Implementing JWT with Uniface

This document describes the components that implement JSON Web Tokens using Uniface as well as a sample Login screen that returns a token and a simple test harness to create and verify token creation.

The files included with this document are Uniface 10 component exports and they need to be imported into a Uniface 10 project/environment. The following files are included:

Filename	Description
cpt_jwt	JWT implementation component
cpt_jwtbuild	The JWT verification/encoding interactive component
cpt_jwttest	The JWT login test component
cpt_timeutil	Uniface Time service for creating/decoding Unix epoch times

These components need to be imported into Uniface 10 and compiled. For additional information on JWT please refer to https://tools.ietf.org/html/rfc7519 for the complete definition of JWT.

JSON Web Tokens Lecture series presentation: https://www.slideshare.net/Uniface/uniface-lectures-webinar-application-infrastructure-security-json-web-tokens

Video Recording : https://www.youtube.com/watch?v=tDFjowsQg5A&feature=youtu.be

JWT Server Page

JWT Operations

The following tables list the operations exposed by the JWT type.

Operations

Name	Description
AddToHeaders(string JWT)	Add the string JWT token to the HTTP Request headers
Authorize	Inspect the HTTP headers or token request param and validate token
CheckExpiration(string JWT)	Check the expiration, not before and evidence of tampering
Create(	Creates a JWT token based upon the elements presented. It is encrypted using the SECRET_KEY specified in the SECRET_KEY logical in the [LOGICALS] settings of the assignment file.

Datatype	Element	Direction
string	Issuer	In
string	Subject	In
string	Audience	In
datetime	Expiration	In
datetime	Notbefore	In
string	otherParams	In
boolean	includeIssuedAt	In
raw	outputToken	out

)

decodeJWT(

Datatype	Element	Direction
string	JWT	In
string	Secret_key	In
string	header	out
string	payload	out
datetime	signature	out
boolean	Verified	out

)

Given a JWT token and optionally the SECRET_KEY – decode the token into its distinct pieces. If the SECRET_KEY is specified Verified will also be set

Login(

Datatype	Element	Direction
string	username	In
string	password	In
string	outputToken	out

)

Sample Login operation that given a username of ‘sample’ and a password of ‘123123’ –

- Creates a JWT token
- Adds the JWT token to the headers (AddToHeaders)
- Checks the token for validity (Authorize)
- Returns the token in the Response

See Also

TIMEUTIL Service

TIMEUTIL Operations

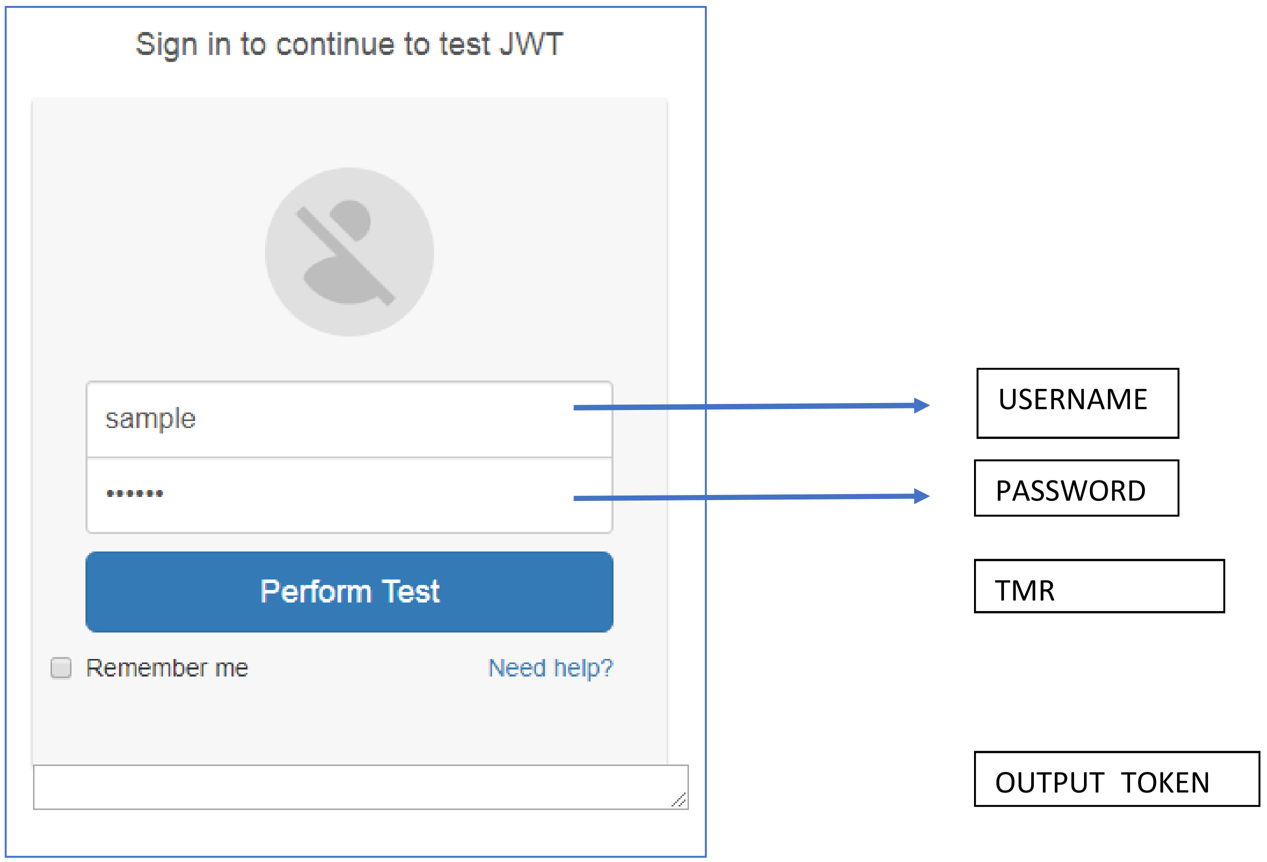
The following tables list the operations exposed by the TIMEUTIL service.

Operations

Name	Description
FromDateTime(datetime InputDate)	Returns a Unix epoch integer from an input date
ToDateTime(Datatype / Element / Direction Numeric / Inputdate / In Datetime / outputDate / Out)	Returns a datetime from a Unix epoch integer.

Datatype	Element	Direction
Numeric	Inputdate	In
Datetime	outputDate	Out

JWTTEST Server Page - Layout



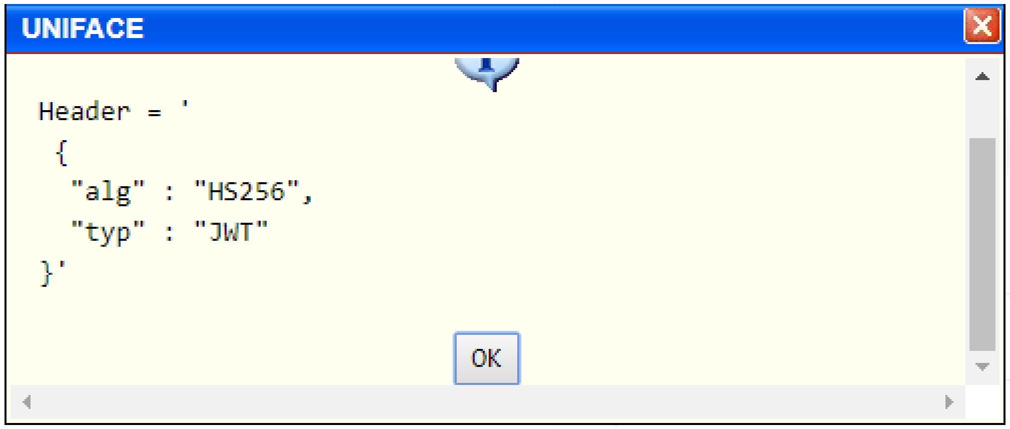
JWTTEST ServerPage

The following tables list the operations exposed by the JWTTEST server page.

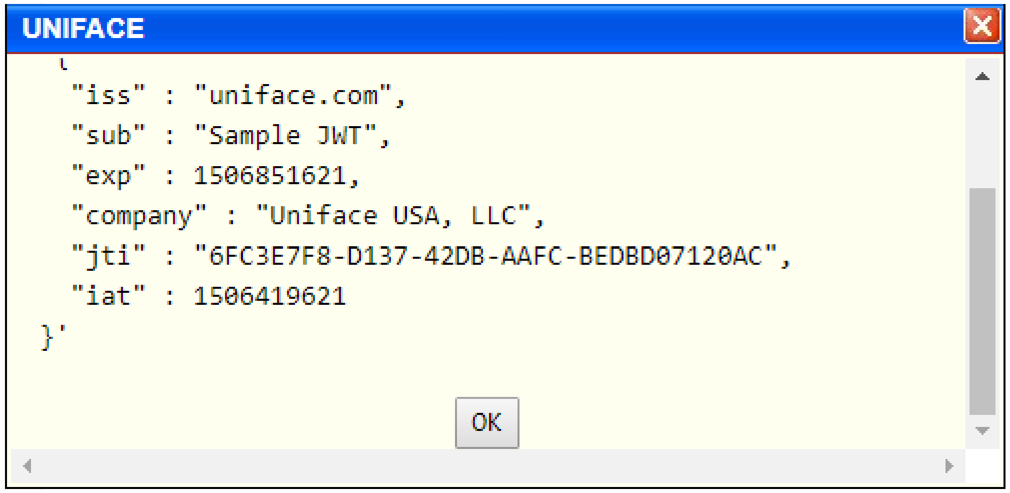
Triggers

Field	Code
TMR **(Perform Test)**	see code below

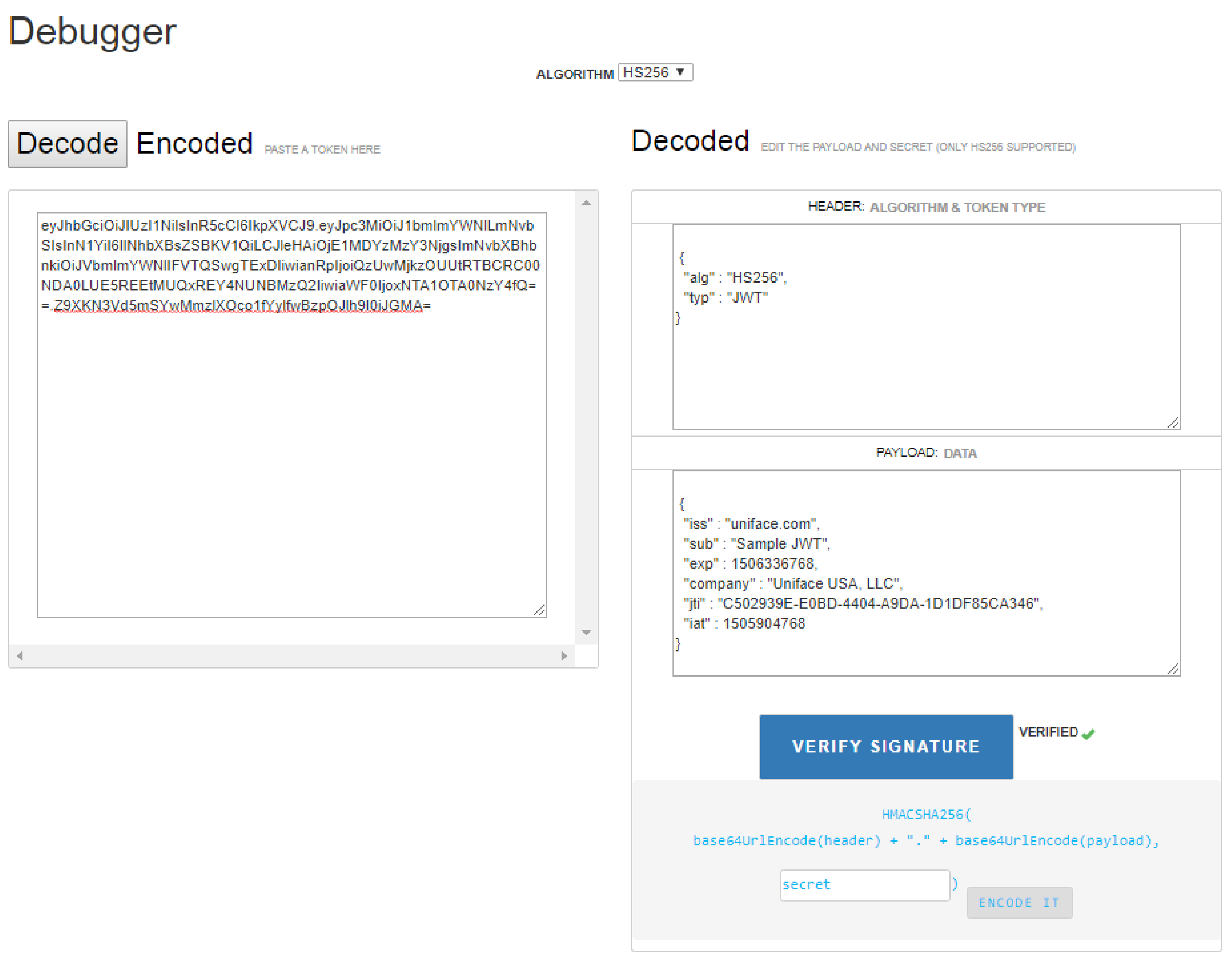
```
trigger detail
public web
    variables
        string output_token
        string header
        string payload
        string signature
        string secret_key
        boolean verified
        handle JWTHandle
    endvariables

    secret_key = ""

    newinstance "JWT", JWTHandle

    JWTHandle->login(username,password,output_token)
    JWTHandle->decodeJWT( output_token, secret_key, header, payload, signature, verified )
    webmessage/info "Header = '%%header'"
    webmessage/info "Payload = '%%payload'"
    returntoken = output_token
end
```



```
Header = '
 {
  "alg" : "HS256",
  "typ" : "JWT"
}'
```



```
  "iss" : "uniface.com",
  "sub" : "Sample JWT",
  "exp" : 1506851621,
  "company" : "Uniface USA, LLC",
  "jti" : "6FC3E7F8-D137-42DB-AAFC-BEDBD07120AC",
  "iat" : 1506419621
}'
```



Debugger

ALGORITHM HS256 ▼

Decode Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ1bmlmYWNlLmNvb
SIsInN1YiI6IlNhbXBsZSBKV1QiLCJleHAiOjE1MDYzMzY3NjgsImNvbXBhb
nkiOiJVbmlmYWNlIFVTQSwgTExDIiwianRpIjoiQzUwMjkzOUUtRTBCRC00
NDA0LUE5REEtMUQxREY4NUNBMzQ2IiwiaWF0IjoxNTA1OTA0NzY4fQ=
=.Z9XKN3Vd5mSYwMmzlXOco1fYylfwBzpOJlh9l0iJGMA=
```

Decoded EDIT THE PAYLOAD AND SECRET (ONLY HS256 SUPPORTED)

HEADER: ALGORITHM & TOKEN TYPE

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

PAYLOAD: DATA

```
{
 "iss" : "uniface.com",
 "sub" : "Sample JWT",
 "exp" : 1506336768,
 "company" : "Uniface USA, LLC",
 "jti" : "C502939E-E0BD-4404-A9DA-1D1DF85CA346",
 "iat" : 1505904768
}
```

VERIFY SIGNATURE VERIFIED ✓

```
HMACSHA256(
base64UrlEncode(header) + "." + base64UrlEncode(payload),
secret )
```

ENCODE IT

This component allows you to test the JWT encoding and decoding process.

	This takes the JWT token entered or pasted and decodes the Header, Payload and Signature and places the decoded information into the fields on the right side of the page.
	This uses the secret key specified below it to verify the signature. The only way to verify the token is to have the encoding key.
	This uses the key specified, the Header and the Payload to create a token and places it in the left hand side of the page. This is the corollary to the Decode button.