Signature

Open API for FSP Interoperability Specification

Signature

Open API for FSP Interoperability Specification

Table of Contents

Table of Tables

1 Preface

This section contains information about how to use this document.

1.1 Conventions Used in This Document

This document uses the notational conventions for BASE64URL(OCTETS), UTF8(STRING), ASCII(STRING), and **||** defined in RFC 7515[1].

The following conventions are used in this document to identify the specified types of information

Type of Information	Convention	Example
Elements of the API, such as resources	Boldface	**/authorization**
Variables	Italics within angle brackets	*<ID>*
Glossary terms	Italics on first occurrence; defined in *Glossary*	The purpose of the API is to enable interoperable financial transactions between a *Payer* (a payer of electronic funds in a payment transaction) located in one *FSP* (an entity that provides a digital financial service to an end user) and a *Payee* (a recipient of electronic funds in a payment transaction) located in another FSP.
Library documents	Italics	User information should, in general, not be used by API deployments; the security measures detailed in *Signature* and *Encryption* should be used instead.

[1] https://tools.ietf.org/html/rfc7515#section-1.1 – JSON Web Signature (JWS) - Notational Conventions

1.2 Document Version Information

Version	Date	Change Description
1.0	2018-03-13	Initial version
1.1	2020-05-09	This version contains the below changes: - Sections 3.1, 3.2 and 3.3 have been updated based on ”Solution Proposal 12 - Clarify usage of FSPIOP-Destination” - ExstensionList elements in Section 4 have been updated based on the issue ”Interpretation of the Data Model for the ExtensionList element”, to fix the data model of the extensionList Object.

2 Introduction

This document details the security methods to be implemented for Open API for FSP Interoperability (hereafter cited as the API) to ensure *integrity* and *non-repudiation* between the API client and the API server.

In information security, *data integrity* means maintaining and assuring the accuracy and completeness of data over its entire lifecycle. For the API, data integrity means that an API message cannot be modified in an unauthorized or undetected manner by parties involved in the API communication.

In legal terms, *non-repudiation* means that a person intends to fulfill their obligations to a contract. It also means that one party in a transaction cannot deny having received the transaction, nor can the other party deny having sent the transaction. For the API, non-repudiation means that an API client cannot deny having sent an API message to a counterparty. JSON Web Signature (JWS), as defined in RFC 7515[2], must be applied to the API to provide message integrity and non-repudiation for either component fields of an API payload or the full API payload. Whenever an API client sends an API message to a counterparty, the API client should sign the message using its private key. After the counterparty receives the API message, the counterparty must validate the signature with the API client's public key. Only the HTTP request message of an API message need to be signed, any HTTP response message of the APIs SHALL NOT be signed.

Note: The corresponding public key should either be shared in advance with the counterparty or retrieved by the counterparty (for example, the local scheme Certificate Authority).

Because intermediary fees are not supported in the current version of the API, intermediaries involved in API message-transit may not modify the API message payload. Thus, the signature at full payload level is used to protect the integrity of the full payload of an API message from end-to-end. Regardless of how many intermediaries there are in transit, the original payload cannot be modified by the intermediaries. The final recipient of the API message must validate the signature generated by the original API client based on the message payload received.

Note: Whether the signature needs to be validated by the intermediaries in transit is determined by the internal implementation of each intermediary or the local schema.

Note: In a future version of the API, intermediary fees may be supported; at that time, signature-at-field-level may also be supported. However, both features are out-of-scope for the current version of the API.

[2] https://tools.ietf.org/html/rfc7515 - JSON Web Signature (JWS)

2.1 Open API for FSP Interoperability Specification

The Open API for FSP Interoperability Specification includes the following documents.

2.1.1 General Documents

- *Glossary*

2.1.2 Logical Documents

- *Logical Data Model*
- *Generic Transaction Patterns*
- *Use Cases*

2.1.3 Asynchronous REST Binding Documents

- *API Definition*
- *JSON Binding Rules*
- *Scheme Rules*

2.1.4 Data Integrity, Confidentiality, and Non-Repudiation

- *PKI Best Practices*
- *Signature*
- *Encryption*

3 API Signature Definition

This section introduces the technology used by the API signature, including the data exchange format for the signature of an API message and the mechanism used to generate and verify a signature.

3.1 Signature Data Model

The API uses a customized HTTP header parameter **FSPIOP-Signature** to represent the signature that is produced by the initiating API client for the API message. The data model for this parameter is described in Table 1.

The entire message-body is used for the signature, meaning that nothing in the message-body is allowed to be changed by an intermediate system. Any modification of the message-body will result in a signature verification error.

It is the message-initiator that decides which HTTP headers should be protected in addition to the mandatory protected headers detailed in Table 1. Intermediate systems are allowed to add new HTTP headers, but any added HTTP header will not be protected. Similarly, unprotected HTTP headers can also be changed by intermediate systems. A modified protected HTTP header will result in a signature verification error.

Note: Currently the API does not support intermediaries in an API message; only the message-initiator can sign a message. If this is required in the future, there will be new customized HTTP header parameter, but this is out-of-scope for the current version of the API.

Name	Cardinality	Type	Description
protectedHeader	1	String(1..32768)	This element indicates the HTTP header parameters that are protected by the signature. Its value must be BASE64URL(UTF8(JWS Protected Header)). According to JWS specification, the **alg** header parameter must be present to identify the cryptographic algorithm used to secure the JWS. A customized parameter **FSPIOP-URI** that represents the URI path and query parameters of HTTP request message of the APIs must be present. A customized parameter **FSPIOP-HTTP-Method** that holds the HTTP method used in the HTTP message must be present. A customized parameter **FSPIOP-Source** that represents the system which sent the API request must be present. The customized HTTP header parameter **FSPIOP-Destination** is mandatory in **protectedHeader** if the destination FSP is known by the message-initiator. Otherwise this header must not be protected as it can be changed by intermediate systems. See API Definition for more information regarding which services that the header **FSPIOP-Destination** is optional for.
signature	1	String(1..512)	This element indicates the signature. Its value is part of JWS serialization; that is, BASE64URL(JWS Signature).

Table 1 – Data model of HTTP header field FSPIOP-Signature

3.2 Generating a Signature

To create the signature for an API message, the following steps are performed. The order of the steps is not significant in cases where there are no dependencies between the inputs and outputs of the steps.

1. Create the content to be used as the JWS Payload. Because the signature is currently at full payload level, the full HTTP body of the API message is the JWS Payload.
2. Compute the encoded payload value BASE64URL(JWS Payload).
3. Create the JSON object or objects containing the desired JWS Protected Header.
   A. The **alg** JWS Protected Header parameter must be present. In the API, the available algorithms for the signature are **RS256**, **RS384**, **RS512**. A key of size 2048 bits or larger must be used with these algorithms.
   B. Other parameters registered in the IANA *JSON Web Signature and Encryption Header Parameters*[3] are optional.
   C. The customized parameter **FSPIOP-URI** must be included in JWS Protected Header to protect the URI path and query parameters of the APIs.
   D. The customized parameter **FSPIOP-HTTP-Method** must be included in JWS Protected Header to protect the HTTP request operation method.
   E. The parameter **FSPIOP-Source** must be present, and its value comes from the corresponding HTTP header parameter **FSPIOP-Source**.
   F. The parameter **FSPIOP-Destination** must be present if the destination FSP is known by the message-initiator, and its value must then be the same as the HTTP header parameter **FSPIOP-Destination**.
   G. Other HTTP Header parameters of the APIs are recommended to be included in JWS Protected Header, but they are optional in this JWS Protected Header. It is not allowed to have multiple parameters in the JWS Protected Header with the same name in different cases. As an example, it is not allowed to have one parameter named **Date** and another one named **dATE**.
4. Compute the encoded header value BASE64URL(UTF8(JWS Protected Header)).
5. Compute the JWS Signature according to the JWS specification using the output of Step 2 and Step 4.
6. Compute the encoded signature value BASE64URL(JWS Signature).
7. Compute the value for the HTTP header parameter **FSPIOP-Signature** as described in Section 3.1. The value for this **FSPIOP-Signature** is a JSON Object Serialization string.

Note: If JSON Web Encryption (JWE) is used to encrypt some fields of the payload (for more information, see *Encryption*), then the API client should first encrypt the desired fields, then replace the plain text of those fields with the encoded cipher text in the payload, and then finally sign the payload.

[3] https://www.iana.org/assignments/jose/jose.xhtml#web-signature-encryption-header-parameters - JSON Web Signature and Encryption Header Parameters

3.3 Validating Signature

When validating the signature of an API request, the following steps are performed. Please note that as stated in *API Definition*, the HTTP header field names are case-insensitive. The order of the steps is not significant in cases where there are no dependencies between the inputs and outputs of the steps. If any of the listed steps fails, then the signature cannot be validated.

1. Parse the HTTP header parameter **FSPIOP-Signature** to get the components **protectedHeader** and **signature**.
2. Use BASE64URL to decode the encoded representation of the JWS Protected Header. Verify that the resulting octet sequence is a UTF-8-encoded representation of a completely valid JSON object conforming to JSON Data Interchange Format, defined in RFC 7159[4].
3. Verify the parameters in the JWS Protected Header.
   a) The parameter **alg** must be present and its value must be one of **RS256**, **RS384**, **RS512**.
   b) Other parameters registered in the IANA *JSON Web Signature and Encryption Header Parameters* are optional.
   c) The parameter **FSPIOP-URI** must be present and its value must be the same as the input URL value of the request.
   d) The parameter **FSPIOP-HTTP-Method** must be present and its value must be same as the operation method of the request.
   e) The parameter **FSPIOP-Source** must be present, and its value must be the same as the corresponding HTTP header parameter **FSPIOP-Source**.
   f) If the parameter **FSPIOP-Destination** is present in the JWS Protected Header, then its value must be same as the corresponding HTTP header parameter **FSPIOP-Destination**.
   g) If there are other HTTP header parameters present in JWS Protected Header, then their values must be validated with the corresponding HTTP header values.
4. Compute the encoded payload value BASE64URL(JWS Payload). Because the current signature is at full payload level, the full HTTP body of the API message is the JWS Payload.
5. Validate the JWS Signature against the JWS Signing Input ASCII(BASE64URL(UTF8(JWS Protected Header)) || '.' || BASE64URL(JWS Payload)) in the manner defined for the algorithm being used, which must be accurately represented by the value of the **alg** (algorithm) Header Parameter. As HTTP header field names are case-insensitive and the signature validation of the **protectedHeader** is case-sensitive, the **protectedHeader** must be used exactly as it was received from the other FSP when validating the JWS signature.
6. Record whether the validation succeeded.

[4] https://tools.ietf.org/html/rfc7159 - The JavaScript Object Notation (JSON) Data Interchange Format

4 API Signature Examples

This section uses a typical quote process to explain how the API signature is implemented using JWS. The FSPs in the API can verify that their internal implementation for API signature is correct using the following case.

The case in this section uses RS256 as the signature algorithm. The RSA key used for the signature example is represented in JSON Web Key (JWK), defined in RFC 7517[5], format below (with line breaks and indentation within values for display purposes only):


```
{
    "kty": "RSA",
    "n": "ofgWCuLjybRlzo0tZWJjNiuSfb4p4fAkd_wWJcyQoTbji9k0l8W26mPddx
          HmfHQp-Vaw-4qPCJrcS2mJPMEzP1Pt0Bm4d4QlL-yRT-SFd2lZS-pCgNMs
          D1W_YpRPEwOWvG6b32690r2jZ47soMZo9wGzjb_7OMg0LOL-bSf63kpaSH
          SXndS5z5rexMdbBYUsLA9e-KXBdQOS-UTo7WTBEMa2R2CapHg665xsmtdV
          MTBQY4uDZlxvb3qCo5ZwKh9kG4LT6_I5IhlJH7aGhyxXFvUK-DWNmoudF8
          NAco9_h9iaGNj8q2ethFkMLs91kzk2PAcDTW9gb54h4FRWyuXpoQ",
    "e": "AQAB",
    "d": "Eq5xpGnNCivDflJsRQBXHx1hdR1k6Ulwe2JZD50LpXyWPEAeP88vLNO97I
          jlA7_GQ5sLKMgvfTeXZx9SE-7YwVol2NXOoAJe46sui395IW_GO-pWJ1O0
          BkTGoVEn2bKVRUCgu-GjBVaYLU6f3l9kJfFNS3E0QbVdxzubSu3Mkqzjkn
          439X0M_V51gfpRLI9JYanrC4D4qAdGcopV_0ZHHzQlBjudU2QvXt4ehNYT
          CBr6XCLQUShb1juUO1ZdiYoFaFQT5Tw8bGUl_x_jTj3ccPDVZFD9pIuhLh
          BOneufuBiB4cS98l2SR_RQyGWSeWjnczT0QU91p1DhOVRuOopznQ",
    "p": "4BzEEOtIpmVdVEZNCqS7baC4crd0pqnRH_5IB3jw3bcxGn6QLvnEtfdUdi
          YrqBdss1l58BQ3KhooKeQTa9AB0Hw_Py5PJdTJNPY8cQn7ouZ2KKDcmnPG
          BY5t7yLc1QlQ5xHdwW1VhvKn-nXqhJTBgIPgtldC-KDV5z-y2XDwGUc",
    "q": "uQPEfgmVtjL0Uyyx88GZFF1fOunH3-7cepKmtH4pxhtCoHqpWmT8YAmZxa
          ewHgHAjLYsp1ZSe7zFYHj7C6ul7TjeLQeZD_YwD66t62wDmpe_HlB-TnBA
          -njbglfIsRLtXlnDzQkv5dTltRJ11BKBBypeeF6689rjcJIDEz9RWdc",
    "dp": "BwKfV3Akq5_MFZDFZCnW-wzl-CCo83WoZvnLQwCTeDv8uzluRSnm71I3Q
          CLdhrqE2e9YkxvuxdBfpT_PI7Yz-FOKnu1R6HsJeDCjn12Sk3vmAktV2zb
          34MCdy7cpdTh_YVr7tss2u6vneTwrA86rZtu5Mbr1C1XsmvkxHQAdYo0",
    "dq": "h_96-mK1R_7glhsum81dZxjTnYynPbZpHziZjeeHcXYsXaaMwkOlODsWa
          7I9xXDoRwbKgB719rrmI2oKr6N3Do9U0ajaHF-NKJnwgjMd2w9cjz3_-ky
          NlxAr2v4IKhGNpmM5iIgOS1VZnOZ68m6_pbLBSp3nssTdlqvd0tIiTHU",
    "qi": "IYd7DHOhrWvxkwPQsRM2tOgrjbcrfvtQJipd-DlcxyVuuM9sQLdgjVk2o
          y26F0EmpScGLq2MowX7fhd_QJQ3ydy5cY7YIBi87w93IKLEdfnbJtoOPLU
          W0ITrJReOgo1cq9SbsxYawBgfp_gh6A5603k2-ZQwVK0JKSHuLFkuQ3U"
}
```


[5] https://tools.ietf.org/html/rfc7517 - JSON Web Key (JWK)

4.1 Generating a Signature

The following message text is an example of **POST /quotes** without a signature sent by Payer FSP to a counterparty (line breaks and indentation within values for display purposes only).

```
POST /quotes HTTP/1.1
Accept:application/vnd.interoperability.quotes+json;version=1.0
FSPIOP-Source:1234
FSPIOP-Destination:5678
Content-Length:975
Date:Tue, 23 May 2017 21:12:31 GMT
Content-Type:application/vnd.interoperability.quotes+json;version=1.0

{
    "amount": { "amount": "150", "currency": "USD" },
    "transactionType": {
        "scenario": "TRANSFER", "initiator": "PAYER",
        "subScenario": "P2P Transfer across MM systems",
        "initiatorType": "CONSUMER"
    },
    "transactionId": "36629a51-393a-4e3c-b347-c2cb57e1e1fc",
    "quoteId": "59e331fa-345f-4554-aac8-fcd8833f7d50",
    "expiration": "2017-05-24T08:40:00.000-04:00",
    "payer": {
        "personalInfo": {
            "dateOfBirth": "1986-02-14",
            "complexName": { "middleName": "Ben",
                "LastName": "Lee", "firstName": "Bill" } },
        "name": "Bill Lee",
        "partyIdInfo": { "fspId": "1234", "partyIdType": "MSISDN",
            "partySubIdOrType": "RegisteredCustomer",
            "partyIdentifier": "16135551212" }
    },
    "payee": {
        "partyIdInfo": { "fspId": "5678",
            "partyIdType": "MSISDN",
            "partyIdentifier": "15295558888" }
    },
    "fees": { "amount": "1.5", "currency": "USD" },
    "extensionList": {
       "extension": [
           { "value": "value1", "key": "key1" },
           { "value": "value2", "key": "key2" },
           { "value": "value3", "key": "key3" } ]
       },
    "note": "this is a sample for POST /quotes",
    "geoCode": {
        "longitude": "125.520001", "latitude": "57.323889" },
    "amountType": "RECEIVE"
}
```

4.1.1 Computing Signature Input

According to JWS specification, the signature input is BASE64URL(UTF8(JWS Protected Header)) || '.' || BASE64URL(JWS Payload).

Assuming the HTTP header parameters **Date** and **FSPIOP-Destination** are protected by the signature, and the algorithm RS256 is used to sign the message, the JWS Protected Header in this case is as follows (line breaks and indentation within values for display purposes only):

```
{
  "alg":"RS256",
  "FSPIOP-Destination":"5678",
  "FSPIOP-URI":"/quotes",
  "FSPIOP-HTTP-Method":"POST",
  "Date":"Tue, 23 May 2017 21:12:31 GMT",
  "FSPIOP-Source":"1234"
}
```

Encoding this JWS Protected Header as BASE64URL(UTF8(JWS Protected Header)) gives this value:

```
eyJhbGciOiJSUzI1NiIsIkZTUElPUC1EZXN0aW5hdGlvbiI6IjU2NzgiLCJGU1BJT1AtVVJJIjo
iL3F1b3RlcyIsIkZTUElPUC1IVFRQLU1ldGhvZCI6IlBPU1QiLCJEYXRlIjoiVHVlLCAyMyBNYX
kgMjAxNyAyMToxMjozMSBHTVQiLCJGU1BJT1AtU291cmNlIjoiMTIzNCJ9
```

In this case, JWS Payload is the HTTP Body described in Section 4.1. Encoding this JWS Payload as BASE64URL(JWS Payload) gives this value:

```
eyJwYXllZSI6eyJwYXJ0eUlkSW5mbyI6eyJwYXJ0eUlkVHlwZSI6Ik1TSVNETiIsInBhcnR5SWR
lbnRpZmllciI6IjE1Mjk1NTU4ODg4IiwiZnNwSWQiOiI1Njc4In19LCJhbW91bnRUeXBlIjoiUk
VDRUlWRSIsInRyYW5zYWN0aW9uVHlwZSI6eyJzY2VuYXJpbyI6IlRSQU5TRkVSIiwiaW5pdGlhd
G9yIjoiUEFZRVIiLCJzdWJTY2VuYXJpbyI6IlAyUCBUcmFuc2ZlciBhY3Jvc3MgTU0gc3lzdGVt
cyIsImluaXRpYXRvclR5cGUiOiJDT05TVU1FUiJ9LCJub3RlIjoidGhpcyBpcyBhIHNhbXBsZSB
mb3IgUE9TVCAvcXVvdGVzIiwiYW1vdW50Ijp7ImFtb3VudCI6IjE1MCIsImN1cnJlbmN5IjoiVV
NEIn0sImZlZXMiOnsiYW1vdW50IjoiMS41IiwiY3VycmVuY3kiOiJVU0QifSwiZXh0ZW5zaW9uT
GlzdCI6W3sidmFsdWUiOiJ2YWx1ZTEiLCJrZXkiOiJrZXkxIn0seyJ2YWx1ZSI6InZhbHVlMiIs
ImtleSI6ImtleTIifSx7InZhbHVlIjoidmFsdWUzIiwia2V5Ijoia2V5MyJ9XSwiZ2VvQ29kZSI
6eyJsYXRpdHVkZSI6IjU3LjMyMzg4OSIsImxvbmdpdHVkZSI6IjEyNS41MjAwMDEifSwiZXhwaX
JhdGlvbiI6IjIwMTctMDUtMjRUMDg6NDA6MDAuMDAwLTA0OjAwIiwicGF5ZXIiOnsicGVyc29uY
WxJbmZvIjp7ImNvbXBsZXhOYW1lIjp7ImZpcnN0TmFtZSI6IkJpbGwiLCJtaWRkbGVOYW1lIjoi
QmVuIiwiTGFzdE5hbWUiOiJMZWUifSwiZGF0ZU9mQmlydGgiOiIxOTg2LTAyLTE0In0sInBhcnR
5SWRJbmZvIjp7InBhcnR5SWRUeXBlIjoiTVNJU0ROIiwicGFydHlTdWJJZE9yVHlwZSI6IlJlZ2
lzdGVyZWRDdXN0b21lciIsInBhcnR5SWRlbnRpZmllciI6IjE2MTM1NTUxMjEyIiwiZnNwSWQiO
iIxMjM0In0sIm5hbWUiOiJCaWxsIExlZSJ9LCJxdW90ZUlkIjoiNTllMzMxZmEtMzQ1Zi00NTU0
LWFhYzgtZmNkODgzM2Y3ZDUwIiwidHJhbnNhY3Rpb25JZCI6IjM2NjI5YTUxLTM5M2EtNGUzYy1
iMzQ3LWMyY2I1N2UxZTFmYyJ9
```

4.1.2 Producing Signature

Use the given RSA Private Key, the JWS Protected Header and the JWS Payload to generate the signature, then encoding the signature as BASE64URL(JWS Signature) produces this value:

```
dz2ntyS0_rDyA0pLeWluG--tBcYYrlvG99ffkXcEB-dz2ntyS0_rDyA0pLeWluG--tBcYYrlvG9
9ffkXcEB-uve5Qzvzyn0ZUi82J7h17RsdfHPuTnbEGvCeU9Y4Bg0nIZHGL4icswaaO09T5hPPY-
KBTzVQeHkokLmL4dXpHdr1ggSEpu3WEU3nfgOFGGAdOq355i1iGuDbhqm_lSfVHaqdVCEhkJ2Y_
r2glO2QpdZrcbvsBV39derj_PlfISBBGjdh0dIPxnFIVcZuPHiq9Ha2MslrBHfqwFfNeU_xhErB
d2PywkDQJbKOlfqdkmFC9bS8Ofx0O6Mg7qdFGw-QkseJTfp0HMbH1d9e6H0cocY8xfuDNGaZpOJ
hxiYtiPLg
```

4.1.3 Re-produce API Request with Signature

As described in Section 3.1, the API signature is represented by a customized HTTP header parameter **FSPIOP-Signature**; thus the API request with the signature in this case is the following message text (line breaks and indentation within values for display purposes only).

Signature

Open API for FSP Interoperability Specification

```
POST /quotes HTTP/1.1
FSPIOP-Destination:5678
Accept:application/vnd.interoperability.quotes+json;version=1.0
Content-Length:975
Date:Tue, 23 May 2017 21:12:31 GMT
FSPIOP-Source:1234
Content-Type:application/vnd.interoperability.quotes+json;version=1.0
FSPIOP-Signature: {"signature":
               "dz2ntyS0_rDyA0pLeWluG--tBcYYrlvG99ffkXcEB-
               uve5Qzvzyn0ZUi82J7h17RsdfHPuTnbEGvCeU9Y4Bg0nIZHGL4icswaaO09T5hPPY-
               KBTzVQeHkokLmL4dXpHdr1ggSEpu3WEU3nfgOFGGAdOq355i1iGuDbhqm_lSfVHaqdVCEh
               kJ2Y_r2glO2QpdZrcbvsBV39derj_PlfISBBGjdh0dIPxnFIVcZuPHiq9Ha2MslrBHfqwF
               fNeU_xhErBd2PywkDQJbKOlfqdkmFC9bS8Ofx0O6Mg7qdFGw-
               QkseJTfp0HMbH1d9e6H0cocY8xfuDNGaZpOJhxiYtiPLg", "protectedHeader":
               "eyJhbGciOiJSUzI1NiIsIkZTUElPUC1EZXN0aW5hdGlvbiI6IjU2NzgiLCJGU1BJT1AtV
               VJJIjoiL3F1b3RlcyIsIkZTUElPUC1IVFRQLU1ldGhvZCI6IlBPU1QiLCJEYXRlIjoiVHV
               lLCAyMyBNYXkgMjAxNyAyMToxMjozMSBHTVQiLCJGU1BJT1AtU291cmNlIjoiMTIzNCJ9"
}

{
    "amount": { "amount": "150", "currency": "USD" },
    "transactionType": {
        "scenario": "TRANSFER", "initiator": "PAYER",
        "subScenario": "P2P Transfer across MM systems",
        "initiatorType": "CONSUMER" },
    "transactionId": "36629a51-393a-4e3c-b347-c2cb57e1e1fc",
    "quoteId": "59e331fa-345f-4554-aac8-fcd8833f7d50",
    "expiration": "2017-05-24T08:40:00.000-04:00",
    "payer": {
        "personalInfo": { "dateOfBirth": "1986-02-14",
            "complexName": { "middleName": "Ben",
                "LastName": "Lee", "firstName": "Bill" } },
        "name": "Bill Lee",
        "partyIdInfo": { "fspId": "1234", "partyIdType": "MSISDN",
            "partySubIdOrType": "RegisteredCustomer",
            "partyIdentifier": "16135551212" } },
    "payee": { "partyIdInfo": { "fspId": "5678",
            "partyIdType": "MSISDN",
            "partyIdentifier": "15295558888" } },
    "fees": { "amount": "1.5", "currency": "USD" },
    "extensionList": {
            "extension": [
                    { "value": "value1", "key": "key1" },
                    { "value": "value2", "key": "key2" },
                    { "value": "value3", "key": "key3" } ]
            },
    "note": "this is a sample for POST /quotes",
    "geoCode": { "longitude": "125.520001", "latitude": "57.323889" },
    "amountType": "RECEIVE"
}
```

4.2 Validating Signature

After the Payee FSP receives the **POST /quotes** API message from Payer FSP, the Payee FSP must validate the signature signed by the Payer FSP.

4.2.1 Parse FSPIOP-Signature

1. Parse the HTTP header parameter **FSPIOP-Signature** to get the components **protectedHeader** and signature. In this case, the value of **protectedHeader** is:

```
eyJhbGciOiJSUzI1NiIsIkZTUElPUC1EZXN0aW5hdGlvbiI6IjU2NzgiLCJGU1BJT1AtVVJJIjo
iL3F1b3RlcyIsIkZTUElPUC1IVFRQLU1ldGhvZCI6IlBPU1QiLCJEYXRlIjoiVHVlLCAyMyBNYX
kgMjAxNyAyMToxMjozMSBHTVQiLCJGU1BJT1AtU291cmNlIjoiMTIzNCJ9
```

2. Use BASE64URL to decode the encoded representation of the JWS Protected Header. Verify that the resulting octet sequence is a UTF-8-encoded representation of a completely valid JSON object conforming to JSON Data Interchange Format, defined in RFC7159. In this case, the decoded JSON object is:

```
{
  "alg":"RS256",
  "FSPIOP-Destination":"5678",
  "FSPIOP-URI":"/quotes",
  "FSPIOP-HTTP-Method":"POST",
  "Date":"Tue, 23 May 2017 21:12:31 GMT",
  "FSPIOP-Source":"1234"
}
```

3. Verify that the **alg** parameter is valid for the API. That means it must be in the list of **RS256**, **RS384**, **RS512**. In this case, the value of **alg** is **RS256**, which is valid.
4. Verify that the value of the parameter **FSPIOP-URI** is same as the input URL of this API message.
5. Verify that the value of the parameter **FSPIOP-HTTP-Method** is same as the HTTP method of this API message.
6. Verify that the value of the HTTP header parameter **FSPIOP-Source** is the same as the corresponding value listed in this JWS Protected Header.
7. Verify that the values for the HTTP header parameter **FSPIOP-Destination** are the same as the corresponding values stated in this JWS Protected Header.
8. Verify the other protected HTTP header parameters. In this case, the **Date** parameter is protected by JWS Protected Header. If the parameters **Date** in the HTTP header of this API message and **Date** in the JWS Protected Header are equal, then the validation is successful. Both **Date** parameters in the example should be the following value:

```
"Tue, 23 May 2017 21:12:31 GMT"
```

   The validation is passed.

4.2.2 Verify JWS Signature

1. In this case, the JWS Payload is the full HTTP body of the API message, that is (line breaks and indentation within values for display purposes only):

```
{
    "amount": { "amount": "150", "currency": "USD" },
    "transactionType": { "scenario": "TRANSFER", "initiator": "PAYER",
        "subScenario": "P2P Transfer across MM systems",
        "initiatorType": "CONSUMER"
    },
    "transactionId": "36629a51-393a-4e3c-b347-c2cb57e1e1fc",
    "quoteId": "59e331fa-345f-4554-aac8-fcd8833f7d50",
    "expiration": "2017-05-24T08:40:00.000-04:00",
    "payer": {
        "personalInfo": { "dateOfBirth": "1986-02-14",
            "complexName": { "middleName": "Ben",
                "LastName": "Lee", "firstName": "Bill" } },
        "name": "Bill Lee",
        "partyIdInfo": { "fspId": "1234",
            "partyIdType": "MSISDN",
            "partySubIdOrType": "RegisteredCustomer",
            "partyIdentifier": "16135551212" } },
    "payee": {
        "partyIdInfo": { "fspId": "5678",
            "partyIdType": "MSISDN",
            "partyIdentifier": "15295558888" } },
    "fees": { "amount": "1.5", "currency": "USD" },
    "extensionList": {
            "extension": [
                    { "value": "value1", "key": "key1" },
                    { "value": "value2", "key": "key2" },
                    { "value": "value3", "key": "key3" } ]
            },
    "note": "this is a sample for POST /quotes",
    "geoCode": { "longitude": "125.520001", "latitude": "57.323889" },
    "amountType": "RECEIVE"
}
```


2. Compute the encoded payload value BASE64URL(JWS Payload). Get the encoded value as:

```
eyJwYXllZSI6eyJwYXJ0eUlkSW5mbyI6eyJwYXJ0eUlkVHlwZSI6Ik1TSVNETiIsInBhcnR5SWR
lbnRpZmllciI6IjE1Mjk1NTU4ODg4IiwiZnNwSWQiOiI1Njc4In19LCJhbW91bnRUeXBlIjoiUk
VDRUlWRSIsInRyYW5zYWN0aW9uVHlwZSI6eyJzY2VuYXJpbyI6IlRSQU5TRkVSIiwiaW5pdGlhd
G9yIjoiUEFZRVIiLCJzdWJTY2VuYXJpbyI6IlAyUCBUcmFuc2ZlciBhY3Jvc3MgTU0gc3lzdGVt
cyIsImluaXRpYXRvclR5cGUiOiJDT05TVU1FUiJ9LCJub3RlIjoidGhpcyBpcyBhIHNhbXBsZSB
mb3IgUE9TVCAvcXVvdGVzIiwiYW1vdW50Ijp7ImFtb3VudCI6IjE1MCIsImN1cnJlbmN5IjoiVV
NEIn0sImZlZXMiOnsiYW1vdW50IjoiMS41IiwiY3VycmVuY3kiOiJVU0QifSwiZXh0ZW5zaW9uT
GlzdCI6W3sidmFsdWUiOiJ2YWx1ZTEiLCJrZXkiOiJrZXkxIn0seyJ2YWx1ZSI6InZhbHVlMiIs
ImtleSI6ImtleTIifSx7InZhbHVlIjoidmFsdWUzIiwia2V5Ijoia2V5MyJ9XSwiZ2VvQ29kZSI
6eyJsYXRpdHVkZSI6IjU3LjMyMzg4OSIsImxvbmdpdHVkZSI6IjEyNS41MjAwMDEifSwiZXhwaX
JhdGlvbiI6IjIwMTctMDUtMjRUMDg6NDA6MDAuMDAwLTA0OjAwIiwicGF5ZXIiOnsicGVyc29uY
WxJbmZvIjp7ImNvbXBsZXhOYW1lIjp7ImZpcnN0TmFtZSI6IkJpbGwiLCJtaWRkbGVOYW1lIjoi
QmVuIiwiTGFzdE5hbWUiOiJMZWUifSwiZGF0ZU9mQmlydGgiOiIxOTg2LTAyLTE0In0sInBhcnR
5SWRJbmZvIjp7InBhcnR5SWRUeXBlIjoiTVNJU0ROIiwicGFydHlTdWJJZE9yVHlwZSI6IlJlZ2
lzdGVyZWRDdXN0b21lciIsInBhcnR5SWRlbnRpZmllciI6IjE2MTM1NTUxMjEyIiwiZnNwSWQiO
iIxMjM0In0sIm5hbWUiOiJCaWxsIExlZSJ9LCJxdW90ZUlkIjoiNTllMzMxZmEtMzQ1Zi00NTU0
LWFhYzgtZmNkODgzM2Y3ZDUwIiwidHJhbnNhY3Rpb25JZCI6IjM2NjI5YTUxLTM5M2EtNGUzYy1
iMzQ3LWMyY2I1N2UxZTFmYyJ9
```


3. Validate the JWS Signature against the JWS Signing Input (that is, the JWS Protected Header, JWS Payload) with the specified algorithm **RS256** (specified in the JWS Protected Header), and the public key. Record whether the validation succeeded or not.