

NetVista 分布式质量监测系统

JWT介绍

|  |  |  |
| --- | --- | --- |
| 文件状态：  [ ] 草稿  [ ] 正式发布  [ √ ] 正在修改 | 文件标识： | VIXTEL-NETVISTA-PRODUCT-JWT |
| 当前版本： |  |
| 作 者： |  |
| 完成日期： |  |



飞思达技术（北京）有限公司

Vixtel Technologies Limited

版本修订历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本/状态 | 作者 | 参与者 | 起止日期 | 备注 |
|  |  |  |  |  |

目录

[什么是jwt 3](#_Toc505875695)

[Jwt说明 3](#_Toc505875696)

[基本流程如下： 3](#_Toc505875697)

[JWT格式 4](#_Toc505875698)

[JWT的构成 4](#_Toc505875699)

[Jwt接口说明(Jwt库为静态库) 5](#_Toc505875700)

[公共登录接口中jwt使用 7](#_Toc505875701)

# 什么是jwt

Json web token(JWT)，是为了在网络环境中传递声明而执行的一种基于JSON的开放标准(RFC 7519)。该token被设计为紧凑且安全的，特别适用于分布式站点的单点登录场景。JWT的声明一般被用来在身份提供者和服务提供者间传递被认证的用户身份信息，以便于从资源服务器获取资源，也可以增加一些额外的其它业务逻辑所必须的声明信息，该token也可以直接被用于认证，也可以被加密。

# Jwt说明

## 基本流程如下：

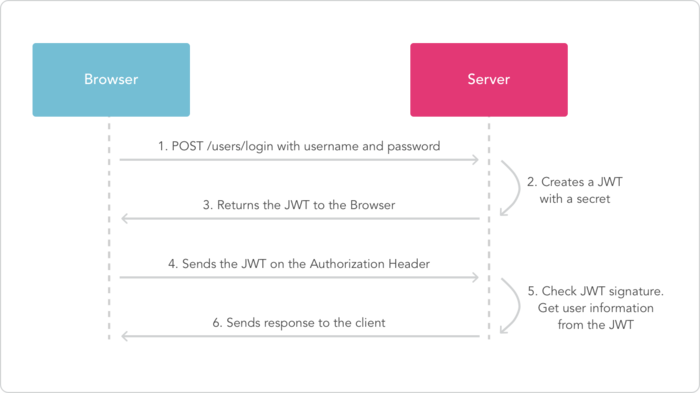
用户使用用户名和密码来请求服务器

服务器进行验证用户的信息

服务器通过验证发送给用户一个token

客户端存储token，并在每次请求时附送上这个token

服务端验证token值，并返回数据



## JWT格式

JWT是由三段信息构成的，将这三段信息用”.”（点）连接一起就构成了JWT字符串，如下示例：

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9.TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ

## JWT的构成

第一部分我们称它为头部(header)，第二部分我们称其为载荷(payload)，第三部分是签证(signature)。

1. header---JWT的头部承载量部分信息：

声明类型，这里是jwt

声明加密的算法，通常直接使用HMAC SHA256(也可以使用其它加密算法)

完整的头部就像下面的JSON：

{

'typ': 'JWT',

'alg': 'HS256'

}

然后将头部进行base64编码，就构成了第一部分

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9

1. payload---JWT的载荷部分就是存放有效信息的部分，这些有效信息包含三个部分
2. 标准中注册的声明

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 含义 | 类型 | 说明 |
| iss | Jwt签发者 | String | 自定义字符串 |
| sub | Jwt所面向的用户 | String | 自定义字符串 |
| aud | 接收jwt的一方 | String | 自定义字符串 |
| exp | Jwt的过期时间 | Int64 | 时间戳 |
| nbf | Jwt的生效时间 | Int64 | 时间戳 |
| iat | Jwt的签发时间 | Int64 | 时间戳 |
| jti | Jwt的唯一身份标识 | String | 自定义字符串 |

1. 公共的声明

公共的声明可以添加任何的信息，一般添加用户的相关信息或其他业务需要的必要信息，但不建议添加敏感信息，因为该部分在客户端可解密。

1. 私有的声明

私有声明是提供者和消费者所共同定义的声明，一般不建议存放敏感信息，因为base64是对称解密的，意味着该部分信息可以归类为明文信息。

定义一个payload

{

"sub": "1234567890",

"name": "John Doe",

"admin": true

}

然后将其进行base64编码，得到JWT的第二部分

eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9

3.signature---jwt的第三部分是一个签证信息，这个签证信息有三部分组成：

Header(base64后的)

Payload(base64后的)

Secret（加盐字符串，自定义）

这个部分需要base64编码后的header和base64编码后的payload使用”.”连接组成的字符串，然后通过header中声明的字符串进行加盐组合加密，然后就构成了jwt的第三部分

**注意：secret是保存在服务器端的，jwt的签发生成也是在服务器端的，secret就是用来进行jwt的签发和jwt的验证，所以，它就是你服务器端的私钥，在任何场景都不应该流露出去。一旦客户端得知这个secret，那就意味着客户端可以自我签发jwt了。**

4.总结

优点：

因为json的通用性，所以JWT是可以进行跨语言支持的

因为有了payload部分，所以JWT可以在自身存储一些其它业务逻辑所必要的非敏感信息

便于传输，jwt的构成非常简单，字节占用很小，所以便于传输

不需要再服务器端保存会话信息，所以易于扩展

安全相关

不应该再jwt的payload部分存放敏感信息，因为该部分是客户端可解密的部分

保护好secret私钥

# Jwt接口说明(Jwt库为静态库)

1. 接口名称：SetJsonWebTokenTimeout(const int64&)

接口说明：设置JWT生存周期，单位秒

参数说明：int64整数

返回值：void

1. 接口名称：SetJsonWebTokenAddWords(const string&)

接口说明：设置JWT第三部分加盐字符串

参数说明：string加盐字符串

返回值：void

1. 接口名称：SetJsonWebTokenIss(const string&)

接口说明：设置JWT签发者

参数说明：string签发者字符串

返回值：void

1. 接口名称：SetJsonWebTokenIssueds(const string&)

接口说明：设置服务器端信任的签发者

接口参数：string 以逗号分隔

例如信任签发者为vixtel和netvista，参数为”vixtel,netvista”

返回值：void

1. 接口名称：AddExtension(const string&, const string&)

接口说明：向payload部分添加扩展字段

接口参数：第一个参数为字段key，第二个为key对应的value

返回值：void

1. 接口名称：GetExtension(const string&, const string&, string&)

接口说明：获取payload中扩展字段中的某个字段

接口参数：第一个参数为jwt字符串，第二个为要获取的字段名，第三个我字段名对应的值

返回值：存在返回true，不存在返回false

1. 接口名称：DelExtension(const string&)

接口说明：删除payload中扩展字段的某个字段

接口参数：要删除字段的key值

返回值：void

1. 接口名称：ClearExtension()

接口说明：清空payload中扩展字段

接口参数：无

返回值：void

1. 接口名称：CreateJWT(string&, string&, string& int64)

接口说明：创建JWT

接口参数：第一个string为返回创建成功的jwt字符串，第二个为签发者字符串，第三个为登录用户名，第四个为登录用户名对应的用户id

返回值：创建成功返回true，失败返回false

1. 接口名称：ValidateJWT(string&, string&, string&, int64&)

接口说明：验证jwt并返回jwt中的签发者，登录账户和登录账户id

接口参数：第一个string为jwt字符串，第二个存储返回的签发者字符串，第三个存储登录账户名，第四个存储登录账户id

返回值：int32对应的错误码如下表

|  |  |
| --- | --- |
| 错误码 | 错误信息 |
| 0 | 成功 |
| 1 | Jwt超时 |
| 2 | Jwt还未到达生效时间 |
| 3 | 非法签发者 |
| 4 | 无效的JWT |
| 5 | 加密算法不是MD5 |
| 6 | 其它错误 |

# 公共登录接口中jwt使用

1. 流程
2. 访问子系统，子系统对访问权限进行检查，发现无jwt或者jwt非法后返回302跳转，跳转到公共登录接口

<http://xx.xx.xx.xx:3012/?template=index-auth1&subsystem=>http%3a%2f%2f172.30.1.31%3a5052%2f%3fsystemIdentification%3dliveserver

注：subsystem为进行url编码后的子系统登录地址，此url示例: <http://172.30.1.31:5052/?systemIdentification=liveserver>这是未转码前的url，在子系统访问url的基础上添加一个参数systemIdentification表示子系统标识，然后进行url编码后未：

http%3a%2f%2f172.30.1.31%3a5052%2f%3fsystemIdentification%3dliveserver

2.用户输入账户信息后cloudserver进行验证，验证通过后将子系统模块权限信息作为扩展字段添加到payload中，扩展字段名称为”sys”，value为子系统模块权限信息

3.Cloudserver将生成的jwt字段和子系统url进行拼接成一个新的url，返回302跳转，重新访问子系统，url格式为http://172.30.1.31:5052/?systemIdentification=liveserver &jwt={jwt}

4.子系统获取jwt后对jwt进行验证和模块权限信息验证

注：应用Netvista认证登录子系统需要在用户管理中给对应的用户配置登录子系统的权限，如liveserver子系统登录，首先要在用户管理中配置对应子系统权限，如下图：

