Bynder Asset Uploader Technical Documentation

Overview

The Bynder Asset Uploader by Lettuce Commerce is a file-driven app that uploads remote assets into Bynder in bulk. It is designed for users who already have a Bynder account connected in the platform and want to create Bynder media records from a spreadsheet rather than uploading files manually one by one.

The app accepts a .csv or .xlsx input file. Each row represents one asset to upload. At minimum, each row must include a filename and a url. Optional columns can be used to set core Bynder fields such as asset name, tags, copyright, audit information, public visibility, as well as customer metaproperties. The uploader processes assets in parallel, writes a status back for every row, and attaches a generated results file to the run.

Primary Use Case

This app is intended for bulk onboarding of assets that already exist at accessible remote URLs. Instead of uploading a local binary directly through the UI, the system retrieves each source asset from the provided URL, stages it through Bynder's upload endpoints, finalizes the import, and then saves the media record with optional metadata.

Prerequisites

Before a user can start a run, the following conditions must be true:

- The user must be registered in Lettuce's Iceberg platform.
- A Bynder account must be connected for the current user.
- The user must upload a valid .csv or .xlsx file.
- The spreadsheet must include the required headers:
    - filename, url

If no Bynder account is connected, the UI blocks execution and prompts the user to connect Bynder first. The app uses the user's current Bynder instance as the target environment for the run.

File & Upload Requirements

- Input File Format Supported File Types
    - csv
    - xlsx
- Required columns:
    - filename: the file name to use during the Bynder upload process
    - url: the remote source URL where the asset binary can be downloaded
- Optional standard columns:
    - name: display name for the media record; if omitted, filename is used
    - tags: tag string sent to Bynder
    - copyright
    - audit (0 or 1, 1 being true, if uploaded assets should automatically be delivered into the waiting room instead of the asset bank)
    - isPublic: visibility flag passed through to Bynder (0 or 1, 1 being true)
- Optional metaproperty columns:
    - Any column prefixed with “metaproperty.” is treated as a custom Bynder metaproperty input.
    - Example:
        - metaproperty.Brand
        - metaproperty.Product Type
    - At runtime, the app strips the “metaproperty.” prefix, normalizes the remainder, looks up the corresponding metaproperty in Bynder, and sends it in the save request.

Processing Flow

The uploader follows this sequence for each valid row:

- Parse the uploaded spreadsheet.
- Validate that required headers exist.
- Transform row data:
    - Trim blank keys
    - Normalize empty strings to nil
    - Collect “metaproperty.*” columns into a dedicated metadata object
- Queue rows for processing.
- Upload assets in parallel, using batches of up to 5 concurrent uploads.
- For each asset:
    - Request the closest Bynder upload endpoint

    - Initialize the upload in Bynder
    - Retrieve the source asset from the provided url
    - Split the file into chunks when needed
    - Upload each chunk to the returned storage endpoint
    - Register each uploaded chunk with Bynder
    - Finalize the upload
    - Poll Bynder until the import is complete
    - Save the media record and attach metadata
    - Update run progress after each batch.
- Generate and attach an output status file containing the original row data plus processing results.

Chunking and Performance

The app uploads files in chunks of 10 MB. If the remote file size can be determined through an HTTP HEAD request, the uploader calculates the number of chunks before transfer begins. If size cannot be determined, it falls back to a single-chunk upload attempt.

Parallelism is fixed at 5 concurrent uploads. This gives the app reasonable throughput while still allowing per-batch progress reporting.

Metaproperty Handling

Metaproperties are resolved dynamically against the target Bynder environment. The app calls Bynder's metaproperties endpoint and builds a lookup keyed by normalized metaproperty name and label. This means the spreadsheet can reference either the technical name or the label, if it matches a metaproperty available in Bynder.

If a metaproperty is not found:

- The asset can still be saved
- The run marks the row as partial_success
- The missing metaproperty names are recorded in the output

This behavior is important because metadata mismatches do not necessarily block the entire upload.

Run Status and Output

Each processed row ends in one of three states:

- Saved: asset uploaded and saved successfully

- Partial_success: asset saved, but one or more metaproperties were not found and were skipped
- failed: upload or save process failed

At the end of the run, the app stores summary counts in the run metadata:

- Successful assets
- Partial-success assets
- Failed assets

A generated results file is attached to the run. That file contains:

- All original input columns
- Any added processing columns such as bynder_media_id
- Missing_metaproperties_count
- Missing_metaproperties
- Status
- Error

This output file is the primary audit artifact for operational review.

Error Handling

The uploader fails fast on structural input issues, such as:

- Missing required headers
- Unsupported file extension

At the row level, failures are isolated where possible. A single asset failure does not stop the full run. Typical row-level failures include:

- Source file download failure
- Empty source chunk
- S3 chunk upload failure
- Chunk registration failure
- Upload finalization failure
- Upload polling timeout
- Media save failure
- Missing mediaid in Bynder response

When an exception occurs for a specific row, the row is marked failed and the error message is written to the output file.

Security and Integration Notes

The app authenticates against Bynder using the user's connected Bynder instance. The Bynder client derives:

- base_url from the selected/current Bynder instance
- OAuth access token from the stored Bynder connection

The uploader sends the token as a bearer token on Bynder API requests. The source asset itself is downloaded from the external url provided in the spreadsheet, so those URLs must be reachable by the application at runtime.