



**SPECIFICHE TECNICHE PER L'ACCESSO AI SERVIZI COOPERATIVI DI ANSC
DETTAGLI SU COME CREARE E USARE JWT/JWS**

INDICE

1. INTRODUZIONE	3
2. COSTRUZIONE DEL TOKEN JWT	4
3. COSTRUZIONE DEL JWS	11
4. ESEMPIO DI RICHIESTA CON JWS DETACHED	17
5. LIBRERIE PER LA FIRMA/VERIFICA DI TOKEN	22
6. ESEMPIO DI CLIENT JAVA	23
7. COME RICHIEDERE SUPPORTO	24
8. FAQ	25
9. DISCLAIMER	27

FIGURE:

Figura 1 - Applicazione Web OTP da utilizzare per generare il valore del codice OTP	6
Figura 2 - Esempio di validazione JWT/JWS con il tool online jwt.io	14
Figura 3 - Esempio di chiamata al servizio di Upload Allegato - body della richiesta	17
Figura 4 - Esempio di chiamata al servizio di Upload Allegato - Bearer Token JWT	18
Figura 5 - Esempio di chiamata al servizio di Upload Allegato - impostazione header JWS	18
Figura 6 - Esempio di chiamata al servizio di Upload Allegato - risposta del servizio con esito positivo	19
Figura 7 - Lista di alcune delle librerie disponibili per Java	22
Figura 8 - Apertura del file PKCS#12 di postazione con il tool KeyStore Explorer	25

TABELLE:

Tabella 1 - End-Point URL servizi cooperativi	3
Tabella 2 - Struttura dell'header per il token JWT di accesso ai servizi cooperativi	4
Tabella 3 - Struttura del payload del token JWT	5
Tabella 4 - Tabella delle FAQ	26

CODICE:

Code 1 - Esempio di risposta dei servizi cooperativi nel caso in di OTP scaduto	6
Code 2 - Esempio di risposta dei servizi cooperativi nel caso in di OTP non valido	7
Code 3 - Esempio di Token JWT nella forma encoded (header, payload e signature)	9
Code 4 - Token JWT nelle parti di header e payload nella forma decoded	10
Code 5 - Header che specifica il tipo di algoritmo di firma e media-type	12
Code 6 - Esempio di JWS nella forma Compact Serialization con la signature RS256	12
Code 7 - Esempio dell'Header e Payload in forma decoded	13

Code 8 - Esempio di JWS Detached da inserire all'interno dell'header HTTP JWS.....	15
Code 9 - Chiamata al servizio di Upload Allegato in modalità JWS Detached utilizzando il tool cURL	20
Code 10 - Esempio di risposta nel caso in cui dovesse fallire la validazione del JWS.....	21

1. INTRODUZIONE

I servizi cooperativi esposti dal sistema ANSC¹ e dedicati all'integrazione con i sistemi gestionali in dotazione ai comuni, sono di tipo REST² (descritti via OpenAPI³) il cui accesso in sicurezza è fondato sull'uso dello standard JWT⁴.

Gli end-point dei servizi cooperativi sono disponibili alle software house per gli scopi d'integrazione e test degli stessi su distinti ambienti che sono: validazione e pre-produzione.

L'accesso ai servizi in ambiente di validazione può avvenire utilizzando il classico JWT, al contrario, sull'ambiente di pre-produzione, l'accesso richiede un ulteriore livello di sicurezza con l'adozione del cosiddetto JWS⁵ Detached.

Ambiente	End-Point URL	JWT	JWS
Validazione	https://anscserviceval.anpr.interno.it/services/service	SI	NO
Pre-Produzione	https://anscservicepre.anpr.interno.it/services/service	SI	SI

Tabella 1 - End-Point URL servizi cooperativi

Le specifiche OpenAPI dei servizi cooperativi e relativa documentazione sono state rilasciate il 27/12/2022 con numero di versione [1.3.1](#).

L'obiettivo di questo documento è quello d'illustrare come costruire il JWT e il JWS Detached affinché questi possano essere utilizzati dai client REST per interagire in modo corretto con i servizi cooperativi esposti dal sistema ANSC.

¹ Archivio Nazione dello Stato Civile

² Representational state transfer (REST) è uno stile architetturale per sistemi distribuiti. L'espressione "representational state transfer" e il suo acronimo, REST, fu introdotto nel 2000 nella tesi di dottorato di Roy Fielding. Il termine REST rappresenta un sistema di trasmissione di dati su HTTP senza ulteriori livelli, quali ad esempio SOAP.

³ OpenAPI è uno standard per la descrizione delle interfacce di programmazione, ovvero le Application Programming Interfaces (API). La specifica OpenAPI definisce un formato di descrizione aperto e non soggetto a licenza per i servizi API. In particolare, tramite OpenAPI si possono descrivere, sviluppare, testare e documentare le API REST.

⁴ JSON Web Tokens standard di settore open definito dall'[RFC 7519](#) utilizzato come metodo per rappresentare in modo sicuro le richieste tra due parti.

⁵ Una firma Web JSON (abbreviata in JWS) è uno standard proposto da IETF ([RFC 7515](#)) per la firma di dati arbitrari. Questo viene utilizzato come base per una varietà di tecnologie basate sul Web, tra cui JSON Web Token.

2. COSTRUZIONE DEL TOKEN JWT

Affinché sia possibile accedere ai servizi cooperativi, è necessario che la richiesta sia corredata da un token JWT generato in modo corretto sia dal punto di vista formale sia dal punto di vista del contenuto, ovvero, dell'header e del payload.

Il token JWT è costituito da tre parti che sono: header, payload e signature. Le due tabelle a seguire mostrano la struttura dell'header e del payload. Non è oggetto di questo documento il processo di creazione della signature del JWT; fare riferimento all'[RFC 7519](#) o alla documentazione del tool kit (o framework) JWT specifico.

Attributo	Descrizione	Valore
alg	L'algoritmo principale in uso per firmare e/o de-crittografare il token JWT.	È possibile indicare diverse famiglie di algoritmi (fare riferimento a JSON Web Algorithms (JWA)). Quello attualmente supportato è RS256.
tpy	Tipo di media-type del JWT.	JWT
x5c	Catena dei certificati X.509. Ogni certificato deve essere nella codifica Base64 della relativa rappresentazione DER PKIX. Il primo certificato nell'array deve essere quello utilizzato per firmare il JWT, seguito dal resto dei certificati nella catena di certificazione.	Un array JSON di certificati X.509 utilizzati per firmare il token. Un esempio (troncato): "x5c": ["MIIE6zCCAtOgAwIBAgIIZt..."]

Tabella 2 - Struttura dell'header per il token JWT di accesso ai servizi cooperativi

Il payload deve rispecchiare la struttura indicata dalla tabella seguente.

Attributo	Descrizione	Valore
sub	L'identificativo dell'utente che effettua l'operazione. Nel caso in cui il valore specificato fosse errato, il servizio risponde con un codice di errore HTTP/500.	Esempio: MSRNTN77H15C351X
sede	Codice ISTAT del comune per il quale si opera. 1. Nel caso in cui il valore specificato fosse errato, il servizio risponde con un codice di errore HTTP/401. 2. Il valore è relazionato a quello dell'attributo <i>postazione</i> , in caso d'inconsistenza il servizio risponde con un codice di errore HTTP/401.	Esempio: 016017
postazione	Nome del certificato utilizzato. In genere corrisponde al CN del certificato di postazione. Nel caso in cui il valore specificato fosse errato, il servizio risponde con un codice di errore HTTP/500.	Esempio: 016017-PC-000
otp	Codice OTP (One Time Password) di sicurezza fornito da apposita applicazione web.	Esempio: 123456
jti, exp e iat	Identificativo del token, data/ora di scadenza del token e data/ora inizio validità del token.	Fare riferimento all' RFC-7519 per il formato dei valori

Tabella 3 - Struttura del payload del token JWT

Il valore dell'OTP è possibile generarlo tramite l'apposita applicazione web, oppure, nel qual caso avrà la validità prevista. Alternativamente, per il solo ambiente di test è possibile usare il valore 123456, che sarà sempre valido. In ambiente di produzione sarà disponibile la sola prima opzione, ovvero, la generazione dell'OTP tramite l'applicazione web. L'OTP generato ha una durata di quattro ore. La figura a seguire mostra

l'applicazione web che genera il valore da inserire sull'attributo OTP del payload del token JWT.




Figura 1 - Applicazione Web OTP da utilizzare per generare il valore del codice OTP

I possibili errori che i servizi cooperativi potrebbero restituire a riguardo il codice OTP sono due, esattamente:

1. *codice OTP scaduto* nel caso in cui il codice non sia più valido perché ha superato la durata del tempo di vita che al momento è impostato a quattro ore;
2. *codice OTP non valido* nel caso in cui il codice non sia mai stato generato dal sistema.

A seguire sono mostrati due esempi di risposta nel caso di errori che riguardano il codice OTP e descritti in precedenza.

```
{
  "testataRisposta": {
    "idComune": 580,
    "idOperazioneComune": "APP2022032210000",
    "idOperazione": "710241",
    "idEsito": 999
  },
  "datiPaginazione": null,
  "errors": [{
    "code": "406002",
    "severity": "E",
    "text": "Codice OTP scaduto"
  }],
  "success": null,
  "infos": null,
  "warnings": null
}
```

Code 1 - Esempio di risposta dei servizi cooperativi nel caso di OTP scaduto

```
{
  "testataRisposta": {
    "idComune": 580,
    "idOperazioneComune": "APP2022032210000",
    "idOperazione": "710241",
    "idEsito": 999
  },
  "datiPaginazione": null,
  "errors": [{
    "code": "406001",
    "severity": "E",
    "text": "Codice OTP non valido"
  }],
  "success": null,
  "infos": null,
  "warnings": null
}
```

Code 2 - Esempio di risposta dei servizi cooperativi nel caso di OTP non valido

A seguire è mostrato un esempio completo di token JWT in tutte le sue parti (header, payload e signature) nella forma *encoded*.

Per ragioni di leggibilità, quanto mostrato in Code 3, riportata le parti separate sempre dal carattere di *dot* (.) seguito da un *CRLF*.

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsIng1YyI6WyJNSU1FNnpDQ0F0T2dBd0lCQWdJSVp0cHRGUjZVd3lJd
0RRWUpLb1pJaHJzTkFRRUxUUF3V2pFYk1Ca0dBMVVFQXd3U1EwRwVRz16ZEdGNmFXOXVhU0JUZG1sc01RMHdDd1
1EV1FRT
ERBUk3UbeJTTVI4d0hRWURWUvFLREJaTmFXNXBjM1JsY204Z1pHVNniQ2RKYm5SbGNTNXZNUN3Q1FZFRZRUUDf0
pKvKRBZUZ3MH1NakEzTwpZeE1ETBNVFPphRncweU5URXdNRFV4TVRBNU1qUmFNSUdaTVJjd0ZRURWUvFEREE0d01
UwXdNVGN0VUV
NdE1EQXdNVEVktUJzR0ExVUVDd3dVVUc5emRHRjZhVz11YVNCa2FTQnNZWFp2Y204eE1UQXZCZ05WQkFzTUtFRnVZ
V2R5WVdabE1FNWhlbWx2Ym1Gc1pTQ1FiM0J2YkdGNmFXOXVaU0JTWh0cFpHVnVkr1V4SHpBZEJnT1ZCQW9NRmsxc
GJtbHpkR1Z5Y
n1Ca1pXeHNKM6x1ZEdeWwJtOHhDekFKQmdOVkzBWRBa2xvTU1JQk1qQU5CZ2txaGtpRz13MEJBUUvGQUFPQ0FR0E
FNSU1CQ2dLQ0FRRUFOY2hyK0d4NnVBSkxFCWtUWkM4aT1DFTZaK0kwZGZmQTZKT0VyUFDQWV2WFXb1dCMTkycEd
LK3lys21FQ1d
4YkNzaUpvSjlRbHJfCEfjODJyTERBb1pONHpvVldmMmhGSHVXWk80YjBRQnc2ZFNZSFBUNGHaM20yL1BGNkNKQ01a
QktXNGHvt2xmTW5kNFd0ZEEvY0Q1S0hhrHJJTnQ0dzRVS2kxamJ3SnJ5Mm1TbU0zYmNpZz1xVy9oTlo2ODNNM1dIa
XdEMjB1bmtBV
WF0RHRtTERySUhMbXm1Uz1rWfo1N31TWEZhyXI3WgZUjdxU091K3hyaXhWUFVSU1JzYmttNkRmRmphUDhEek5MV1
VqY21DwnFqdk1JTjl6dEd6dHNOQTEycjdURWtkT1BDbUhyVw9POGdobnpjZ3g4MERTeTk4Y0p0Z3hBNm1OSkF3UWp
DaFFJREFRQUJ
vM1V3Y3pBTUJnT1ZIUK1CQWY4RUFQQUFNQjhHQTFVZEl3UUVlNQmFBRkFZTG44cWgxNW10MUNMdXMDxhGS1BJVjV0
K01CTUdBMVVKs1FRTU1Bb0dDQ3NHQVFRkzJ3TUNNQjBHQTFVZERnUUVdCQ1Jhenh6wjFaaWo3U1E3R2dQdU2ZRWhTc
TdzT2pBT0JnT
1ZIUThCQWY4RUJBTUNCNEF3RFFZSkvWk1odmNOQVFfTEJRQRnZ01CQUt1MVZ6eERQVVVOawtK2K2cydzxhVHZwbX
V5dDdJdHVLZmt4cWJla29ZbFJJL1NMV1hSWGpyT1N6am9nQnE2M0ZnTj10UjJvBGFdGNMfHir2hGM1VGyM5FUKU
1WdZ0UkxjbVY
zaG5KNVYrWlA5SEdPNmhaRm1HK2hrZ2FFRm9xY3VuMUpCOUtycFRPVG5xOU9yc1BYSGRNM1ZDWWFoU0tEeT1pL21R
NnpjVVBqd0wRZDVR0VE3L2tDUSs3ZnJaZ3NyKzZWU3hDZUZhU4zaDh5ajU2S2EraVNDcz1FSEFZaUzMbXQvZXdyV
FBaWmxj0HVUC
G9US1hOY31MVXV5MkRTR0o30E1CSjRwOXDPSFdIMGRoa11EU2tIempIeT1jc3BVUDNpwjZxNkhGRE41NzR5S5jdWU1
p1TnJFdvFKclcwdfRVUnV0WHPHYVE0MkVFR1YxbXVrbzJpSDdaTX1mTEJ6VXZoQXZrR3p2VXVkuJjNuNgTJc3lqOUp
3TzdpdkEvTjN
keFp1MzMwTTYzR0FyYkhLSXFnZxhCRWtsN1dsdzBiUXdENFo5Q1Racw9BRFERQitPZE5TZVhJSFdv3pkOC9TTzI0
VUP4ejB0bk1XU1Yzcf04NzZ5bGczd3UwSEdm1V6ZVZMOC9tQ2xTYUg4RHo1UmIrNUpOZmZPYTV4eVZGU0I1ZER5U
DVTt0ZHa1dzU
HNQNZlhSktyS2EyanpxU1NnellpdTBwT3U2MFJBNjk4N2JQU3hZCwXsdVIwcGNMODN1Uldab1RW0VF5TT1TRm1UbT
JJUXdWnjdXYnYyS25yQnRYb1hpNE5YY0ZYMzZsb0xYEWwyMkdpwVRRVi9SbGxNS112Wk9iVnFRMGZvVTNOBTfoN3d
tcnVIAxyzenV
BdTLGe1prUjVWNGVHI119
.
eyJzdWIiOiJNU1JOVE43N0gxNUMzNTFYIiwiaWF0IjoxNjcyMzAyOTA1LCJleHAiOiJlZ2NzI1MTg5MDUsIm5iZiI6M
TY3MjUxODkwNSwianRpIjoiTnRyOXhkcE5STmtmNjRPR09XWnFGQSIInNlZGU0IiIwMTYwMTciLCJw
b3N0YXppb251IjoimDE2MDE3LVBDLTAWMDEiLCJvdHAiOiIxmMjM0NTYifQ
.
X1lp7c7Ryq0DcM5GuYh6Kc3c1mm0rXCE-y0S9PgDqbL1dMe8h-
vq0NrLPRd5Q72fgjH00xnuFcktvB5x1k6VqL435zE8leqKK1_mjDSOCf1H0AnFev27N1FTAWefMp8qI8H
eLGfitXY0FEaTTYFFMQ_dNSiN_Xv1_uo3JRwn30t1V_GTNZraySB-XE3U4iCgQW5CLa64P4hwnr-
R8nCKM9KY8qTo2lq1joM77pGRNOI9g7WthV9SgI9QajT10KFMmiaZcV_21XLqZKsU7WT90bDzPG7aioJ3i5jwJIE1
K_m5p7qAe0dagGmPSyOw-d5TJ
rEx_EsJjEWItFwr2NiKHw

Code 3 - Esempio di Token JWT nella forma encoded (header, payload e signature)

A seguire è mostrato un esempio completo di token JWT (quello di Code 3) nelle parti di header e payload nella forma *decoded*.

Header
<pre>{ "alg": "RS256", "typ": "JWT", "x5c": ["MIIIE6zCCAt0gAwIBAgIIZtptFR6UwyIwDQYJKoZIhvcNAQELBQAwwjEbmBkGA1UEAwSQ0EGUG9zdGF6aW9uaSBTdm1sMQ0wCwYDVQQLDARBT1BSMR8wHQYDVQQKDBZNaW5pc3Rlcm8gZGVsbCJbnRlcm5vMQswCQYDVQQGEwJJVDAeFw0yMjA3MjYxMDM0MTZaFw0yNTEwMDUxMTA5MjRaMIGZMRcwFQYDVQQDDA4wMTYwMTctUEMtMDAwMTedMBsGA1UECwwUUG9zdGF6aW9uaSBkaSBsYXZvcn8xMTAvBgNVBAsMKEFuYWdyYWZlIE5hem1vbWFsZSBQb3BvbG9uZSBzZXNpZGVudGUxHzAdBgNVBAoMFk1pbmlzdGVybyBkZWxsJ01udGVybm8xCzAJBgNVBAYTAklUMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEhchr+Gx6uAJLEqkTZC8i9CLVZ+I0dffA6J0ErPWEAevYqWnWB192pGK+yrKmECWxbCsiJoJ9Q1rEpAc82rLDAoZN4zoVwf2hFHUwZ04b0QBw6dSYHPT4hZ3m2/PF6CJCIZBKW4ho01fMnd4WtdA/cD5KHaDrInt4w4UKi1jBwJry2mSmM3bcig9qW/hNZ683M3WHiWd20unkAUatDtmLDrIHLms5S9kXZ57ySXFaar7Xh3R7qS0u+xrrixVPURRRsbkm6DFFjaP8DzNLVUjcmCZqjvMIN9ztGztshA12r7TEkdOPCmHrUo08ghnzcgx80D5y98cJtgxA6iNJAwQjChQIDAQABo3UwczAMBgNVHRMBAf8EAjAAMB8GA1UdIwQYMBaAFAYLn8qh15mt1CLus4uxFJPIV5t+MBMGA1UdJQQMMAoGCCsGAQUFBwMCMB0GA1UdDgQWBBRazxzZ1Zij7RQ7GgPuFvEhmq7s0jA0BgNVHQ8BAf8EBAMCB4AwDQYJKoZIhvcNAQELBQAAdggIBAKu1VzxDPUNikP+g2w8qTvpmyt7ItuKfKxqbekoY1RI/SLWXRXjr0SszjogBq63FgN9tR2UleEtC0X0bGhF2UFbnERE5X6tRLcmV3hnJ5V+ZP9HG06hZFiG+hkgaEfoqcun1JB9KrpT0Tnq90rsPXHdM2VCXQhSKDy9i/iQ6zcUPjwL+d5k9Q7/kCQ+7FrZgsr+6VSxCeFauN3h8yj56Ka+iSCs9EHAYiFLmt/ewrTPZZ1c8uTpoTJXNcyLUuy2DSGJ78MBJ4p9wOHWH0dhkYDSkHzzHy9cspUP3iZ6q6HFDN574yJ7VRZuNrEuQJrW0tTURutXzaaQ42EEFV1muko2iH7ZMyfLBzUvhAvkGzvUudR3n4kIsy9Jw07ivA/N3dxZe330M63GArbHKIqgexBEk17WlW0bQwD4Z9BTZqoADQ+B+0dnSeXcHWokzd8/SO24UJxz0tnIWRV3pZ876y1g3wu0HGFwUzeVL8/mC1SaH8Dz5Rb+5JNff0a5xyVFSB5dDyP5S0FGkwsPsP79aJKrKa2jzqSSgzYiu0p0u60RA6987bP5xYq1RuR0pCL83uRWZoTV9QyM9SfiTm2IQwV67Wbv2KnrBtXoXi4NXcFX361oLDa122GiYTQV/Rl1MKYvZ0bVqQ0foU3Nm1h7wmruHiv3zuAu9FzZkR5V4eG"] }</pre>
Payload
<pre>{ "sub": "MSRNTN77H15C351X", "iat": 1672302905, "exp": 1672518905, "nbf": 1672518905, "jti": "Ntr9xdpNRNkf640G0WZqFA", "sede": "016017", "postazione": "016017-PC-0001", "otp": "123456" }</pre>

Code 4 - Token JWT nelle parti di header e payload nella forma decoded

Per le operazioni di verifica della firma o per semplice attività di debug, è possibile utilizzare il tool online jwt.io.

3. COSTRUZIONE DEL JWS

JSON Web Signatures (o JWS) è probabilmente una delle caratteristiche più utili dello standard JWT. Combinando un semplice formato di dati (come JSON) con una serie ben definita di algoritmi di firma, i token JWT sono rapidamente diventati l'ideale formato per la condivisione sicura dei dati tra client e intermediari.

È stato deciso di adottare JWS per consentire a una o più parti di stabilire l'autenticità del JWT. Autenticità che in questo contesto significa che i dati contenuti nel JWT non sono stati manomessi.

Qualsiasi soggetto che possa eseguire un controllo della firma può fare affidamento sui contenuti forniti dal JWT. È importante sottolineare che una firma non impedisce ad altre parti di leggere il documento contenuto all'interno del JWT.

I JWT firmati sono definiti nella specifica JSON Web Signature, [RFC 75151](#).

Esistono due serializzazioni definite per i JWS: una compatta, quella utilizzata dai servizi cooperativi e una JSON.

Il JWS serializzato compatto è una stringa contenente tre parti (in ordine) unite da un punto ("."). Ogni parte è codificata in Base64URL.

1. Header
2. Payload
3. Signature

L'**Header** (o intestazione) descrive la firma digitale o il codice di autenticazione del messaggio (MAC⁶) applicato al payload e facoltativamente proprietà aggiuntive del JWS. A seguire (vedi Code 5) è mostrata la parte dell'header prima della codifica: è una struttura JSON.

⁶ In crittografia un Message Authentication Code (MAC) è un piccolo blocco di dati utilizzato per garantire l'autenticazione e integrità di un messaggio digitale, generato secondo un meccanismo di crittografia simmetrica: un algoritmo MAC accetta in ingresso una chiave segreta e un messaggio da autenticare di lunghezza arbitraria, e restituisce un MAC (alle volte chiamato anche tag). In ricezione il destinatario opererà in maniera identica sul messaggio pervenuto in chiaro ricalcolando il MAC con lo stesso algoritmo e la stessa chiave: se i due MAC coincidono si ha autenticazione e integrità del messaggio inviato.

```
{
  "alg": "RS256",
  "typ": "JWT"
}
```

Code 5 - Header che specifica il tipo di algoritmo di firma e media-type

Per i valori dell'attributo *alg* fare riferimento a quanto riportato in Tabella 2. Il valore attualmente supportato per l'attributo *alg* è RS256.

Il **Payload** può essere qualunque tipo di contenuto (anche binario); nello specifico uso dei servizi cooperativi, il contenuto rispecchia il body della richiesta al servizio REST.

La **Signature** è calcolata nel modo definito per il particolare algoritmo utilizzato (e dichiarato nell'header) da:

```
Base64URLEncoded(header).Base64URLEncoded(payload).Base64URLEncoded(
Signature of Base64URLEncoded(header).Base64URLEncoded(payload))
```

A seguire è riportato un esempio completo di JWS (in tutte le sue parti) sia nella forma *encoded* sia nella forma *decoded*.

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9
.
ewoidGVzdGF0YVJpY2hpZXN0YSI6IHsKImlkQ29tdW51IjogMzY1NSwKImlkT3BlcmF6aW9uZUNvbXVvZSI6ICJBUF
AyMDIyMDMyMjEwMDAwIiwKImlRhdGFpcmlFSAwNoaWVzdGEiOiAiMjAyMi0wOC0xNyIsCiJub21lQXBwbGljYXRpdM8i
OiAiQXBwbGljYXRpdM8gRGVtb2dyYWZpY28iLAoidmVyc2l2bmVBcHBsawNhdG12byI6IClXljAuMCIsc1Jmb3JuaX
RvcmluYXBcHBsawNhdG12byI6ICJOb21lIEZvcm5pdG9yZSIKfSwKImlFsbGVnYXRvSW5wdXQiOiB7CiJjb250ZW51dG8i
OiAiUVU1VFF3PT0iLAoidm9tZWZpbGUiOiAiZW50ZWZpbGUiLAoidG1wb0ZpbGUiOiAiMSIsCiJ0aXBvQWxsZW
dhdG8iOiAiMSIKfQp9
.
fs0oNxltdgZbvsVQhzbjhICU0Bzj7L1rkXSCrfGaIdigU2UQQYzyfiTFd6YSjKXSH_QFqw0NwsH0WQxjBZRWZeyze0
Ups0L3zoZRA3MBdYw1TbEkYsZRL1Z31BScSa8pXuqP8dPgfxeG7YsbkgFIDemDlnEsNJ0BgNFR6EwxDHmpsJfXdh0X
goEdVRYoVusN2YTHd9DqBIOU6Kq3nafg0To14SvewE7NGqRwbvaq1LKCVrsZFT1tssRgDMdhapQVKbPkhekYp9R6cT
SM4Zt9DfGRqR5-SYcJ9_fjVwbj0JDGEar-V8wcYRiDEGhqQTFH0iqTn5pH0FoyEEGqG9_RiQ
```

Code 6 - Esempio di JWS nella forma Compact Serialization con la signature RS256

Header
<pre>{ "alg": "RS256", "typ": "JWT" }</pre>
Payload
<pre>{ "testataRichiesta": { "idComune": 580, "idOperazioneComune": "APP2022032210000", "dataOraRichiesta": "2022-08-17", "nomeApplicativo": "Applicativo Demografico", "versioneApplicativo": "1.0.0", "fornitoreApplicativo": "Nome Fornitore" }, "allegatoInput": { "contenuto": "QU5TQw==", "nomefile": "mionomefile", "tipoFile": "1", "tipoAllegato": "1" } }</pre>

Code 7 - Esempio dell'Header e Payload in forma decoded

In questo specifico caso, il payload mostrato in Code 7 rappresenta il body in formato JSON della chiamata al servizio di *Upload Allegato* dei servizi cooperativi.

Anche per il caso del JWS, è possibile utilizzare il tool online jwt.io per verificare che il JWS creato sia corretto, in particolar modo per quel che riguarda la firma. La figura a seguire mostra un esempio del tool utilizzato per la validazione del JWS.

[illegible]

Figura 2 - Esempio di validazione JWT/JWS con il tool online jwt.io

Il **JWS Detached**, richiesto per accedere ai servizi cooperativi, è una variante di JWS che consente di firmare il contenuto (corpo o body) della richiesta o risposta HTTP senza la sua modifica.

Alla richiesta HTTP deve essere aggiunta l'intestazione HTTP **JWS**, che contiene dati che consentono di verificare se il messaggio non è stato modificato nel tragitto dal mittente al destinatario.

L'algoritmo di generazione JWS Detached è molto semplice e consiste in:

- a) generare un JWS standard in formato serializzazione compatta utilizzando il corpo HTTP come payload;
- b) eliminare la parte centrale del JWS, ovvero, il Payload;
- c) inserire la stringa finale nell'intestazione HTTP JWS.

Con riferimento all'esempio di JWS trattato in precedenza (vedi Code 6), il JWS Detached da inserire all'interno dell'header HTTP JWS è quello mostrato a seguire.

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9..Cn3Xu8Lbn9TKoZHC4AadHXP0EzYEsOX2u81xo_3vdzP9vTkxU78f6uMrP6cswqaaMBA3DqWIahqiqLKp29_E7uWKS0wN8eV7SKemtuvwt_r5y9K3CRAPg1N0uc1Q9aEPx4H0qB0unh2_4fV0q0QvizSg5vvuTZ7S00y7at1r2u4bFn6vs103vdArXbrUTn2PUNsQ7V2kH1e57BXUiVAVYf1AY_KT64diqeLC-lVS11NKY9_70CH72ZAegU1VJ_IDGLg_DyShq5y8xG4SEegVQtmkzE9b_T9fvNNe_fvWTjy0ViXn38q-Ynox6UBmd6FSomIixLy89-d3vpJ2_zg5AQ
```

Code 8 - Esempio di JWS Detached da inserire all'interno dell'header HTTP JWS

Riassumendo, il flusso da seguire per generare correttamente una chiamata a uno dei servizi cooperativi in modalità JWS Detached è indicato a seguire:

- 1) generazione del KeyPair⁷ partendo dal PKCS⁸#12⁹ di postazione consegnato;
- 2) estrazione del certificato pubblico dal KeyPair;
- 3) lettura del payload in formato JSON che rappresenta il body della richiesta da inviare al servizio cooperativo;
- 4) generazione del JWT specifico dei servizi cooperativi di ANSC e firmato con la chiave privata del KeyPair;

⁷ Le coppie di chiavi (chiave pubblica e chiave privata) sono generate con algoritmi crittografici basati su problemi matematici definiti funzioni unidirezionali o one-way. La sicurezza della crittografia a chiave pubblica dipende dal mantenere segreta la chiave privata; la chiave pubblica può essere distribuita apertamente senza compromettere la sicurezza.

⁸ PKCS (acronimo di Public-Key Cryptography Standards) sono delle specifiche utilizzate nella crittografia informatica prodotte dai laboratori della RSA inc., in collaborazione con sviluppatori sparsi in tutto il mondo, al fine di incrementare l'uso della tecnica di cifratura a chiave Asimmetrica.

⁹ Standard per la sintassi dello scambio d'informazioni personali. Definisce il formato del file usato per conservare la chiave privata e i certificati pubblici, proteggendoli con una parola chiave.

- 5) generazione del JWS in Compact Serialization e firmato con la chiave privata del KeyPair;
- 6) generazione del JWS Detached partendo dal JWS in Compact Serialization;
- 7) invocazione del servizio cooperativo di ANSC passando il JWS Detached come HTTP header (di nome JWS) della richiesta.

4. ESEMPIO DI RICHIESTA CON JWS DETACHED

Dopo aver generato il token JWT (secondo le specifiche indicate in Tabella 2 e Tabella 3) per l'**Authentication Bearer Token**¹⁰ HTTP header e il JWS (vedi 3 *Costruzione del JWS*) da inserire all'interno dell'apposito HTTP header di nome JWS, è possibile procedere con la costruzione della richiesta verso uno dei servizi cooperativi.

In questo esempio di richiesta verso il servizio di *Upload Allegato*, sarà utilizzato il comunissimo tool [Insomnia](#). Le figure a seguire illustrano un esempio di chiamata verso il servizio indicato in precedenza.

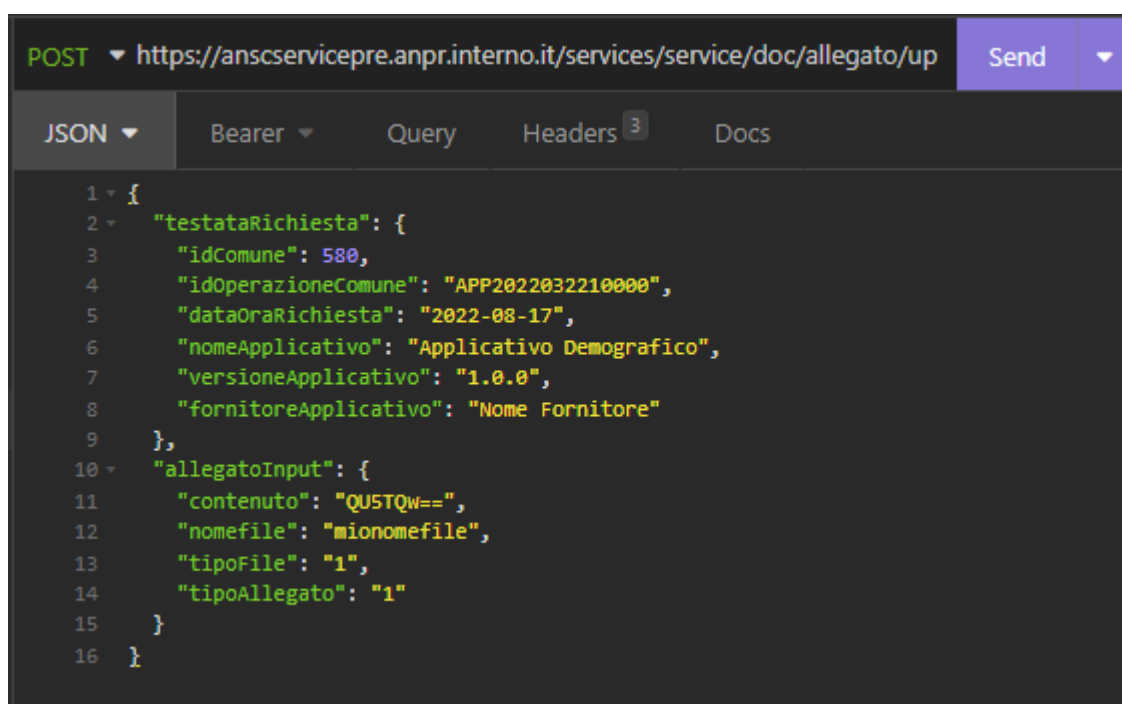


Figura 3 - Esempio di chiamata al servizio di Upload Allegato - body della richiesta

¹⁰ Per maggiori informazioni fare riferimento [all'RFC 6750 The OAuth 2.0 Authorization](#)

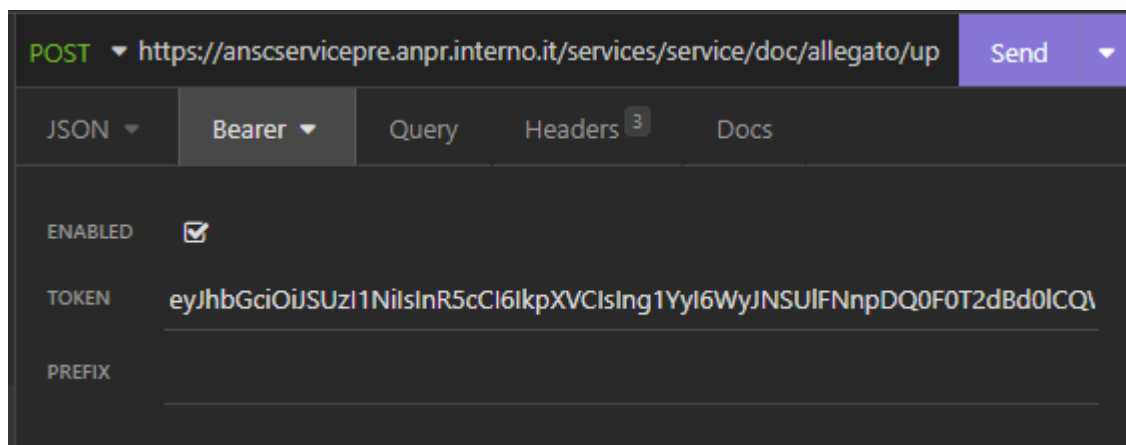


Figura 4 - Esempio di chiamata al servizio di Upload Allegato - Bearer Token JWT

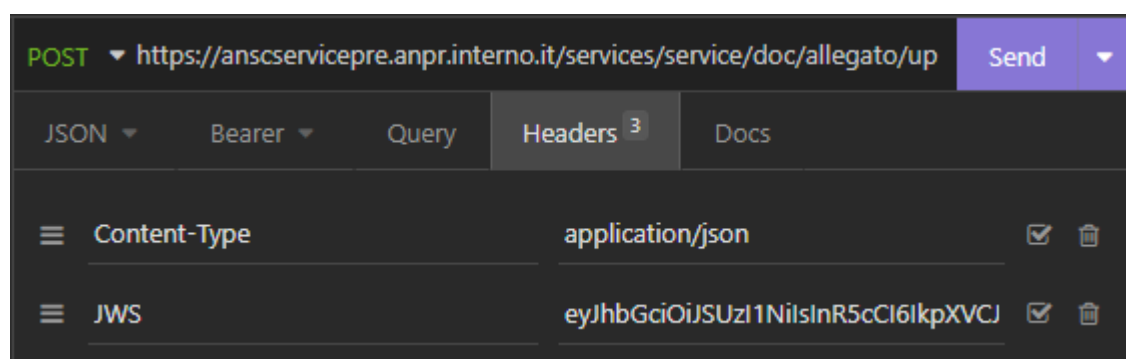
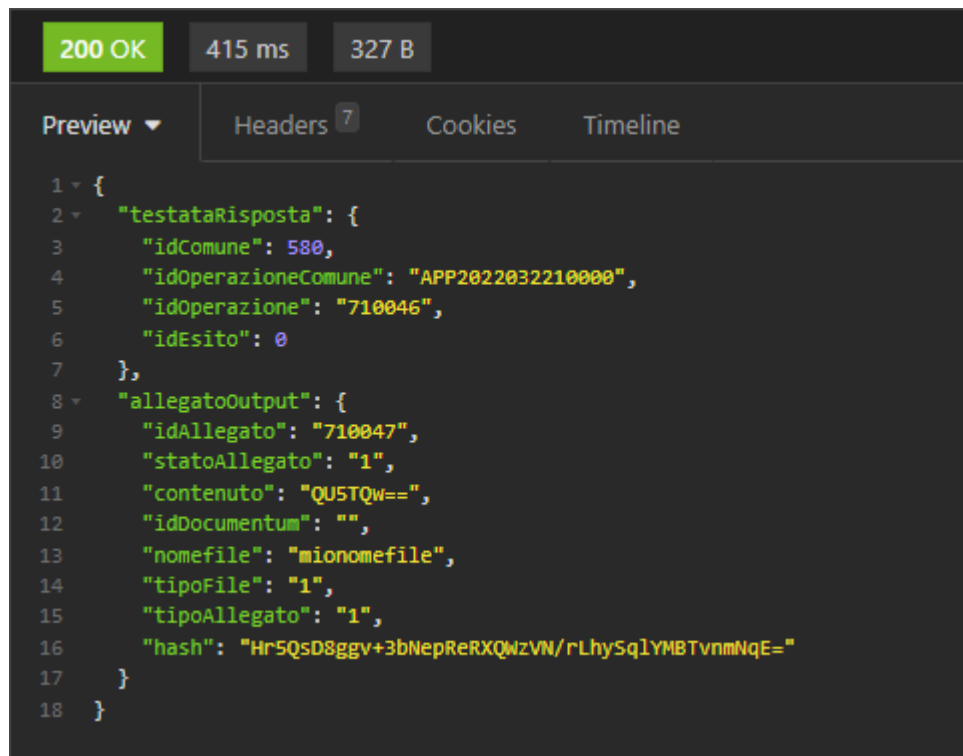


Figura 5 - Esempio di chiamata al servizio di Upload Allegato - impostazione header JWS



```
200 OK 415 ms 327 B
Preview Headers 7 Cookies Timeline
1 {
2   "testataRisposta": {
3     "idComune": 580,
4     "idOperazioneComune": "APP2022032210000",
5     "idOperazione": "710046",
6     "idEsito": 0
7   },
8   "allegatoOutput": {
9     "idAllegato": "710047",
10    "statoAllegato": "1",
11    "contenuto": "QUSTQW==",
12    "idDocumentum": "",
13    "nomefile": "mionomefile",
14    "tipoFile": "1",
15    "tipoAllegato": "1",
16    "hash": "Hr5QsD8ggv+3bNepReRXQWzVN/rLhySqlYMBTvmNqE="
17  }
18 }
```

Figura 6 - Esempio di chiamata al servizio di Upload Allegato - risposta del servizio con esito positivo

Per ragioni di test è possibile utilizzare tanti altri tipi di tool, come per esempio [Postman](#) o anche [cURL](#) di cui a seguire è possibile vedere un esempio.

```

curl -v --request POST \
  --url https://anscservicepre.anpr.interno.it/services/service/doc/allegato/upload/1 \
  --header 'Authorization: Bearer
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsIng1YyI6WyJNSU1FNnpDQ0F0T2dDd0lCQWdJSVp0cHRGUjZVd3l3JdO
RRWUplb1pJaHJjTtkFRRUxUUF3V2pFYk1Ca0dBWVVFQXQ3U1EwRWRVRz16ZEdGNmFXOXVhU0JUZG1sc0lRMHdDd11E
VlFRTERBUKJUBETTVI4d0hRWURWUWFLREJaTmFXNXBjM1J1S204Z1pHVnNiQ2RKYm5SbGNTNXZUNXN3Q1FZRFZRUU
dFd0pKVkRBZUZ3MH1NakEzTwpZeE1ETTBNVFphRncweU5URXdnRFV4TVRBNU1qUmFNSUdaTVJjd0ZRURWUWFEREE0
d01UWXdnVGN0VUVNdE1EQXdNVEVKTUJzR0ExVUVDd3dVVUc5emRHRjZHVz11YVNCa2FTQnNZWFp2Y204eE1UQXZCZ0
5WQkFzTUTFRnVZV2R5WVdabE1FNWh1bWx2Ym1Gc1pTQ1FiM0J2YkdGNmFXOXVhU0JTWlhoCfPHVnVkr1V4SHpBZEJn
TlZCQW9NRmsxcGJtbHpkR1Z5Yn1Ca1pXhNKMgX1ZEdeWwJtOHhDekFKQmdOVKBWVRBa2xvTU1JQklqQU5CZ2ZtaG
tpRz13MEJBUUVGQUFPOQFR0EFNSU1CQ2dLQ0FRRUFOY2hyK0d4NnVBSkxwFwUWkM4aT1DTFZaK0kwZGZmQTZKT0Vy
UFDQWV2WFXbldCMTkycEdLK31yS21FQ1d4YkNzaUpvSj1RbHJFCEfjODJyTERBb1pONHpvVldmMmHGSVHVWk80Yj
BRQnc2ZFNZSFBUNghaM20yL1BGnKNKQ0laQktXNGhvT2xmT5kNFd0ZEevY0Q1S0hhRHJjTnQ0dzRVS2kxamJ3SnJ5
Mm1TbU0zYmNpZz1xVy9oTlo2ODNNM1dIaXdEMjB1bmtBVWF0RHRTTERySUhMbXN1Uz1rWfoN131TWEZHYX13WgZUj
dxU091K3hyaXhWUfVSU1JzYmttNkRmRmphUDhEek5MV1VqY21DwnFqdk1JTj16dEd6dHNOQTEycjdURWkt1BDBuhy
VW9POGdobnpjZ3g4MERTEtk4Y0p0Z3hBNm1OSkF3UWpDaFFJREFRQUJvM1V3Y3pBTUJnTlZiUk1CQWY4RUfQQUFNQj
hHQTfVZE13UV1NQMfBRKFZTG44cWgXNW10MUNMdxM0dXhGS1BJVjV0K01CTudBMVVK1FRTU1Bb0dDQ3NHQVfVRk3J3
TUNNJbHQTFVZERnUVDQ1Jhenh6WjFaawo3U1E3R2dQdUZ2RWhTcTdzT2pBT0JnTlZiUthCQWY4RUJBtUNCNEF3RF
FZSkvtvklodmNOQVFFTEJRQRnZ01CQUt1MVZ6eERQVVOawtQK2cydzxhVHZwbXV5dDdJdHVLZmt4cWJla29ZbFJJ
L1NMV1hSWGpyT1N6am9nQnE2M0ZnTj10UjJvBGFVdGNyMFhIR2hGM1VGym5FUKU1WDZ0UkxjbYyag5KNVYrW1A5SE
dPNmhaRm1HK2hrZ2FFRm9xY3VumUpCOUtycFRPVG5x0U9yc1BYSGRNM1ZDWFfoU0tEeT1pL2LRNnpjVVBqd0wrZDVR
OVE3L2tDUSs3ZnJaZ3NyKzWU3hDZUhdU4zaDh5ajU2S2EraVNDcZ1FSEFZaUZMbXQvZXdyVFBaWmxjOHVUCG9US1
hOY3LMVXV5MkRTR0o30E1CSjRwOXDPSFdIMGRoa1lEU2tIempIeTljc3BVUDNpWjZxNkhGRE41NzR5S5jdwU1p1TnJF
dVFKc1cWdFRVUnV0WPhYVE0MkVFR1YxbXVrbzJpSDdaTX1mTEJ6VXZoQXZrR3p2VXVkuJNuNGtJc3lqOUp3Tzdpdk
EvTjNkeFp1MzMwTtYzR0FyYkhLSXFfNzXhCRWtsN1dsdzBiUXdENFo5Q1RacW9BRFERQitPZE5TZVhjSFdva3pkOC9T
TzI0VUp4eJb0bklXU1YzcFo4NzZ5bGczd3UwSEdm1V6ZVMOC9tQ2xTYUg4RHo1UmIrNUpOZmZPYTV4eVZGU0I1ZE
R5UDVT0ZHa1dzUHNQNz1hSktyS2EyanpxU1Nne1lpdTbwT3U2MFJBNjk4N2JQU3hZCwXsdVWcGNMODN1Uldab1RW
OVF5TT1TRmlubTJJUXdWNjdXYnYyS25yQnRYb1hpNE5YY0ZYMzZsb0xEYwYmKdpWVRRV9SbGxNS1l2Wk9iVnFRMG
ZvVTN0bTfoN3dtcnVIAxYzenVBdTLGe1prUjVWNGVH119.eyJzdWIiOiJNU1JOVE43N0gxNUMzNTFYIiwiaWF0Ij0j
xNjcYmZk3MDE4L3JleHAiOiJlE2NzI2MTMwMTgsIm5iZiI6MTY3MjMjYmZxOAcwianRpiJoiJWVWcmJfZ2dnBiRm9mTHdFa
mxkVnZWUSisInNlZGUiOiIwMTYwMTciLCJwb3N0YXppb251IjoimDE2MDE3LVBDLTAWMDEiLCJvdHAiOiIwMTYwMTY
ifQ. bF-F7X6dAAtrypIBrpd0LfZrIHk-KgmF6OUefG3MxicJuZlomkNpxZvm6DCBvk7K6rQNw2RuqLe-
dL_h9awjjyYIOxPzQ130jpARQemXNw7u_JpGWLvtglJ9sRjN8yrL9tPitwnze0aZxYGSwUaMx10Zp0UCOPeozfB3
SZqdh9ZUQjINZjYiBH_Do1Lnksq83E3wD9ONLtnL3z1AVIJK8ZtYVjYCiBZhUDeDnrMMWXMN8iRbjs6P7b0oAeMck
6EBxFeQNoJh6w_9QqXDku05tgbfvXrp560XbCLnnEYLLZYiCbREAbA-YyKnvW85YHoMJ5YHb0xyAtVCjpwmfIA' \
  --header 'Content-Type: application/json' \
  --header 'JWS:
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9..Cn3Xu8Lbn9TKoZHC4AadHXP0EzYE5oX2u81xo_3vdzP9vTkxU78f
6uMrP6cswqaaMBA3DQWIAhqiLkP29_E7uWKS0Wn8eV7SKemtuvwt_r5y9K3CRAPg1N0uc1Q9aEPx4H0qB0unh2_4f
V0qQ0vzSgSvvuTZ7S00y7at1r2u4bFn6vs103vdArXbrUTn2PUNsQ7V2kH1e57BXUuIAVYf1AY_KT64diqelC-
1VS11NKY9_70CH72ZAegU1VJ_IDGLg_DyShq5y8xG4SEegVQtmkzE9b_T9fvNNe_fwvTjy0VixN38q-
Ynox6UBmd6FSomIixLy89-d3vpJ2_zg5AQ' \
  --cookie 14d163860c1376f46d3a6ddce34f2cab=d606e9e345d81080eead2391076aaa39 \
  --data '{
    "testataRichiesta": {
      "idComune": 580,
      "idOperazioneComune": "APP2022032210000",
      "dataOraRichiesta": "2022-08-17",
      "nomeApplicativo": "Applicativo Demografico",
      "versioneApplicativo": "1.0.0",
      "fornitoreApplicativo": "Nome Fornitore"
    },
    "allegatoInput": {
      "contenuto": "QU5TQw==",
      "nomefile": "mionomefile",
      "tipoFile": "1",
      "tipoAllegato": "1"
    }
  }'

```

Code 9 - Chiamata al servizio di Upload Allegato in modalità JWS Detached utilizzando il tool cURL

Qualora dovessero esserci problemi con la validazione del JWS, processo eseguito dal componente di sicurezza dell'intera infrastruttura, in risposta si otterrà un errore con un codice HTTP 401 e il seguente contenuto.

```
{
  "operation": "/services/service/doc/allegato/upload/1",
  "error": "401",
  "error_description": "Errore nella validazione del JWS"
}
```

Code 10 - Esempio di risposta nel caso in cui dovesse fallire la validazione del JWS

5. LIBRERIE PER LA FIRMA/VERIFICA DI TOKEN

Il token JWT è uno standard open di settore definito dall'[RFC 7519](#) e utilizzato come metodo per rappresentare in modo sicuro le richieste tra due parti. Grazie alla grande diffusione di questo standard, esistono una miriade di librerie per quasi la totalità delle piattaforme di sviluppo.

La pagina [Libraries for Token Signing/Verification](#) del sito di [jwt.io](#), riporta le librerie disponibili per linguaggio e/o piattaforma di sviluppo. L'uso delle librerie è consigliato per agevolare lo sviluppo di codice che riguarda la gestione dei token.

La figura a seguire mostra alcune delle librerie disponibili per la piattaforma Java, dove per ognuna sono indicate le caratteristiche dello standard supportate.

The screenshot displays the 'Libraries for Token Signing/Verification' page on the jwt.io website. The page has a dark header with a colorful logo and the title 'Libraries for Token Signing/Verification'. Below the header, there's a filter dropdown set to 'Java'. Three library cards are shown, each with a Java logo and a list of supported JWT claims. The first card is for 'Auth0' (5042 stars), the second for 'Brian Campbell' (jose4j), and the third for 'connect2id' (nimbus-jose-jwt). Each card lists claims like Sign, Verify, iss check, sub check, aud check, exp check, nbf check, iat check, jti check, typ check, HS256, HS384, HS512, PS256, PS384, PS512, RS256, RS384, RS512, ES256, ES256K, ES384, ES512, and EdDSA, with green checkmarks indicating support.

Library	Sign	Verify	iss check	sub check	aud check	exp check	nbf check	iat check	jti check	typ check	HS256	HS384	HS512	PS256	PS384	PS512	RS256	RS384	RS512	ES256	ES256K	ES384	ES512	EdDSA
Auth0 (5042 stars)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Brian Campbell (jose4j)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
connect2id (nimbus-jose-jwt)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Figura 7 - Lista di alcune delle librerie disponibili per Java

6. ESEMPIO DI CLIENT JAVA

Allo scopo di facilitare l'integrazione con i servizi cooperativi di ANSC, sono stati implementati due tool (in linguaggio Java), che utilizzano la libreria [jose4j](#) sviluppata da [Bitbucket](#) (tra quelle consigliate da jwt.io). Questi due tool consentono in breve di:

1. generare e validare un JSON Web Signature (JWS). Questo tool è rappresentato dalla main class *JWSGeneratorAndValidate*;
2. generare JWT, JWS, HTTP header del JWS detached e successiva chiamata al servizio cooperativo di ANSC. Questo tool è rappresentato dalla main class *TestCallRestServiceWithJWS*.

Il JWT generato dal tool è conforme alle specifiche introdotte dal capitolo 2 *Costruzione del token JWT*.

L'intero progetto (codice sorgente) del client Java è disponibile al seguente indirizzo [jws-signature-verify](#). Il file README.md contiene tutte le informazioni utili per l'uso dei tool menzionati in precedenza.



Il client Java fornito, ha esclusivamente lo scopo di mostrare come creare il JWT e JWS secondo le specifiche indicate in questo documento e di conseguenza facilitare/velocizzare l'integrazione tra i gestionali in dotazione ai comuni e i servizi di ANSC. Non è assolutamente previsto nessun tipo di supporto sul tool fornito.

7. COME RICHIEDERE SUPPORTO

Nel caso in cui dovessero emergere dei problemi d'integrazione con i servizi cooperativi di ANSC e nello specifico questioni inerenti JWT o JWS Detached, è possibile scrivere alla casella supporto-ansc@sogei.it.

Prima di scrivere al supporto di ANSC, è consigliato consultare la sezione 8 FAQ che potrebbe contenere già la soluzione al problema riscontrato.

Il contenuto della email deve descrivere in modo chiaro il problema riscontrato nell'integrazione con i servizi cooperativi e in particolare riportare le seguenti informazioni:

1. ambiente: validazione, pre-produzione o produzione;
2. end-point completo del servizio cooperativo;
3. data e ora dell'avvenuta richiesta;
4. indirizzo IP (pubblico) da cui la richiesta "esce";
5. header http di richiesta. Gli header indubbiamente più rilevanti al fine dell'analisi, sono: Authorization e JWS. Il primo contiene il token JWT, il secondo il JWS Detached;
6. cookie inviati;
7. contenuto del body http di richiesta;
8. JWS in forma Compact Serialization.

Se fattibile, l'ideale sarebbe fornire in allegato alla mail l'intero dump della richiesta http e eventuali altri allegati che siano utili a descrivere ulteriormente il problema riscontrato.

8. FAQ

1 Q È possibile verificare e/o generare il token JWT/JWS con qualche tool disponibile online?

A Esiste il debugger disponibile all'indirizzo jwt.io. Tramite questo tool è possibile verificare token già esistenti o generare di nuovi

2 Q È possibile ottenere in modo semplice il PEM del certificato x509 di postazione?

A Esistono diversi tool per raggiungere l'obiettivo; quelli più comuni sono [OpenSSL](#) e [KeyStore Explorer](#).

Per ottenere informazioni tra cui il PEM del certificato x509, è possibile utilizzare il comando OpenSSL indicato a seguire.

```
openssl pkcs12 -info -in <certificato-di-postazione>.p12
```

L'esecuzione del comando richiederà l'inserimento della password fornita al momento della consegna del certificato di postazione. Una volta inserita la password, OpenSSL restituirà una serie d'informazioni tra cui il PEM del certificato.

È possibile ottenere lo stesso risultato aprendo il file PKCS#12 con il tool KeyStore Explorer. A seguire un esempio.

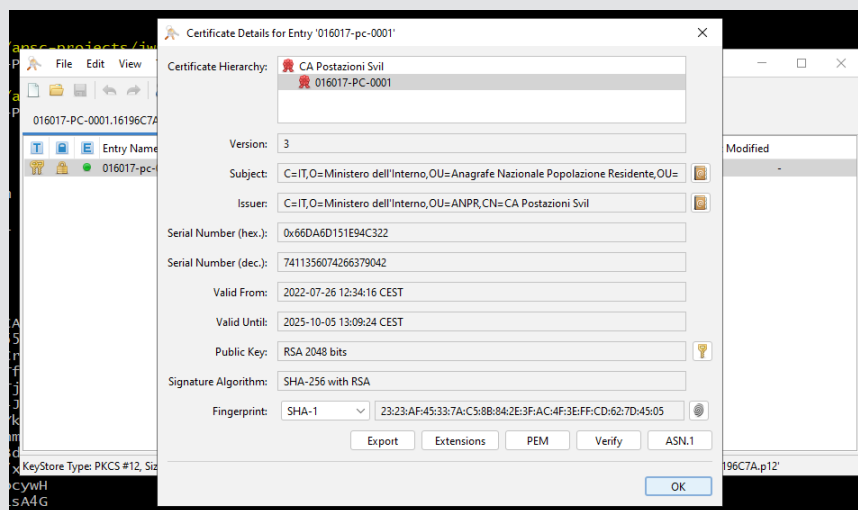


Figura 8 - Apertura del file PKCS#12 di postazione con il tool KeyStore Explorer

3 Q Nonostante il JWS sia valido il servizio continua a rispondere con il codice di errore HTTP 401 e il messaggio: *Errore nella validazione del JWS*. Cos'è possibile controllare della richiesta?

A Se il token JWT e JWS è senza ombra di dubbio valido (verificati con il tool online o programmaticamente utilizzando una delle librerie standard), i controlli da eseguire sono i seguenti:

1. verificare che il contenuto dell'header HTTP su cui è inserito il JWS Detached sia nel formato corretto vedi Code 8. Il nome dell'header è in-case-sensitive (può per esempio essere passato come jws);
2. verificare il che il body della richiesta HTTP sia esattamente il contenuto che ha subito il processo di firma e questo non abbia subito alterazioni prima dell'invio all'endpoint del servizio cooperativo;
3. il body della richiesta HTTP deve essere il JSON previsto dallo specifico servizio cooperativo e non il JWS nella forma Compact Serialization.

4 Q Nonostante il JWS sia valido il servizio continua a rispondere con il codice di errore HTTP 500 e il body riporta il path relativo dell'operation richiesta. Cos'è possibile controllare della richiesta?

- A Solitamente questo genere di errore potrebbe capitare per diversi motivi, di cui i più comuni sono:
1. il nome dell'header HTTP che contiene il JWS Detached non è corretto;
 2. manca l'header HTTP che contiene il JWS Detached;
 3. l'header del JWS Detached non è valido, per esempio a causa di un errato encoding in Base64URL;
 4. l'header del JWS Detached specifica un algoritmo di signature non supportato.

Tabella 4 - Tabella delle FAQ

9. **DISCLAIMER**

In questa fase sono forniti messaggi dettagliati sugli errori di invocazione dei servizi al fine di favorire tutte le parti coinvolte nell'integrazione con il sistema ANSC. In futuro i messaggi potrebbero essere rivisti allo scopo di evitare l'esposizione d'informazioni utili a favorire eventuali attacker (vedi [OWASP A3:2017-Sensitive Data Exposure](#)). I codici di errore rimarranno stabili.