Robert Feldman

FeldmanR@si.edu

Revised April 27, 2015

CDIS 2.0

Collections-DAMS Integration System - v2.0 Configuration and User Guide

# Introduction

The Collections-DAMS Integration System (CDIS) is a batch process which fully integrates the Smithsonian’s Institution’s central media repository (DAMS) with a CIS system (such as TMS). There are many museums (SI units) within the Smithsonian Institution, each of which catalog different types of media and each have varying ways of organizing that media. CDIS will be used to automate the integration of all of this data from the CIS with the DAMS. While this may sound like a complex task, CDIS 2.0 is fully configurable and is able to handle different scenarios for different units, and even different scenarios within a unit’s collection. Currently CDIS only supports image data.

There are three different configurable areas which alter the flow of data through CDIS. These are the config.ini file, the SQL-XML file and the enhanced CDIS table. The idea behind this design is to bring CDIS away from having unit specific coding, but rather have different options available to any group of a unit’s collection.

## Invoking CDIS

With CDIS 2.0, there are executable batch files which invoke the CDIS process. The batch file is a simple one line file that specifies the CDIS operation type, the proper jar file (executable java file), config.ini (configuration file) to use, and where to send any output. The batch file may be found on the Windows server in the CDIS directory, and is invoked by double clicking on the batch file.

The batch files can be executed manually, or put into window scheduler to run on an automated basis every night. It is recommended to put each units’ batch files into the windows scheduler, so all new additions and changes in the DAMS and/or the CIS are picked up automatically.

## Overview of config.ini, SQL-XML and CDIS table

The config.ini file contains database connection settings, general settings as well as very specific settings that alter the behavior of CDIS. Typically each unit will have their own CDIS.ini file, named appropriately for the unit (example NASM\_config.ini). This file will replace the old style config files. An important distinction is for CDIS 2.0 there will be only one config file per unit, rather than one config file for each operation type for each unit. This will eliminate the need for updating multiple files within a unit when a single setting is changed. Specific options for the config.ini are detailed in the many sections below. Also refer to appendix A for the sample config.ini file.

The SQL-XML file is contained in the conf directory along with the config.ini file. This file contains the queries that drive all of the basic CDIS operations. These operations and the function of each query is detailed in the sections that follow. Each unit will have their own SQL-XML file, named appropriately for the unit (example NASM\_SQL.xml). See Appendix C for a sample SQL-XML file.

With CDIS 2.0 there have been additional fields added to the CDIS table. The CDIS table can be manipulated to alter the flow of the CDIS process. This table is detailed in Appendix B.

# CDIS Operation Types

The CDIS batch process has been broken down into five operation types. The operation type is specified when the batch process is invoked. The operation types are as follows:

*IngestToCIS*

*IngestToDAMS*

*LinkToCIS*

*Sync*

*Thumbnailsync*

Each operation type is detailed in the following sections:

## IngestToCIS

The IngestToCIS operation type is used where the media image exists in DAMS, but the rendition is not present in the Collections system. The IngestToCIS operation type is used as the first step in the integration process for these types of image/renditions.  In this operation, a media record is created in the CIS. This is done by

1. Directly inserting rows to several tables into the CIS database. This will add the media record into the CIS, then
2. A stream is initiated in the CDIS java code to copy the thumbnail image from the DAMS to the Thumbnail BLOB in the newly created CIS media record. The transfer is performed byte by byte (with no alterations whatsoever).
3. The CDIS table is populated with the UOI\_ID (DAMS unique identifier), the rendition id of the newly created rendition in the CIS, the rendition number from CIS, and the objectID from the CIS.
4. The Source system Identifier in DAMS is updated with the RenditionNumber from the CIS showing that the rendition has been successfully linked with the image in DAMS.

### 2.1.1 IngestToCIS SQL-XML configuration Options

There exists full control over which images are to come from DAMS and be integrated with the collections system. The selection of DAMS images are configurable in the SQL.xml file.

In each SQL.xml file there should be an <ingestToCIS> tag (Refer to Appendix C). Within the <ingestToCIS> area there are several query tags. If the query type is “type="DAMSSelectList", then the query contained in this tag will be used to obtain the DAMS images that need to be incorporated into TMS. The query can be a based on Date, media name or any other criteria that can be selected from the DAMS database. The select list can be limited to as many records as desired. Upon execution of the IngestToCIS operation, CDIS will obtain that list of UOI\_IDs, and UANs from DAMS that could be integrated with the CIS.

Each of the selected UOI\_IDs and the UANS from the DAMSSelectList is then checked to see if there are already existing media renditions in the CIS for the one that is to be created. This is where the next query tag in the ingestToCIS area of the xml file is used (the CheckForExistingTMSRendition) query type tag. Typically, we do not wish to create renditions for one that is already there, and this query when properly configured will prevent that.

### 2.1.2 IngestToCIS config.ini Options

There are several settings to be found in the config.ini file which impact the ingestToCIS logic. They are listed as follows

A set of four options allows CDIS the ability to find the appropriate object to attach the new media to in the CIS in various ways. These are *mapFileNameToBarcode*, *mapFileNameToObjectNumber*, and *mapFileNameToObjectID*. Each unit may use one or more of these options in each batch (so multiple selections are possible).

*mapFileNameToBarcode:* When set to ‘true’, CDIS will interpret a portion of the DAMS image file name as a barcode number. CDIS will locate the appropriate object in the CIS to add the new rendition to based on a lookup of this barcode. When the mapFileNameToBarcode is set to ‘false’, the filename in DAMS is assumed to not be related to a barcode number, and there will be no lookup of the object based on the barcode number.

*mapFileNameToObjectNumber*: When set to ‘true’, CDIS will interpret part of the DAMS image file name as the ObjectNumber in the CIS. CDIS will locate the appropriate object in the CIS to add the new rendition to based on a lookup of this ObjectNumber.

*mapFileNameToObjectNumber*: When set to ‘true’, CDIS will interpret part of the DAMS image file name as the object ID in the CIS. CDIS will locate the appropriate object in the CIS to add the new rendition to based on a lookup of this ObjectNumber.

*locateByLetterRange:* When set to ‘true’, CDIS will attempt to find the object based on the letter range methodology. At times units have named the objects with letter ranges, and all renditions in that range will be mapped to the object (for example 2014-02-01b will be mapped to 2014-02-01ag)

*damsDelimiter*: Specifies the delimiter in the DAMS image filename. (for example for image filename 2014-25-02-01.tif the delimiter should be specified as ‘-‘. Using the correct delimiter will help CDIS to find the proper object to create the media on (when used in conjunction with *tmsDelimiter and imageObjectTrunc).*

*tmsDelimiter*: Specifies the delimiter in the TMS Rendition Number. Using the correct delimiter will help CDIS to find the proper object to create the new media on. Also when the tmsDelimiter is different from the DAMS delimiter, it will impact the naming of the new media in the CIS. For Example when the new media is created in the CIS for DAMS filename 2014-25-02-01.tif, the CIS rendition will be renamed 2014.25.02.01). This does not rename anything in the DAMS.

*imageObjectTrunc*: specifies the relationship between the CIS rendition and the CIS object number. This option works off the tmsDelimiter setting above. Negative numbers will truncate the Rendition number from the right the number of delimiters specified, and positive numbers from the left. For example if CDIS needs to find the object associated to TMS Rendition 2014.25.02.01 with an imageObjectTrunc of -1 and tmsDelimiter of ‘.’, then CDIS will attempt to create the media on object 2014.25.02 (removes everything after the last ‘.’ from the right to map to the opject.

*mediaFormatID: This option is expected to be added, and will be documented after testing is complete.*

*mediaTypeID: This option is expected to be added, and will be documented after testing is complete.*

*mediaStatusID: This option is expected to be added, and will be documented after testing is complete.*

*appendTimeToNumber:* adds a timestamp (time in hours, minutes and seconds) to the end of the newly created renditionNumber in the CIS. This option has been requested to add unique-ness to the RenditionNaming conventions.

## IngestToDAMS

In the cases where the media rendition exists in the CIS, but the image is not present in DAMS, the IngestToDAMS operation type is used as the first step in the integration process. In this operation, the new media is created in the DAMS database. The steps of this process include:

1. The media (such as an image file) is copied off the TMS media drive and placed into the CDIS work Folder location. This workfolder (as specified in the config.ini file) is located on the same server as hotfolder for the unit, and is typically named ‘TEMP-XFER’.
2. Once all files in the batch are copied to the workfolder location, the files are moved out of the workfolder and placed into the hotfolder location.
3. A zero-byte file with the name ‘ready.txt’ is created in the hotfolder directory.
4. An automated process outside of the scope of CDIS picks up this new media and will create the DAMS image record for these TMS renditions.

### 2.2.1 IngestToDAMS SQL-XML configuration Options

Upon execution of the IngestToDAMS operation, CDIS will generate a list of RenditionIDs and Filenames from TMS that need to be created in DAMS. This list of CIS renditions to be added to the DAMS with the IngestToDAMS operation are configurable in the SQL.xml file.

In the SQL-XML file, within the <ingestToDAMS> tag the query contained in the “type="TMSSelectList”, will be used to obtain the CIS records that need to be incorporated into DAMS. The query can be a based on any criteria that can be selected from the DAMS database. The select list can limited to as many records as you configure.

Each of the renditions from the TMSSelectList result set are checked against the other query tag in the ingestToCIS area of the xml file. This is the CheckForExistingDAMSRendition query type tag. This second query checks the DAMS database if there are already existing media renditions for the one that is to be created. It is important to check to see if there exist any DAMS renditions from the TMSSelectList are already in DAMS. We do not create a media rendition in TMS for one that is already there. See Appendix B for a Sample XML file showing these tags.

### 2.2.2 IngestToDAMS config.ini Options

There are several settings in the config.ini file that are used exclusively for the ingestToDams process. They are as follows:

workFolder: Indicates the work folder for media files that are identified as needing to be ingested into DAMS. To ensure quickest file copy speed possible, this workfolder should reside on the same server as the hotFolder.

hotFolderMaster: This setting indicates the server and folder that new media for DAMS is dropped off. This is typically the directory with the folder name ‘MASTER’.

## LinkToCIS

The linkToCIS operation type associates a DAMS image with a rendition in the CIS system. Once this association is performed they are said to be linked’. The linkToCIS process is to be used in two different scenarios. They are:

* Where the same rendition was created independently of each other in both the CIS and DAMS (outside of the CDIS process).
* Following ingestToDAMS (detailed in section 2.2), the linkToCIS process must be run to associate the newly created DAMS image to the TMS rendition. Note: While the linkToCIS must be run following the ingestToDAMS, the linkToCIS does *not* need to run following the ingestToCIS because linkToCIS is contained as part of the ingestToCIS.

The steps of the LinkToCIS operation type include:

1. The CDIS table is populated with the UOI\_ID (DAMS unique identifier), the rendition id of the newly created rendition in the CIS, the rendition number from CIS, and the objectID from the CIS.
2. The Source system Identifier in DAMS is updated with the RenditionNumber from the CIS showing that the rendition has been successfully linked with the image in DAMS.

The DAMS image and the collection system rendition must be linked if the metadata is expected to flow from the CIS to DAMS. The LinkToCIS process establishes this link.

### 2.3.1 LinkToCIS SQL-XML configuration Options

As with the other CDIS operation types, CDIS offers full control over how to map the DAMS image to the rendition in the CIS. This is configurable in the SQL.xml file within the <linkToCIS> tag.

Like the other execution types, within the <linkToCIS> tag there is several query tags. The query tag with the type of retrieveDamsImages indicates the starting point for the CDIS process to obtain a list of renditions to link to the CIS. This query will obtain a list of uoi\_ids and UANs that may possibly need to be linked.

Once this list of uoi\_ids and UANS is retrieved, the CIS searches for the corresponding record to link to by using the query in the checkAgainstCollections tag.

### LinkToCIS config.ini Options

There is one setting in the config.ini file that are used exclusively for the link process:

updateTMSThumbnail: When this is set to TRUE, the last step of the linking process will also include updating thumbnail in TMS with the thumbnail in DAMS. This will replace the existing thumbnail in TMS. This option is only to be used in special cases, and only with the SI unit’s consent.

setForDamsFlag: When set to TRUE, the CDIS process will update the checkbox ‘forDAMS’ in the CIS to true when the dams image is linked to the CIS rendition. In TMS this is the column ‘isColor’ in the mediaRenditions table.

## 2.4 Sync

The final step in the typical CDIS flow is the sync process.  This sync step includes two separate processes: a metadata sync, and the ‘IDS sync’ process. The metadata sync involves copying the metadata from the CIS and making that metadata available in the DAMS. The IDS sync process updates the CIS to ensure that the IDS derivative is viewable from the CIS.

### 2.4.1 Sync SQL-XML config.ini Options

CDIS collects metadata from the CIS. CDIS then inserts this metadata into the SI\_ASSET\_METADATA table in the DAMS database. Because each SI unit utilizes the CIS differently, and each unit expects different results in the DAMS, the metadata process has been configured separately for each unit. It is important to note that there is a core data model which must be followed when determining how to map the CIS metadata to DAMS. The Core data model specifies which fields in DAMS are visible in the Media Manager GUI, and also specifies guidelines that detail what the fields in the DAMS are used for.

The metadata mapping is configurable in the SQL.xml file. The metadata mapping is specified within the sync section of this file. Each unit can have one or many queries to obtain the required metadata. CDIS is designed to have all of the metadata mapping logic available in the SQL-XML file.

There are currently two metadata query types. "singleResult" and “cursorAppend”.

Most metadata queries are of the singleResult type. singleResult queries return a single row of metadata from the CIS. (One result from the CIS is mapped to one field in the DAMS).

The second type of query is the cursorAppend query. This type of query is used when there may be multiple results returned from the CIS that need to go into a single field in DAMS. This is useful where lists of things such as keywords and captions need to be retained. Several units indicated that they wish for a list of constituents in the CIS system as a single field (keyword) in DAMS. When the cursorAppend query is used, the delimiter is required to be specified in the SQL-XML file as well. For example if the CIS column has the following display\_names returned from a cursorAppend query with a delimiter of “, “:

Louis H. Draper

Miles Davis

Ron Carter

CDIS will make the corresponding column into Louis H. Draper, Miles Davis, Ron Carter.

### 2.4.2 Sync config.ini Options

*FlagforIDS:*  This option indicates whether or not IDS should be notified of the DAMS image changes following the sync. This is an important enhancement because IDS does not have to know of every metadata update. Unnecessarily notifying IDS of each metadata sync update has a potential of overloading the IDS system. There are currently four different options:

*default*: IDS should only be notified if there is a change in restrictions (size restriction or public access restrictions). This option will reduce the number of notifications sent to IDS, but also make sure IDS is notified when it needs to incorporate restriction changes.

*always*: IDS should be notified when any metadata sync is performed.

*never*: IDS should not be notified when the metadata sync is performed. This option can be used in new projects when IDS already has the required renditions.

*ifRestricted*: IDS should only be notified if the rendition is restricted in any manner (whether by size or by the IS\_RESTRICTED flag). This option is to be used when IDS already has all of the images, but in some cases may not have the restrictions found in the CIS.

### 2.4.3 The Sync Process and the CDIS table

The metadata sync process collects of a list of DAMS images to sync by determining which images have never been synced before, and also by detecting which renditions have been recently modified in the CIS. This is done by checking the metaDataSyncDate in the CDIS table. If the metaDataSyncDate is null, or the metadataSyncDate is older than the date of the last change in the CIS, the sync process will pick up the metadata from the CIS and perform a sync.

## ThumbnailSync

ThumbnailSync is more of a utility used in special cases, rather than part of the typical CDIS flow. The thumbnailSync updates the thumbnail in the CIS system with a small size image (or thumbnail) of what is found in DAMS. This can be useful in two different scenarios:

1. When the image was placed in DAMS independently of the rendition in TMS, outside of the CDIS process integration, the CIS thumbnail and the IDS image may offer different views of the same object. The thumbnail sync will ensure sure that the CIS thumbnail matches exactly what is in DAMS. This option is also available as part of the linkToCIS step when updateTMSThumbnail is set to true.

2. When the thumbnail image in the CIS database is null (empty), corrupted or otherwise problematic. There may also be legacy renditions in the CIS that do not have the thumbnail populated. These renditions can have their thumbnail refreshed in the CIS with the thumbnailSync operation.

Because this is outside the normal flow, thumbnailSync should only be used when need dictates, and with the SI units approval.

# 3. Other Global Configuration Settings

There are several other important settings in the config.ini file

Database settings:

* damsDriver : Used by java to connect to the DAMS database.
* damsUrl : Connection settings which allows java to connect to the DAMS Database.
* damsUser: Contains the userid for the application to connect to DAMS database.
* damsPass**:** Contains an encrypted password for the damsUser
* tmsDriver**:** Used by java to connect to the TMS database
* tmsUrl: Connection settings which allows java to connect to the TMS Database.
* tmsUser:Contains the userid for the application to connect to TMS database
* tmsPass:Contains an encrypted password for the TMSUser

Other confi.ini Settings used for multiple operation types

* siUnit: indicates the SI unit. The java application code does not use setting for logic processing, but does display the unit on email reports and in naming logfiles.
* IDSPathId: Relates to the PathID in TMS for IDS. The value is obtainable in the TMS MediaPaths table.
* emailReportTo: contains one or several email addresses to email reports upon completion of batch file execution.
* xmlSQLFile: points to the correct SQL-XML file.

# 4. Glossary

Barcode: A sequence number placed on an object at time it is photographed to be incorporated into Digital management systems. Barcodes are used to quickly assign identifiers to objects in rapid capture projects.

CDIS: Collections-DAMS Integration System. A batch process used to integrate images in the DAMS with the CIS.

CIS:

DAMS: Digital Asset Management System. The centralized repository for media belonging to all SI units.

Hot Folder: A location where media is placed where an automated process will pick up the media and add it to the DAMS system.

IDS: Image Delivery Service. Used to serve images to the public via the internet, and to provide internal SI image requests on demand.

Meta Data: Data that describes a specific object or rendition (creator or work, date of creation, etc).

NodeID:

Object ID: The unique identifier for the Object. Each object is assigned an objectID in the CIS.

Object Number: The SI unit specified name or cataloging number for a particular object in the CIS. Sometimes relates to the accession ID.

Rendition: A particular view of the object. Objects may have multiple renditions, each rendition corresponds to a different view of the object (top, bottom, side, etc…or in the case of manuscripts each rendition can be a different page.)

RenditionID: The unique identifier for the rendition. Each rendition is assigned a renditionID in the CIS.

RenditionNumber: The SI unit specified name or cataloging number for a particular rendition in the CIS. The RenditionNumber often contains the object Number along with extra information that specify the specific rendition.

UAN: The Unique Asset Name. The UAN is used by IDS

UOIID: The unique identifier for the DAMS image. Each image is assigned a renditionID in the DAMS.

**Appendix A: config.ini file**

# CDIS.ini file for National Museum of African American History and Culture (NMAAHC)

####################### Configuration for All Operation Types #########################

siUnit = NMAAHC

#### Database Connectivity section

damsDriver = oracle.jdbc.OracleDriver

damsUrl=jdbc:oracle:thin:@si-osmpd3.si.edu:1521:DAMS7

damsUser = towner

damsPass = XXXXXXX

tmsDriver = net.sourceforge.jtds.jdbc.Driver

tmsUrl=jdbc:jtds:sqlserver://160.111.103.118/AAHCTMS;instance=v4;

tmsUser = DAMSUSER

tmsPass = XXXXXXXX

#####

IDSPathId = 9

emailReportTo = FeldmanR@si.edu

xmlSQLFile = conf/NMAAHC\_SQL.xml

################# Configuration for CollectionsIngest Operation #################################

damsImageNameToTMSObjectTrunc = lastUnderscore

locateByBarcode = false

newRenditionNameFormat = underscoreToDot

tmsObjectNumberFormat = underscoreToDot

mediaFormatID = 20

mediaTypeID = 15

mediaStatusID = 0

####################### Configuration for DAMSIngest Operation #################################

hotFolderMaster = W:\\MASTER

workFolder = W:\\TEMP-XFER

############################## Configuration for Link Operation #################################

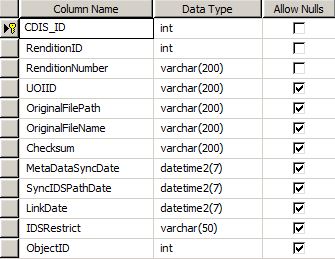
updateTMSThumbnail = false

############################## Configuration for Sync Operation #################################

#flag for IDS, legit values=always, never, ifRestricted, ifRestrictUpdated

flagForIDS=default

**Appendix B: CDIS table**

****

The CDIS table has two main purposes:

1. Tracks which images/renditions have been through CDIS and retains timestamps of when key events have last occurred. Vital tool for reporting or troubleshooting.
2. Used as an integral part of the logic of and processing of the TMS application.

*CDIS\_ID:* The unique sequence for a particular row in the CDIS table.

*RenditionID*: The renditionID from the CIS.

*RenditionNumber*: The RenditonNumber from the CIS

*UOIID*: Contains the UOI\_ID from DAMS for the particular media.

*OriginalFilePath*: Not currently in use by CDIS 2.0.

*OriginalFileName*: Not currently in use by CDIS 2.0.

*Checksum*: Not currently in use by CDIS 2.0.

*MetaDataSyncDate*: New for CDIS 2.0. This field holds the timestamp of the last metadata sync for this rendition/image. Important not only for tracking, but critical part of CDIS logic. Helps determine which renditions to sync.

*SyncIDSPathDate*: New for CDIS 2.0. This field holds the date that the rendition in TMS has had its filepath updated to point to the IDS derivative.

*LinkDate*: New for CDIS 2.0. This field holds the timestamp that the image in DAMS has been linked to the rendition in the CIS.

*IDSRestrict*: New for CDIS 2.0. Contains the restrictions for IDS. This will contain size restriction size for IDS (in which case it will be a numeric value indicating the maximum number of pixels in an image served to the public), as well as external restriction information. Retaining this information allows CDIS to resend the information to IDS only under certain conditions

*ObjectID*: New for CDIS 2.0. Contains the CIS ObjectID, or unique object identifier to which the Rendition belongs to. Established at the time of linking, and later used in the metadata sync.

**Appendix C: Sample XML file**

**<?xml version="1.0"?>**

**<!-- 1/2015 RFeldman -->**

**<!-- This is the SQL mapping for NMAAHC.**

**The key field is enclosed in '?' and is substituted within the java code for the**

**actual value.**

**-->**

**<NMAAHC>**

**<ingestToCIS>**

**<query type="DAMSSelectList">**

**select UOI\_ID, OWNING\_UNIT\_UNIQUE\_NAME**

**from SI\_ASSET\_METADATA**

**where UPPER(PUBLIC\_USE) = 'YES'**

**and SOURCE\_SYSTEM\_ID IS null**

**and UOI\_ID in (select UOI\_ID from NODES\_FOR\_UOIS where NODE\_ID = 116788)**

**order by OWNING\_UNIT\_UNIQUE\_NAME**

**</query>**

**<query type="CheckForExistingTMSRendition">**

**SELECT 'X'**

**WHERE NOT EXISTS (**

**SELECT 'For IDS file'**

**FROM MediaFiles mf**

**WHERE FileName = '?owning\_unit\_unique\_name?')**

**AND NOT EXISTS (**

**select 'Non-IDS match on Rendition-Number-Filename'**

**From MediaRenditions a,**

**MediaFiles b**

**where a.RenditionID = b.RenditionID**

**and a.isColor = 1**

**and 'NMAAHC-' + RenditionNumber =**

**REPLACE('?owning\_unit\_unique\_name?','\_','.'))**

**</query>**

**</ingestToCIS>**

**<ingestToDAMS>**

**<query type="TMSSelectList">**

**SELECT mr.RenditionID, mf.FileName**

**FROM MediaRenditions mr,**

**MediaFiles mf**

**WHERE mr.RenditionID = mf.RenditionID**

**AND mr.isColor = 0**

**AND NOT EXISTS (**

**SELECT 'X'**

**FROM CDIS**

**where CDIS.RenditionID = mr.RenditionID)**

**AND mr.RenditionID = 1077**

**</query>**

**<query type="CheckForExistingDAMSImage">**

**SELECT 'X'**

**FROM dual**

**WHERE NOT EXISTS (**

**SELECT 'X'**

**FROM si\_asset\_metadata si**

**WHERE OWNING\_UNIT\_UNIQUE\_NAME = ?fileName? )**

**</query>**

**</ingestToDAMS>**

**<linkToCIS>**

**<query type="RetrieveDamsImages">**

**select uoi\_id, owning\_unit\_unique\_name**

**from SI\_ASSET\_METADATA**

**where UPPER(PUBLIC\_USE) = 'YES'**

**and SOURCE\_SYSTEM\_ID IS null**

**and OWNING\_UNIT\_UNIQUE\_NAME like 'NMAAHC-%'**

**order by OWNING\_UNIT\_UNIQUE\_NAME**

**</query>**

**<query type="** **checkAgainstCollections">**

**select a.RenditionID, a.RenditionNumber**

**from MediaRenditions a,**

**MediaFiles b**

**where a.RenditionID = b.RenditionID**

**and a.isColor = '1'**

**and b.FileName = '?owning\_unit\_unique\_name?'**

**and b.PathID = 9**

**and not exists (Select 'X' from CDIS c where a.RenditionID = c.RenditionID)**

**and b.FileName like '%NMAAHC-%'**

**union**

**select a.RenditionID, a.RenditionNumber**

**From MediaRenditions a,**

**MediaFiles b**

**where a.RenditionID = b.RenditionID**

**and a.isColor = 1**

**and 'NMAAHC-' + RenditionNumber = REPLACE('?owning\_unit\_unique\_name?','\_','.')**

**and b.PathID = 6**

**and not exists (Select 'X' from CDIS c where a.RenditionID = c.RenditionID)**

**</query>**

**</linkToCIS>**

**<sync>**

**<query type="singleResult">**

**SELECT mm.Restrictions AS other\_constraints,**

**REPLACE(REPLACE (mr.Remarks,'[MAX IDS SIZE =',''),']','')**

**AS max\_ids\_size,**

**mr.Remarks AS digital\_item\_notes,**

**CASE WHEN mm.PublicAccess = 1 THEN 'No' ELSE 'Yes' END**

**AS is\_restricted**

**FROM MediaRenditions mr,**

**MediaMaster mm**

**WHERE mr.MediaMasterID = mm.MediaMasterID**

**AND mr.RenditionID = ?RenditionID?**

**</query>**

**<query type="singleResult">**

**SELECT ob.CreditLine AS credit,**

**ob.Description + ' ' + ob.Dimensions AS description,**

**ob.portfolio AS group\_title,**

**ob.ObjectNumber AS source\_system\_id,**

**ob.Title AS title,**

**ob.Dated AS work\_creation\_date**

**FROM Objects ob**

**WHERE ObjectID = ?ObjectID?**

**</query>**

**<query type="singleResult">**

**SELECT obr.Copyright AS rights\_holder,**

**obrt.ObjRightsType AS terms\_and\_restrictions**

**FROM ObjRights obr,**

**ObjRightsTypes obrt**

**WHERE obr.ObjRightsTypeID = obrt.ObjRightsTypeID**

**AND ObjectID = ?ObjectID?**

**</query>**

**<query type="cursorAppend" delimiter=", ">**

**SELECT DISTINCT c.DisplayName AS keywords**

**FROM Constituents c, ObjConXrefs o**

**WHERE c.ConstituentID = o.ConstituentID**

**AND o.ObjectID = ?ObjectID?**

**</query>**

**</sync>**

**</NMAAHC>**

**Appendix D: CDIS 2.0 Report File**

**Example 1:**

CDIS 2.0: Synchronization Report and Statistics

siUnit: NMAAHC

Batch Number: 20150316-133658

Renditions to metadata sync not synced before: 1505

Renditions where DAMS needs metadata changes: 4

Total Number of Renditions to MetaData sync: 1509

Renditions to IDS path sync in Collections DataBase: 1505

================================================================

UOI\_ID / Rendition Number Pairs:

5b77345a5cdd539bdf2d98d4dd30a60566ceb9b9 metaData 2013.223.77.001

2c5d0099a70edf1c7a0f4aaff27b0aedafe74a64 metaData 2009.27.5.001

d16d3f125d27326896b2857e4a976976a28ac892 metaData 2012.154.6.001

.

.

4d670c358cea06a69b45b21221e84193aa357d04 idsPath 2009.14.7.002

eb6232f08b22de6263bd87ed2a31f2dc24501a71 idsPath 2009.14.7.001

d03cb95aa9ee341752bb2b542ac846e01d68d3d2 idsPath 2009.14.2.001

39bb728b63b434d21e1e021e398260944fd75e48 idsPath 2009.14.8.001

.

.

================================================================

Failed UOI\_ID / Rendition Number Pairs:

No Failures recorded

**Example 2:**

CDIS 2.0: DAMS/Collections Link Report and Statistics

siUnit: NMAAHC

Batch Number: 20150316-112217

Unlinked DAMS Renditions: 10131

================================================================

UOI\_ID / Rendition Number Pairs:

5b77345a5cdd539bdf2d98d4dd30a60566ceb9b9 link 2013.223.77.001

2c5d0099a70edf1c7a0f4aaff27b0aedafe74a64 link 2009.27.5.001

95304866fdf96f4962415b11256c06baf3a145a3 link 2014.37.30.1.001

.

.

.

================================================================

Failed UOI\_ID / Rendition Number Pairs:

No Failures recorded