Sample Code

Uploading App Previews

Upload your app previews, including video files, to App Store Connect by using the asset upload APIs.

Download

Overview

This sample code includes a script written in Python that uploads a video as an app preview for an app. To run this sample, you must have an app in App Store Connect with a version in the “Prepare for Submission” state, and an app preview file to upload. You’ll also need your API key.

The sample’s script first generates an authentication token, so it can call the App Store Connect API. It then creates an app preview set or uses an existing one, if one exists. The script then reserves a new app preview associated with the app preview set, uploads the binary data, and commits the upload.

For more information about the process of uploading assets, see Uploading Assets to App Store Connect.

Configure the Sample Code Project

This project requires Python 3, which is pre-installed on macOS 10.15. It uses Python’s `pipenv` dependency management system, which you need to install. Follow these steps to install `pipenv` and the required Python modules:

1. Open the Terminal app by typing Command-space and entering “Terminal.app” in the Spotlight search field.
2. In Terminal, enter:

```
pip3 install --user pipenv
```

If you're prompted to install Xcode command line tools, follow the prompts. When installation is complete, run the above command again.

3. In Terminal, navigate to the folder that contains this sample project:

```
 cd UploadingAppPreviews
```

4. Install the Python modules by entering:

```
~/Library/Python/3.7/bin/pipenv install
```

If you've installed your own version of Python, the `pipenv` path may be different.

Add the API Key to the Script and Prepare the Upload

Prepare to run the script by following these steps:

1. Add your API key to the script by editing the *upload-preview.py* file and adding your API Key information to the KEY CONFIGURATION section at the top of the file. To find your API key, log in to App Store Connect and open the Users & Access module. The key is found in the Keys tab. For more information, see Creating API Keys for App Store Connect API.

2. Make sure the video file you'd like to upload is available on your local machine. For more information about app previews, including technical specifications, see Show More with App Previews.

3. Your app must be in App Store Connect with a version in the "Prepare for Submission" state. To upload the app preview, you'll need the `bundle id`, `platform`, `version number`, and `locale` of this app.

Provide Command-Line Parameters to Run the Script

On the command line, run this command, replacing the parameters with actual values:

```
~/Library/Python/3.7/bin/pipenv run python3 upload-preview.py {bundle id} {platform
```

The command-line parameters are:

- `{bundle id}`—Your app's bundle ID.

- `{platform}`—Your app's platform. See Platform for valid values.
- `{version}`—The version of your app to which the preview applies.
- `{locale}`—The locale to which the preview applies. See Create an App Info Localization for valid values.
- `{preview type}`—The type of preview. See Preview Types for valid values.
- `{path to video file}`—The path to the video file on your computer.

The following example replaces `{bundle id}` with `com.mycompany.MyApp`; `{platform}` with `IOS`; `{version}` with `1.0`; `{locale}` with en-US; `{preview type}` with `IPHONE_65`; and `{path to video file}` with `../my-video.mp4`.

```
~/Library/Python/3.7/bin/pipenv run python3 upload-preview.py com.mycompany.MyApp IO
```

Generate an Authentication Token

First, the script creates a JSON Web Token (JWT) that it uses to authenticate each App Store Connect API request. The JWT includes an expiration timestamp, along with an Issuer ID, Key ID, and the `appstoreconnect-v1` audience. The token is signed using the ES256 algorithm and the private key.

```
def create_token():
    expiry = datetime.datetime.utcnow() + datetime.timedelta(minutes=20)
    token_data = jwt.encode(
        {
            'iss': ISSUER_ID,
            'aud': 'appstoreconnect-v1',
            'exp': expiry
        },
        PRIVATE_KEY,
        headers={
            'kid': KEY_ID
        },
        algorithm='ES256'
    )
    return token_data.decode('UTF-8')
```

Find or Create the App Preview Set

The App Store Connect API requires the addition of an app preview to an *app preview set*. Create one app preview set for each preview type and each locale. Each set includes up to three previews. For example, a set might have a 6.5 inch iPhone in the US English locale, and another in the Mexican Spanish locale.

Before the sample creates the app preview, it needs to find the right preview set or create a new set. The script first searches for the app by bundle ID.

```
app_response = make_http_request(
    "GET",
    f"https://api.appstoreconnect.apple.com/v1/apps?filter[bundleId]={bundleId}",
    headers={
        "Authorization": auth_header
    }
)
```

Then, the script searches for the version by platform and version number:

```
version_response = make_http_request(
    "GET",
    f"https://api.appstoreconnect.apple.com/v1/apps/{app['id']}/appStoreVersions?fil
    headers={
        "Authorization": auth_header
    }
)
```

If the version doesn’t have a localization for the given locale, the script creates a new one:

```
selected_localization_response = make_http_request(
    "POST",
    "https://api.appstoreconnect.apple.com/v1/appStoreVersionLocalizations",
    headers={
        "Authorization": auth_header,
        "Content-Type": "application/json"
    },
    body=json.dumps({
        "data": {
            "type": "appStoreVersionLocalizations",
            "attributes": {
                "locale": locale
            },
            "relationships": {
                "appStoreVersion": {
                    "data": {
                        "type": "appStoreVersions",
                        "id": version['id']
                    }
                }
            }
        }
    })
)
```

Finally, if the localization doesn't have an app preview set of the given type, the script creates one:

```
preview_set_response = make_http_request(
    "POST",
    "https://api.appstoreconnect.apple.com/v1/appPreviewSets",
    headers={
        "Authorization": auth_header,
        "Content-Type": "application/json"
    },
    body=json.dumps({
        "data": {
            "type": "appPreviewSets",
            "attributes": {
                "previewType": previewType
            },
            "relationships": {
                "appStoreVersionLocalization": {
                    "data": {
                        "type": "appStoreVersionLocalizations",
                        "id": selected_localization['id']
                    }
                }
            }
        }
    })
)
```

Reserve an App Preview

Once the sample project finds or creates the app preview set, it reserves a new app preview associated with the app preview set ID. The reservation tells App Store Connect the name and size of the file the script intends to upload.

```
reserve_preview_response = make_http_request(
    "POST",
    "https://api.appstoreconnect.apple.com/v1/appPreviews",
    headers={
        "Authorization": auth_header,
        "Content-Type": "application/json"
    },
    body=json.dumps({
        "data": {
            "type": "appPreviews",
            "attributes": {
                "fileName": os.path.basename(filePath),
                "fileSize": os.path.getsize(filePath)
            },
            "relationships": {
                "appPreviewSet": {
                    "data": {
                        "type": "appPreviewSets",
                        "id": selected_preview_set['id']
                    }
                }
            }
        }
    })
)
```

Upload the Binary Data

In response to the reservation request, App Store Connect responds with a set of upload operations. Use the upload operations to upload a file, potentially in multiple parts. The sample uses the `offset` and `length` in the upload operation to extract a portion of the source file.

```
with open(filePath, mode='rb') as file:
    file.seek(upload_operation['offset'])
    bytes = file.read(upload_operation['length'])
```

The `method`, `url` and `requestHeaders` in the upload operation are used for uploading the chunk of data.

```
make_http_request(
    upload_operation['method'],
    upload_operation['url'],
    headers={h['name']: h['value'] for h in upload_operation['requestHeaders']},
    body=bytes
)
```

Commit the App Preview

Once all the parts of the file are uploaded, the sample code then demonstrates how to commit the app preview by marking it uploaded and providing an MD5 checksum of the original source file.

```
make_http_request(
    "PATCH",
    f"https://api.appstoreconnect.apple.com/v1/appPreviews/{preview['id']}",
    headers={
        "Authorization": auth_header,
        "Content-Type": "application/json"
    },
    body=json.dumps({
        "data": {
            "type": "appPreviews",
            "id": preview['id'],
            "attributes": {
                "uploaded": True,
                "sourceFileChecksum": hashlib.md5(open(filePath,'rb').read()).hexdig
            }
        }
    })
)
```

Verify the App Preview

App Store Connect processes the asset asynchronously. Check the upload's status in App Store Connect or by re-requesting the `/v1/appPreview/{id}` and examining the response.

See Also

Managing App Previews

Uploading Assets to App Store Connect

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

App Preview Sets

Create sets of app previews to upload to App Store Connect.

App Previews

Upload and download app previews for an app locale and display target.