revIgniter User Guide Version 2.1.7

Search User Guide Search...

1. revIgniter Home
2. User Guide Home
3. JSON Web Token Helper
4. Table of Contents Page

- revIgniter
- User Guide Home
- Table of Contents Page
- Basic Info
- Server Requirements
- License Agreement
- Change Log
- Credits
- Installation
- Downloading revIgniter
- Installation Instructions
- Upgrading from a Previous Version
- Troubleshooting
- Introduction
- Getting Started
- revIgniter at a Glance
- revIgniter Cheatsheets
- Supported Features
- Application Flow Chart
- Model-View-Controller
- HMVC - Extending MVC
- Architectural Goals

- General Topics
- revIgniter URLs
- Controllers
- Reserved Names
- Views
- Models
- Helpers
- Using revIgniter Libraries
- Creating Your Own Libraries
- Creating Core Libraries
- Using Stacks
- Hooks - Extending the Core
- Auto-loading Resources
- Common Handlers
- Scaffolding

JSON Web Token Helper

The JSON Web Token Helper contains handlers which allows you to generate, decode and verify JSON Web Tokens. It follows the industry standard RFC 7519 which is a means for representing claims as a JSON object to be transferred between two parties. The JWT helper uses the HMAC SHA256, SHA384 or SHA512 hashing algorithm to sign tokens, RSA public/private key pairs are currently not supported. If you are not familiar with JSON Web Tokens you can read about the concept at https://jwt.io.

Note: This helper requires the LiveCode Builder (library) extension com.livecode.library.json which needs to be stored in `application/extensions`. So, the path to the extension is `application/extensions/com.livecode.library.json/module.lcm`. Please read about how to include/load the LiveCode JSON extension in chapter "Extensions".

Note: HMAC SHA384 and SHA512 hashing algorithms are not supported on LiveCode server prior to version 9.

Loading this Helper

This helper is loaded using the following code:

```
rigLoadHelper "jwt"
```

The Key

The key (secret) used to sign the token should be saved to a file named "jwt.lc" in `application/config`. Preferably this key is a binary string which implicitly needs to be base64 encoded. If you don't specify a key it will be generated for you using the LiveCode *randomBytes(64)* function and saved to `application/config/jwt.lc` on calling the handler to encode a token. Keep in mind that if your server is

not totally under your control it's impossible to ensure key security so you may want to think carefully before using it for anything that requires high security.

Handler Reference

The following handlers are available:

rigJWTencode(*pHeader*, *pPayload*, *pNumBits*)

This function generates and returns a JSON Web Token. The first parameter which is used to build the header object is optional and is generated automatically if not defined. Usually you can leave this parameter empty or else it needs to be a comma delimited key value list or an array. The header generated by revIgniter consists of the type of the token, which is "JWT" and the appropriate hashing algorithm ("HS256" on LiveCode below version 9, otherwise the algorithm is determined by the value of the third optional parameter). Like the first parameter the mandatory second parameter can be a comma delimited key value list or an array. The optional third parameter is used to determine the hashing algorithm. Valid values are "256", "384", "512" or empty. If the parameter is empty the default value on LiveCode server below version 9 is "256", otherwise "512" is used. Example using HS256 on LIveCode server below vesion 9 and HS512 on LiveCode server version 9 or higher:

Copy

```
put the seconds into tIAT
put tIAT + 10 into tNBF
put tIAT + 60 into tEXP
put uuid() into tJTI
put "sub, jwt test, name," && tUsername & ", admin, true, iat," && tIAT & ", nbf," \
        && tNBF & ", exp," && tEXP & ", jti," && tJTI into tJWTpayload

put rigJWTencode( , tJWTpayload) into tJWT

put tJWT into tTokenA["jwt"]

# rigSetHeader DOES NOT WORK WITH AJAX, USE: put header ...
# GENERATES AN OBJECT ON CLIENT SIDE. SO, DON'T USE JSON.parse() TO PROCESS THE DATA
put header "Content-type: application/json; charset=UTF-8"
put JsonExport(tTokenA)
```

Following a variant using an array as second parameter:

Copy

```
put the seconds into tIAT
put tIAT + 10 into tNBF
put tIAT + 60 into tEXP

# NOTE: SECONDS NEED TO BE A STRING, OTHERWISE THE DATA OF JSONEXPORT IS NOT VALID.
put tIAT & "" into tJWTpayloadA["iat"]
```

```
put tNBF & "" into tJWTpayloadA["nbf"]
put tEXP & "" into tJWTpayloadA["exp"]
put "jwt test" into tJWTpayloadA["sub"]
put tUsername into tJWTpayloadA["name"]
put "true" into tJWTpayloadA["admin"]
put uuid() into tJWTpayloadA["jti"]

put rigJWTencode( , tJWTpayloadA) into tJWT

put tJWT into tTokenA["jwt"]

# rigSetHeader DOES NOT WORK WITH AJAX, USE: put header ...
# GENERATES AN OBJECT ON CLIENT SIDE. SO, DON'T USE JSON.parse() TO PROCESS THE DATA
put header "Content-type: application/json; charset=UTF-8"
put JsonExport(tTokenA)
```

These two samples above assume that the client side provides code which handles the response in form of a JSON object.

rigJWTdecode(*pToken*, *pNumBits*)

Decode and validate a JSON Web Token. The parameter representing a token is optional. If not provided the function assumes that the client side has sent the JWT in an "Authorization" header using the "Bearer" schema (Authorization: Bearer <token>). The optional second parameter is used to determine the hashing algorithm. Valid values are "256", "384", "512" or empty. If the parameter is empty the default value on LiveCode server below version 9 is "256", otherwise "512" is used. Example using HS256 on LIveCode server below vesion 9 and HS512 on LiveCode server version 9 or higher:

Copy

```
put "The secret data." into tSecretData

put rigJWTdecode() into tJWTA

if tJWTA["valid"] is TRUE then

  put the seconds into tCurrentTime
  put TRUE into tTimeValid

  if tCurrentTime < tJWTA["payload"]["nbf"] then
    put "Token not yet valid." into tA["responsedata"]
    put FALSE into tTimeValid
  else if tCurrentTime > tJWTA["payload"]["exp"] then
    put "Token expired." into tA["responsedata"]
    put FALSE into tTimeValid
  end if

  if tTimeValid is TRUE then
    if tJWTA["valid"] then
      put tSecretData into tA["responsedata"]
    else
```

```
      put tJWTA["response"] into tA["responsedata"]
    end if
  end if

else-- if tJWTA["valid"] is TRUE
  # RESPONSE IS ERROR "HTTP/1.0 400 Bad Request" OR "HTTP/1.0 401 Unauthorized"
  put tJWTA["response"] into tA["secretstring"]
  rigSetStatusHeader word 2 of tJWTA["response"]
end if -- if tJWTA["valid"] is TRUE

# rigSetHeader DOES NOT WORK WITH AJAX, USE: put header ...
# GENERATES AN OBJECT ON CLIENT SIDE. SO, DON'T USE JSON.parse() TO PROCESS THE DATA
put header "Content-type: application/json; charset=UTF-8"
put JsonExport(tA)
```

The sample above assumes that the client side provides code which handles the response in form of a JSON object.

Data Returned

This function returns an **array** including the following keys:

- "header"
  (the value is the token header as array)
- "payload"
  (the value is the token payload as array)
- "valid"
  (the value is a boolean, this refers to the integrity of the token)
- "response"
  (the value is empty in case the token is valid, otherwise it is a string "HTTP/1.0 401 Unauthorized", in case the token can not be decoded the string is "HTTP/1.0 400 Bad Request" - use these strings as response header if something went wrong and the integrity of the token could not be determined)

Previous Topic: HTML Helper :: Top of Page :: User Guide Home :: Next Topic: Language Helper