Function 3: Export Set to DCAP01 CSV

Overview

Function 3 exports bibliographic records from Alma to a CSV file containing comprehensive Dublin Core metadata. This function is designed to extract detailed metadata from Alma bib records and format it according to the DCAP01 (Dublin Core Application Profile) specification used by Digital Grinnell.

What It Does

This function takes a loaded set of bibliographic records (MMS IDs) and exports all their Dublin Core metadata to a timestamped CSV file. The export includes 62 different metadata fields, providing a complete snapshot of the records' descriptive metadata.

Key Features

- **Batch API Processing**: Uses efficient batch API calls (100 records per request) instead of individual calls
- **Comprehensive Metadata**: Extracts 62 different Dublin Core and custom fields
- **Progress Tracking**: Shows real-time progress during export
- **Automatic Filename**: Generates timestamped filenames (`alma_export_YYYYMMDD_HHMMSS.csv`)
- **Error Resilience**: Continues processing even if individual records fail

CSV Output Structure

Column Headings (62 fields)

The exported CSV includes the following columns:

Identification & Administrative

- `group_id` - Collection grouping identifier
- `collection_id` - Collection identifier
- `mms_id` - Alma MMS ID (unique record identifier)
- `originating_system_id` - Original system identifier
- `compoundrelationship` - Compound object relationship (custom field)

Title Fields

- `dc:title` - Primary title
- `dcterms:alternative` - Alternative titles
- `oldalttitle` - Legacy alternative title field

Identifier Fields

- `dc:identifier` - General identifiers (semicolon-separated if multiple)
- `dcterms:identifier.dcterms:URI` - URI/URL identifier

Content Description

- `dcterms:tableOfContents` - Table of contents
- `dc:creator` - Creator/author
- `dc:contributor` - Contributors
- `dc:description` - General description
- `dcterms:abstract` - Abstract/summary
- `dcterms:provenance` - Provenance information
- `dcterms:bibliographicCitation` - Citation information

Subject Fields

- `dc:subject` - Primary subject
- `dcterms:subject.dcterms:LCSH` (×12) - Library of Congress Subject Headings (up to 12)

Publication Information

- `dcterms:publisher` (×2) - Publisher information
- `dc:date` - General date
- `dcterms:created` - Creation date
- `dcterms:issued` - Issue/publication date
- `dcterms:dateSubmitted` - Submission date
- `dcterms:dateAccepted` - Acceptance date

Format & Type

- `dc:type` - Resource type
- `dc:format` - Format information
- `dcterms:extent` (×2) - Physical extent/size
- `dcterms:medium` - Physical medium
- `dcterms:format.dcterms:IMT` - Internet Media Type
- `dcterms:type.dcterms:DCMIType` - DCMI Type vocabulary

Language & Relations

- `dc:language` - Language(s)
- `dc:relation` - Related resources
- `dcterms:isPartOf` (×3) - Parent collections/series

Coverage

- `dc:coverage` - General coverage
- `dcterms:spatial` - Geographic coverage
- `dcterms:spatial.dcterms:Point` - Geographic point
- `dcterms:temporal` - Temporal coverage

Rights & Source

- `dc:rights` - Rights statements
- `dc:source` - Source information

Custom Fields

- `bib custom field` - General custom field
- `googlesheetsource` - Google Sheets source reference (custom)
- `dginfo` - Digital Grinnell information (custom)

Digital Representation Fields

- `rep_label` - Representation label
- `rep_public_note` - Public note
- `rep_access_rights` - Access rights
- `rep_usage_type` - Usage type
- `rep_library` - Library
- `rep_note` - Internal note
- `rep_custom field` - Representation custom field

File Information

- `file_name_1` - First file name
- `file_label_1` - First file label
- `file_name_2` - Second file name
- `file_label_2` - Second file label

How It Works

Prerequisites

1. **Load a set of records** using one of these methods:
    - Load Set by ID (Function loads members from an Alma set)
    - Load MMS IDs from CSV (Imports MMS IDs from a CSV file)
2. **Verify the set is loaded** - The status area will show the number of loaded members

Execution Steps

1. **Click Function 3** from the function dropdown
2. **Function executes automatically** (no confirmation required - this is read-only)
3. **Monitor progress** via the progress bar and status updates
4. **Locate output file** in the project directory when complete

Processing Flow

1. Initialization

- Generates timestamped filename: `alma_export_YYYYMMDD_HHMMSS.csv`
- Opens CSV file for writing

- Writes column headers

2. Batch Fetching

- Divides MMS IDs into batches of 100
- Makes batch API calls to Alma: `/almaws/v1/bibs?mms_id=ID1,ID2,...,ID100`
- Logs API efficiency (e.g., "10 batch calls vs 1000 individual calls")

3. Metadata Extraction

For each record:

- Parses the `anies` field (contains Dublin Core XML)
- Extracts fields using namespace-aware XML parsing
- Handles both `dc:` (http://purl.org/dc/elements/1.1/) and `dcterms:` (http://purl.org/dc/terms/) namespaces
- Extracts custom fields from institution-specific namespace

4. Data Mapping

- Maps extracted values to CSV columns
- Handles multiple values (joins with semicolons)
- Fills empty fields with blank strings
- Writes each record as a CSV row

5. Completion

- Displays summary: "X succeeded, Y failed"
- Shows filename in status area
- Logs API efficiency statistics

Technical Details

API Endpoints Used

Batch Bib Records Endpoint:

```
GET /almaws/v1/bibs?mms_id={comma_separated_ids}&view=full&expand=None
```

Parameters:

- `mms_id`: Comma-separated list of up to 100 MMS IDs
- `view=full`: Returns complete record data
- `expand=None`: No additional expansion
- `apikey`: API authentication

XML Parsing

The function uses Python's `xml.etree.ElementTree` with namespace support:

```
namespaces = {
    'dc': 'http://purl.org/dc/elements/1.1/',
    'dcterms': 'http://purl.org/dc/terms/'
}
```

Custom fields use the institution-specific namespace:

```
grinnell_ns = f"http://alma.exlibrisgroup.com/dc/{originating_system}"
```

Multiple Value Handling

When a field has multiple values:

- Values are joined with `"; "` (semicolon-space separator)
- Example: `"Subject 1; Subject 2; Subject 3"`

For repeated column names (like `dcterms:subject.dcterms:LCSH`):

- First value goes in first column
- Subsequent values go in subsequent columns
- Empty if no value available

Performance Optimization

Batch API Approach:

- **Old method**: 1000 records = 1000 API calls
- **New method**: 1000 records = 10 batch API calls
- **Savings**: 99% reduction in API calls

Benefits:

- Faster execution
- Reduced API quota usage
- Lower network overhead
- Better rate limit compliance

Output Example

The CSV file will look like this:

```
group_id,collection_id,mms_id,originating_system_id,compoundrelationship,dc
:title,...
```

```
,,991234567890104641,grinnell:12345,,Title of the Item,...
,,991234567890204641,grinnell:12346,isPartOf:parent123,Another Title,...
```

Use Cases

1. Metadata Analysis

Export records to analyze patterns in metadata:

- Count which fields are populated
- Identify records missing required fields
- Review subject heading usage

2. Batch Metadata Review

Create snapshots for manual review:

- Export a set before bulk edits
- Compare before/after states
- Share with catalogers for review

3. Integration with Other Tools

Use exported data in:

- Spreadsheet applications (Excel, Google Sheets)
- Data analysis tools (Python pandas, R)
- Digital collection platforms
- Metadata quality assessment tools

4. Preparation for Function 4

Export records, then filter for specific criteria:

1. Function 3: Export full set
2. Function 4: Filter for records 95+ years old
3. Result: Subset of public domain candidates

5. Documentation & Reporting

Create metadata reports:

- Collection statistics
- Metadata completion rates
- Rights statement distribution
- Date range analysis

Error Handling

Common Issues

No set loaded:

- **Error**: "Please load a set first"
- **Solution**: Use Load Set by ID or Load MMS IDs from CSV first

API failures:

- Failed records are logged with error details
- Batch continues processing remaining records
- Summary shows count of failures

Invalid MMS IDs:

- Records not found are logged as warnings
- Counted in failure statistics
- Does not stop batch processing

Recovery

If export fails:

- Check log file in `logfiles/` directory for details
- Verify API key has "Bibs" read permissions
- Ensure network connectivity to Alma API
- Try smaller batch (use Limit field)

Performance Considerations

Large Sets

For sets with thousands of records:

- **Progress updates**: Every 50 records logged
- **Progress bar**: Real-time percentage completion
- **Memory usage**: Processes in batches to minimize memory footprint
- **Time estimate**: ~1000 records in ~30 seconds (depends on network)

API Rate Limits

The batch approach helps stay within Alma API limits:

- Standard limit: 25 requests/second
- Batch processing: Reduces request count by 99%
- Large sets complete without hitting limits

Output File Management

File Naming Convention

```
alma_export_YYYYMMDD_HHMMSS.csv
```

Example: `alma_export_20251119_154447.csv`

- Created: November 19, 2025 at 3:44:47 PM

File Location

- Saved in project root directory
- Not automatically deleted
- Included in `.gitignore` pattern (`*_export_*.csv`)

File Retention

- Keep exports for documentation purposes
- Use for before/after comparisons
- Archive older exports periodically

Best Practices

1. **Test with small sets first**: Use the Limit field to process 10-20 records initially
2. **Review sample output**: Open CSV and verify column mapping meets expectations
3. **Monitor progress**: Watch for errors in the log panel during export
4. **Save strategically**: Export before major batch operations for backup
5. **Clean up old exports**: Periodically remove outdated CSV files

Integration with Other Functions

Function 1: Fetch and Display XML

- Use to examine individual record structure before export
- Verify which fields are populated
- Understand namespace usage

Function 2: Clear dc:relation Collections

- Export before bulk deletion for backup
- Export after to verify results

Function 4: Filter CSV for Records 95+ Years Old

- Function 3 creates the input file
- Function 4 filters it
- Common workflow for rights review

Function 6: Replace dc:rights

- Export before batch rights updates
- Export after to document changes
- Compare CSVs to verify updates

Limitations

- **Read-only**: This function does not modify Alma records
- **No representation data**: File fields currently not populated (placeholders for future)
- **Subject limit**: Maximum 12 LCSH subject headings exported
- **Flat structure**: Complex nested metadata flattened to CSV format
- **No binary data**: Images, PDFs, etc. not included (metadata only)

Related Documentation

- **Alma Bibs API**: https://developers.exlibrisgroup.com/alma/apis/bibs/
- **Dublin Core Metadata**: https://www.dublincore.org/specifications/dublin-core/
- **DCMI Metadata Terms**: https://www.dublincore.org/specifications/dublin-core/dcmi-terms/