Manage access tokens for API requests

09/25/2017 • 4 minutes to read •

In this article

Applies to: Machine Learning Server, Microsoft R Server 9.x

Machine Learning Server, formerly known as Microsoft R Server, uses tokens to identify and authenticate the user who is sending the API call within your application. Users must authenticate when making an API call. They can do so with the 'POST /login HTTP/1.1' API call, after which Machine Learning Server issues a bearer token to your application for this user. Alternately, if the organization is using Azure Active Directory (AAD), users receive a bearer token from AAD when they authenticate.

This bearer token is a lightweight security token that grants the “bearer” access to a protected resource, in this case, Machine Learning Server's core APIs for operationalizing analytics. After a user has been authenticated, the application must validate the user’s bearer token to ensure that authentication was successful.

> **Important**
>
> For proper access token signing and verification across your configuration, ensure that the JWT settings are exactly the same for every web node. These JWT settings are defined on each web node in the configuration file, appsetting.json. Check with your administrator. **Learn more...**

Security Concerns

Despite the fact that a party must first authenticate to receive the token, tokens can be intercepted by an unintended party if the token is not secured in transmission and

storage. While some security tokens have a built-in mechanism to protect against unauthorized parties, these tokens do not and must be transported in a secure channel such as transport layer security (HTTPS).

If a token is transmitted in the clear, a man-in the middle attack can be used by a malicious party to acquire the token to make an unauthorized access to a protected resource. The same security principles apply when storing or caching tokens for later use. Always ensure that your application transmits and stores tokens in a secure manner.

> You can revoke a token if a user is no longer permitted to make requests on the API or if the token has been compromised.

Create tokens

The API bearer token's properties include an access_token / refresh_token pair and expiration dates.

Tokens can be generated in one of two ways:

- If Active Directory LDAP or a local administrator account is enabled, then send a 'POST /login HTTP/1.1' API request to retrieve the bearer token.
- If Azure Active Directory (AAD) is enabled, then the token comes from AAD.

Learn more about these authentication methods.

Example: Token creation request

- **Request**

Copy

```
POST /login HTTP/1.1
{
  "username": "my-user-name",
  "password": "$ecRetPas$1"
}
```

- **Response**

Copy

```
{
  "token_type":"Bearer",
  "access_token":"eyJhbGci....",
  "expires_in":3600,
  "expires_on":1479937454,
  "refresh_token":"0/LTo...."
}
```

Token Lifecycle

The bearer token is made of an access_token property and a refresh_token property.

	The "access_token" Lifecycle	The "refresh_token" Lifecycle
Gets Created	Whenever the user logs in, or a refreshToken api is called	Whenever the user logs in
Expires	After 1 hour (3660 seconds) of inactivity	After 336 hours (14 days) of inactivity
Becomes Invalid	If the refresh_token was revoked, or If not used for 336 hours (14 days), or When a new pair of access_token/refresh_token has been created	If not used for 336 hours (14 days), or When the refresh_token expires, or When a new access_token/refresh_token pair was created, or If the refresh_token was revoked

Use tokens

As defined by HTTP/1.1 [RFC2617], the application should send the access_token directly in the Authorization request header.

You can do so by including the bearer token's access_token value in the HTTP request body as 'Authorization: Bearer {access_token_value}'.

When the API call is sent with the token, Machine Learning Server attempts to validate that the user is successfully authenticated and that the token itself is not expired.

- If an authenticated user has a bearer token's access_token or refresh_token that is expired, then a '401 - Unauthorized (invalid or expired refresh token)' error is returned.
- If the user is not successfully authenticated, a '401 - Unauthorized (invalid credentials)' error is returned.

Examples

Example HTTP header for session creation:

Copy

```
POST /sessions HTTP/1.1
    Host: mrs.contoso.com
    Authorization: Bearer eyJhbGci....
    ...
```

Example HTTP header for publishing web service:

Copy

```
POST /api/{service}/{version} HTTP/1.1
    Host: mrs.contoso.com
    Authorization: Bearer eyJhbGci....
    ...
```

Renew tokens

A valid bearer token (with active access_token or refresh_token properties) keeps the user's authentication alive without requiring him or her to re-enter their credentials frequently.

The access_token can be used for as long as it’s active, which is up to one hour after login or renewal. The refresh_token is active for 336 hours (14 days). After the access_token expires, an active refresh_token can be used to get a new access_token / refresh_token pair as shown in the following example. This cycle can continue for up to 90 days after which the user must log in again. If the refresh_token expires, the tokens cannot be renewed and the user must log in again.

To refresh a token, use the 'POST /login/refreshToken HTTP/1.1' API call.

Example: Refresh access_token

- Example request:

```
POST /login/refreshToken HTTP/1.1
  Connection: Keep-Alive
  Content-Type: application/json; charset=utf-8
  Accept-Encoding: gzip, deflate
  Content-Length: 370
  Host: mrs.contoso.com

  {
    "refreshToken": "0/LTo...."
  }
```

- Example response:

```
{
  "token_type":"Bearer",
  "access_token":"eyJhbGci....",
  "expires_in":3600,
  "expires_on":1479937523,
  "refresh_token":"ScW2t...."
}
```

Revoke refresh tokens

A refresh_token should be revoked:

- If a user is no longer permitted to make requests on the API, or
- If the access_token or refresh_token have been compromised.

Use [the 'DELETE /login/refreshToken?refreshToken={refresh_token_value} HTTP/1.1' API call] to revoke a token.

Example: Revoke token

- Example request:

```
DELETE https://mrs.contoso.com/login/refreshToken?refreshToken=ScW2t
HTTP/1.1
  Connection: Keep-Alive
```

```
  Accept-Encoding: gzip, deflate
  Host: mrs.contoso.com
```

- Example response:

Copy

```
HTTP 200 Success
```