Bynder -> InRiver Technical Overview

Summary

This integration pulls approved assets from Bynder, transforms them into an InRiver Content Onboarding CSV, and imports them into InRiver as Resource records. It is implemented in Iceberg as an InRiver job type that reuses the platform's existing Bynder asset retrieval patterns and adds an InRiver-specific delivery layer.

The current implementation is designed around InRiver Content Onboarding rather than direct entity-by-entity API updates. This gives us deterministic batch imports, native Resource upsert behavior, and support for InRiver's filename-based auto-linking to Products or Items.

Runtime Flow

At runtime, the integration executes the following sequence:

1. Iceberg starts the configured integration run.
2. The job class initializes:
   - the Bynder client
   - the InRiver Content Onboarding client
3. The job fetches assets from Bynder using the configured `client_metadata_query` and any run-time filters such as `last_run_timestamp`.
4. Each Bynder asset is transformed into a single CSV row matching the InRiver `SourceImporter` mapping.
5. Iceberg generates a temporary CSV file and attaches it to the run as an output artifact.
6. Iceberg uploads the CSV to the configured InRiver `SourceImporter` using the Content Onboarding `landing/upload` endpoint.
7. Iceberg polls the resulting `runImportId` until InRiver returns a terminal status.
8. The run stores a structured summary including assets processed, resources added or updated, links added or updated, warnings, errors, and the InRiver import ID.
9. Iceberg rolls those run results into cumulative analytics shown on the integration dashboard.

Required Configuration

Client Credentials

The integration expects both Bynder and InRiver credentials in `client_credentials`:

```
{
  "bynder_base_url": "https://<tenant>.bynder.com",
  "bynder_api_key": "<bynder-api-key>",
  "inriver_base_url": "https://onboardinguse.productmarketingcloud.com/api/v1",
  "inriver_api_key": "<inriver-onboarding-api-key>"
}
```

Core Settings

The most important job settings are:

```
{
  "source_importer_id": "<inriver-source-importer-id>",
  "integration_model_name": "INTEGRATION_MODEL_BYNDER_ASSETS",
  "filename_source_metafield": "property_InRiver_Item_ID",
  "name_source_metafield": "name",
  “media_type_value”: “Image”,
  // or “media_type_source_metafield”: “property_value”,
  “image_angle_default”: ”Hero”,
  // or “image_angle_source_metafield”: “property_value”,
  "client_metadata_query": {
        "property_Send_to_Inriver": "Yes"
  },
  "importable_metadata": {
        "property_Lettuce_Variety": "ResourceLettuceVariety"
  }
}
```

These settings drive the import as follows:

- source_importer_id
  - identifies the InRiver SourceImporter that receives the CSV
- integration_model_name
  - documents which InRiver integration mapping the importer is based on
- filename_source_metafield
  - controls the ResourceFilename value used by InRiver for auto-linking
- name_source_metafield
  - controls the ResourceName value shown in InRiver
- client_metadata_query
  - scopes which Bynder assets are eligible for export
- importable_metadata
  - maps Bynder metaproperties to additional InRiver CSV columns

CSV Mapping Behavior

Each Bynder asset is converted into a CSV row containing the InRiver columns expected by the configured importer. The integration currently populates:

- ResourceExternalAssetId
  - Bynder asset UUID
- ResourceFilename
  - derived from filename_source_metafield
- ResourceName
  - derived from name_source_metafield
- ResourceMediaType
  - derived from “media_type_value” or “media_type_source_metafield”
- ResourceImageAngle
  - derived from “image_angle_default” or “image_angle_source_metafield”
- sys_MediaUrls
  - public Bynder asset URL
- additional mapped columns
  - from importable_metadata

filename_source_metafield supports both a single field and concatenated values. This allows customers to build a linkable filename from multiple Bynder fields when InRiver auto-linking requires a compound identifier.

Examples:

```
"filename_source_metafield": "property_InRiver_Item_ID"
```

```
"filename_source_metafield": {
  "values": [
        "property_InRiver_Item_ID",
        "property_otherValue",
        "name"
  ],
  "concatenation_character": "_"
}
```

InRiver Import Behavior

The integration uses POST /landing/upload/{sourceImporterId} on the Content Onboarding API. A successful upload returns a runImportId, which Iceberg then polls until the import completes or fails.

InRiver behavior confirmed during testing:

- Resources can be upserted by external asset ID
- repeated imports update existing Resources instead of creating duplicates
- ResourceFilename can be used for automatic Product or Variation linking
- the uploaded file name and import name appear in InRiver workareas

This means the integration is effectively batch-upsert behavior rather than create-only behavior.

Analytics and Run Outputs

Each run stores:

- generated CSV file
- processed asset count
- Resources added or updated
- links added or updated
- warning count
- error count
- InRiver runImportId

The analytics view currently focuses on:

- total assets sent
- assets sent in the last 30 days
- daily trend for assets sent

These metrics are derived from completed non-dry-run job runs.