Article

Uploading Assets to App Store Connect

Upload screenshots, app previews, attachments for App Review, and routing app coverage files to App Store Connect.

Overview

While managing your App Store apps, you may need to upload various assets or files to App Store Connect. These assets may include screenshots, app previews, attachments for App Review, and routing app coverage files. The uploading APIs are designed to ensure reliability for large files, even over an unreliable connection. The API instructs you to divide individual large asset files into multiple upload requests. When you tell the API you've finished uploading, App Store Connect begins processing your files.

There are four steps to upload an asset:

1. Make an asset reservation.

2. Upload an asset, potentially in multiple parts.

3. Commit the upload.

4. Verify that App Store Connect processed the asset successfully.

This workflow is the same for every asset type you manage and upload using the following API resources:

Asset Type	API Resource
App Store screenshots	App Screenshot Sets App Screenshots
App event screenshots	App Event Screenshots
App previews	App Preview Sets App Previews

Asset Type	API Resource
App Clip card images	App Clip Header Images Advanced App Clip Experience Images
Attachments for the App Review team	App Store review attachments
Game center activity images	Game Center activity images
Game center challenge images	Game Center challenge images
In-App purchase App Store review screenshots	In-App Purchase App Store Review Screenshots
In-app purchase images	In-app purchase images
Subscription App Store review screenshots	Subscription App Store Review Screenshots
Subscription images	Subscription images
Routing app coverage files	Routing App Coverages

Make an Asset Reservation

The first step to upload an asset is making an asset reservation. You send a request that specifies the type of asset you'll be uploading, its filename, and size in bytes. A successful response provides a set of operations you will use to upload the asset. If your asset is large, you may receive multiple operations in the response that ask you to divide the asset into parts for uploading.

For example, to make a reservation to upload an app screenshot called "my_screenshot.png" with a file size of 11097 bytes, the request is:

```
POST /v1/appScreenshots
{
    "data": {
        "type": "appScreenshots",
        "attributes": {
            "fileSize": 11097,
            "fileName": "my_screenshot.png"
        },
        "relationships": {
            "appScreenshotSet": {
```

```
                "data": {
                    "type": "appScreenshotSets",
                    "id": "54594240-5c4c-4a0f-add1-b4cb7e52d166"
                }
            }
        }
    }
}
```

A successful response includes these fields to note:

- A unique ID that identifies the reservation throughout the delivery workflow.
- A set of operations that you use to upload the parts of the asset to App Store Connect.
- Attributes that describe the file.
- The asset delivery state, which will be `AWAITING_UPLOAD`.

The full response is:

```
{
    "data" : {
        "type" : "appScreenshots",
        "id" : "4d62262c-4ec1-4d89-b82c-c7b7a402e866",
        "attributes" : {
            "fileSize" : 11097,
            "fileName" : "my_screenshot.png",
            "sourceFileChecksum" : null,
            "imageAsset" : null,
            "assetToken" : "PurpleSource62/v4/c4/c6/5f/c4c65fe0-b616-0454-d71b-7771b
            "assetType" : "SCREENSHOT",
            "uploadOperations" : [ {
                "method" : "PUT",
                "url" : "https://store-030.blobstore.apple.com/assets-massilia-03000
                "length" : 11097,
                "offset" : 0,
                "requestHeaders" : [ {
                    "name" : "Content-Type",
                    "value" : "image/png"
                } ]
           } ],
           "assetDeliveryState" : {
               "errors" : [ ],
               "state" : "AWAITING_UPLOAD"
```

```
            }
        },
        "relationships" : {
            "appScreenshotSet" : {
                "links" : {
                    "self" : "https://api.appstoreconnect.apple.com/v1/appScreenshots
                     "related" : "https://api.appstoreconnect.apple.com/v1/appScreens
                }
            }
        },
        "links" : {
            "self" : "https://api.appstoreconnect.apple.com/v1/appScreenshots/4d62262
        }
    },
    "links" : {
         "self" : "https://api.appstoreconnect.apple.com/v1/appScreenshots"
     }
}
```

Note the upload operations. In this example, the file can be uploaded in a single operation:

```
"uploadOperations" : [ {
    "method" : "PUT",
    "url" : "https://store-030.blobstore.apple.com/assets-massilia-030001/PurpleSou
    "length" : 11097,
    "offset" : 0,
    "requestHeaders" : [ {
        "name" : "Content-Type",
        "value" : "image/png"
    } ]
} ]
```

After you make a successful reservation, your asset moves into the `AWAITING_UPLOAD` state, which you can see in the response.

If your reservation request fails, the API returns an error code (`ErrorResponse`). For more information, see Interpreting and Handling Errors.

Upload the Asset

The next step in delivering an asset to App Store Connect is uploading the binary data that is the asset or file. Use the upload operations returned in the response to your reservation request to

upload your asset to App Store Connect.

> **Important**
>
> You have limited time to complete the upload. In general, plan to finish an upload within a week of creating the asset reservation. See `Resolve failures due to an expired reservation` for more information.

If your asset is large enough, you will receive multiple upload operations, each specifying the length in bytes and the byte offset into the file. Divide the large asset into binary data parts as specified by those instructions. One option, using the `length` 11097, is:

```
split -b 11097 sample_image.png
```

The resulting split files are have a sequential naming convention, you need to upload all splits of the original file. The system will combine the splits into one image.

Next you:

- Make an HTTP request using the method, URL, and request headers specified in the operation.
- In the body of the request, include the binary data from the given byte range of the original file.

The provided upload URLs are unauthenticated and time-limited. You don't need to supply a JWT; don't share the URLs.

You can upload parts of your asset concurrently and in any order to improve performance. If one part doesn't upload correctly, you can resend it at any time before you commit the asset. See `Resolve upload failures caused by lost connections including power outages` for more information.

The asset state remains as `AWAITING_UPLOAD` while you upload the parts that make up your file.

Commit the Reservation

After you've uploaded all parts of your asset to App Store Connect, you must commit the reservation. The commit tells App Store Connect that you've finished uploading and the asset is ready to be processed.

To commit the reservation, provide the following attributes in the PATCH request:

- `uploaded`
- `sourceFileChecksum` - the MD5 checksum of your original, entire asset file

```
PATCH /v1/appScreenshots/4d62262c-4ec1-4d89-b82c-c7b7a402e866
{
    "data": {
        "type": "appScreenshots",
        "id": "4d62262c-4ec1-4d89-b82c-c7b7a402e866",
        "attributes": {
            "uploaded": true,
            "sourceFileChecksum": "1a79a4d60de6718e8e5b326e338ae533"
        }
    }
}
```

App Store Connect compares the bytes received with the total bytes you described for your asset in the asset reservation request (`fileSize`). If the total bytes received don't match the file size value in the reservation, the commit request fails.

A successful commit request changes the asset state to `UPLOAD_COMPLETE`.

> **Note**
>
> Once you commit the asset, you can no longer upload component parts. If you need to update your asset after committing it, you must delete it and create another upload, beginning with an asset reservation.

Verify the Upload Succeeded

After receiving your commit request, App Store Connect validates and processes the uploaded asset. Processing is asynchronous and can take from a few seconds up to a few minutes, depending on the size and complexity of the asset. Your asset remains in the `UPLOAD_COMPLETE` state until App Store Connect finishes processing it. Use the API to re-fetch the asset at any time to check on its status.

If your asset processes successfully, the system changes the asset's state to `COMPLETE`. This is the final state for a successful asset upload.

If your asset's processing fails, the system changes the asset's state to `FAILED`. `FAILED` is a terminal state. To upload your asset you must delete the asset record and retry your upload, beginning with making an asset reservation.

Validate Your Upload

App Store Connect validates uploads by confirming that the checksum of the final asset received matches the `sourceFileChecksum` you specified in your commit request. App Store Connect also checks other criteria, like file formats and screenshot dimensions.

You can further validate your uploads by viewing your app's metadata in App Store Connect or by using the API to download the final processed asset for review.

Resolve Upload Failures

You may encounter errors at any stage of the upload process: during the upload due to power outages or an expired reservations, or after you've uploaded the asset if the App Store Connect processing fails.

If you experience a lost connection while uploading an asset or a part of an asset, you can retry the upload for just the failed parts using the upload operations. You can continue uploading the asset parts, as needed, until you commit the reservation.

If an asset reservations expires, attempting an upload or commit returns an error. You have a limited window of time to finish your upload after you have created an asset reservation. In general, plan to finish an upload within a week of creating the asset reservation. If the time window expires before you finish uploading and committing the asset, delete the asset reservation and begin again, starting with requesting a new asset reservation. The exact expiration time for each upload operation is found in the `Expires` parameter in the upload URL. This value is a Unix timestamp, in UTC.

If your asset fails processing at App Store Connect, the asset state will change to `FAILED`. `FAILED` is a terminal state. Review the error messaging provided by the API so you can identify and resolve the issue. Then delete the asset record and retry your upload, beginning by making an asset reservation.

See Also

Essentials

- Creating API Keys for App Store Connect API
  Create API keys to sign JSON Web Tokens (JWTs) and authorize API requests.
- Generating Tokens for API Requests
  Create JSON Web Tokens (JWTs) signed with your private key to authorize API requests.
- Revoking API Keys
  Revoke unused, lost, or compromised private keys.

Identifying Rate Limits

Recognize the rate limits that REST API responses provide and handle them in your code.

App Store Connect API Release Notes

Learn about new features and updates in the App Store Connect API.