Robert Feldman

FeldmanR@si.edu

Revised April 6, 2015

CDIS 2.0

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

# Introduction

The Collections-DAMS Integration System 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 integrate all of this data from the CIS into the DAMS…and in some cases synchronize and add data into the CIS as well. 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 for different renditions 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 and have NO unit specific code anywhere in the CDIS java code.

With CDIS 2.0, there are batch files which invoke the CDIS process. The batch file specifies the proper config.ini file to use. The config.ini contains database connection settings, general settings as well as very specific settings that alter the behavior of CDIS. 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. See appendix A for the sample config.ini 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 file into the windows scheduler after the initial testing with the unit produces positive results, so any new additions or changes in DAMS and/or the CIS are picked up automatically.

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. These new fields are detailed in Appendix B.

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.

# CDIS Operation Types

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

*IngestToCIS*

*IngestToDAMS*

*Link*

*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 TMS. This is done by

1. Directly inserting rows to several tables into TMS database. This will add the media record into TMS, then
2. The thumbnail image from the thumbnail location in the DAMS is fed into a stream in the CDIS java code byte by byte to a Thumbnail BLOB in the newly created TMS media table.
3. The CDIS table is populated with the UOI\_ID (DAMS unique identifier), the rendition id of the newly created rendition in TMS, the rendition number from TMS, and the objectID from TMS to show that the link exists between the DAMS image and the TMS rendition.
4. The Source system Identifier in DAMS is updated with the RenditionNumber from TMS showing that the TMS 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 added to the collections system as part of this process. The records to be selected from the DAMS are configurable in the SQL.xml file as follows:

Within the <ingestToCIS> tag there are several query tags. Each query tag will have a type. 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 as small or as large as needed, and can be limited to as many records as you configure it to be. Upon execution of the IngestToCIS type, CDIS will obtain a list of UOI\_IDs, and OWNING\_UNIT\_UNIQUE\_NAMEs from DAMS that could be brought over to the collections system.

Once we have obtained the UOI\_IDs and the UANS from the DAMSSelectList result, each DAMS image is checked to see if there are already existing media renditions 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. This is the CheckForExistingTMSRendition query type tag. This second query is the sql that checks the TMS database to see if there exist any TMS renditions from the DAMSSelectList that are already in TMS. Often times, we do not wish to create renditions for one that is already there, and this query when properly configured will prevent that. See Appendix C for a Sample XML file showing these tags.

### 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

*locateByBarcode:* Newly configurable option available with CDIS 2.0. locateByBarcode indicates whether to step through the barcode logic to find the object in TMS that maps to the DAMS image. When locateByBarcode is set to ‘true’, CDIS will interpret the DAMS image file name as a barcode number. CDIS will attempt to find the TMS object based a lookup of barcode. When the locateByBarcode is set to ‘false’, the filename in DAMS is assumed to not be related to a barcode number, and there will be no barcode lookup for the object.

*appendTimeToRenditionName:* Newly configurable option to add a timestamp (time in hours, minutes and seconds) at the end of the renditionName. This option has been requested to add unique-ness to the RenditionNaming conventions.

*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.*

*renditionToObjectTrunc: This option is expected to be removed/ changed significantly in the next week to streamline and simplify Rendition To Object identification options, and will be documented after testing is complete.*

*newRenditionNameFormat. This option is expected to be removed/ changed significantly in the next week to streamline and simplify Rendition To Object identification options, and will be documented after testing is complete*

## IngestToDAMS

In the cases where the media rendition exists in TMS, 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 (image file) is copied off the TMS media drive and placed into the DAMS media drive at a hotfolder location.
2. An automated process outside 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

As with the IngestToCIS process in the earlier section, there is full control over which renditions are to come from the collections system and be added as images in DAMS. The records to be selected from the collections system are configurable in the SQL.xml file as follows

Within the <ingestToDAMS> tag there are several query tags. Each query tag will have a type. If the query type is “type="TMSSelectList”, then the query contained in this tag will be used to obtain the TMS records that need to be incorporated into DAMS. 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 as small or as large as needed, and can be limited to as many records as you configure it to be. Upon execution of the IngestToDAMS type, CDIS will generate a list of RenditionIDs and Filenames from TMS that need to be created in DAMS.

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 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:

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’.

workFolder: Indicates the work folder for media and xml 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.

## Link

The link process associates a DAMS image with a TMS rendition. Once this association is performed it is known as a ‘link’. The link process is to be used in two different scenarios. They are:

* In the cases where the same rendition was created in both TMS and DAMS by the unit (outside of CDIS), and CDIS did not create the media rendition in either one.
* Following an ingest to DAMS (detailed in section 2.2), the linking process must be run to associate the newly created DAMS image to a TMS rendition. Note: The link process does not need to be run following the ingestToTMS procedure, because the ingestToTMS also perfoms the link.

The DAMS image and the collection system rendition must be linked if the metadata is expected to flow from TMS to DAMS. The Link process establishes the link between DAMS and TMS by updating the image in DAMS with the source\_system\_id from TMS and then Inserting the linkage information into the CDIS table. Refer to Appendix B for the CDIS table.

### 2.3.1 Link SQL-XML configuration Options

As with the other CDIS execution types, because each SI unit may use different naming convention and methodology, CDIS offers full control over how to map the DAMS image to the TMS rendition. This is configurable in the SQL.xml file within the <link> tag.

Like the other execution types, within the <link> 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. 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, each uoi\_id/uan pair is checked in TMS to see if it can find if the rendition is there already….and make sure it has not already been linked by CDIS.

### Link 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.

## 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 involves going back to the collections database to ensure that the collections database will have the IDS derivative image available.

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

CDIS collects metadata from various tables in the collections system via a SQL join. Once the data is collected via a join, CDIS inserts this metadata into the SI\_ASSET\_METADATA table in the DAMS database. Because each SI unit uses 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 much be followed when determining how to link the collections metadata to DAMS. The Core data model specifies which fields in DAMS are visible in the MediaManager application, and also guidelines that specify what the fields in the DAMS are used for.

The metadata mapping is configurable for each unit in the SQL.xml file. There will be a sync section in the SQL-xml file. Each unit can have one or many queries to obtain the required metadata. The queries will query the Collections database for the required metadata and identify which column in the DAMS table SI\_ASSET\_METADATA it will map to. All of the mapping logic is found in this file. There can be case statements, appending two columns together and other logic. The idea is to have as much logic as possible in the SQL-XML file.

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

Most metadata queries are of the singleResult type. The singleResult is expected to return only a single row of metadata from the collections system. One result is TMS is mapped to one rendition.

The second type of query is the cursorAppend query. This type of query is used when there may be multiple results returned from the collections system that all need to go into a single field in DAMS. This is useful for things such as keywords and captions. For instance, 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 expected to be specified as well, as we do not want all the results running into each other. 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 new configurable option for CDIS 2.0 will indicate 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. Notifying IDS of the update too much can cause unnecessary load on the IDS system. There are currently four different options:

default: IDS should only be notified if there is a change in restrictions. This option will reduce the number of notifications sent to IDS, but also make sure IDS will get notified when it needs to know about 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 should be used especially in new projects when IDS already has the required renditions, and the high volume of metadata syncing can flood the IDS server.

ifRestricted: IDS should only be notified if the rendition is restricted in any manner (whether by size or by the is\_Restricted limitations). 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 sync process determines which assets renditions require sync by detecting which assets have been modified in DAMS, but also determining which assets had never been synced before. The CDIS table now contains a *MetaDataSyncDate*. If the metaDataSync date is null, the sync process will pick up the rendition and perform a sync.

The same goes for *SyncIDSPathDate*. One can configure which renditions need to be synced to point to the correct IDS rendition. CDIS will only attempt to sync the collection system’s file name and path in cases where the SyncIDSPathDate is null.

Instead of relying on the ‘last\_sync\_date’ as per the old design, this new design allows someone to easily pick and choose which renditions need to be re-synced at any time.

## ThumbnailSync

ThumbnailSync is more of a utility used in special cases, rather than a 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. It has been found that in some cases, the thumbnail and the IDS image offer different views of the same object when they were linked after the fact. The thumbnail sync will ensure sure that the thumbnail matches exactly what is in DAMS. This option is also available as part of the link step when updateTMSThumbnail is set to true.

2. When the thumbnail image in the CIS database is null (empty), corrupted or otherwise problematic. There are many legacy renditions in TMS that do not have the thumbnail populated. Many of these renditions will require their thumbnail blob to be refreshed.

Note that when run independently or as part of the link, because this is outside the normal flow, this 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 siUnit. As mentioned earlier, the java application code does not use this 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.

**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 #################################

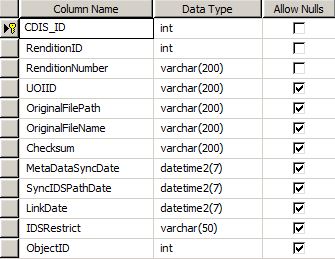
updateTMSThumbnail = false

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

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

flagForIDS=default

**Appendix B: CDIS table**

****

Purpose: The CDIS table has two main purposes:

1. Tracks which images/renditions have been through CDIS and when particular events happened. Can be used for reporting or troubleshooting renditions.
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 TMS for the particular rendition.

*RenditionNumber*: The RenditonNumber from TMS for the particular rendition.

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

*OriginalFilePath*: Part of earlier version of CDIS, Not currently in use by CDIS 2.0.

*OriginalFileName*: Part of earlier version of CDIS, Not currently in use by CDIS 2.0.

*Checksum*: Never used by CDIS, may be added in future.

*MetaDataSyncDate*: New for CDIS 2.0. This field holds the date that metaDataSync has last occurred for this rendition/image file. Important not only for tracking, but critical part of CDIS logic. Helps determine which renditions have been synced and which need to be synced for the first time. Gives control over resyncing images with renditions like never before.

*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. Gives control over resyncing renditions like never before.

*LinkDate*: New for CDIS 2.0. This field holds the date that the rendition in TMS has been first linked with the image file in DAMS.

*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 TMS ObjectID of the Rendition. Established at the time of linking, can be used later 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>**

**<link>**

**<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>**

**</link>**

**<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