal-api.example.com`.