Article

Generating JWS to sign App Store requests

Create signed JSON Web Signature (JWS) strings on your server to authorize your API requests in your app.

Overview

JWS is an open standard (RFC 7515) that defines a way to securely transmit information. You create the JWS header and payload using information specific to the App Store feature. You sign the JWS on your server using a private API key that you download from App Store Connect.

The following App Store features require a JWS Compact Serialization to authorize calls in StoreKit:

- Advanced Commerce API in-app requests
- Promotional offer signatures
- Introductory offer eligibility

The data you sign when generating the JWS is a JSON Web Token (JWT). The JWT contains name-value pairs called *claims*, including claims specific to the feature you're using. Your server returns the JWS Compact Serialization to your app. Your app uses the JWS in the relevant StoreKit API call.

Tip

The App Store Server Library provides a signing utility that creates JWS specific for each of these uses. For more information, see Simplifying your implementation by using the App Store Server Library.

To get started, you need your key ID and issuer ID from App Store Connect. Then, to generate a signed JWS, you perform these steps on your server:

1. Create the JWS header.
2. Create the JWS payload. The payload includes the base payload claims, and may also require feature-specific claims.
3. Sign the JWS.

Get your key ID and issuer ID from App Store Connect

First you need your key ID for the JWS header and your issuer ID for the JWS payload. Sign in to App Store Connect to get these values.

To get the key ID:

1. Select Users and Access, then select the Keys tab.
2. Hover the cursor next to a key ID to display the Copy Key ID link. The key IDs appear in a column under the Active heading.
3. Click Copy Key ID.

If you have more than one API key, copy the key ID of the private key that you use to sign the JWS. For information about creating keys, see Creating API keys to authorize API requests.

To get the issuer ID:

1. Select Users and Access, then select the Keys tab.
2. To copy the issuer ID that appears near the top of the page, click Copy next to the ID.

Create the JWS header

The App Store features all use the same JWS header.

To create a JWS to communicate with the App Store, use the following fields and values in the header:

Header field	Value
`alg` - Encryption Algorithm	`ES256` You need to sign JWS with ES256 encryption.

Header field	Value
kid - Key ID	Your private key ID from App Store Connect (example: 2X9R4HXF34)
typ - Token Type	JWT

Here's an example of a JWS header:

```
{
  "alg": "ES256",
  "kid": "2X9R4HXF34",
  "typ": "JWT"
}
```

Create the JWS payload

The JWS payload contains base claims specific to the App Store, such as an issuer ID and issuance time. It also includes custom claims that are specific to App Store features. The following table shows the base claims. The next sections shows the feature-specific claims that you also include in the JWS payload.

Include the following base claims in the JWS payload:

Payload field	Value
iss - Issuer	Your issuer ID from the Keys page in App Store Connect (example: "57246542-96fe-1a63-e053-0824d011072a")
iat - Issued At	The UNIX time, in seconds, that you issue the token, which the App Store server uses to calculate an expiration time (example: 1623085200)
aud - Audience	A value that depends on the feature you're using (see the table below)
bid - Bundle ID	Your app's bundle ID (example: "com.example.testbundleid")
nonce - Nonce	A one-time-use UUID that identifies this request (example: "368f3888-dcd8-11ef-b3c8-325096b39f47")

Don’t include an `exp` field in your payload, because including it makes the request fail. The App Store server enforces an expiration time based on your `iat` value.

Choose the `aud` value to match the feature you’re using:

Feature	aud value
Advanced Commerce API in-app request	`“advanced-commerce-api"`
Promotional offer signature	`“promotional-offer"`
Introductory offer eligibility	`“introductory-offer-eligibility"`

Here’s an example of the base payload, without any feature-specific fields:

```
{
  "iss": "57246542-96fe-1a63e053-0824d011072a",
  "iat": 1741043663,
  "aud": "promotional-offer",
  "bid": "com.example.testbundleid",
  "nonce": "6584bedf-2ed0-4c01-93ed-c0c64a1670cc"
}
```

Be sure to include feature-specific custom claims in the JWS payload as well.

Include custom claims for Advanced Commerce API in-app requests

For custom claims for Advanced Commerce in-app requests, use the following values:

Payload field	Value
`request`	Base64-encoded request data

For more information on generating the base64-encoded request data, see Create the base64-encoded request data.

Here’s an example of a payload for an Advanced Commerce API in-app request:

```
{
  "iss": "57246542-96fe-1a63e053-0824d011072a",
  "iat": 1741043663,
  "aud": "advanced-commerce-api",
  "bid": "com.example.testbundleid",
  "nonce": "df2b8374-95a1-425b-a6a5-77a4d7648333",
  "request": "<base64-encoded request data>"
}
```

For more information about making Advanced Commerce API requests in StoreKit, see Sending Advanced Commerce API requests from your app.

Include custom claims for promotional offer signatures

For promotional offer signature custom claims, use the following values:

Payload field	Value
`productId`	The unique identifier of the product (for more information, see `id`)
`offerIdentifier`	The promotional offer identifier that you set up in App Store Connect
`transactionId`	The unique identifier of any transaction that belongs to the customer. You can use the customer's `appTransactionID`, even for customers who haven't made any In-App Purchases in your app. This field is optional, but recommended.

Here's an example of a payload for a promotional offer signature:

```
{
    "iss": "57246542-96fe-1a63e053-0824d011072a",
    "iat": 1741043663,
    "aud": "promotional-offer",
    "bid": "com.example.testbundleid",
    "nonce": "368f3088-dcd5-11ef-b3c8-325096b39f46",
    "productId": "com.example.product",
    "offerIdentifier": "com.example.product.offer",
    "transactionId": "1000011859217"
}
```

Include custom claims for introductory offer eligibility

For introductory offer eligibility custom claims, use the following values:

Payload field	Value
`productId`	The unique identifier of the product (for more information, see `id`)
`allowIntroductoryOffer`	A Boolean value, `true` or `false`, that determines whether the customer is eligible for an introductory offer
`transactionId`	The unique identifier of any transaction that belongs to the customer. You can use the customer's `appTransactionID`, even for customers who haven't made any In-App Purchases in your app.

Here's an example of a payload for introductory offer eligibility:

```
{
  "iss": "57246542-96fe-1a63e053-0824d011072a",
  "iat": 1741043663,
  "aud": "introductory-offer-eligibility",
  "bid": "com.example.testbundleid",
  "nonce": "cfb43594-4f92-4fe2-8b06-d947a848adaa",
  "productId": "com.example.product",
  "allowIntroductoryOffer": false,
  "transactionId": "1000011859217"
}
```

For more information about providing the introductory offer eligibility signed value in a purchase option, see `Product.PurchaseOption`.

Sign the JWS

On your server, use the private key associated with the key ID you specified in the header to generate the signature using ES256 encryption. The process of signing combines the JWS Header and JWS Payload into a single signed string.

There are a variety of open source libraries available online for creating and signing JWT tokens; see JWT.io for more information. Consider using the App Store Server Library to create the JWS. For more information, see Simplifying your implementation by using the App Store Server Library.

After signing, you should have a string in JWS compact serialization format. Send that string to your app.

> **Note**
>
> Always use a secure connection when sending data, including the signature, between your app and server. For more information on ensuring your data's security, see Preventing Insecure Network Connections.

See Also

Advanced Commerce API interactions

`struct AdvancedCommerceProduct`

A product configured as a generic SKU in App Store Connect for use with the Advanced Commerce API.

Sending Advanced Commerce API requests from your app

Send Advanced Commerce API requests from your app that you authorize with a JSON Web Signature (JWS) you generate on your server.