# PCC YouTube Upload CLI - Design Document

## 1. Project Overview

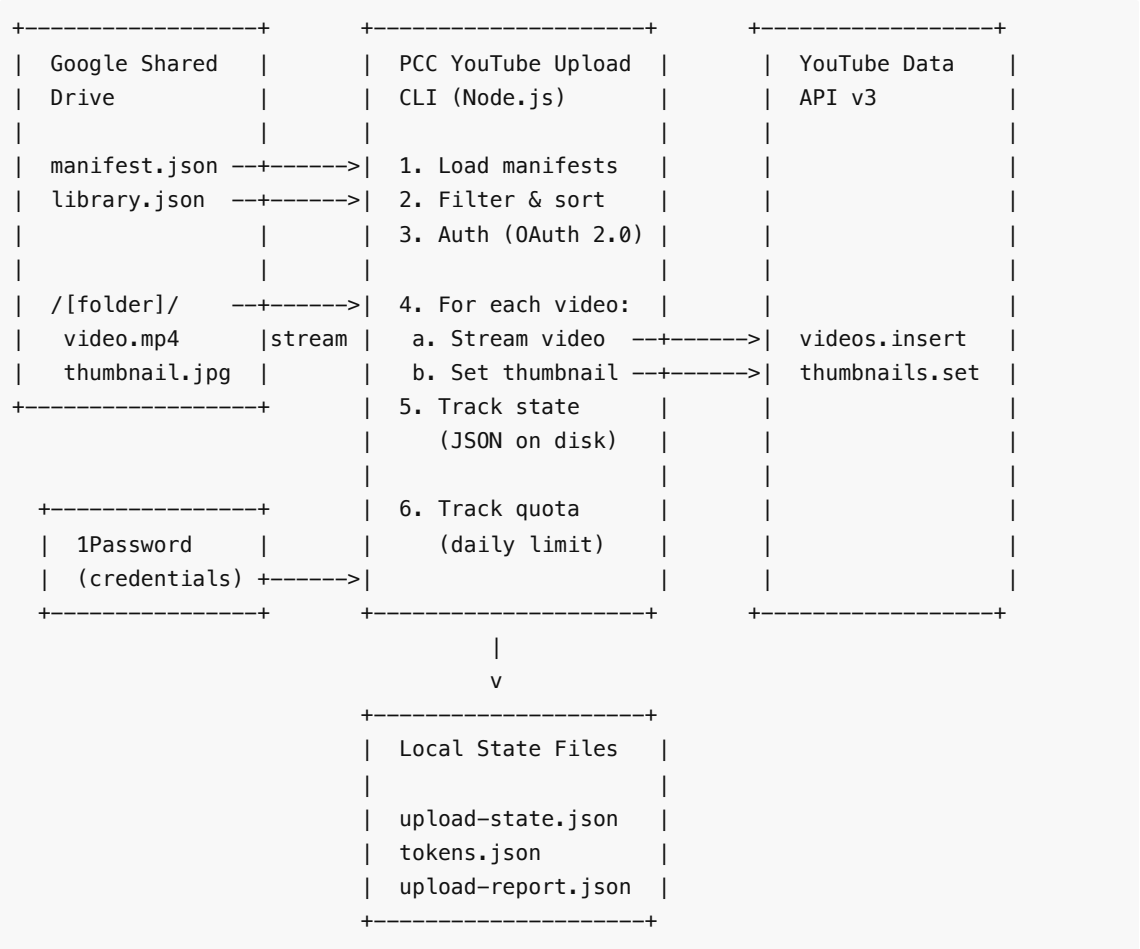
**Organization:** Pacific Crossroads Church (501(c)(3) nonprofit, Los Angeles, CA) **Project:** PCC YouTube Upload CLI **Google Cloud Project Number:** 150130945051 **Purpose:** One-time bulk upload of a sermon video archive to YouTube

Pacific Crossroads Church maintains an archive of sermon recordings on a Google Shared Drive. This CLI tool automates uploading 505 sermon videos to the church's YouTube channel with custom thumbnails. Playlist organization is handled manually in YouTube Studio. The tool is designed for a single migration project and will not be used for ongoing operations once the archive upload is complete.

### Scope

| Item | Count |
|---|---|
| Videos | 505 |
| Thumbnails | 505 |

## 2. Architecture Overview



## 3. Authentication Flow

The tool uses **OAuth 2.0** with the following scopes:

- `youtube.upload` - Upload videos
- `youtube` - Manage thumbnails
- `drive.readonly` - Read manifests and video files from Shared Drive

**Credential storage:** OAuth client ID and secret are stored in **1Password** and retrieved at runtime via the 1Password SDK. They are never written to disk or committed to source control.

**Token lifecycle:**

1. First run: `auth` command opens a browser for Google OAuth consent, starts a local HTTP server on port 3000 to receive the callback
2. User selects a channel (supports brand accounts via `channels.list`)
3. Tokens (access + refresh) are saved to `tokens.json` (gitignored)
4. Subsequent runs: tokens are loaded from disk, auto-refreshed when expired via an `oauth2Client.on('tokens')` listener

## 4. Data Sources

Two JSON manifest files are loaded from Google Shared Drive at the start of each run:

| File | Contents |
|---|---|
| manifest.json | All media items with file references and status |
| library.json | Metadata: speaker, scriptures, tags, summaries |

**Filtering:** Only items where `files.video_original` exists AND `status === 'complete'` are selected for upload (466 of 1,176 total items currently ready; up to 505 total have video content).

**Sorting:** Items are sorted by date ascending (oldest sermons uploaded first).

## 5. Upload Pipeline

For each video, the tool performs these steps sequentially:

```
1. Check quota          Can we afford 1,650 units?
       |                      |
       | no                   | yes
       v                      v
Sleep until midnight    2. Stream video from Google Drive
Pacific, then retry            |
                               v
                        3. youtube.videos.insert
                           (1,600 units)
                           - title, description, tags
                           - category: Nonprofits & Activism
                           - privacy: private
                           - recording date from manifest
                              |
```

```
                              v
                        4. youtube.thumbnails.set
                           (50 units)
                           - priority: image_wide > image_banner > thumbnail_01
                              |
                              v
                        5. Save state to disk
                              |
                              v
                        6. Sleep 5 seconds
                              |
                              v
                        Next video...
```

**Video metadata constructed per upload:**

- **Title:** Sermon title from manifest
- **Description:** Series name, speaker, scripture references, summary text, church footer
- **Tags:** "sermon", "church", "Pacific Crossroads Church", speaker name, scripture references, custom tags
- **Category ID:** 29 (Nonprofits & Activism)
- **Privacy:** Private (can be changed to public manually after review)
- **Recording date:** Original sermon date
- **Subscriber notification:** Disabled ( `notifySubscribers: false` )

## 6. YouTube API Usage

| API Method | Cost (units) | When Used | Frequency |
|---|---|---|---|
| `youtube.videos.insert` | 1,600 | Upload each video | 505 times |
| `youtube.thumbnails.set` | 50 | Set thumbnail per video | 505 times |
| `youtube.channels.list` | 1 | Select brand account (auth only) | 1 time |

**Total estimated API cost:** ~833,250 units

**No read-heavy operations:** The tool does not poll, scrape, or repeatedly read YouTube data.

## 7. Quota Management

The tool tracks quota usage locally and respects the daily limit:

**Constants (from `src/quota.js` ):**

- `DAILY_QUOTA` : 10,000 units (will be updated if quota increase is granted)
- `VIDEO_TOTAL_COST` : 1,650 units (upload + thumbnail)

**Tracking:**

- `quotaUsedToday` counter incremented after each API call
- `quotaResetDate` stores the current date in Pacific timezone (America/Los_Angeles)
- When a new day is detected, `quotaUsedToday` resets to 0

**Daily limit enforcement:**

- Before each upload, `canUploadMore()` checks if 1,650 units remain
- If quota is exhausted, the tool calculates milliseconds until midnight Pacific and sleeps
- Adds a 60-second buffer after midnight before resuming

**Rate limiting:**

- 5-second delay between consecutive uploads
- Single-threaded: only one API call in flight at a time
- Peak QPS: 1

## 8. State Management

All state is persisted to `upload-state.json` after every operation:

```
upload-state.json
{
  "startedAt": "2025-01-15T...",
  "updatedAt": "2025-01-15T...",
  "quotaUsedToday": 8500,
  "quotaResetDate": "2025-01-15",
  "videos": {
    "<itemId>": {
      "status": "complete",        // pending | uploading | uploaded | complete |
failed
      "youtubeVideoId": "dQw4...",
      "youtubeUrl": "https://youtu.be/dQw4...",
      "uploadedAt": "...",
      "thumbnailUploaded": true
    }
  }
}
```

**Resumption:** On restart, `getNextPendingItem()` finds the first item with status `pending` or `uploading` and continues from there. No work is repeated.

**Partial failure handling:** If a video uploads but the thumbnail fails, the video is still marked `complete` with `thumbnailUploaded: false`.

## 9. Error Handling

**Retry logic (from `src/youtube.js`):**

- Max retries: 3
- Backoff delays: 5s, 30s, 120s
- Retryable errors: `ECONNRESET`, `ETIMEDOUT`, `EPIPE`, HTTP 500, HTTP 503, HTTP 403 with `rateLimitExceeded` reason

**Non-retryable errors:** The video is marked `failed` in state with the error message, and the tool moves to the next video.

**Dry-run mode:** `--dry-run` flag validates the manifest and shows what would be uploaded without making any API calls or persisting state changes.

## 10. Security

- **Credentials:** OAuth client ID and secret are stored in 1Password and fetched at runtime via the 1Password SDK. They are never stored in source code or config files.
- **Tokens:** OAuth access and refresh tokens are stored in `tokens.json`, which is listed in `.gitignore` and never committed.
- **Environment variables:** Sensitive values (`OP_SERVICE_ACCOUNT_TOKEN`) are loaded from `.env` (also gitignored).
- **Privacy:** All videos are uploaded as **private**. They can be reviewed and made public manually afterward.
- **Shared Drive access:** The tool uses `drive.readonly` scope — it can only read files, never modify or delete them.
- **No public-facing surface:** The tool is a CLI run locally by an administrator. The only network listener is a temporary HTTP server on localhost:3000 during the initial OAuth flow.

## 11. CLI Commands

| Command | Description |
|---|---|
| `auth` | Run OAuth 2.0 flow, select channel, save tokens |
| `upload` | Start or resume bulk upload |
| `upload --item <id>` | Upload a single video by manifest ID |
| `upload --dry-run` | Preview what would be uploaded without making changes |
| `status` | Show current quota usage and upload progress |
| `report` | Generate JSON report of upload progress |

## 12. Dependencies

| Package | Version | Purpose |
|---|---|---|
| `@googleapis/youtube` | 20.0.0 | YouTube Data API v3 client |
| `@googleapis/drive` | 8.0.0 | Google Drive API v3 client |
| `google-auth-library` | 9.0.0 | OAuth 2.0 authentication |
| `@1password/sdk` | 0.3.1 | Secure credential retrieval |
| `commander` | 14.0.3 | CLI argument parsing |
| `chalk` | 5.6.2 | Terminal colored output |
| `pino` | 10.1.0 | Structured JSON logging |
| `dotenv` | 17.2.3 | Environment variable loading (dev) |