Timetoknow Create SSO

Specification – Initial Draft

**April 2014**

**Modified: January 2015**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

# Table of Contents

[Table of Contents 2](#_Toc387227262)

[1 Intro. 3](#_Toc387227263)

[2 High Level Overview 3](#_Toc387227264)

[3 Creating a JWT Token 4](#_Toc387227265)

[3.1 Forming JWS Header 4](#_Toc387227266)

[3.2 Forming the JWT claim set 4](#_Toc387227267)

[3.3 Computing the Signature 6](#_Toc387227268)

[4 Login to T2K Create 8](#_Toc387227269)

[5 Integration Prerequisites 8](#_Toc387227270)

# Intro.

This document describes the CGS SSO integration specification which is based on the JWT OAuth2 specification.

In a typlical SSO scenario a user which is logged in to an external application would like to lauch the CGS application from within that application without the need to authenticate again.

This document describes how to implement such a flow.

# High Level Overview

The SSO mechanism used to laucn T2K Create is based on [JWT](http://tools.ietf.org/html/draft-ietf-oauth-json-web-token-19) (JSON Web Token) OAuth2 specification to authorize users from external applications.



The SSO flow is described by the following steps:

1. A User logs into an external application
2. The login process is authorized by the external application server
3. The user launches the T2K Create (e.g. from a link in the external application)
4. The external application client requests a JWT token from its server
5. The external application server generates a signed [JWT](#_Creating_a_JWT) token.
6. The external application server returns the JWT token to it’s client
7. The external client application launches the T2K Create. It invoks a request to a login endpoint along with the signed JWT token.
8. If this is the first time ever that the user signs into T2K Create then a new user account will be created for him. Otherwise, the user will sign into his exiting account.

# Creating a JWT Token

JSON Web Token (JWT) is a compact URL-safe means of representing claims to be transferred between two parties. The claims in a JWT are encoded as a JavaScript Object Notation (JSON) object that is used as the payload of a JSON Web Signature (JWS) structure or as the plaintext of a JSON Web Encryption (JWE) structure, enabling the claims to be digitally signed or MACed and/or encrypted.

For more information regarding JWT see specification: <http://tools.ietf.org/html/draft-ietf-oauth-json-web-token-19>

A JSON Web Token (JWT, pronounced "jot") is composed of three parts: a header, a claim set, and a signature. The header and claim set are JSON objects. These JSON objects are serialized to UTF-8 bytes, then encoded using the Base64url encoding. This encoding provides resilience against encoding changes due to repeated encoding operations. The header, claim set, and signature are concatenated together with a period (.) character.

A JWT is composed as follows:

{Base64url encoded header}.{Base64url encoded claim set}.{Base64url encoded signature}

The base string for the signature is as follows:

{Base64url encoded header}.{Base64url encoded claim set}

## Forming JWS Header

The header consists of two fields that indicate the signing algorithm and the format of the assertion. Both fields are mandatory, and each field has only one value.

{"alg":"HS256","typ":"JWT"}

The T2K Create JWT is signed using the HMAC-SH1 algorithm as seen in the above header specification.

The Base64url representation of this is as follows:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9

## Forming the JWT claim set

The JWT claim set contains information about the JWT, including the permissions being requested (scopes), the target of the token, the issuer, the time the token was issued, the lifetime of the token and the username (oauth2 subject) which is attempting to login .   
Like the JWT header, the JWT claim set is a JSON object and is used in the calculation of the signature.

#### Required claims

The claims in the JWT claim set are shown below. They may appear in any order in the claim set.

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Type** |
| iss | The client\_id of the application making the access token request. | String |
| scope | A space-delimited list of the permissions that the application requests. For T2K Create authentication, the scope will hold the following value:  [https://cgs.timetoknow.com/cgs/auth/login/${ROLE}](https://cgs.timetoknow.com/cgs/auth/login/$%7bROLE%7d)  Where role can have the following values:   * EDITOR * AUTHOR | String |
| aud | A descriptor of the intended target of the assertion. When making an access token request this value is https://${t2k create subdomain}.timetoknow.com/cgs/auth/login-oauth2 | String |
| exp | The expiration time of the assertion, measured in seconds since 00:00:00 UTC, January 1, 1970. This value has a maximum of 1 hour from the issued time. | Int (long) |
| iat | The time the assertion was issued, measured in seconds since 00:00:00 UTC, January 1, 1970. | Int (long) |
| sub | The **unique** id of the user attempting to login | String |
| username | The **unique** username of the user attempting to login (without white spaces) | String |
| first\_name | The first name of the user attempting to login |  |
| last\_name | The last name of the user attempting to login |  |
| t2k.ext.catalog.url | (Optional) The URL of an external catalog to which the user can publish his content | String |
| t2k.ext.catalog.type | (Optional) The external catalog type. Currently Create only supports: blossom type | String |

An example JSON representation of the required fields in a JWT claim set is shown below:

{

"iss":"2",

"scope":"https://cgs.timetoknow.com/cgs/login/EDITOR",

"aud":"https://cgs-demo.timetoknow.com/cgs/auth/oauth2/login",

"exp":1328554385,

"iat":1328550785,

"sub":"1234",

"username":"john.smith",

"first\_name":"john",

"last\_name":"smith",

"t2k.ext.catalog.url":"http://lms-brazil.timetoknow.com/WebServices/content/Upload\_ExtID/",

"t2k.ext.catalog.type":"blossom"

}

## Computing the Signature

JSON Web Signature (JWS) is the specification that guides the mechanics of generating the signature for the JWT. The *input* for the signature is the byte array of the following content:

{Base64url encoded header}.{Base64url encoded claim set}

The signing algorithm in the JWT header must be used when computing the signature. The only signing algorithm supported by T2K Create is HMAC using SHA256 hashing algorithm.

Sign the UTF-8 representation of the *input* with the shared secret key using HMAC-SHA256

You can read more about HMAC-SHA256 specification at:

* <http://www.ietf.org/rfc/rfc2104.txt>
* <http://self-issued.info/docs/draft-ietf-oauth-json-web-token.html>

The signature must then be Base64url encoded. The signature is then concatenated with a ‘.’ character to the end of the Base64url representation of the input string. The result is the JWT.

It should be the following (line breaks added for clarity):

{Base64url encoded header}.

{Base64url encoded claim set}.

{Base64url encoded signature}

Below is an example of a JWT before Base64url encoding:

{"alg":"HS256","typ":"JWT"}.

{

"iss":"2",

"scope":"https://cgs.timetoknow.com/cgs/login/EDITOR",

"aud":"https://cgs-demo.timetoknow.com/cgs/auth/oauth2/login",

"exp":1328554385,

"iat":1328550785,

"sub":"1234",

"username":"john.smith"

}.

[signature bytes]

Below is an example of a JWT that has been signed and is ready for transmission (The message was signed with the shared secret key: *secret*):

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiIyIiwic2NvcGUiOiJodHRwczovL2Nncy50aW1ldG9rbm93LmNvbS9jZ3MvbG9naW4vRURJVE9SIiwiYXVkIjoiaHR0cHM6Ly9jZ3MtZGVtby50aW1ldG9rbm93LmNvbS9jZ3MvYXV0aC9vYXV0aDIvbG9naW4iLCJleHAiOjEzMjg1NTQzODUsImlhdCI6MTMyODU1MDc4NSwic3ViIjoiMTIzNCIsInVzZXJuYW1lIjoiam9obi5zbWl0aCJ9.ZI7mTlCYdwg1weeMJavWe5RVPDe8BcfwPZuirHoUkGs=

# Login to T2K Create

After generating the JWT token an application may use it in order to login a user to the T2K Create.   
The application should peform the following GET request in the taget window/tab in which T2K Create will be opened.

https://${T2K Create subdomain}.timetoknow.com/cgs/auth/login-oauth2?jwt=${JWT Token}

For example:

The token above will be sent by the following request for the cgs on cgs-production-qa server (note the jwt query encoding)

https://cgs-production-qa.timetoknow.com/cgs/auth/login-oauth2?jwt=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiIxIiwic2NvcGUiOiJodHRwczovL2Nncy50aW1ldG9rbm93LmNvbS9jZ3MvbG9naW4vRURJVE9SIiwiYXVkIjoiaHR0cHM6Ly8xMjMxMjMxMi50aW1ldG9rbm93LmNvbS9jZ3MvYXV0aC9vYXV0aDIvbG9naW4iLCJleHAiOjI5MTI3LCJpYXQiOjEzOTk4OTQwNDc1MDQsInN1YiI6InVuaXF1ZUlkIiwidXNlcm5hbWUiOiJ0ZXN0VXNlciJ9.WFpx7G9GvkTUlKwh46MbF6Kql1nADHVh9dJNVZOSYwQ%3D

# Integration Prerequisites

Contact T2K to get the shared secret and client ID parameter values