Function 9: Validate Handle URLs and Export Results

Overview

Function 9 validates Handle System URLs in bibliographic records by testing each URL's accessibility and exporting the results to a CSV file. This function is essential for quality assurance, identifying broken links, and maintaining the integrity of persistent identifiers in your digital collections.

What It Does

This function processes all records in a loaded set and:

- Extracts Handle URLs (dc:identifier starting with `http://hdl.handle.net/`)
- Tests each Handle URL with an HTTP HEAD request
- Records the HTTP status code (200, 404, etc.)
- Follows redirects to capture the final destination URL
- Validates that the redirect URL contains the correct MMS ID
- Queries Primo API to retrieve and compare titles (when MMS ID matches)
- Exports results to a CSV file with Handle, title, status, validation, and title comparison information
- Identifies broken or problematic links for remediation

Key Features

- **HTTP validation**: Tests actual URL accessibility
- **Status reporting**: Records HTTP status codes (200, 404, 301, 500, etc.)
- **Redirect tracking**: Captures final destination URL after following redirects
- **MMS ID verification**: Confirms the Handle redirects to the correct record
- **Primo API integration**: Queries Primo to get the title from the discovery system
- **Title comparison**: Compares Alma dc:title with Primo display title
- **Error detection**: Identifies timeouts and connection errors
- **Title inclusion**: Includes dc:title for context
- **CSV export**: Easy-to-analyze spreadsheet format with 8 columns
- **Batch processing**: Efficient API calls for Alma records (100 records per call)
- **Progress tracking**: Real-time progress updates
- **Filter-friendly output**: Easy to find problems (filter by status code, MMS ID match, or title match)
- **Kill switch support**: Can interrupt long-running validations

The Need for This Function

Handle System Quality Assurance

Handle URLs are persistent identifiers meant to provide stable, long-term access to digital objects. However:

- **Migration issues**: URLs may break during system migrations
- **Configuration errors**: Incorrect Handle server settings
- **Content removal**: Objects deleted but Handles remain

- **Server problems**: Handle resolver or target server down
- **Redirect issues**: Unexpected redirects or moved content

Function 9 helps identify:

- 404 Not Found: Handle exists but target missing
- 500 Server Error: Handle resolver or target server problems
- Timeouts: Network or server performance issues
- Connection errors: DNS or routing problems
- Unexpected redirects: Changed or moved content

Proactive Link Checking

Rather than waiting for users to report broken links:

- **Regular audits**: Run monthly or quarterly
- **Post-migration validation**: Verify all Handles after system changes
- **Quality metrics**: Track link health over time
- **Remediation planning**: Prioritize fixes based on error types

How It Works

Step-by-Step Process

1. **Load Set**: User loads a set of records to validate

2. **Extract Handles**: For each record:

    - Fetch bibliographic record from Alma
    - Parse Dublin Core metadata
    - Find dc:identifier starting with `http://hdl.handle.net/`
    - Extract dc:title for context

3. **Test Handle URL**:

    - Send HTTP HEAD request to Handle URL
    - Allow redirects (follow 301/302)
    - Record HTTP status code
    - If status is 200, send GET request to capture final URL
    - Extract final redirect destination URL
    - Check if MMS ID appears in the redirect URL
    - Set 10-second timeout
    - Map status code to message

4. **Validate MMS ID**:

    - For successful redirects (200 status):
        - Extract the final destination URL
        - Check if the MMS ID is contained in the URL
        - Mark as TRUE if MMS ID found, FALSE if not
    - For other status codes:

        - Mark as N/A (not applicable)

5. **Query Primo API for Title** (when MMS ID matches):

    - If MMS ID found in redirect URL (step 4 = TRUE):
        - Construct Primo API URL: `https://grinnell.primo.exlibrisgroup.com/primaws/rest/pub/pnxs/undefined/alma{MMS_ID}?vid=01GCL_INST:GCL&lang=en&lang=en`
        - Send GET request to Primo API
        - Parse JSON response
        - Extract title from `pnx.display.title[0]`
        - Compare with dc:title from Alma record (case-insensitive)
        - Mark as TRUE if titles match, FALSE if different
    - If MMS ID not found or API error:
        - Mark as N/A

6. **Export Results**:

    - Write CSV with 8 columns:
        - MMS ID
        - Handle URL
        - dc:title
        - HTTP Status Code
        - Status Message
        - Final Redirect URL
        - Returned Correct MMS ID
        - Titles Match!

7. **Progress Updates**:

    - Update progress bar after each record
    - Log batch completion
    - Show final statistics with status code counts

HTTP Request Details

Request Type: HEAD followed by GET for successful responses, plus Primo API query

- **HEAD request**: Gets headers only (faster, less load)
    - Determines HTTP status code
    - Checks if Handle resolves
    - Sufficient for error detection
- **GET request**: For 200 status codes only
    - Captures final redirect URL
    - Verifies MMS ID in destination URL
    - Used for validation, not content parsing
- **Primo API request**: When MMS ID matches
    - Queries Primo's public REST API
    - Retrieves JSON record with display fields

- Extracts title for comparison with Alma metadata

Primo API Integration

When a Handle successfully resolves (200 status) and the redirect URL contains the correct MMS ID, Function 9 makes an additional API call to Primo:

API Endpoint Pattern:

```
https://grinnell.primo.exlibrisgroup.com/primaws/rest/pub/pnxs/undefined/alma{MMS_ID}?vid=01GCL_INST:GCL&lang=en&lang=en
```

Example:

```
https://grinnell.primo.exlibrisgroup.com/primaws/rest/pub/pnxs/undefined/alma991011506418804641?vid=01GCL_INST:GCL&lang=en&lang=en
```

Response Format: JSON object with PNX (Primo Normalized XML) structure

```
{
  "pnx": {
    "display": {
      "title": ["Data Repository for Reproducible Research"],
      "creator": [...],
      "type": [...]
    },
    "search": {...},
    "control": {...}
  }
}
```

Title Extraction: The function extracts the title from `pnx.display.title[0]` and compares it with the `dc:title` from the Alma bibliographic record. This verifies that:

1. The Handle points to the correct MMS ID
2. The Primo discovery system has the same title as Alma
3. Metadata synchronization between Alma and Primo is working correctly

Why Not Extract Page Titles?

An earlier version of this function attempted to extract and compare page titles from the returned HTML. This approach was abandoned because:

1. **JavaScript-Rendered Pages**: Grinnell's Handle URLs redirect to Primo (Ex Libris discovery system), which is a JavaScript-based single-page application. The HTML returned by a simple HTTP request contains empty `<title>` tags that get populated by JavaScript after the page loads:

```
<title id="primoExploreTitle"></title>
```

2. **Empty Meta Tags**: Meta tags that might contain title information are also empty in the initial HTML:

```
<meta id="ogTitle" property="og:title" content="">
```

3. **Would Require Browser Automation**: To extract dynamically-loaded titles would require:
   - Headless browser (Selenium, Playwright, Puppeteer)
   - JavaScript execution
   - Much slower processing (~10-30 seconds per URL)
   - Significantly more complex code
   - Higher resource usage

4. **Better Alternative**: Instead, the function now verifies that the Handle redirects to a URL containing the correct MMS ID. Grinnell's Handle URLs redirect to Primo with this pattern:

```
https://grinnell.primo.exlibrisgroup.com/discovery/fulldisplay/alma991
011506418804641/01GCL_INST:GCL
```

   The presence of the MMS ID in the URL confirms the Handle points to the correct record.

Redirects: `allow_redirects=True`

- Follows 301/302 redirects automatically
- Reports final status code after redirect chain
- Captures final destination URL

Timeout: 10 seconds

- Prevents hanging on slow servers
- Recorded as "Timeout" if exceeded

Error Handling:

- Timeout: Status code 0, message "Timeout"
- Connection error: Status code 0, message "Connection Error"
- Other exceptions: Status code 0, message includes error details

Usage

Basic Validation

Step 1: Load Set

1. Enter set ID or load from CSV

    - Example: `7071087320004641` (DCAP01 set)
2. Click "Load Set"
3. Verify set loaded: "Set loaded: 2,847 records"

Step 2: Select Function

1. Open function dropdown
2. Select "Validate Handle URLs and Export Results"
3. Function 9 button becomes active

Step 3: Execute Validation

1. Click Function 9 button
2. Progress bar appears
3. Note warning: "⚠️ This will make HTTP requests to each Handle URL"
4. Watch progress: "Validated 1 of 2,847 records"
5. Wait for completion

Important: This function makes external HTTP requests, so:

- **Slower than other exports**: ~1-2 seconds per Handle
- **Network dependent**: Requires internet connectivity
- **Respectful**: Uses HEAD requests to minimize server load

Step 4: Locate Output File

1. Check CABB project directory
2. Find file: `handle_validation_YYYYMMDD_HHMMSS.csv`
3. Example: `handle_validation_20241204_143022.csv`

Step 5: Analyze Results

1. Open CSV in spreadsheet application
2. Filter by "HTTP Status Code" column
3. Find problems:
    - Status code 404: Broken links
    - Status code 500: Server errors
    - Status code 0: Timeouts/connection errors
4. Check "Returned Correct MMS ID" column:
    - FALSE: Handle points to wrong record (critical!)
    - TRUE: Handle correctly redirects
    - N/A: Could not verify (due to error)
5. Check "Titles Match!" column:
    - FALSE: Primo title differs from Alma (may need re-publish)
    - TRUE: Titles match (expected)
    - N/A: Could not compare (Handle failed or API error)

Filtering for Problems

Find All Problems (Excel/Google Sheets):

1. Select all data
2. Create filter
3. Filter "HTTP Status Code" ≠ 200 OR "Returned Correct MMS ID" = FALSE OR "Titles Match!" = FALSE

Find Broken Links Only:

1. Filter "HTTP Status Code" column
2. Select only: 404, 500, 0
3. Shows unreachable Handles

Find Wrong Redirects:

1. Filter "Returned Correct MMS ID" column
2. Select only: FALSE
3. Shows Handles pointing to wrong records

Find Title Mismatches:

1. Filter "Titles Match!" column
2. Select only: FALSE
3. Shows records where Primo title differs from Alma
4. May indicate need to re-publish from Alma to Primo

Excel Filter:

1. Select column D (HTTP Status Code)
2. Filter → Number Filters → Not Equal to 200
3. Shows only problematic Handles

Google Sheets Filter:

1. Select all data
2. Data → Create a filter
3. Click filter arrow on "HTTP Status Code"
4. Uncheck 200
5. Shows only errors

SQL Query (if imported to database):

```sql
SELECT * FROM handle_validation
WHERE http_status_code != 200
ORDER BY http_status_code;
```

Output File Format

Filename Convention

Pattern: `handle_validation_YYYYMMDD_HHMMSS.csv`

Examples:

- `handle_validation_20241204_143022.csv`
- `handle_validation_20241204_090000.csv`

CSV Structure

Header Row:

```
MMS ID,Handle URL,dc:title,HTTP Status Code,Status Message,Final Redirect
URL,Returned Correct MMS ID,Titles Match!
```

Example Data Rows:

```
991234567890104641,http://hdl.handle.net/11084/12345,Historic Campus
Photo,200,OK,https://grinnell.primo.exlibrisgroup.com/discovery/fulldisplay
/alma991234567890104641/01GCL_INST:GCL,TRUE,TRUE
991234567890204641,http://hdl.handle.net/11084/12346,Student Yearbook
1925,404,Not Found,,N/A,N/A
991234567890304641,http://hdl.handle.net/11084/12347,Faculty
Portrait,301,Moved Permanently,,N/A,N/A
991234567890404641,http://hdl.handle.net/11084/12348,Annual
Report,0,Timeout,,N/A,N/A
991234567890504641,http://hdl.handle.net/11084/12349,Historic
Document,200,OK,https://grinnell.primo.exlibrisgroup.com/discovery/fulldisp
lay/alma991234567890504641/01GCL_INST:GCL,TRUE,FALSE
```

Column Details

Column	Description	Example Values
MMS ID	Alma record identifier	991234567890104641
Handle URL	Full Handle URL from dc:identifier	http://hdl.handle.net/11084/12345
dc:title	Title from Dublin Core metadata	Historic Campus Photo
HTTP Status Code	Numeric HTTP status	200, 404, 301, 500, 0
Status Message	Human-readable status	OK, Not Found, Timeout
Final Redirect URL	Destination URL after redirects (200 only)	https://grinnell.primo.exlibrisgroup.com/...
Returned Correct MMS ID	MMS ID found in redirect URL	TRUE, FALSE, N/A
Titles Match!	Alma title matches Primo title	TRUE, FALSE, N/A

Column	Description	Example Values
Handle URL	Full Handle URL from dc:identifier	http://hdl.handle.net/11084/12345
dc:title	Title from Dublin Core metadata	Historic Campus Photo
HTTP Status Code	Numeric HTTP status	200, 404, 301, 500, 0
Status Message	Human-readable status	OK, Not Found, Timeout

HTTP Status Codes

Success:

- **200 OK**: Handle resolves successfully

Redirects:

- **301 Moved Permanently**: Permanent redirect (expected for Handles)
- **302 Found**: Temporary redirect

Client Errors:

- **403 Forbidden**: Access denied
- **404 Not Found**: Handle exists but target missing (PROBLEM)

Server Errors:

- **500 Internal Server Error**: Target server error (PROBLEM)
- **502 Bad Gateway**: Proxy/gateway error
- **503 Service Unavailable**: Server temporarily down

Connection Issues:

- **0 Timeout**: Request took >10 seconds (PROBLEM)
- **0 Connection Error**: DNS, network, or routing issue (PROBLEM)

MMS ID Validation Results

The "Returned Correct MMS ID" column indicates whether the Handle redirects to the correct record:

TRUE: ✅ Handle correctly points to the record

- MMS ID found in the final redirect URL
- Example: Handle for 991011506418804641 redirects to URL containing "alma991011506418804641"
- This is the expected behavior

FALSE: ⚠️ Handle points to wrong record (SERIOUS PROBLEM)

- MMS ID NOT found in redirect URL
- Handle may point to different record
- Requires investigation and correction

- Could indicate Handle configuration error

N/A: Not applicable

- Status code was not 200 (OK)
- No redirect URL captured
- Cannot verify MMS ID for failed requests
- Examples: 404, 500, timeouts, connection errors

How to Find Mismatched Handles:

```
Filter column G (Returned Correct MMS ID) = FALSE
```

These require immediate attention as they point to wrong records.

Title Comparison Results

The "Titles Match!" column (column 8) indicates whether the Primo discovery system has the same title as the Alma bibliographic record:

TRUE: ✅ Titles match perfectly

- Primo API title matches dc:title from Alma
- Case-insensitive comparison (ignores uppercase/lowercase differences)
- Whitespace trimmed before comparison
- Indicates proper metadata synchronization
- This is the expected result

FALSE: ⚠️ Title mismatch detected (POTENTIAL PROBLEM)

- Primo has a different title than Alma
- Could indicate:
  - Outdated Primo index (hasn't synced recent Alma changes)
  - Title was edited in Alma but not yet published to Primo
  - Different title normalization/display rules
  - Data inconsistency requiring investigation
- Review these records to determine if re-publishing is needed

N/A: Not applicable

- MMS ID did not match (column 7 = FALSE)
- Handle did not resolve successfully (status ≠ 200)
- Primo API query failed or timed out
- No title field found in Primo JSON response
- Cannot compare titles when Handle or API fails

How to Find Title Mismatches:

```
Filter column H (Titles Match!) = FALSE
```

These may indicate records that need to be re-published from Alma to Primo.

Combined Problem Filter:

```
Filter: Status Code ≠ 200 OR MMS ID = FALSE OR Titles Match! = FALSE
```

This shows all records with any validation issue.

Use Cases

1. Post-Migration Link Validation

Scenario: After migrating from Digital Grinnell to Alma, verify all Handles work

Workflow:

1. Load DCAP01 set (all migrated digital objects)
2. Run Function 9
3. Review results:
    - Count of 200 OK responses
    - List of 404 errors
    - Any timeouts or connection errors
4. Investigate and fix problems
5. Re-run to verify fixes

Success Metrics:

```
Total Handles tested: 2,847
Status 200 (OK): 2,798 (98.3%)
Status 404 (Not Found): 42 (1.5%)
Status 500 (Error): 3 (0.1%)
Timeouts: 4 (0.1%)
```

2. Regular Quality Assurance Audits

Scenario: Monthly audit to catch link degradation

Workflow:

1. Schedule Function 9 run on first Monday of month
2. Export results to CSV
3. Compare with previous month:
    - New 404s: Investigate immediately
    - Resolved 404s: Document fix

    - Persistent 404s: Prioritize remediation
4. Generate monthly report
5. Track trends over time

Tracking Spreadsheet:

```
Month       | Total | 200 OK | 404   | 500 | Timeouts | % Success
------------|-------|--------|-------|-----|----------|----------
Jan 2024    | 2847  | 2798   | 42    | 3   | 4        | 98.3%
Feb 2024    | 2847  | 2815   | 28    | 1   | 3        | 98.9%
Mar 2024    | 2847  | 2835   | 10    | 0   | 2        | 99.6%
```

3. Troubleshooting User Reports

Scenario: User reports "Handle doesn't work"

Workflow:

1. Get MMS ID from user report
2. Create temporary set with just that MMS ID
3. Run Function 9
4. Check result:
    - 200: User error or caching issue
    - 404: Legitimate broken link
    - 301/302: Redirect (may be confusing user)
    - Timeout: Network/server issue
5. Take appropriate action
6. Document in ticket

4. Handle Server Configuration Testing

Scenario: Handle server upgraded, verify configuration

Workflow:

1. Before upgrade: Run Function 9, save baseline
2. Perform server upgrade
3. After upgrade: Run Function 9 again
4. Compare results:
    - All previous 200s still 200? ✓
    - New errors appeared? Investigate
    - Different redirect behavior? Review
5. Rollback if major issues found

5. Identifying Patterns in Broken Links

Scenario: Many 404s, need to find pattern

Workflow:

1. Run Function 9
2. Export CSV
3. Filter for 404 status
4. Analyze Handle URLs:
    - All from specific collection?
    - Similar ID patterns?
    - All migrated on same date?
5. Identify root cause
6. Implement systematic fix

Example Analysis:

```
import csv
import re

# Count 404s by Handle prefix
errors_by_prefix = {}

with open('handle_validation_20241204_143022.csv', 'r') as f:
    reader = csv.DictReader(f)
    for row in reader:
        if row['HTTP Status Code'] == '404':
            # Extract prefix: http://hdl.handle.net/11084/grinnell:NNNNN
            match = re.search(r'hdl\.handle\.net/\d+/([^:]+):', row['Handle
URL'])
            if match:
                prefix = match.group(1)
                errors_by_prefix[prefix] = errors_by_prefix.get(prefix, 0)
+ 1

print("404 errors by collection prefix:")
for prefix, count in sorted(errors_by_prefix.items(), key=lambda x: x[1],
reverse=True):
    print(f"  {prefix}: {count} errors")
```

6. Pre-Publication Validation

Scenario: Before publishing new collection, verify all Handles

Workflow:

1. Add new records to Alma
2. Create set of new records
3. Run Function 9
4. Verify all return 200
5. Fix any issues before publication
6. Re-validate
7. Publish when 100% success

Technical Details

HTTP Request Implementation

Python Code:

```python
import requests

try:
    response = requests.head(
        handle_url,
        allow_redirects=True,
        timeout=10
    )
    status_code = response.status_code
except requests.exceptions.Timeout:
    status_code = 0
    status_message = "Timeout"
except requests.exceptions.ConnectionError:
    status_code = 0
    status_message = "Connection Error"
```

Why HEAD not GET:

- HEAD requests only fetch headers, not content
- Much faster for large files (images, PDFs)
- Same status codes as GET
- Respectful to servers (less bandwidth)

Timeout Setting:

- 10 seconds chosen as balance
- Typical Handle resolution: <1 second
- Slow servers get up to 10 seconds
- Prevents indefinite hanging

Performance Considerations

Time Estimates:

- Alma API batch fetch: ~1 second per 100 records
- HTTP HEAD request: ~0.5-2 seconds per Handle
- Total per record: ~1-3 seconds
- 100 records: 2-5 minutes
- 1,000 records: 20-50 minutes
- 2,847 records: 1-2.5 hours

Network Impact:

- Outbound HTTP requests to Handle servers

- May trigger rate limiting on Handle resolver
- Respectful timing built in (sequential, not parallel)

Factors Affecting Speed:

- Network latency to Handle servers
- Handle server response time
- Number of redirects
- Timeouts (add 10 seconds each)

Error Handling

Record-Level Errors:

- Continue processing on individual failures
- Log errors for review
- Count as "failed" in statistics

Network Errors:

- Each Handle tested independently
- One timeout doesn't stop others
- Partial results still exported

Alma API Errors:

- Same as other functions
- 404/401/403 logged and counted

Interpreting Results

Good Results

Ideal Outcome:

```
All handles: 200 OK
or
Most handles: 200 OK
Some handles: 301 Moved Permanently (expected for Handle system)
```

Concerning Results

Requires Investigation:

404 Not Found:

- Handle resolves but target missing
- Object may have been deleted
- Handle configuration incorrect
- **Action**: Check target URL, restore object, or update Handle

500 Internal Server Error:

- Target server having problems
- Database connection issues
- Application error
- **Action**: Check server logs, contact IT

Timeout:

- Server very slow or unresponsive
- Network connectivity issues
- **Action**: Test manually, check server status

Connection Error:

- DNS resolution failed
- Network routing problem
- Server completely down
- **Action**: Check DNS, verify server running

Status Code Priorities

Fix Immediately:

1. 404 Not Found (broken user experience)
2. 500 Server Error (system problem)
3. Connection Error (complete failure)

Investigate Soon: 4. Timeout (performance issue) 5. 403 Forbidden (access problem)

Monitor: 6. 301/302 Redirect (expected, but verify targets) 7. 200 OK (success!)

Best Practices

Before Validation

1. **Test small set first**: 10-20 records to verify function works
2. **Check network**: Ensure stable internet connection
3. **Off-peak hours**: Run large sets during low-usage times
4. **Note baseline**: Document current known issues
5. **Backup results**: Keep previous validation CSVs for comparison

During Validation

1. **Monitor progress**: Check for unusual patterns (many timeouts)
2. **Don't interrupt**: Let process complete
3. **Check logs**: Review for systematic errors
4. **Network stability**: Ensure connection remains stable

After Validation

1. **Analyze results**: Filter by status code
2. **Prioritize issues**: 404s before timeouts
3. **Investigate patterns**: Group by collection, date, etc.
4. **Document findings**: Note any systemic issues
5. **Plan remediation**: Create fix schedule
6. **Track over time**: Compare with previous runs

Regular Auditing

1. **Monthly schedule**: First of each month
2. **Consistent sets**: Use same set for trend analysis
3. **Archive results**: Keep CSVs in dated folders
4. **Trend tracking**: Chart success rate over time
5. **Automated alerts**: Flag significant changes

Troubleshooting

All Handles Timeout

Symptoms: Every Handle shows "Timeout"

Possible Causes:

- Network connection down
- Firewall blocking outbound requests
- Handle resolver down

Solutions:

- Check internet connectivity
- Try accessing Handle URL in browser
- Check firewall rules
- Contact Handle system administrator

Many 404 Errors After Migration

Symptoms: Large percentage of 404s in validation

Possible Causes:

- Handle targets not updated during migration
- Objects not properly migrated
- Handle server configuration incorrect

Solutions:

- Compare Handle targets with actual object URLs
- Verify objects exist in new system
- Update Handle records to point to new URLs
- Contact migration team

Validation Very Slow

Symptoms: Taking much longer than expected

Possible Causes:

- Network latency
- Slow Handle servers
- Many redirects

Solutions:

- Run during off-peak hours
- Check network speed
- Use smaller batches
- Consider running overnight

Inconsistent Results

Symptoms: Same Handle returns different status on retests

Possible Causes:

- Intermittent server issues
- Load balancer behavior
- CDN caching variations

Solutions:

- Run validation multiple times
- Note inconsistent Handles
- Test manually at different times
- Contact server administrator

Comparison with Other Functions

Function 9 vs. Function 8

Aspect	Function 8	Function 9
Purpose	Export identifier fields	Validate Handle URLs
Output	4 columns (identifiers)	5 columns (Handle + status)
Speed	Fast (~30-45 min for 2,847)	Slow (~1-2.5 hours)
External requests	No	Yes (HTTP to Handles)
Use case	Identifier inventory	Link quality assurance

Use Together:

1. Run Function 8 first to get Handle inventory

2. Run Function 9 to validate those Handles
3. Cross-reference results

Function 9 vs. Manual Testing

Aspect	Manual Testing	Function 9
Speed	Very slow	Automated
Coverage	Sample only	Complete
Documentation	Informal notes	CSV export
Repeatability	Difficult	Easy (re-run anytime)
Trend analysis	Manual tracking	Compare CSV files

Integration with Other Functions

After Function 7 (Add Grinnell Identifiers)

If Function 7 added Grinnell:* identifiers, some records may still need Handles:

1. Run Function 8 to see Handle coverage
2. Run Function 9 to validate existing Handles
3. Identify records needing Handle registration
4. Register new Handles
5. Re-run Function 9 to verify

With Function 1 (View Single XML)

To investigate specific Handle issues:

1. Run Function 9, find problematic Handle
2. Copy MMS ID from validation CSV
3. Use Function 1 to view full record
4. Examine all dc:identifier fields
5. Check if Handle is correct in metadata

Before Major System Changes

Create validation baseline:

1. Run Function 9 before change
2. Perform system upgrade/migration
3. Run Function 9 after change
4. Compare CSV files
5. Identify any new issues caused by change

Related Documentation

- **Handle System**: https://www.handle.net/

- **HTTP Status Codes**: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status
- **Dublin Core dc:identifier**: https://www.dublincore.org/specifications/dublin-core/dcmi-terms/#identifier
- **Function 8**: Export Identifier CSV (companion function)

Version History

- **Initial Implementation**: December 2024
- **Purpose**: Handle URL quality assurance and broken link detection
- **Status**: Active, production-ready