Flow 14

Flow 14 is the successor of both flow 1 and flow 4. Please consult the documentation of Flow 1 and Flow 4 for more information about these flows.

Introduction

As a result of various digitization projects and digitization requests by customers, the digitized objects have to be validated and ingested to the Shared Object Repository. Flow 14 will perform the operations that can be done automatically. This flow does not cover continues scans with the A0 printer (see Flow 2) and does not cover new acquisitions (see Flow 3). In all cases, the success or failure of a started procedure will be emailed.

Structure of items

EAD records

Digitized objects with an EAD description have to be delivered according to the following structure:

Type	Description	Convention	Example(s)
Parent folder	This folder has the number of the archive.	/[naming authority]/[archive number]	/10622/ARCH12345
Concordance table	CSV file that describes the digitized objects.	/[naming authority]/[archive number]/[archive number].csv	/10622/ARCH12345/ARCH12345.csv
EAD document	The EAD record. (Optional)	/[naming authority]/[archive number]/[archive number].xml	/10622/ARCH12345/ARCH12345.xml
Files per group	Files are mapped per group. This means that all master files, level1 derivatives and the different text layers all have their own parent folder.	/[naming authority]/[archive number]/[group name]	/10622/ARCH12345/master /10622/ARCH12345/level1 /10622/ARCH12345/text transcription /10622/ARCH12345/text translation nl
Files per objnr	Files are next mapped per objnr. Each objnr matches an invnr in the EAD document.	/[naming authority]/[archive number]/[group name]/[objnr]	/10622/ARCH12345/master/1 /10622/ARCH12345/master/2 /10622/ARCH12345/master/3
Files	The files to be ingested.	[archive number]_[objnr]_[volgnr].[extension]	ARCH12345_1_0001.tif ARCH12345_1_0002.tif ARCH12345_1_0003.tif

A concordance table should be delivered in CSV format which describes the content. Each row describes a master file. The following information is expected to be included:
Note: The order of the headers is no longer of importance. Unknown columns are ignored.

Header	Description
ID	The EAD consists of various invnrs. The ID of the invnr to which the master file belongs is recorded in this column.
objnr	Due to folder naming restrictions, each ID is mapped to a numeric objnr.
master	Specifies the location of the master file. The location is irrespective of the file format. (Such as images, audio, video, pdf...)
level1	Specifies the location of the level 1 derivative. (Optional)
text [type] [language]	Specifies the location of the plain text / ALTO xml file describing the text content of the image. (Optional) *Type* is required, but *language* is optional. In case of multiple text layers, multiple columns may be added. Not only are these columns completely optional, not every image may have text content. Therefore, empy values in the concordance table are allowed.
volgnr	To keep the master files in sequential order, a sequence number is required. May start with either 0 or 1.

An example of a concordance table:

Concordance table example

```
objnr,ID,master,level1,volgnr,text transcription,text translation nl
1,A1,/ARCH12345/master/1/ARCH12345_1_0001.tif,/ARCH12345/level1/1/ARCH12345_1_0001.jpg,1,,
1,A1,/ARCH12345/master/1/ARCH12345_1_0002.tif,/ARCH12345/level1/1/ARCH12345_1_0002.jpg,2,,
1,A1,/ARCH12345/master/1/ARCH12345_1_0003.tif,/ARCH12345/level1/1/ARCH12345_1_0003.jpg,3,/ARCH12345/text
transcription/1/ARCH12345_1_0003.xml,
1,A1,/ARCH12345/master/1/ARCH12345_1_0004.tif,/ARCH12345/level1/1/ARCH12345_1_0004.jpg,4,/ARCH12345/text
transcription/1/ARCH12345_1_0004.xml,
2,A2,/ARCH12345/master/2/ARCH12345_2_0001.wav,/ARCH12345/level1/2/ARCH12345_2_0001.mp3,1,/ARCH12345/text
transcription/2/ARCH12345_2_0001.xml,/ARCH12345/text translation nl/2/ARCH12345_2_0001.txt
2,A2,/ARCH12345/master/2/ARCH12345_2_0002.wav,/ARCH12345/level1/2/ARCH12345_2_0002.mp3,2,/ARCH12345/text
transcription/2/ARCH12345_2_0002.xml,/ARCH12345/text translation nl/2/ARCH12345_2_0002.txt
```

More examples can be found on the subpage: EAD ingest examples

MARC records

Digitized objects with an MARC description have to be delivered according to the following structure:

Type	Description	Convention	Example(s)
Parent folder	This folder has the name of the barcode to which the items belong.	/[naming authority]/[852p]	/10622/30051234567890
Files per group	Files are mapped per group. This means that all master files, level1 derivatives and the different text layers all have their own parent folder. An important difference with the EAD structure is that the name of the folder will correspond with the name of the file in the resulting METS document.	/[naming authority]/[852p]/[group name]	/10622/30051234567890/archive image /10622/30051234567890/archive audio /10622/30051234567890/archive video /10622/30051234567890/archive pdf
Text layers	Text layers (plain text / ALTO xml) should be recorded using a more strict convention. *Type* is required, but *language* is optional.	/[naming authority]/[852p]/text [type] [language]	/10622/30051234567890/text transcription /10622/30051234567890/text translation nl
Derivatives	Any derivatives should have the same file name as the master file.	/[naming authority]/[852p]/[group name]/.level1 /[naming authority]/[852p]/[group name]/.level2 /[naming authority]/[852p]/[group name]/.level3	/10622/30051234567890/archive image/.level1 /10622/30051234567890/archive audio/.level2 /10622/30051234567890/archive video/.level3
Files	The files to be ingested.	[852p].[sequence number].[extension]	30051234567890.1.jpg 30051234567890.2.jpg 30051234567890.1.xml 30051234567890.2.xml

Various examples can be found in on the subpage: MARC ingest examples

Validation of the concordance table

As the concordance table is only used in combination with EAD records, ingests with a MARC record should skip this step.

The validation procedure can be started by placing an empty file with the filename *validate.txt* in the parent folder. The procedure will start by locking the folder. During the procedure no files/folders can be moved, deleted, added etc. After the procedure has been completed, the folder will be unlocked automatically.

The validation procedure will start by producing a DROID report (file identification) of all the files. The DROID report and the concordance table are both used to produce a validation report. The following checks are included:

- Are all delivered files recorded in the concordance table?
- Are all files mentioned in the concordance table found on disk?
- Does the number of files per objnr folder correspond to the number of files per objnr in the concordance table?
- Do the master file, its derivative and its text files have the same name?
- The sequence numbering contains no missing numbers?
- Are the files at least 1000 bytes? (Text files may be smaller, but at least 1 byte, as it is not unlikely to have text files with a size of only a couple of bytes)
- Is there no mismatch between the extension and the content of the *master* file?
- Is there no mismatch between the extension and the content of the *level1* file?
- Is the file type of the *level1* file comparable to the file type of the *master* file?
- Is the *text* file really either a plain text file or an XML file?

The results are recorded in the file *report.txt*.

In addition, if an EAD document is provided, the invnr IDs in the EAD document are compared to the invnr IDs in the concordance table. The HTML file *ead.report.html* is produced with a table listing invnr IDs recorded in the EAD, but missing in the concordance table and the other way around.

Starting an ingest

EAD records

The ingest procedure can be started by placing an empty file with the filename *ingest_ead.txt* in the parent folder. The procedure will start by locking the folder. During the procedure no files/folders can be moved, deleted, added etc. After the procedure has been completed, the folder will be unlocked automatically. You cannot start an ingest if the validation procedure has not been started. However, the result of the validation procedure does not limit you to start the ingest procedure.

Before it actually starts with the ingest, the directory structure is first transformed to correspond with the directory structure used for MARC records. However, instead of [852p], [archive number].[objnr] is used instead. The reason for this transformation is to make sure the ingest procedures (and the METS creation procedures) of both the EAD and the MARC can use the same scripts. This results in the following structure for the given example concordance table:

- ARCH12345
    - ARCH12345.1
        - archive image
            - ARCH12345.1.1.tif
            - ARCH12345.1.2.tif
            - ARCH12345.1.3.tif
            - ARCH12345.1.4.tif
            - .level1
                - ARCH12345.1.1.jpg
                - ARCH12345.1.2.jpg
                - ARCH12345.1.3.jpg
                - ARCH12345.1.4.jpg
        - text transcription
            - ARCH12345.1.3.xml
            - ARCH12345.1.4.xml
    - ARCH12345.2
        - archive audio
            - ARCH12345.2.1.wav
            - ARCH12345.2.2.wav
            - .level1
                - ARCH12345.2.1.mp3
                - ARCH12345.2.2.mp3
        - text transcription
            - ARCH12345.2.1.xml
            - ARCH12345.2.2.xml
        - text translation nl
            - ARCH12345.2.1.txt
            - ARCH12345.2.2.txt

The script will then start the ingest procedure for each invnr, based on a new DROID report, with the instruction to the SOR te perform the following tasks:

- Ingest the master (and level 1 deriviative, if provided)
- Create level 1 to 3 derivatives (only if a) in the case of a level 1 derivative, the derivative was not provided and b) the SOR knows how to produce the derivative)
- Bind the PIDs created for each individual file

The script will take care of binding the PID for each invnr, with references to:

- The first master file (sequence nr 2) *(for instance, for archive ARCH03210 and invnr 1: http://hdl.handle.net/10622/ARCH03210.1?locatt=view:master)*
- The derivatives of the first master file (sequence nr 2) *(for instance, for archive ARCH03210 and invnr 1: http://hdl.handle.net/10622/ARCH03210.1?locatt=view:level1)*
- The PDF *(for instance, for archive ARCH03210 and invnr 1: http://hdl.handle.net/10622/ARCH03210.1?locatt=view:pdf)*
- The METS *(for instance, for archive ARCH03210 and invnr 1: http://hdl.handle.net/10622/ARCH03210.1?locatt=view:mets)*

In addition, specific to the EAD ingest, it will register a PID for the the complete archive with references to:

- The EAD description *(for instance, for archive ARCH03210: http://hdl.handle.net/10622/ARCH03210?locatt=view:ead)*
- The catalog reference *(for instance, for archive ARCH03210: http://hdl.handle.net/10622/ARCH03210?locatt=view:catalog)*
- The first master file (sequence nr 2 from the first invnr) *(for instance, for archive ARCH03210: http://hdl.handle.net/10622/ARCH03210?locatt=view:master)*
- The derivatives of the first master file (sequence nr 2 from the first invnr) *(for instance, for archive ARCH03210: http://hdl.handle.net/10622/ARCH03210?locatt=view:level1)*

MARC records

The ingest procedure can be started by placing an empty file with the filename ingest_marc.txt in the parent folder. The procedure will start by locking the folder. During the procedure no files/folders can be moved, deleted, added etc. After the procedure has been completed, the folder will be unlocked automatically.

Before the ingest procedure starts, it makes sure there are no empty files found and the folder structure corresponds with the expected structure. It also makes a call to the SRU service to make sure a MARC record with the specified barcode (852p) exists.

The script will then start the ingest procedure, based on a DROID report, with the instruction to the SOR te perform the following tasks:

- Ingest the master (and level 1-3 deriviatives, if provided)
- Create level 1 to 3 derivatives (only if a) the derivatives are not provided and b) the SOR knows how to produce the derivative)
- Bind the PIDs created for each individual file.

The script will take care of binding the PID of the complete item, with references to:

- The first master file (sequence nr 1)
- The derivatives of the first master file (sequence nr 1)
- The PDF
- The METS

Access status

The access statuses known by the ingest and the METS procedure are:

Access status	Master	Level 1	Level 2	Level 3
open	closed	open	open	open
restricted	closed	closed	open	open
minimal	closed	closed	closed	open
pictoright	closed	closed	closed	open
closed	closed	closed	closed	closed

In the case of text layers, the access status is determined by the access status of the master file. If the level 1 derivative of the master file is open, then the text layer (a master file) is open as well. So, only when the master file has an access status of 'open', the text layers are open as well.

The access status of the material is automatically obtained from the MARC record. In the case of EAD records the access status is by default 'open'. By placing a file with the name .access.txt in the parent folder, the access status can be changed for the complete ingest. The .access.txt file can also be placed in the directory with master files for a specific objnr to change the access status on an item level. For example in the folder /10622/ARCH12345/Tiff /2 in order to change the access status for all files belonging to the item with objnr 2.

⚠ DODS-91 - Jira project doesn't exist or you don't have permission to view it.

If the access is set to a level other than 'open', this information is added to the text file. i.e. create an instruction '.access.txt' with the content "restricted" in the top folder.

METS creation

The METS creation is a new addition in flow 14 and non-existant in both flow 1 and flow 4. Before, the METS was automatically created on the fly by the Object Repository when requested. However, as the METS has become more complex, the creation of the METS is now created before and ingested.

At the moment the METS creation procedure should be started by placing an empty file with the filename mets_marc.txt in the parent folder in the case of MARC records and an empty file with the filename mets_ead.txt in the case of EAD records. The only difference is that in the case of EAD records, multiple METS documents are created and ingested. (One for each item in the EAD) As the METS creation needs information about the derivatives that were created by the SOR, the METS creation procedure cannot be started while the SOR is still processing the ingest. There is a feature request for an API in the SOR which allows flow 14 to request the status of the ingest.

⚠ OCRP-13 - Jira project doesn't exist or you don't have permission to view it.

Until this has been resolved, the METS creation has to be triggered manually.

Information recorded in the METS

Based on the DROID procedure and the conventions, the following information is recorded in the created METS:

- All master files and the given/created derivatives with information from the DROID report.
- File groups created based on the given group names.
- The open/closed access status of each file group using Eprints DC XML.
- A physical structure map, listing
    - a) The sequential order of the files.
    - b) Groups master files, derivatives and text layers together.
- A descriptive metadata section with the language code (Dublin Core) for a given file group with text layers.

For information about the derivatives, the metadata API of the SOR is used.

The file groups are created based on the given group names. In the case of derivatives, the METS creation procedure needs to know in what group to place the derivatives. Currently, this is determined by the name of the file group of the master file. As the EAD structure does not include a file group, the file group is determined by the content type of the master file:

Content type	Master file group	Level 1 file group	Level 2 file group	Level 3 file group
image/...	archive image	hires reference image	reference image	thumbnail image
audio/...	archive audio	reference audio		
video/..., application/mxf	archive video	reference video	stills video	thumbnail video
application/pdf, application/x-pdf	archive pdf	hires reference pdf	reference pdf	thumbnail pdf

If the METS procedure found a derivative, but cannot place it in a file group, it will end the procedure with an error.

In the case of text layers, the file group name is created as follows: *[text type] [format] text. Text type* is determined by the name of the folder in the case of MARC records and the name of the column in the case of EAD records. The *format* can be either 'plain' in the case of plain text or 'alto' in the case of XML files. Some examples are:

- transcription alto text
- transcription plain text
- transliteration plain text

Other procedures

Remove

At the moment the remove procedure should be started by placing an empty file with the filename remove.txt in the parent folder. Just like the METS procedure, the removal procedure needs to know when the SOR has finished processing both the ingest and the METS ingest procedure. There is a feature request for an API in the SOR which allows flow 14 to request the status of the ingest.

⚠ OCRP-13 - Jira project doesn't exist or you don't have permission to view it.

Until this has been resolved, the removal procedure has to be triggered manually.

Checksum

If you are provided with an MD5 checksum, it is possible to start a checksum procedure which generates a checksum of the fileset as well. This allows you to compare checksums and to make sure the fileset was delivered without file corruption. The checksum procedure can be started by placing an empty file with the filename checksum.txt in the parent folder.

The checksum procedure requires a file named *checksum.md5* in the parent folder containing the checksums:

checksum.md5

```
61bfc67dbfec972648363d39d8d58066  path/to/fileA.txt
4ee914f58a9f1bbd7ea9a31382419e96  path/to/fileB.txt
cfa3a4e384e53f5d6d27e2276b24e055  path/to/fileC.txt
```

The results are placed in a file *[archiefnummer].report.txt*:

ARCH12345.report.txt

```
path/to/fileA.txt: OK
path/to/fileB.txt: OK
path/to/fileC.txt: OK
```

Concordance

The concordance procedure can be started by placing an empty file with the filename concordance.txt in the parent folder. The procedure reconstructs a CSV concordance table based on a conventional folder setup.

Workarounds

Updating the METS

The procedures described here do not take into account digital objects that have been ingested to the SOR before. In general it is not a big problem as one can just add new digital objects to an existing collection in the SOR. However, the METS creation procedure creates and ingests a completely new METS file based on a DROID analysis that was created ONLY for the new files. As a result, all files that were ingested before are missing from the new METS file.

There is a workaround created to solve this problem and to merge the old METS with the new METS. This is archieved by transforming the old METS file to a DROID analysis, combining the DROID analysis with the actual DROID analysis, and then let the METS creation procedure produce a METS from this newly created DROID file.

1. In the case of EAD, create a concordance table listing **only** the files that will be added. (Even if no new masters are added, the '*master*' column should remain present, but empty)
2. In the case of EAD, the concordance table will most likely NOT pass validation. For this reason, a special validation procedure, '**validate_merge**', was added which can be started by placing an empty file with the filename *validate_merge.txt* in the parent folder.
3. Run the ingest procedures as usual.
4. Before starting the METS creation procedure, the DROID analysis produced during the ingest procedure should be updated with information from the old METS.
   In the case of EAD, this procedure, '**merge_ead**', can be started by placing an empty file with the filename *merge_ead.txt* in the parent folder.
   In the case of MARC, this procedure, '**merge_marc**', can be started by placing an empty file with the filename *merge_marc.txt* in the parent folder.
5. Run the METS creation procedures as usual.
6. If the handle URL still points to the 'old', automatically created METS, then the PID binding of the METS URL has to be updated to link to the new METS file.
   In the case of EAD, this procedure, '**bind_mets_ead**', can be started by placing an empty file with the filename *bind_mets_ead.txt* in the parent folder.
   In the case of MARC, this procedure, '**bind_mets_marc**', can be started by placing an empty file with the filename *bind_mets_marc.txt* in the parent folder.

Please note: This is implemented as a workaround and should be used as a workaround.

⚠ OCRP-14 - Jira project doesn't exist or you don't have permission to view it.

The limitations of this workaround:

- This workaround will only work with original METS files that were either produced by this flow or METS files that were originally produced automatically by the SOR.
- This workaround will omit the METS *dmdSec* section with text layer language information that was present in the original METS file.

Future additions

A METS file describes a single object, which corresponds to a single item in the case of an EAD record and corresponds to a single barcode (852$p) in the case of a MARC record. However, it is possible to have multiple sequences for a single object. Let's say for example, you have the following material:

- 4 wavs (a four part audio interview)
- 4 plain text files (a transcription for each audio file)
- 2 tiffs (the front and back cover)

Rather than one sequence, these are actually two different sequences:

1. The four part interview and its transcriptions
2. The front and back cover

At the moment, it is not possible to create a METS document consisting of multiple sequences. In order to make this work, two pieces of information are missing: which files belong to which group and how should the groups be prioritized? The following describes a **possible** solution for this problem by extending the current convention how the files should be structured:

EAD records

Type	Description	Convention	Example(s)
Parent folder	This folder has the number of the archive.	/[naming authority]/[archive number]	/10622/ARCH12345
Concordance table	CSV file that describes the digitized objects.	/[naming authority]/[archive number]/[archive number].csv	/10622/ARCH12345 /ARCH12345.csv
EAD document	The EAD record. (Optional)	/[naming authority]/[archive number]/[archive number].xml	/10622/ARCH12345 /ARCH12345.xml
Files per group	Files are mapped per group. This means that all master files, level1 derivatives and the different text layers all have their own parent folder.	/[naming authority]/[archive number]/[sequence group]/[group name]	/10622/ARCH12345/master /10622/ARCH12345/level1 /10622/ARCH12345/text transcription /10622/ARCH12345/text translation nl
Files per objnr	Files are next mapped per objnr. Each objnr matches an invnr in the EAD document.	/[naming authority]/[archive number]/[sequence group]/[group name]/[objnr]	/10622/ARCH12345/master/1 /10622/ARCH12345/master/2 /10622/ARCH12345/master/3
Files	The files to be ingested.	[archive number]_[objnr]_[volgnr].[extension] ***or in case of a sequence group: (see concordance)*** ***[archive number]_[objnr]_[sequence group]_[volgnr].[extension]***	ARCH12345_1_0001.tif ARCH12345_1_0002.tif ARCH12345_1_0003.tif ***ARCH12345_1_1_0001.tif*** ***ARCH12345_1_1_0002.tif*** ***ARCH12345_1_1_0003.tif***

Header	Description
ID	The EAD consists of various invnrs. The ID of the invnr to which the master file belongs is recorded in this column.
objnr	Due to folder naming restrictions, each ID is mapped to a numeric objnr.
master	Specifies the location of the master file.
level1	Specifies the location of the level 1 derivative. (Optional)
text [type] [language]	Specifies the location of the plain text / ALTO xml file describing the text content of the image. (Optional) *Type* is required, but *language* is optional. In case of multiple text layers, multiple columns may be added. Not only are these columns completely optional, not every image may have text content. Therefore, empy values in the concordance table are allowed.
group	***The sequence group the files belong to. (Optional)*** ***The sequence group has a number,*** ***where the lowest number has the highest priority.***
volgnr	To keep the master files in sequential order, a sequence number is required.

Concordance table example

```
objnr,ID,master,level1,group,volgnr,text transcription,text translation nl
1,A1,/ARCH12345/master/1/ARCH12345_1_1_0001.tif,/ARCH12345/level1/1/ARCH12345_1_1_0001.jpg,1,1,,
1,A1,/ARCH12345/master/1/ARCH12345_1_1_0002.tif,/ARCH12345/level1/1/ARCH12345_1_1_0002.jpg,1,2,,
1,A1,/ARCH12345/master/1/ARCH12345_1_1_0003.tif,/ARCH12345/level1/1/ARCH12345_1_1_0003.jpg,1,3,/ARCH12345/text
transcription/1/ARCH12345_1_1_0003.xml,
1,A1,/ARCH12345/master/1/ARCH12345_1_1_0004.tif,/ARCH12345/level1/1/ARCH12345_1_1_0004.jpg,1,4,/ARCH12345/text
transcription/1/ARCH12345_1_1_0004.xml,
1,A1,/ARCH12345/master/1/ARCH12345_1_2_0001.wav,/ARCH12345/level1/1/ARCH12345_1_2_0001.mp3,2,1,/ARCH12345/text
transcription/1/ARCH12345_1_2_0001.xml,/ARCH12345/text translation nl/1/ARCH12345_1_2_0001.txt
1,A1,/ARCH12345/master/1/ARCH12345_1_2_0002.wav,/ARCH12345/level1/1/ARCH12345_1_2_0002.mp3,2,2,/ARCH12345/text
transcription/1/ARCH12345_1_2_0002.xml,/ARCH12345/text translation nl/1/ARCH12345_1_2_0002.txt
```

MARC records

Type	Description	Convention	Example(s)
Parent folder	This folder has the name of the barcode to which the items belong.	/[naming authority]/[852p]	/10622/30051234567890
Sequence group	***The sequence group the files belong to. The sequence group has a number, where the lowest number has the highest priority.***	***/[naming authority]/[852p]/[sequence group]***	***/10622/30051234567890/1 /10622/30051234567890/2***
Files per group	Files are mapped per group. This means that all master files, level1 derivatives and the different text layers all have their own parent folder. An important difference with the EAD structure is that the name of the folder will correspond with the name of the file in the resulting METS document.	/[naming authority]/[852p]/[sequence group]/[group name]	/10622/30051234567890/1/archive image /10622/30051234567890/1/archive audio /10622/30051234567890/1/archive video /10622/30051234567890/1/pdf
Text layers	Text layers (plain text / ALTO xml) should be recorded using a more strict convention. *Type* is required, but *language* is optional.	/[naming authority]/[852p]/[sequence group]/text [type] [language]	/10622/30051234567890/1/text transcription /10622/30051234567890/1/text translation nl
Derivatives	Any derivatives should have the same file name as the master file.	/[naming authority]/[852p]/[sequence group]/[group name]/.level1 /[naming authority]/[852p]/[sequence group]/[group name]/.level2 /[naming authority]/[852p]/[sequence group]/[group name]/.level3	/10622/30051234567890/1/archive image/.level1 /10622/30051234567890/1/archive audio/.level2 /10622/30051234567890/1/archive video/.level3
Files	The files to be ingested.	[852p].[sequence number].[extension]	30051234567890.1.jpg 30051234567890.2.jpg 30051234567890.1.xml 30051234567890.2.xml