



digital
vision
group

Token Based Access

TRIPsystem
Product Documentation

End User License Agreement

All rights to this software, its documentation and logotypes of the TRIP product family and software (altogether "Software") supplied by DVG Operations GmbH (DVG) are exclusively owned by DVG.

The transfer of this Software, solutions or parts thereof requires the prior written agreement of DVG. Furthermore, the customer has the right to use licensed Software and / or process solutions supplied by DVG to the extent specified in his contract with DVG.

The free-to-use non-commercial version doesn't require a prior written agreement with DVG but such customers, organizations and/or third parties agree by using the software and / or solution of DVG to be strongly obliged to keep all rights to this software, documentation and logotypes of the TRIP product family absolutely un infringed and protected.



Table of Contents

INTRODUCTION	4
DEFINITIONS	4
CONFIGURATION	4
TDBS_REQUIRE_APIKEY	4
TDBS_TOKEN_KEY.....	5
TDBS_ACCESS_TOKEN_EXPIRATION.....	5
TDBS_REFRESH_TOKEN_EXPIRATION.....	5
APPLICATION CONSIDERATIONS	5
KEEP IT SECRET AND SAFE	5
USAGE FROM TRIPNXP AND TRIPJXP	6
USAGE FROM OTHER TRIP API CLIENTS	6
TOKEN AND KEY ADMINISTRATION	6
CREATING AND ROTATING THE TOKEN ENCRYPTION KEY	6
THE TRIPTOK UTILITY	6
<i>API Key Administration.....</i>	<i>7</i>
<i>Token Administration.....</i>	<i>8</i>
OTHER USES OF TRIPTOK	9

Introduction

TRIPsystem 8.4-0 introduces support for access tokens, a common way for applications to access an API without always having to use actual user credentials. TRIP applications can request a token in a session where a non-SYSTEM user has successfully logged on.

In order to better secure access to TRIPsystem, so-called API Keys can also be used in this context. Such keys are assigned by the TRIP administrator to applications that should be able to request and use tokens. If API keys are enabled, only application instances that possess a valid API key can successfully request and use tokens.

Definitions

API Key	An API Key is used as to grant application instances the permission to use token-based access.
Access Token	An Access Token provides access to TRIPsystem for a short time, e.g. 30 minutes or one hour, and can be used for login instead of username and password. Access tokens typically require a valid API Key for both acquisition and use.
Refresh Token	A Refresh Token is returned together with an access token and grants, for the duration of its validity, the application the option to request a new token pair. The Refresh Token has a longer expiration time (e.g. 30 days) but is one-time use only, and usage typically also requires a valid API key.
Token Pair	A pair of one Access Token and one Refresh Token as returned to the application on a successful token creation or refresh request.

Configuration

The following privileged TRIPsystem environment variables can be set in the `tdbs.conf` configuration file to control the behavior of the token subsystem.

TDBS_REQUIRE_APIKEY

This privileged TRIPsystem environment variable is used to determine if and to which degree API Keys are used and required. The following values are valid for this variable:

ALWAYS	An API key is required for creating a new token pair, refreshing a token pair, and for login using an access token. An API key is also required for token administration (e.g. revocation) if requested via the network. This value is the default if the variable is not explicitly set.
TOKENS	An API key is required for creating a new token pair and for refreshing a token pair. Login using an access token does not require an API key.
REFRESH	A valid API key is only required for refreshing a token pair.
NO	API keys are not required by any token operation.

TDBS_TOKEN_KEY

This privileged TRIPsystem environment variable defines the fully qualified path to the server-side file that contains the 256-bit (32-byte) AES encryption key used to encrypt the access and refresh tokens as returned to the calling application.

Rotating this key immediately invalidates all issued access and refresh tokens. For more information on this topic refer to “Token and Key Administration” below.

The absence of a valid value for this variable means that token access is disabled.

TDBS_ACCESS_TOKEN_EXPIRATION

Access tokens are by default set to expire after 3600 seconds, i.e., one hour. To change this, set this privileged TRIPsystem environment variable to an integer value representing the expiration time as the number of seconds since the creation of the token pair.

TDBS_REFRESH_TOKEN_EXPIRATION

Refresh tokens are by default set to expire after 2592000 seconds, i.e., 30 days. To change this, set this privileged TRIPsystem environment variable to an integer value representing the expiration time as the number of seconds since the creation of the token pair.

Application considerations

Keep it Secret and Safe

The access and refresh tokens as obtained by an application as a response to a token creation or refresh call are opaque strings of about 120 byte each. It is very important that these are kept secret and safe. If either is compromised for whatever reason, it is critically important to contact the TRIP administrator so that they immediately can revoke the token pair.

The same goes for the API Key. Since the API key has no expiration time, it is supposed to be rotated by the TRIP administrator on a schedule of their choosing. This makes an API key a semi-permanent, potentially very long-lived object. For this reason, store all copies of an API key as securely as possible, and do not fall for the temptation to store the key in a version management repository since such storage is the same as compromising the security of the key.

If the application is deployed as a Docker container, the API key can for example be stored as a Docker Secret and accessed as such. When the key is rotated by the TRIP administrator, the application container must be configured with the new key. Refer to Docker documentation for details on replacing a Secret.

Usage from TRIPnpx and TRIPjxp

The TRIP SDKs TRIPjxp (for Java) and TRIPnpx (for Microsoft .NET) support tokens from version 8.4-0. The following new methods for token support have been added to the TdbSession class:

- **requestAccessToken**: Call this in a logged in session to acquire a new token pair. May require a valid API key.
- **refreshTokenPair**: Call this to refresh a token pair when the access token has expired. Does not require a logged in session in order to be used, but will normally require a valid API key.
- **revokeToken** and **revokeTokens**: The application can call these methods to revoke (delete) a previously acquired token pair.
- **isTokensEnabled**: Returns true if the connected TRIPsystem server supports tokens. May be called before login.
- **getApiKeyMode**: Returns an indicator for which operations the connected TRIPsystem server requires valid API keys.

Login using a token is, like with the older “login ticket” feature, done via the regular login method. Pass a valid access token as the password. Depending on how TRIPsystem has been configured, the login operation may also require a valid API key. The API key should be passed as the username. If you don’t have an API key and none is required, pass an empty string for the username.

Usage from other TRIP API clients

Except from via the server-side TRIP Toolkit C API, the only supported way to use tokens with a TRIP application is via TRIPnpx and TRIPjxp. The older networked APIs TRIPclient and TRIPjtk are not supported, nor is TRIPhighway.

Token and Key Administration

Creating and Rotating the Token Encryption AES Key

The token encryption key is a 32-byte (256-bit) file referred to by the TDBS_TOKEN_KEY privileged tdb.conf environment variable. This file must contain a cryptographically secure random number, and can be created using the following OpenSSL command:

```
$ openssl rand -out /path/to/my/token/key 32
```

A token encryption key is mandatory unless the privileged TDBS_REQUIRE_APIKEY tdb.conf variable is set to NO.

Rotating the file (i.e., generating a new random key) should be done once in a while as a security measure. Note, however, that this immediately renders all current tokens useless. This also includes refresh tokens. Rotating this key file should therefore be done in parallel to a blanket purge of all tokens, invalid or not.

The TRIPTOK Utility

The TRIPsystem server-side command-line utility program TRIPTOK enables the TRIP administrator to administer API keys and tokens.

API Key Administration

Unless API keys are disabled (see under section “TDBS_REQUIRE_APIKEY”), the TRIP administrator should create one API key per application instance that should be able to access TRIP using tokens. While it technically is possible for several application instances to share an API key, it is not advisable. With each application instance having its own API key it is also easier to revoke and rotate API keys for specific compromised application instances instead.

The TRIPTOK utility can be used to create, list and revoke API keys.

Creating an API Key

Creating an API key using the TRIPTOK utility involves giving it a unique name (via the `-i` option) and a comment (via the `-c` option). The key is the roughly 140-byte string emitted at the end of the program output. This is the only time the key is shown, so make sure to copy it to a secure location.

For example:

```
$ triptok create --key -i KEYNAME -c "Key description"

**** TRIP System Utility TRIPTOK - Token Administration ****
      Version 8.3    07-Oct-2024 10:43:26

hQm1vCXmxdZgMdZBC3/XqgEABABQAEcAHh+125q+5gbsNl8cHeQzDoo+MF4oGR0Nxj
g6BpWzjG+YBMmOHKSqHMDMSaYwydBTH2clAr0oeWAj7IH32gKfHrSn3dxny+9UJXZZ
+rIn/fs=
```

Alternatively, write the key into a JSON file:

```
$ triptok create --key --json -o key.json -i NAME -c "Comment"
```

Listing API Keys

Listing available API keys is done using TRIPTOK as follows. The report includes the key name, the key's comment, the name of the TRIP user who created the key (normally assigned to SYSTEM), and the date and time of creation. The key itself is not shown.

```
$ triptok list --key
```

Or, to produce the report in JSON format:

```
$ triptok list --key --json
```

If you wish to have the report written to a file, you can specify the name of the file using the `-o` option. For example:

```
$ triptok list --key --json -o ~/apikeys.json
```

Revoking an API Key

Revoking a key by its name can be done using TRIPTOK as follows:

```
$ triptok revoke --key -i KEYNAME
```

Token Administration

While tokens are created, refreshed and used from applications, the TRIP administrator can use the TRIPTOK utility to list currently issued tokens, purge expired tokens and revoke compromised tokens.

The TRIP administrator should:

- Run purge regularly to remove fully expired token pairs.
- Keep their eyes open for users with an excessive number of tokens. This may be an indication of a misconfigured application, or of compromised security (e.g. leaked user credentials and API keys).

Listing Tokens

You can list tokens per user or (if you have access to the SYSTEM user) for all users. In addition, user manager users can list the tokens of the users they own. Unprivileged users can only list their own users. For any kind of token listing, TRIPTOK requires the username (`-u`) and password (`-p`) for a TRIP user to log in as. To list a specific user's tokens, specify the name of the user using the `-a` option. The report can optionally be requested in JSON format using the `--json` option, and/or printed to a file using the `-o` option.

As a user with UM (user manager) privileges, list the tokens for user TRIPUSER (password is prompted for if not given on the command line):

```
$ triptok list --token -u TRIPADM -a TRIPUSER
```

As with most of the other TRIPTOK commands, the list command also supports JSON output:

```
$ triptok list --token --json -u TRIPADM -a TRIPUSER
```

Purging Tokens

The purge operation is primarily intended as a way to remove fully expired token pairs. A fully expired token pair is for which both the access token and the refresh token has expired. The purge operation can also be used to remove tokens that are not fully expired using the `-m` option, for which the following values are supported:

- | | |
|----------------|--|
| all | Revoke all token pairs, expired or not |
| access | Purge all token pairs whose access token has expired |
| refresh | Purge all fully expired token pairs (default) |
| created | Revoke tokens created in a particular date interval (options <code>-b</code> and <code>-e</code>) |

For some modes, the purge operation also supports a date range, using the `-b` and `-e` options. If a date range is given, it is assumed to be in UTC and formatted in ISO8601. The *created* mode requires a date range, and is optional for the modes *access* and *refresh*. The *all* mode does not take a date range, as its purpose is to revoke everything.

To purge all fully expired tokens, with output of a list of removed tokens in JSON format:

```
$ triptok purge --token -m refresh --json -u TRIPADM
```

Revoking Tokens

The purpose of the token revoke operation is to remove a specific token pair based on its access or revoke token string, or all tokens for a named user.

To revoke all tokens for user TRIPUSER:

```
$ triptok revoke --token --json -u TRIPADM -a TRIPUSER
```

To revoke a specific token, specify the token with the `-t` option (token string here truncated for convenience):

```
$ triptok revoke --token -u TRIPADM -t "pw0xTlDsryWV4kxzu"
```

Other uses of TRIPTOK

The TRIPTOK utility can also for test purposes be used to acquire and refresh tokens. Note that such creation of tokens is not intended for production use due to the sensitive nature of tokens and the risk involved in having a token string logged in the history of the command shell.