mosromgr

Release 0.9.0

unknown

Jun 21, 2021

CONTENTS

Python library for managing MOS running orders. Pronounced *mos-ro-manager*.

The library provides functionality for classifying MOS file types, processing and inspecting MOS message files, as well as merging MOS files into a running order, and providing a "completed" programme including all additions and changes made between the first message (`roCreate`) and the last (`roDelete`).

This can be used as a library, using the utilities provided in the *mosromgr* module, and the command line command *Command line interface* can be used to process either a directory of MOS files, or a folder within an S3 bucket.

This library was developed by the BBC News Labs team.

> **Warning:** Note that the library is currently in beta. The API and CLI are not yet stable and may change. Once the library reaches v1.0, it will be considered stable. Please consider giving *Feedback* to help stabilise the API.

CHAPTER

ONE

EXAMPLE USAGE

1.1 Command line

List the stories within a running order:

```
$ mosromgr inspect -f roCreate.mos.xml --stories
0828 MIDLANDS TODAY Wed, 11.11.2020

INTRODUCTION-READ

TESTING-OOV

WEATHER-SHORT

END OF PROGRAMME
```

Merge all MOS files in directory *newsnight* and save in `FINAL.xml`:

```
$ mosromgr merge -f newsnight/* -o FINAL.xml
```

1.2 Library

Load a `roCreate` file and view its stories:

```
from mosromgr.mostypes import RunningOrder

ro = RunningOrder.from_file('roCreate.mos.xml')

for story in ro.stories:
    print(story.slug)
```

Merge a single `roStorySend` (*StorySend*) into a `roCreate` (*RunningOrder*) and output the file to a new file:

```
from mosromgr.mostypes import RunningOrder, StorySend

ro = RunningOrder.from_file('roCreate.mos.xml')
ss = StorySend.from_file('roStorySend.mos.xml')

ro += ss
```

(continues on next page)

(continued from previous page)

```
with open('final.mos.xml', 'w') as f:
    f.write(str(ro))
```

If you're automating this process you won't necessarily know which MOS Type to use, so you can construct an object from the base class *MosFile* which will automatically classify your file:

```
>>> from mosromgr.mostypes import MosFile
>>> mf1 = MosFile.from_file('roCreate.mos.xml')
>>> mf1
<RunningOrder 1000>
>>> mf2 = MosFile.from_file('roStorySend.mos.xml')
>>> mf2
<StorySend 1001>
```

Using *MosCollection* will sort and classify multiple MOS types of all given files, allowing you to process a collection of MOS files within a complete or partially complete programme:

```
from mosromgr.moscollection import MosCollection

mos_files = ['roCreate.mos.xml', 'roStorySend.mos.xml', 'roDelete.mos.xml']
mc = MosCollection.from_files(mos_files)

mc.merge()
with open('final.mos.xml', 'w') as f:
    f.write(str(mc))
```

CHAPTER

TWO

DOCUMENTATION

This documentation follows the Diátaxis system, so is split between four modes of documentation: tutorials, how-to guides, technical reference and explanation.

2.1 Getting started

This section shows you how to get started with *mosromgr*.

2.1.1 Installing

Install with pip:

```
$ pip install mosromgr
```

2.1.2 Command line interface check

After installing the module, a simple way to verify it's working is by using the *Command line interface*. First of all, open a terminal and run the command `mosromgr` to be sure it's installed. You should see output like so:

```
$ mosromgr
optional arguments:
  -h, --help            show this help message and exit
  --version             show program's version number and exit

commands:
  {help,detect,inspect,merge}
    help                Displays help about the specified command
    detect              Detect the MOS type of one or more files
    inspect             Inspect the contents of a roCreate file
    merge               Merge the given MOS files
```

Now start by obtaining the MOS files for a single complete programme. In a terminal window, enter the directory containing the MOS files and run the command `mosromgr detect` on a single roCreate file, for example:

```
$ mosromgr detect 123456-roCreate.mos.xml
123456-roCreate.mos.xml: RunningOrder
```

The output shows that it's identified the `roCreate` file as a *RunningOrder*. Try it with some other files to check it can correctly identify a *MosFile* subclass to represent the file.

2.1.3 Using the module in Python code

Now you've tested the ready-made command line program is working with your MOS file, try using the module in some custom Python code.

Open a Python shell and try creating a MOS object from your `roCreate` file:

```
>>> from mosromgr.mostypes import RunningOrder
>>> ro = RunningOrder.from_file('123456-roCreate.mos.xml')
>>> ro
<RunningOrder 123456>
```

This shows you've successfully loaded a MOS file and created a *RunningOrder* from it. The output shows the object representation (`__repr__`) which includes the class name and message ID (this is from the XML contents, not the filename).

The next page will walk through the functionality provided by the module.

2.2 Introduction

This section is a walkthrough of the contents of the module, intended to explain how *mosromgr* works and introduce the concepts.

2.2.1 MOS Types

The *API - MOS Types* section of the module provides a collection of classes for dealing with individual MOS messages. The classes provide easy access to some of the elements within a MOS file, such as a list of stories within a running order, the transmission time of a programme, or its duration.

For example, you can load a running order from a `roCreate` file, print the RO Slug and access some details:

```
>>> from mosromgr.mostypes import RunningOrder
>>> ro = RunningOrder.from_file('123456-roCreate.mos.xml')
>>> ro.ro_slug
'22:45 NEWSNIGHT 54D CORE Thu, 08.04.2021'
>>> ro.message_id
123456
>>> ro.start_time
datetime.datetime(2021, 4, 8, 21, 46, 30)
>>> ro.duration
970.0
>>> len(ro.stories)
10
```

In the case of MOS messages which contain a *change* to a running order, the relevant details are exposed, for example a *StoryInsert* includes access to the *source_stories* and *target_story*.

When dealing with merging *MosFile* objects, this is done by "adding" each file to the *RunningOrder* object by using the + operator:

```
>>> from mosromgr.mostypes import RunningOrder, StoryInsert
>>> ro = RunningOrder.from_file('123456-roCreate.mos.xml')
>>> ss = StoryInsert.from_file('123457-roStoryInsert.mos.xml')
```

(continues on next page)

(continued from previous page)

```
>>> len(ro.stories)
10
>>> ro += ss
>>> len(ro.stories)
11
```

2.2.2 MOS Elements

The *API - MOS Elements* part of the module provides a collection of classes used to provide easy access to certain elements within a *MosFile* object, such as a list of stories within a running order, and the items within a story:

```
from mosromgr.mostypes import RunningOrder

ro = RunningOrder.from_file('123456-roCreate.mos.xml')

print(ro.ro_slug)
for story in ro.stories:
    print(story.slug)
```

Here, `ro.stories` is a list of *Story* objects. Each story has its own set of accessible properties, such as the story's *duration*, *start_time*, *end_time*, *offset* and *items*:

```
>>> story = ro.stories[0]
>>> story.duration
180.0
>>> story.start_time
datetime.datetime(2021, 4, 8, 21, 46, 30)
>>> len(story.items)
3
```

Here, the story contains 3 items, each of these is an *Item* object.

2.2.3 MOS Collection

The *API - MOS Collection* part of the module provides a wrapper class *MosCollection* which stores references to specified MOS files, strings or S3 object keys so the *MosFile* objects can be recreated when needed rather than kept in memory. Rather than using the + operator, a *merge()* method is provided:

```
from mosromgr.moscollection import MosCollection

mc = MosCollection.from_s3(bucket_name=bucket_name, prefix=prefix)

mc.merge()
```

The next page will cover some example problems and solutions to show you how you can use *mosromgr* in practice.

2.3 How-to guide

This section is a series of helpful recipes for how to do things and solve particular problems with *mosromgr*.

Note: These examples deal with MOS messages read from local files, but *MosFile* and *MosCollection* objects can also be constructed using *from_string* and *from_s3*. Refer to *API - MOS Types* and *API - MOS Collection* for more information.

2.3.1 Merging MOS files

When dealing with merging *MosFile* objects, this is done by "adding" each file to the *RunningOrder* object by using the + operator:

```
>>> from mosromgr.mostypes import RunningOrder, StoryInsert
>>> ro = RunningOrder.from_file('123456-roCreate.mos.xml')
>>> si = StoryInsert.from_file('123457-roStoryInsert.mos.xml')
>>> len(ro.stories)
10
>>> ro += si
>>> len(ro.stories)
11
```

To parse and merge a collection of MOS files, you could create a list of files (or use `glob()`), let *MosFile* classify each file, then merge each of them into the *RunningOrder*:

```
from mosromgr.mostypes import MosFile
from glob import glob

files = glob('*.mos.xml')

ro, *mosfiles = sorted(MosFile.from_file(f) for f in files)

for mf in mosfiles:
    ro += mf
```

To access the final XML, simply print the *RunningOrder* object or access the *__str__*:

```
>>> print(ro)
<mos>
  <mosID>MOS ID</mosID>
  <messageID>1234567</messageID>
  ...
>>> s = str(ro)
>>> s
<mos>
  <mosID>MOS ID</mosID>
  <messageID>1234567</messageID>
  ...
```

2.3.2 Merging MOS files using MOSCollection

The *MosCollection* class provides a wrapper for operations dealing with a collection of MOS files as part of one programme. So to merge files like in the previous example, you could do the following instead:

```
from mosromgr.moscollection import MosCollection
from glob import glob

files = glob('*.mos.xml')
mc = MosCollection.from_files(files)

mc.merge()
```

To access the final XML, simply print the *MosCollection* object or access the *__str__*:

```
>>> print(mc)
<mos>
  <mosID>MOS ID</mosID>
  <messageID>1234567</messageID>
  ...
>>> s = str(mc)
>>> s
<mos>
  <mosID>MOS ID</mosID>
  <messageID>1234567</messageID>
  ...
```

2.3.3 Accessing the properties of a running order

For example, a *RunningOrder* object could contain several *Story* objects, each containing a number of *Item* objects:

```
>>> from mosromgr.mostypes import RunningOrder
>>> ro = RunningOrder.from_file('roCreate.mos.xml')
>>> ro.stories
[<Story 1234>, <Story 1235>, <Story 1236>]
>>> [story.duration for story in ro.stories]
[10, 20, 30]
>>> ro.duration
60
>>> story = ro.stories[0]
>>> story.slug
'Some story'
>>> story.items
[<Item ITEM1>, <Item ITEM2>, <Item ITEM3>]
>>> item = story.items[0]
>>> item.slug
'Some item'
```

In the case of a *StoryAppend* object, this would contain a single story:

```
>>> from mosromgr.mostypes import StoryAppend
>>> sa = StoryAppend.from_file('roStoryAppend.mos.xml')
>>> sa.story
```

(continues on next page)

(continued from previous page)

```
<Story STORY1>
>>> sa.duration
20
```

If this *StoryAppend* object was merged with a *RunningOrder* object, the new story would be accessible in the *RunningOrder* *stories* property:

```
>>> from mosromgr.mostypes import RunningOrder, StoryAppend
>>> ro = RunningOrder.from_file('roCreate.mos.xml')
>>> sa = StoryAppend.from_file('roStoryAppend.mos.xml')
>>> len(ro.stories)
3
>>> ro += sa
>>> len(ro.stories)
4
```

Note: Note that these classes should not normally be constructed by the user, but instances of them can be found within *MosFile* objects, so the following documentation is provided as a reference to how they can be used.

Note: Note that additional information may be contained within the XML, and these elements are simply an abstraction providing easy access to certain elements. In the sprit of escape hatches and ejector seats, the original XML in which the element was found is accessible as an `xml.etree.ElementTree.Element` object for further introspection.

2.3.4 Handling Exceptions

This can be useful for handling exceptions in your own code. For example, to handle any exception generated by the library, you can catch the library's base exception *MosRoMgrException*:

```
try:
    main()
except MosRoMgrException as e:
    print(e)
```

To catch a specific exception known to be raised under certain circumstances, each exception can be imported and handled separately if required:

```
from mosromgr.mostypes import MosFile
from mosromgr.exc import MosInvalidXML, UnknownMosFileType

try:
    ro = MosFile.from_file(mosfile)
except MosInvalidXML as e:
    print("Invalid in", mosfile)
except UnknownMosFileType as e:
    print("Unknown MOS file type", mosfile)
```

In some cases, a warning is raised rather than an exception. This means that execution is continued but a warning is output, which can be suppressed using the `warnings` module.

2.3.5 Capturing warnings

If you want to catch warnings and log them (for example, during a merge), you can use `warnings.catch_warnings`:

```
with warnings.catch_warnings(record=True) as warns:
    mc.merge()

warning_messages = [str(w.message) for w in warns]
```

2.3.6 Suppressing warnings

If you are not interested in seeing or capturing warnings, you can either use a warning filter or use `warnings.catch_warnings`:

```
with warnings.catch_warnings() as warns:
    mc.merge()
```

2.3.7 Using the command line interface

The `mosromgr` command provided includes a number of subcommands. Running `mosromgr` alone will show the general help message, and running a subcommand without arguments will show the help message for that subcommand.

Detecting MOS file types

To detect the type of a MOS file, use the *mosromgr detect* command:

```
$ mosromgr detect -f 123456-roCreate.mos.xml
123456-roCreate.mos.xml: RunningOrder
```

Multiple files can be provided as arguments:

```
$ mosromgr detect -f 123456-roCreate.mos.xml 123457-roStorySend.mos.xml
123456-roCreate.mos.xml: RunningOrder
123457-roStorySend.mos.xml: StorySend
```

Wildcards can also be used:

```
$ mosromgr detect *
123456-roCreate.mos.xml: RunningOrder
123457-roStorySend.mos.xml: StorySend
...
9148627-roDelete.mos.xml: RunningOrderEnd
bbcProgrammeMetadata.xml: Unknown MOS file type
cricket: Invalid
FINAL.json: Invalid
FINAL.xml: RunningOrder (completed)
```

You can also read files from an S3 bucket. Either a specific file by key:

```
$ mosromgr detect -b my-bucket -k newsnight/20210101/123456-roCreate.mos.xml
INFO:botocore.credentials:Found credentials in environment variables.
OPENMEDIA_NCS.W1.BBC.MOS/OM_10.1253459/5744992-roCreate.mos.xml: RunningOrder
```

Or a whole folder by prefix:

```
$ mosromgr detect -b bbc-newslabs-slicer-mos-message-store -p newsnight/20210101/
INFO:botocore.credentials:Found credentials in environment variables.
newsnight/20210101/123456-roCreate.mos.xml: RunningOrder
newsnight/20210101/123457-roStorySend.mos.xml: StorySend
newsnight/20210101/123458-roStorySend.mos.xml: StorySend
newsnight/20210101/123459-roStorySend.mos.xml: StorySend
...
```

Inspecting a running order

To inspect the contents of a `roCreate` file, use the *mosromgr inspect* command:

```
$ mosromgr inspect -f 123456-roCreate.mos.xml
22:45 NEWSNIGHT 54D CORE Thu, 08.04.2021
```

Many options are available which allow for inspecting a file from an S3 bucket (`-b` and `-k`) instead of a local file (`-f`); and others which affect the output such as `-t` (start time), `-d` (duration), `-s` (stories):

```
$ mosromgr inspect -b my-bucket -k newsnight/20210804/123456-roCreate.mos.xml -tds
22:45 NEWSNIGHT 54D CORE Thu, 08.04.2021
Start time: 2021-04-08 21:46
Duration: 0:35:09

MENU START

MENU-PRE TITLE TEASE

MENU-TITLES

MENU-POST TITLE "ALSO TONIGHT"

NORTHERN IRELAND-INTRO

NORTHERN IRELAND-LEWIS PACKAGE

...

END OF PROGRAMME
```

Merging MOS files

To merge a set of MOS files for a programme, use the *mosromgr merge* command.

Merging local files:

```
$ mosromgr merge -f *.mos.xml -o FINAL.xml
...
INFO:mosromgr.moscollection:Merging RunningOrderEnd 123499
INFO:mosromgr.moscollection:Completed merging 99 mos files
Writing merged running order to FINAL.xml
```

Or files in an S3 bucket folder by prefix:

```
$ mosromgr merge -b my-bucket -p newsnight/20210101/ -o
...
INFO:mosromgr.moscollection:Merging RunningOrderEnd 123499
INFO:mosromgr.moscollection:Completed merging 99 mos files
Writing merged running order to FINAL.xml
```

2.4 API - MOS Types

This part of the module provides the classes required for classifying and managing MOS files.

MOS Type classes are typically imported like so:

```
from mosromgr.mostypes import MosFile
```

MOS objects are constructed using one of three classmethods. Either from a file path:

```
ro = RunningOrder.from_file('roCreate.mos.xml')
```

from an XML string:

```
with open('roCreate.mos.xml') as f:
    xml = f.read()

ro = RunningOrder.from_string(xml)
```

or from an S3 file key:

```
ro = RunningOrder.from_s3(bucket_name='newsnight', mos_file_key='20200101/roCreate.mos.
↪xml')
```

Similarly, objects constructed using these classmethods on the *MosFile* base class will be automatically classified and an instance of the relevant class will be created:

```
>>> ro = MosFile.from_file('roCreate.mos.xml')
>>> ro
<RunningOrder 1000>
>>> ss = MosFile.from_file('roStorySend.mos.xml')
>>> ss
<StorySend 1001>
>>> ro = MosFile.from_string(xml1)
>>> ro
<RunningOrder 1000>
>>> ss = MosFile.from_string(xml2)
>>> ss
<StorySend 1001>
```

Even `roElementAction` files, which require a number of different subclasses, can be classified this way:

```
>>> ea1 = MosFile.from_file('roElementAction1.mos.xml')
>>> ea1
<EAStorySwap 1012>
```

(continues on next page)

(continued from previous page)

```
>>> ea2 = MosFile.from_string(xml)
>>> ea2
<EAItemMove 1013>
```

2.4.1 MOS message classes

The following classes are used to parse and manage specific types of MOS messages.

RunningOrder

class mosromgr.mostypes.**RunningOrder**

Bases: *mosromgr.mostypes.MosFile*

A `RunningOrder` object is created from a `roCreate` MOS file and can be constructed using classmethods *from_file()*, *from_string()* or *from_s3()*.

Specification: Create Running Order http://mosprotocol.com/wp-content/MOS-Protocol-Documents/MOSProtocolVersion40/index.html#calibre_link-32

__add__(*other*)

`RunningOrder` objects can be merged with other MOS files which implement a `merge` method by using the + operator, for example:

```
ro = RunningOrder.from_file('roCreate.mos.xml')
ss = StorySend.from_file('roStorySend.mos.xml')
ro += ss
```

__gt__(*other*)

Sort by *message_id* i.e. `ss > ro` or `sorted([ro, ss])`

__lt__(*other*)

Sort by *message_id* i.e. `ro < ss` or `sorted([ro, ss])`

__str__()

The XML string of the MOS file

classmethod from_file(*mos_file_path*)

Construct from a path to a MOS file

Parameters **mos_file_path** (*str*) – The MOS file path

classmethod from_s3(*bucket_name*, *mos_file_key*)

Construct from a MOS file in an S3 bucket

Parameters

- **bucket_name** (*str*) – The name of the S3 bucket
- **mos_file_key** (*str*) – A MOS file key within the S3 bucket

classmethod from_string(*mos_xml_string*)

Construct from an XML string of a MOS document

Parameters **mos_xml_string** (*str*) – The XML string of the MOS document

inspect()

Print an outline of the key file contents

property base_tag
The base tag (xml.etree.ElementTree.Element) within the *xml*, as determined by *base_tag_name*

property base_tag_name
The name of the base XML tag for this file type (str)

property body
A list of elements found in the story bodies. Each item in the list is either a string (representing a <p> tag) or an *Item* object (representing an <item> tag). Unlike *script*, this does not exclude empty paragraph tags.

property completed
Whether or not the running order has had a *RunningOrderEnd* merged (bool)

property dict
Convert XML to dictionary using xmltodict library. Useful for testing. (*dict*)

property duration
Total running order duration in seconds (int)

property end_time
Transmission end time (datetime.datetime)

property message_id
The MOS file's message ID (int)

property ro_id
The running order ID (str)

property ro_slug
The running order slug (str)

property script
A list of strings found in paragraph tags within the story bodies, excluding any empty paragraphs or technical notes in brackets.

property start_time
Transmission start time (datetime.datetime) or None if not available in the XML

property stories
A list of *Story* objects within the running order

property xml
The XML element of the MOS file (xml.etree.ElementTree.Element)

StorySend

class mosromgr.mostypes.**StorySend**
Bases: *mosromgr.mostypes.MosFile*

A StorySend object is created from a roStorySend MOS file and can be constructed using classmethods *from_file()*, *from_string()* or *from_s3()*.

StorySend objects can be merged with a *RunningOrder* by using the + operator. This behaviour is defined in the *merge()* method in this class.

Specification: Send Story information, including Body of the Story http://mosprotocol.com/wp-content/MOS-Protocol-Documents/MOSProtocolVersion40/index.html#calibre_link-49

__gt__(*other*)
Sort by *message_id* i.e. `ss > ro` or `sorted([ro, ss])`

__lt__(*other*)
Sort by *message_id* i.e. `ro < ss` or `sorted([ro, ss])`

__str__()
The XML string of the MOS file

classmethod from_file(*mos_file_path*)
Construct from a path to a MOS file

Parameters **mos_file_path** (*str*) – The MOS file path

classmethod from_s3(*bucket_name*, *mos_file_key*)
Construct from a MOS file in an S3 bucket

Parameters

- **bucket_name** (*str*) – The name of the S3 bucket
- **mos_file_key** (*str*) – A MOS file key within the S3 bucket

classmethod from_string(*mos_xml_string*)
Construct from an XML string of a MOS document

Parameters **mos_xml_string** (*str*) – The XML string of the MOS document

inspect()
Print an outline of the key file contents

merge(*ro*)
Merge into the *RunningOrder* object provided.

Replaces the story tag in the running order with the one in the `roStorySend` message.

property base_tag
The base tag (`xml.etree.ElementTree.Element`) within the *xml*, as determined by *base_tag_name*

property base_tag_name
The name of the base XML tag for this file type (`str`)

property dict
Convert XML to dictionary using `xmltodict` library. Useful for testing. (*dict*)

property message_id
The MOS file's message ID (`int`)

property ro_id
The running order ID (`str`)

property story
The *Story* object being sent

property xml
The XML element of the MOS file (`xml.etree.ElementTree.Element`)

StoryReplace

class mosromgr.mostypes.**StoryReplace**
Bases: *mosromgr.mostypes.MosFile*

A `StoryReplace` object is created from a `roStoryReplace` MOS file and can be constructed using classmethods *from_file()*, *from_string()* or *from_s3()*.

`StoryReplace` objects can be merged with a *RunningOrder* by using the + operator. This behaviour is defined in the *merge()* method in this class.

Specification: Replace a Story with Another in a Running Order http://mosprotocol.com/wp-content/MOS-Protocol-Documents/MOS_Protocol_Version_2.8.5_Final.htm#roStoryReplace

__gt__(*other*)
Sort by *message_id* i.e. `ss > ro` or `sorted([ro, ss])`

__lt__(*other*)
Sort by *message_id* i.e. `ro < ss` or `sorted([ro, ss])`

__str__()
The XML string of the MOS file

classmethod from_file(*mos_file_path*)
Construct from a path to a MOS file

Parameters **mos_file_path** (*str*) – The MOS file path

classmethod from_s3(*bucket_name*, *mos_file_key*)
Construct from a MOS file in an S3 bucket

Parameters

- **bucket_name** (*str*) – The name of the S3 bucket
- **mos_file_key** (*str*) – A MOS file key within the S3 bucket

classmethod from_string(*mos_xml_string*)
Construct from an XML string of a MOS document

Parameters **mos_xml_string** (*str*) – The XML string of the MOS document

inspect()
Print an outline of the key file contents

merge(*ro*)
Merge into the *RunningOrder* object provided.

Replaces the story tag in the running order with the one in the `roStoryReplace` message.

property base_tag
The base tag (`xml.etree.ElementTree.Element`) within the *xml*, as determined by *base_tag_name*

property base_tag_name
The name of the base XML tag for this file type (`str`)

property dict
Convert XML to dictionary using `xmltodict` library. Useful for testing. (*dict*)

property message_id
The MOS file's message ID (`int`)

property ro_id
The running order ID (`str`)

property stories
A list of replacement *Story* objects

property story
The *Story* object being replaced

property xml
The XML element of the MOS file (xml.etree.ElementTree.Element)

StoryInsert

class mosromgr.mostypes.**StoryInsert**
Bases: *mosromgr.mostypes.MosFile*

A StoryInsert object is created from a roStoryInsert MOS file and can be constructed using classmethods *from_file()*, *from_string()* or *from_s3()*.

StoryInsert objects can be merged with a *RunningOrder* by using the + operator. This behaviour is defined in the *merge()* method in this class.

Specification: Insert Stories in Running Order http://mosprotocol.com/wp-content/MOS-Protocol-Documents/MOS_Protocol_Version_2.8.5_Final.htm#roStoryInsert

__gt__(*other*)
Sort by *message_id* i.e. `ss > ro` or `sorted([ro, ss])`

__lt__(*other*)
Sort by *message_id* i.e. `ro < ss` or `sorted([ro, ss])`

__str__()
The XML string of the MOS file

classmethod from_file(*mos_file_path*)
Construct from a path to a MOS file

Parameters **mos_file_path** (*str*) – The MOS file path

classmethod from_s3(*bucket_name*, *mos_file_key*)
Construct from a MOS file in an S3 bucket

Parameters

- **bucket_name** (*str*) – The name of the S3 bucket
- **mos_file_key** (*str*) – A MOS file key within the S3 bucket

classmethod from_string(*mos_xml_string*)
Construct from an XML string of a MOS document

Parameters **mos_xml_string** (*str*) – The XML string of the MOS document

inspect()
Print an outline of the key file contents

merge(*ro*)
Merge into the *RunningOrder* object provided.

Inserts the story tags from the roStoryInsert message into the running order.

property base_tag
The base tag (xml.etree.ElementTree.Element) within the *xml*, as determined by *base_tag_name*

property base_tag_name
The name of the base XML tag for this file type (str)

property dict
Convert XML to dictionary using `xmltodict` library. Useful for testing. (*dict*)

property message_id
The MOS file's message ID (`int`)

property ro_id
The running order ID (`str`)

property source_stories
A list of *Story* objects to be inserted

property target_story
The *Story* object above which the source stories are to be inserted

property xml
The XML element of the MOS file (`xml.etree.ElementTree.Element`)

StoryAppend

class mosromgr.mostypes.**StoryAppend**
Bases: *mosromgr.mostypes.MosFile*

A `StoryAppend` object is created from a `roStoryAppend` MOS file and can be constructed using classmethods *from_file()*, *from_string()* or *from_s3()*.

`StoryAppend` objects can be merged with a *RunningOrder* by using the + operator. This behaviour is defined in the *merge()* method in this class.

Specification: Append Stories to Running Order http://mosprotocol.com/wp-content/MOS-Protocol-Documents/MOS_Protocol_Version_2.8.5_Final.htm#roStoryAppend

__gt__(*other*)
Sort by *message_id* i.e. `ss > ro` or `sorted([ro, ss])`

__lt__(*other*)
Sort by *message_id* i.e. `ro < ss` or `sorted([ro, ss])`

__str__()
The XML string of the MOS file

classmethod from_file(*mos_file_path*)
Construct from a path to a MOS file

Parameters **mos_file_path** (*str*) – The MOS file path

classmethod from_s3(*bucket_name*, *mos_file_key*)
Construct from a MOS file in an S3 bucket

Parameters

- **bucket_name** (*str*) – The name of the S3 bucket
- **mos_file_key** (*str*) – A MOS file key within the S3 bucket

classmethod from_string(*mos_xml_string*)
Construct from an XML string of a MOS document

Parameters **mos_xml_string** (*str*) – The XML string of the MOS document

inspect()
Print an outline of the key file contents

merge(*ro*)
Merge into the *RunningOrder* object provided.

Adds the story tag in the `roStoryAppend` message onto the end of the running order.

property base_tag
The base tag (`xml.etree.ElementTree.Element`) within the *xml*, as determined by *base_tag_name*

property base_tag_name
The name of the base XML tag for this file type (`str`)

property dict
Convert XML to dictionary using `xmltodict` library. Useful for testing. (*dict*)

property message_id
The MOS file's message ID (`int`)

property ro_id
The running order ID (`str`)

property stories
A list of *Story* objects to be appended

property xml
The XML element of the MOS file (`xml.etree.ElementTree.Element`)

StoryMove

class mosromgr.mostypes.**StoryMove**
Bases: *mosromgr.mostypes.MosFile*

A `StoryMove` object is created from a `roStoryMove` MOS file and can be constructed using classmethods *from_file()*, *from_string()* or *from_s3()*.

`StoryMove` objects can be merged with a *RunningOrder* by using the + operator. This behaviour is defined in the *merge()* method in this class.

Specification: Move a story to a new position in the Playlist http://mosprotocol.com/wp-content/MOS-Protocol-Documents/MOS_Protocol_Version_2.8.5_Final.htm#roStoryMove

__gt__(*other*)
Sort by *message_id* i.e. `ss > ro` or `sorted([ro, ss])`

__lt__(*other*)
Sort by *message_id* i.e. `ro < ss` or `sorted([ro, ss])`

__str__()
The XML string of the MOS file

classmethod from_file(*mos_file_path*)
Construct from a path to a MOS file

Parameters **mos_file_path** (*str*) – The MOS file path

classmethod from_s3(*bucket_name*, *mos_file_key*)
Construct from a MOS file in an S3 bucket

Parameters

- **bucket_name** (*str*) – The name of the S3 bucket
- **mos_file_key** (*str*) – A MOS file key within the S3 bucket

classmethod from_string(*mos_xml_string*)
Construct from an XML string of a MOS document

Parameters **mos_xml_string** (*str*) – The XML string of the MOS document

inspect()
Print an outline of the key file contents

merge(*ro*)
Merge into the *RunningOrder* object provided.

Moves the source story to the position above the target story.

property base_tag
The base tag (`xml.etree.ElementTree.Element`) within the *xml*, as determined by *base_tag_name*

property base_tag_name
The name of the base XML tag for this file type (`str`)

property dict
Convert XML to dictionary using `xmltodict` library. Useful for testing. (*dict*)

property message_id
The MOS file's message ID (`int`)

property ro_id
The running order ID (`str`)

property source_story
The *Story* object to be moved

property target_story
The *Story* object above which the source story is to be moved

property xml
The XML element of the MOS file (`xml.etree.ElementTree.Element`)

StoryDelete

class mosromgr.mostypes.**StoryDelete**
Bases: *mosromgr.mostypes.MosFile*

A `StoryDelete` object is created from a `roStoryDelete` MOS file and can be constructed using classmethods *from_file()*, *from_string()* or *from_s3()*.

`StoryDelete` objects can be merged with a *RunningOrder* by using the + operator. This behaviour is defined in the *merge()* method in this class.

Specification: Delete Stories from Running Order http://mosprotocol.com/wp-content/MOS-Protocol-Documents/MOS_Protocol_Version_2.8.5_Final.htm#roStoryDelete

__gt__(*other*)
Sort by *message_id* i.e. `ss > ro` or `sorted([ro, ss])`

__lt__(*other*)
Sort by *message_id* i.e. `ro < ss` or `sorted([ro, ss])`

__str__()
The XML string of the MOS file

classmethod from_file(*mos_file_path*)
Construct from a path to a MOS file

Parameters **mos_file_path** (*str*) – The MOS file path

classmethod from_s3(*bucket_name*, *mos_file_key*)
Construct from a MOS file in an S3 bucket

Parameters

- **bucket_name** (*str*) – The name of the S3 bucket
- **mos_file_key** (*str*) – A MOS file key within the S3 bucket

classmethod from_string(*mos_xml_string*)
Construct from an XML string of a MOS document

Parameters **mos_xml_string** (*str*) – The XML string of the MOS document

inspect()
Print an outline of the key file contents

merge(*ro*)
Merge into the *RunningOrder* object provided.

Removes any story tags from the running order which are included in the `roStoryDelete` message.

property base_tag
The base tag (`xml.etree.ElementTree.Element`) within the *xml*, as determined by *base_tag_name*

property base_tag_name
The name of the base XML tag for this file type (`str`)

property dict
Convert XML to dictionary using `xmltodict` library. Useful for testing. (*dict*)

property message_id
The MOS file's message ID (`int`)

property ro_id
The running order ID (`str`)

property stories
A list of *Story* objects to be deleted

property xml
The XML element of the MOS file (`xml.etree.ElementTree.Element`)

MetaDataReplace

class mosromgr.mostypes.**MetaDataReplace**
Bases: *mosromgr.mostypes.MosFile*

A `MetaDataReplace` object is created from a `roMetadataReplace` MOS file and can be constructed using classmethods *from_file()*, *from_string()* or *from_s3()*.

`MetaDataReplace` objects can be merged with a *RunningOrder* by using the + operator. This behaviour is defined in the *merge()* method in this class.

Specification: Replace RO metadata without deleting the RO structure http://mosprotocol.com/wp-content/MOS-Protocol-Documents/MOSProtocolVersion40/index.html#calibre_link-34

__gt__(*other*)
Sort by *message_id* i.e. `ss > ro` or `sorted([ro, ss])`

__lt__(*other*)
Sort by *message_id* i.e. `ro < ss` or `sorted([ro, ss])`

__str__()
The XML string of the MOS file

classmethod from_file(*mos_file_path*)
Construct from a path to a MOS file

Parameters **mos_file_path** (*str*) – The MOS file path

classmethod from_s3(*bucket_name*, *mos_file_key*)
Construct from a MOS file in an S3 bucket

Parameters

- **bucket_name** (*str*) – The name of the S3 bucket
- **mos_file_key** (*str*) – A MOS file key within the S3 bucket

classmethod from_string(*mos_xml_string*)
Construct from an XML string of a MOS document

Parameters **mos_xml_string** (*str*) – The XML string of the MOS document

inspect()
Print an outline of the key file contents

merge(*ro*)
Merge into the *RunningOrder* object provided.

Replaces the metadata tags in the running order with the ones in the `MetaDataReplace` message.

property base_tag
The base tag (`xml.etree.ElementTree.Element`) within the *xml*, as determined by *base_tag_name*

property base_tag_name
The name of the base XML tag for this file type (`str`)

property dict
Convert XML to dictionary using `xmltodict` library. Useful for testing. (*dict*)

property message_id
The MOS file's message ID (`int`)

property ro_id
The running order ID (`str`)

property ro_slug
The running order slug (`str`)

property xml
The XML element of the MOS file (`xml.etree.ElementTree.Element`)

ItemDelete

class mosromgr.mostypes.**ItemDelete**
Bases: *mosromgr.mostypes.MosFile*

An `ItemDelete` object is created from a `roItemDelete` MOS file and can be constructed using classmethods *from_file()*, *from_string()* or *from_s3()*.

`ItemDelete` objects can be merged with a *RunningOrder* by using the + operator. This behaviour is defined in the *merge()* method in this class.

Specification: Delete Items in Story http://mosprotocol.com/wp-content/MOS-Protocol-Documents/MOS_Protocol_Version_2.8.5_Final.htm#roItemDelete

__gt__(*other*)
Sort by *message_id* i.e. `ss > ro` or `sorted([ro, ss])`

__lt__(*other*)
Sort by *message_id* i.e. `ro < ss` or `sorted([ro, ss])`

__str__()
The XML string of the MOS file

classmethod from_file(*mos_file_path*)
Construct from a path to a MOS file

Parameters **mos_file_path** (*str*) – The MOS file path

classmethod from_s3(*bucket_name*, *mos_file_key*)
Construct from a MOS file in an S3 bucket

Parameters

- **bucket_name** (*str*) – The name of the S3 bucket
- **mos_file_key** (*str*) – A MOS file key within the S3 bucket

classmethod from_string(*mos_xml_string*)
Construct from an XML string of a MOS document

Parameters **mos_xml_string** (*str*) – The XML string of the MOS document

inspect()
Print an outline of the key file contents

merge(*ro*)
Merge into the *RunningOrder* object provided.

Deletes any item tags with the IDs specified in the `roItemDelete` message from the running order.

property base_tag
The base tag (`xml.etree.ElementTree.Element`) within the *xml*, as determined by *base_tag_name*

property base_tag_name
The name of the base XML tag for this file type (`str`)

property dict
Convert XML to dictionary using `xmltodict` library. Useful for testing. (*dict*)

property items
A tuple of the two *Item* objects being deleted

property message_id
The MOS file's message ID (`int`)

property ro_id
The running order ID (`str`)

property story
The *Story* object containing the items being deleted

property xml
The XML element of the MOS file (`xml.etree.ElementTree.Element`)

ItemInsert

class mosromgr.mostypes.**ItemInsert**
Bases: *mosromgr.mostypes.MosFile*

An `ItemInsert` object is created from a `roItemInsert` MOS file and can be constructed using classmethods *from_file()*, *from_string()* or *from_s3()*.

`ItemInsert` objects can be merged with a *RunningOrder* by using the + operator. This behaviour is defined in the *merge()* method in this class.

Specification: Insert Items in Story http://mosprotocol.com/wp-content/MOS-Protocol-Documents/MOS_Protocol_Version_2.8.5_Final.htm#roItemInsert

__gt__(*other*)
Sort by *message_id* i.e. `ss > ro` or `sorted([ro, ss])`

__lt__(*other*)
Sort by *message_id* i.e. `ro < ss` or `sorted([ro, ss])`

__str__()
The XML string of the MOS file

classmethod from_file(*mos_file_path*)
Construct from a path to a MOS file

Parameters **mos_file_path** (*str*) – The MOS file path

classmethod from_s3(*bucket_name*, *mos_file_key*)
Construct from a MOS file in an S3 bucket

Parameters

- **bucket_name** (*str*) – The name of the S3 bucket
- **mos_file_key** (*str*) – A MOS file key within the S3 bucket

classmethod from_string(*mos_xml_string*)
Construct from an XML string of a MOS document

Parameters **mos_xml_string** (*str*) – The XML string of the MOS document

inspect()
Print an outline of the key file contents

merge(*ro*)
Merge into the *RunningOrder* object provided.

Inserts the item tags from the `roItemInsert` message into the relevant story in the running order.

property base_tag
The base tag (`xml.etree.ElementTree.Element`) within the *xml*, as determined by *base_tag_name*

property base_tag_name
The name of the base XML tag for this file type (`str`)

property dict
Convert XML to dictionary using `xmltodict` library. Useful for testing. (*dict*)

property item
The *Item* object above which the items are to be inserted

property items
A list of *Item* objects to be inserted

property message_id
The MOS file's message ID (int)

property ro_id
The running order ID (str)

property story
The *Story* object into which the items are to be inserted

property xml
The XML element of the MOS file (xml.etree.ElementTree.Element)

ItemMoveMultiple

class mosromgr.mostypes.**ItemMoveMultiple**
Bases: *mosromgr.mostypes.MosFile*

An ItemMoveMultiple object is created from a roItemMoveMultiple MOS file and can be constructed using classmethods *from_file()*, *from_string()* or *from_s3()*.

ItemMoveMultiple objects can be merged with a *RunningOrder* by using the + operator. This behaviour is defined in the *merge()* method in this class.

Specification: Move one or more Items to a specified position within a Story http://mosprotocol.com/wp-content/MOS-Protocol-Documents/MOS_Protocol_Version_2.8.5_Final.htm#roItemMoveMultiple

__gt__(*other*)
Sort by *message_id* i.e. `ss > ro` or `sorted([ro, ss])`

__lt__(*other*)
Sort by *message_id* i.e. `ro < ss` or `sorted([ro, ss])`

__str__()
The XML string of the MOS file

classmethod from_file(*mos_file_path*)
Construct from a path to a MOS file

Parameters **mos_file_path** (*str*) – The MOS file path

classmethod from_s3(*bucket_name*, *mos_file_key*)
Construct from a MOS file in an S3 bucket

Parameters

- **bucket_name** (*str*) – The name of the S3 bucket
- **mos_file_key** (*str*) – A MOS file key within the S3 bucket

classmethod from_string(*mos_xml_string*)
Construct from an XML string of a MOS document

Parameters **mos_xml_string** (*str*) – The XML string of the MOS document

inspect()
Print an outline of the key file contents

merge(*ro*)
Merge into the *RunningOrder* object provided.

Moves item tags in the roItemMove message to a new position within the story.

property base_tag
The base tag (xml.etree.ElementTree.Element) within the *xml*, as determined by *base_tag_name*

property base_tag_name
The name of the base XML tag for this file type (str)

property dict
Convert XML to dictionary using xmltodict library. Useful for testing. (*dict*)

property item
The *Item* object above which the items will be moved

property items
A list of *Item* objects to be moved

property message_id
The MOS file's message ID (int)

property ro_id
The running order ID (str)

property story
The *Story* object containing the items being moved

property xml
The XML element of the MOS file (xml.etree.ElementTree.Element)

ItemReplace

class mosromgr.mostypes.**ItemReplace**
Bases: *mosromgr.mostypes.MosFile*

An ItemReplace object is created from a roItemReplace MOS file and can be constructed using classmethods *from_file()*, *from_string()* or *from_s3()*.

ItemReplace objects can be merged with a *RunningOrder* by using the + operator. This behaviour is defined in the *merge()* method in this class.

Specification: Replace an Item with one or more Items in a Story http://mosprotocol.com/wp-content/MOS-Protocol-Documents/MOS_Protocol_Version_2.8.5_Final.htm#roItemReplace

__gt__(*other*)
Sort by *message_id* i.e. `ss > ro` or `sorted([ro, ss])`

__lt__(*other*)
Sort by *message_id* i.e. `ro < ss` or `sorted([ro, ss])`

__str__()
The XML string of the MOS file

classmethod from_file(*mos_file_path*)
Construct from a path to a MOS file

Parameters **mos_file_path** (*str*) – The MOS file path

classmethod from_s3(*bucket_name*, *mos_file_key*)
Construct from a MOS file in an S3 bucket

Parameters

- **bucket_name** (*str*) – The name of the S3 bucket
- **mos_file_key** (*str*) – A MOS file key within the S3 bucket

classmethod from_string(*mos_xml_string*)
Construct from an XML string of a MOS document

Parameters **mos_xml_string** (*str*) – The XML string of the MOS document

inspect()
Print an outline of the key file contents

merge(*ro*)
Merge into the *RunningOrder* object provided.

Replaces the item tag in the story in the running order with the ones in the `roItemReplace` message

property base_tag
The base tag (`xml.etree.ElementTree.Element`) within the *xml*, as determined by *base_tag_name*

property base_tag_name
The name of the base XML tag for this file type (`str`)

property dict
Convert XML to dictionary using `xmltodict` library. Useful for testing. (*dict*)

property item
The *Item* object being replaced

property items
A list of replacement *Item* objects

property message_id
The MOS file's message ID (`int`)

property ro_id
The running order ID (`str`)

property story
The *Story* object containing the item being replaced

property xml
The XML element of the MOS file (`xml.etree.ElementTree.Element`)

ReadyToAir

class mosromgr.mostypes.**ReadyToAir**
Bases: *mosromgr.mostypes.MosFile*

A `ReadyToAir` object is created from a `roReadyToAir` MOS file and can be constructed using classmethods *from_file()*, *from_string()* or *from_s3()*.

`ReadyToAir` objects can be merged with a *RunningOrder* by using the + operator. This behaviour is defined in the *merge()* method in this class.

Specification: Identify a Running Order as Ready to Air http://mosprotocol.com/wp-content/MOS-Protocol-Documents/MOSProtocolVersion40/index.html#calibre_link-41

__gt__(*other*)
Sort by *message_id* i.e. `ss > ro` or `sorted([ro, ss])`

__lt__(*other*)
Sort by *message_id* i.e. `ro < ss` or `sorted([ro, ss])`

__str__()
The XML string of the MOS file

classmethod from_file(*mos_file_path*)
Construct from a path to a MOS file

Parameters **mos_file_path** (*str*) – The MOS file path

classmethod from_s3(*bucket_name*, *mos_file_key*)
Construct from a MOS file in an S3 bucket

Parameters

- **bucket_name** (*str*) – The name of the S3 bucket
- **mos_file_key** (*str*) – A MOS file key within the S3 bucket

classmethod from_string(*mos_xml_string*)
Construct from an XML string of a MOS document

Parameters **mos_xml_string** (*str*) – The XML string of the MOS document

inspect()
Print an outline of the key file contents

merge(*ro*)
Merge into the *RunningOrder* object provided.

Currently unimplemented - has no effect on the running order. TODO: #18

property base_tag
The base tag (`xml.etree.ElementTree.Element`) within the *xml*, as determined by *base_tag_name*

property base_tag_name
The name of the base XML tag for this file type (`str`)

property dict
Convert XML to dictionary using `xmltodict` library. Useful for testing. (*dict*)

property message_id
The MOS file's message ID (`int`)

property ro_id
The running order ID (`str`)

property xml
The XML element of the MOS file (`xml.etree.ElementTree.Element`)

EAStoryReplace

class mosromgr.mostypes.**EAStoryReplace**
Bases: *mosromgr.mostypes.ElementAction*

An `EAStoryReplace` object is created from a `roElementAction` MOS file containing a story replacement, and can be constructed using classmethods *from_file()*, *from_string()* or *from_s3()*.

`EAStoryReplace` objects can be merged with a *RunningOrder* by using the + operator. This behaviour is defined in the *merge()* method in this class.

Specification: Replacing a story http://mosprotocol.com/wp-content/MOS-Protocol-Documents/MOSProtocolVersion40/index.html#calibre_link-43

__gt__(*other*)
Sort by *message_id* i.e. `ss > ro` or `sorted([ro, ss])`

__lt__(*other*)
Sort by *message_id* i.e. `ro < ss` or `sorted([ro, ss])`

__str__()
The XML string of the MOS file

classmethod from_file(*mos_file_path*)
Construct from a path to a MOS file

Parameters **mos_file_path** (*str*) – The MOS file path

classmethod from_s3(*bucket_name*, *mos_file_key*)
Construct from a MOS file in an S3 bucket

Parameters

- **bucket_name** (*str*) – The name of the S3 bucket
- **mos_file_key** (*str*) – A MOS file key within the S3 bucket

classmethod from_string(*mos_xml_string*)
Construct from an XML string of a MOS document

Parameters **mos_xml_string** (*str*) – The XML string of the MOS document

inspect()
Print an outline of the key file contents

merge(*ro*)
Merge into the *RunningOrder* object provided.

Replaces the `element_target` story tag in the running order with any story tags found in the `element_source` in the `roElementAction` message.

property base_tag
The base tag (`xml.etree.ElementTree.Element`) within the *xml*, as determined by *base_tag_name*

property base_tag_name
The name of the base XML tag for this file type (`str`)

property dict
Convert XML to dictionary using `xmltodict` library. Useful for testing. (*dict*)

property message_id
The MOS file's message ID (`int`)

property ro_id
The running order ID (`str`)

property stories
A list of replacement *Story* objects

property story
The *Story* object being replaced

property xml
The XML element of the MOS file (`xml.etree.ElementTree.Element`)

EAItemReplace

class mosromgr.mostypes.**EAItemReplace**
Bases: *mosromgr.mostypes.ElementAction*

An `EAItemReplace` object is created from a `roElementAction` MOS file containing an item replacement, and can be constructed using classmethods *from_file()*, *from_string()* or *from_s3()*.

`EAItemReplace` objects can be merged with a *RunningOrder* by using the + operator. This behaviour is defined in the *merge()* method in this class.

Specification: Replacing an item http://mosprotocol.com/wp-content/MOS-Protocol-Documents/MOSProtocolVersion40/index.html#calibre_link-43

__gt__(*other*)
Sort by *message_id* i.e. `ss > ro` or `sorted([ro, ss])`

__lt__(*other*)
Sort by *message_id* i.e. `ro < ss` or `sorted([ro, ss])`

__str__()
The XML string of the MOS file

classmethod from_file(*mos_file_path*)
Construct from a path to a MOS file

Parameters **mos_file_path** (*str*) – The MOS file path

classmethod from_s3(*bucket_name*, *mos_file_key*)
Construct from a MOS file in an S3 bucket

Parameters

- **bucket_name** (*str*) – The name of the S3 bucket
- **mos_file_key** (*str*) – A MOS file key within the S3 bucket

classmethod from_string(*mos_xml_string*)
Construct from an XML string of a MOS document

Parameters **mos_xml_string** (*str*) – The XML string of the MOS document

inspect()
Print an outline of the key file contents

merge(*ro*)
Merge into the *RunningOrder* object provided.

Replaces the target item tag in the target story in the running order with any item tags found in the `element_source` in the `roElementAction` message.

property base_tag
The base tag (`xml.etree.ElementTree.Element`) within the *xml*, as determined by *base_tag_name*

property base_tag_name
The name of the base XML tag for this file type (`str`)

property dict
Convert XML to dictionary using `xmltodict` library. Useful for testing. (*dict*)

property item
The *Item* object being replaced

property items
A list of replacement *Item* objects

property message_id
The MOS file's message ID (`int`)

property ro_id
The running order ID (`str`)

property story
The *Story* object containing the item being replaced

property xml
The XML element of the MOS file (xml.etree.ElementTree.Element)

EAStoryDelete

class mosromgr.mostypes.**EAStoryDelete**
Bases: *mosromgr.mostypes.ElementAction*

An EAStoryDelete object is created from a roElementAction MOS file containing a story deletion, and can be constructed using classmethods *from_file()*, *from_string()* or *from_s3()*.

EAStoryDelete objects can be merged with a *RunningOrder* by using the + operator. This behaviour is defined in the *merge()* method in this class.

Specification: Deleting stories http://mosprotocol.com/wp-content/MOS-Protocol-Documents/MOSProtocolVersion40/index.html#calibre_link-43

__gt__(*other*)
Sort by *message_id* i.e. ss > ro or sorted([ro, ss])

__lt__(*other*)
Sort by *message_id* i.e. ro < ss or sorted([ro, ss])

__str__()
The XML string of the MOS file

classmethod from_file(*mos_file_path*)
Construct from a path to a MOS file

Parameters **mos_file_path** (*str*) – The MOS file path

classmethod from_s3(*bucket_name*, *mos_file_key*)
Construct from a MOS file in an S3 bucket

Parameters

- **bucket_name** (*str*) – The name of the S3 bucket
- **mos_file_key** (*str*) – A MOS file key within the S3 bucket

classmethod from_string(*mos_xml_string*)
Construct from an XML string of a MOS document

Parameters **mos_xml_string** (*str*) – The XML string of the MOS document

inspect()
Print an outline of the key file contents

merge(*ro*)
Merge into the *RunningOrder* object provided.

Removes any stories specified in element_source in the roElementAction message from the running order.

property base_tag
The base tag (xml.etree.ElementTree.Element) within the *xml*, as determined by *base_tag_name*

property base_tag_name
The name of the base XML tag for this file type (str)

property dict
Convert XML to dictionary using xmltodict library. Useful for testing. (*dict*)

property message_id
The MOS file's message ID (int)

property ro_id
The running order ID (str)

property stories
A list of *Story* objects to be deleted

property xml
The XML element of the MOS file (xml.etree.ElementTree.Element)

EAItemDelete

class mosromgr.mostypes.**EAItemDelete**
Bases: *mosromgr.mostypes.ElementAction*

An `EAItemDelete` object is created from a `roElementAction` MOS file containing an item deletion, and can be constructed using classmethods *from_file()*, *from_string()* or *from_s3()*.

`EAItemDelete` objects can be merged with a *RunningOrder* by using the + operator. This behaviour is defined in the *merge()* method in this class.

Specification: Deleting items http://mosprotocol.com/wp-content/MOS-Protocol-Documents/MOSProtocolVersion40/index.html#calibre_link-43

__gt__(*other*)
Sort by *message_id* i.e. `ss > ro` or `sorted([ro, ss])`

__lt__(*other*)
Sort by *message_id* i.e. `ro < ss` or `sorted([ro, ss])`

__str__()
The XML string of the MOS file

classmethod from_file(*mos_file_path*)
Construct from a path to a MOS file

Parameters **mos_file_path** (*str*) – The MOS file path

classmethod from_s3(*bucket_name*, *mos_file_key*)
Construct from a MOS file in an S3 bucket

Parameters

- **bucket_name** (*str*) – The name of the S3 bucket
- **mos_file_key** (*str*) – A MOS file key within the S3 bucket

classmethod from_string(*mos_xml_string*)
Construct from an XML string of a MOS document

Parameters **mos_xml_string** (*str*) – The XML string of the MOS document

inspect()
Print an outline of the key file contents

merge(*ro*)
Merge into the *RunningOrder* object provided.

Deletes any items specified in the `element_target` in the `roStorySend` message from the specified story in the running order.

property base_tag
The base tag (xml.etree.ElementTree.Element) within the *xml*, as determined by *base_tag_name*

property base_tag_name
The name of the base XML tag for this file type (str)

property dict
Convert XML to dictionary using xmltodict library. Useful for testing. (*dict*)

property items
A list of *Item* objects being deleted

property message_id
The MOS file's message ID (int)

property ro_id
The running order ID (str)

property story
The *Story* object containing the items being deleted

property xml
The XML element of the MOS file (xml.etree.ElementTree.Element)

EAStoryInsert

class mosromgr.mostypes.**EAStoryInsert**
Bases: *mosromgr.mostypes.ElementAction*

An EAStoryInsert object is created from a roElementAction MOS file containing a story insertion, and can be constructed using classmethods *from_file()*, *from_string()* or *from_s3()*.

EAStoryInsert objects can be merged with a *RunningOrder* by using the + operator. This behaviour is defined in the *merge()* method in this class.

Specification: Inserting stories http://mosprotocol.com/wp-content/MOS-Protocol-Documents/MOSProtocolVersion40/index.html#calibre_link-43

__gt__(*other*)
Sort by *message_id* i.e. `ss > ro` or `sorted([ro, ss])`

__lt__(*other*)
Sort by *message_id* i.e. `ro < ss` or `sorted([ro, ss])`

__str__()
The XML string of the MOS file

classmethod from_file(*mos_file_path*)
Construct from a path to a MOS file

Parameters **mos_file_path** (*str*) – The MOS file path

classmethod from_s3(*bucket_name*, *mos_file_key*)
Construct from a MOS file in an S3 bucket

Parameters

- **bucket_name** (*str*) – The name of the S3 bucket
- **mos_file_key** (*str*) – A MOS file key within the S3 bucket

classmethod from_string(*mos_xml_string*)
Construct from an XML string of a MOS document

Parameters **mos_xml_string** (*str*) – The XML string of the MOS document

inspect()
Print an outline of the key file contents

merge(*ro*)
Merge into the *RunningOrder* object provided.

Inserts any story tags found in the `element_source` in the `roElementAction` message into the running order.

property base_tag
The base tag (`xml.etree.ElementTree.Element`) within the *xml*, as determined by *base_tag_name*

property base_tag_name
The name of the base XML tag for this file type (`str`)

property dict
Convert XML to dictionary using `xmltodict` library. Useful for testing. (*dict*)

property message_id
The MOS file's message ID (`int`)

property ro_id
The running order ID (`str`)

property stories
The *Story* objects to be inserted

property story
The *Story* object above which the source story will be inserted

property xml
The XML element of the MOS file (`xml.etree.ElementTree.Element`)

EAItemInsert

class mosromgr.mostypes.**EAItemInsert**
Bases: *mosromgr.mostypes.ElementAction*

An `EAItemInsert` object is created from a `roElementAction` MOS file containing an item insertion, and can be constructed using classmethods *from_file()*, *from_string()* or *from_s3()*.

`EAItemInsert` objects can be merged with a *RunningOrder* by using the + operator. This behaviour is defined in the *merge()* method in this class.

Specification: Inserting items http://mosprotocol.com/wp-content/MOS-Protocol-Documents/MOSProtocolVersion40/index.html#calibre_link-43

__gt__(*other*)
Sort by *message_id* i.e. `ss > ro` or `sorted([ro, ss])`

__lt__(*other*)
Sort by *message_id* i.e. `ro < ss` or `sorted([ro, ss])`

__str__()
The XML string of the MOS file

classmethod from_file(*mos_file_path*)
Construct from a path to a MOS file

Parameters **mos_file_path** (*str*) – The MOS file path

classmethod from_s3(*bucket_name*, *mos_file_key*)
Construct from a MOS file in an S3 bucket

Parameters

- **bucket_name** (*str*) – The name of the S3 bucket
- **mos_file_key** (*str*) – A MOS file key within the S3 bucket

classmethod from_string(*mos_xml_string*)
Construct from an XML string of a MOS document

Parameters **mos_xml_string** (*str*) – The XML string of the MOS document

inspect()
Print an outline of the key file contents

merge(*ro*)
Merge into the *RunningOrder* object provided.

Inserts any item tags found in the `element_source` in the `roElementAction` message into the relevant story in the running order.

property base_tag
The base tag (`xml.etree.ElementTree.Element`) within the *xml*, as determined by *base_tag_name*

property base_tag_name
The name of the base XML tag for this file type (`str`)

property dict
Convert XML to dictionary using `xmltodict` library. Useful for testing. (*dict*)

property item
The *Item* object above which the source item is to be be inserted

property items
A list of *Item* objects to be inserted

property message_id
The MOS file's message ID (`int`)

property ro_id
The running order ID (`str`)

property story
The *Story* object into which the item is to be inserted

property xml
The XML element of the MOS file (`xml.etree.ElementTree.Element`)

EAStorySwap

class mosromgr.mostypes.**EAStorySwap**
Bases: *mosromgr.mostypes.ElementAction*

An `EAStorySwap` object is created from a `roElementAction` MOS file containing a story swap, and can be constructed using classmethods *from_file()*, *from_string()* or *from_s3()*.

`EAStorySwap` objects can be merged with a *RunningOrder* by using the + operator. This behaviour is defined in the *merge()* method in this class.

Specification: Swapping stories http://mosprotocol.com/wp-content/MOS-Protocol-Documents/MOSProtocolVersion40/index.html#calibre_link-43

__gt__(*other*)
Sort by *message_id* i.e. `ss > ro` or `sorted([ro, ss])`

__lt__(*other*)
Sort by *message_id* i.e. `ro < ss` or `sorted([ro, ss])`

__str__()
The XML string of the MOS file

classmethod from_file(*mos_file_path*)
Construct from a path to a MOS file

Parameters **mos_file_path** (*str*) – The MOS file path

classmethod from_s3(*bucket_name*, *mos_file_key*)
Construct from a MOS file in an S3 bucket

Parameters

- **bucket_name** (*str*) – The name of the S3 bucket
- **mos_file_key** (*str*) – A MOS file key within the S3 bucket

classmethod from_string(*mos_xml_string*)
Construct from an XML string of a MOS document

Parameters **mos_xml_string** (*str*) – The XML string of the MOS document

inspect()
Print an outline of the key file contents

merge(*ro*)
Merge into the *RunningOrder* object provided.

Swaps the order of the two story tags specified in `element_source` in the `roElementAction` message in the running order.

property base_tag
The base tag (`xml.etree.ElementTree.Element`) within the *xml*, as determined by *base_tag_name*

property base_tag_name
The name of the base XML tag for this file type (`str`)

property dict
Convert XML to dictionary using `xmltodict` library. Useful for testing. (*dict*)

property message_id
The MOS file's message ID (`int`)

property ro_id
The running order ID (`str`)

property stories
A tuple of the two *Story* objects to be swapped

property xml
The XML element of the MOS file (`xml.etree.ElementTree.Element`)

EAItemSwap

class mosromgr.mostypes.**EAItemSwap**
Bases: *mosromgr.mostypes.ElementAction*

An `EAItemSwap` object is created from a `roElementAction` MOS file containing an item swap, and can be constructed using classmethods *from_file()*, *from_string()* or *from_s3()*.

`EAItemSwap` objects can be merged with a *RunningOrder* by using the + operator. This behaviour is defined in the *merge()* method in this class.

Specification: Swapping items http://mosprotocol.com/wp-content/MOS-Protocol-Documents/MOSProtocolVersion40/index.html#calibre_link-43

__gt__(*other*)
Sort by *message_id* i.e. `ss > ro` or `sorted([ro, ss])`

__lt__(*other*)
Sort by *message_id* i.e. `ro < ss` or `sorted([ro, ss])`

__str__()
The XML string of the MOS file

classmethod from_file(*mos_file_path*)
Construct from a path to a MOS file

Parameters **mos_file_path** (*str*) – The MOS file path

classmethod from_s3(*bucket_name*, *mos_file_key*)
Construct from a MOS file in an S3 bucket

Parameters

- **bucket_name** (*str*) – The name of the S3 bucket
- **mos_file_key** (*str*) – A MOS file key within the S3 bucket

classmethod from_string(*mos_xml_string*)
Construct from an XML string of a MOS document

Parameters **mos_xml_string** (*str*) – The XML string of the MOS document

inspect()
Print an outline of the key file contents

merge(*ro*)
Merge into the *RunningOrder* object provided.

Swaps the order of the two item tags specified in `element_source` in the `roElementAction` message in the relevant story in the running order.

property base_tag
The base tag (`xml.etree.ElementTree.Element`) within the *xml*, as determined by *base_tag_name*

property base_tag_name
The name of the base XML tag for this file type (`str`)

property dict
Convert XML to dictionary using `xmltodict` library. Useful for testing. (*dict*)

property items
A tuple of the two *Item* objects to be swapped

property message_id
The MOS file's message ID (`int`)

property ro_id
The running order ID (str)

property story
The *Story* object containing the items being swapped

property xml
The XML element of the MOS file (xml.etree.ElementTree.Element)

EAStoryMove

class mosromgr.mostypes.**EAStoryMove**
Bases: *mosromgr.mostypes.ElementAction*

An EAStoryMove object is created from a roElementAction MOS file containing a story move, and can be constructed using classmethods *from_file()*, *from_string()* or *from_s3()*.

EAStoryMove objects can be merged with a *RunningOrder* by using the + operator. This behaviour is defined in the *merge()* method in this class.

Specification: Moving stories http://mosprotocol.com/wp-content/MOS-Protocol-Documents/MOSProtocolVersion40/index.html#calibre_link-43

__gt__(*other*)
Sort by *message_id* i.e. `ss > ro` or `sorted([ro, ss])`

__lt__(*other*)
Sort by *message_id* i.e. `ro < ss` or `sorted([ro, ss])`

__str__()
The XML string of the MOS file

classmethod from_file(*mos_file_path*)
Construct from a path to a MOS file

Parameters **mos_file_path** (*str*) – The MOS file path

classmethod from_s3(*bucket_name*, *mos_file_key*)
Construct from a MOS file in an S3 bucket

Parameters

- **bucket_name** (*str*) – The name of the S3 bucket
- **mos_file_key** (*str*) – A MOS file key within the S3 bucket

classmethod from_string(*mos_xml_string*)
Construct from an XML string of a MOS document

Parameters **mos_xml_string** (*str*) – The XML string of the MOS document

inspect()
Print an outline of the key file contents

merge(*ro*)
Merge into the *RunningOrder* object provided.

Moves story tags in element_source to the specified location in the running order.

property base_tag
The base tag (xml.etree.ElementTree.Element) within the *xml*, as determined by *base_tag_name*

property base_tag_name
The name of the base XML tag for this file type (str)

property dict
Convert XML to dictionary using `xmltodict` library. Useful for testing. (*dict*)

property message_id
The MOS file's message ID (`int`)

property ro_id
The running order ID (`str`)

property stories
A list of *Story* objects being moved

property story
The *Story* object above which the other stories will be moved

property xml
The XML element of the MOS file (`xml.etree.ElementTree.Element`)

EAItemMove

class mosromgr.mostypes.**EAItemMove**
Bases: *mosromgr.mostypes.ElementAction*

An `EAItemMove` object is created from a `roElementAction` MOS file containing an item move, and can be constructed using classmethods *from_file()*, *from_string()* or *from_s3()*.

`EAItemMove` objects can be merged with a *RunningOrder* by using the + operator. This behaviour is defined in the *merge()* method in this class.

Specification: Moving items http://mosprotocol.com/wp-content/MOS-Protocol-Documents/MOSProtocolVersion40/index.html#calibre_link-43

__gt__(*other*)
Sort by *message_id* i.e. `ss > ro` or `sorted([ro, ss])`

__lt__(*other*)
Sort by *message_id* i.e. `ro < ss` or `sorted([ro, ss])`

__str__()
The XML string of the MOS file

classmethod from_file(*mos_file_path*)
Construct from a path to a MOS file

Parameters **mos_file_path** (*str*) – The MOS file path

classmethod from_s3(*bucket_name*, *mos_file_key*)
Construct from a MOS file in an S3 bucket

Parameters

- **bucket_name** (*str*) – The name of the S3 bucket
- **mos_file_key** (*str*) – A MOS file key within the S3 bucket

classmethod from_string(*mos_xml_string*)
Construct from an XML string of a MOS document

Parameters **mos_xml_string** (*str*) – The XML string of the MOS document

inspect()
Print an outline of the key file contents

merge(*ro*)
Merge into the *RunningOrder* object provided.

Moves item tags in `element_source` to the specified location in the story in the running order.

property base_tag
The base tag (`xml.etree.ElementTree.Element`) within the *xml*, as determined by *base_tag_name*

property base_tag_name
The name of the base XML tag for this file type (`str`)

property dict
Convert XML to dictionary using `xmltodict` library. Useful for testing. (*dict*)

property item
The *Item* object above which the source items will be moved

property items
A list of *Item* objects to be moved

property message_id
The MOS file's message ID (`int`)

property ro_id
The running order ID (`str`)

property story
The *Story* object containing the item being replaced

property xml
The XML element of the MOS file (`xml.etree.ElementTree.Element`)

RunningOrderReplace

class mosromgr.mostypes.**RunningOrderReplace**
Bases: *mosromgr.mostypes.RunningOrder*

An `RunningOrderReplace` object is created from a `roReplace` MOS file and can be constructed using class-methods *from_file()*, *from_string()* or *from_s3()*.

`RunningOrderReplace` objects can be merged with a *RunningOrder* by using the + operator. This behaviour is defined in the *merge()* method in this class.

Specification: Replace Running Order http://mosprotocol.com/wp-content/MOS-Protocol-Documents/MOSProtocolVersion40/index.html#calibre_link-33

__add__(*other*)
`RunningOrder` objects can be merged with other MOS files which implement a `merge` method by using the + operator, for example:

```
ro = RunningOrder.from_file('roCreate.mos.xml')
ss = StorySend.from_file('roStorySend.mos.xml')
ro += ss
```

__gt__(*other*)
Sort by *message_id* i.e. `ss > ro` or `sorted([ro, ss])`

__lt__(*other*)
Sort by *message_id* i.e. `ro < ss` or `sorted([ro, ss])`

__str__()
The XML string of the MOS file

classmethod from_file(*mos_file_path*)
Construct from a path to a MOS file

Parameters **mos_file_path** (*str*) – The MOS file path

classmethod from_s3(*bucket_name*, *mos_file_key*)
Construct from a MOS file in an S3 bucket

Parameters

- **bucket_name** (*str*) – The name of the S3 bucket
- **mos_file_key** (*str*) – A MOS file key within the S3 bucket

classmethod from_string(*mos_xml_string*)
Construct from an XML string of a MOS document

Parameters **mos_xml_string** (*str*) – The XML string of the MOS document

inspect()
Print an outline of the key file contents

merge(*ro*)
Merge into the *RunningOrder* object provided.

Replaces the entire `roCreate` tag in the running order with the one in the `roReplace` message.

property base_tag
The base tag (`xml.etree.ElementTree.Element`) within the *xml*, as determined by *base_tag_name*

property base_tag_name
The name of the base XML tag for this file type (`str`)

property body
A list of elements found in the story bodies. Each item in the list is either a string (representing a <p> tag) or an *Item* object (representing an <item> tag). Unlike *script*, this does not exclude empty paragraph tags.

property dict
Convert XML to dictionary using `xmltodict` library. Useful for testing. (*dict*)

property duration
Total running order duration in seconds (`int`)

property end_time
Transmission end time (`datetime.datetime`)

property message_id
The MOS file's message ID (`int`)

property ro_id
The running order ID (`str`)

property ro_slug
The running order slug (`str`)

property script
A list of strings found in paragraph tags within the story bodies, excluding any empty paragraphs or technical notes in brackets.

property start_time
Transmission start time (`datetime.datetime`) or `None` if not available in the XML

property stories
A list of *Story* objects within the running order

property xml
The XML element of the MOS file (xml.etree.ElementTree.Element)

RunningOrderEnd

class mosromgr.mostypes.**RunningOrderEnd**
Bases: *mosromgr.mostypes.MosFile*

A RunningOrderEnd object is created from a roDelete MOS file and can be constructed using classmethods *from_file()*, *from_string()* or *from_s3()*.

RunningOrderEnd objects can be merged with a *RunningOrder* by using the + operator. This behaviour is defined in the *merge()* method in this class. Once a RunningOrderEnd object has been merged into a *RunningOrder*, the running order is considered "completed" and no further messages can be merged (with the exception of *RunningOrderControl*).

Specification: Delete Running Order http://mosprotocol.com/wp-content/MOS-Protocol-Documents/MOSProtocolVersion40/index.html#calibre_link-35

__gt__(*other*)
Sort by *message_id* i.e. `ss > ro` or `sorted([ro, ss])`

__lt__(*other*)
Sort by *message_id* i.e. `ro < ss` or `sorted([ro, ss])`

__str__()
The XML string of the MOS file

classmethod from_file(*mos_file_path*)
Construct from a path to a MOS file

Parameters **mos_file_path** (*str*) – The MOS file path

classmethod from_s3(*bucket_name*, *mos_file_key*)
Construct from a MOS file in an S3 bucket

Parameters

- **bucket_name** (*str*) – The name of the S3 bucket
- **mos_file_key** (*str*) – A MOS file key within the S3 bucket

classmethod from_string(*mos_xml_string*)
Construct from an XML string of a MOS document

Parameters **mos_xml_string** (*str*) – The XML string of the MOS document

inspect()
Print an outline of the key file contents

merge(*ro*)
Merge into the *RunningOrder* object provided.

Adds a mosromgrmeta tag containing the roDelete tag from the roDelete message to the roCreate tag in the running order.

property base_tag
The base tag (xml.etree.ElementTree.Element) within the *xml*, as determined by *base_tag_name*

property base_tag_name
The name of the base XML tag for this file type (str)

property dict
Convert XML to dictionary using xmltodict library. Useful for testing. (*dict*)

property message_id
The MOS file's message ID (int)

property ro_id
The running order ID (str)

property xml
The XML element of the MOS file (xml.etree.ElementTree.Element)

RunningOrderControl

class mosromgr.mostypes.**RunningOrderControl**
Bases: *mosromgr.mostypes.MosFile*

A RunningOrderControl object is created from a roCtrl MOS file and can be constructed using classmethods *from_file()*, *from_string()* or *from_s3()*.

Specification: Running Order Control http://mosprotocol.com/wp-content/MOS-Protocol-Documents/MOSProtocolVersion40/index.html#calibre_link-47

TODO: generalise this class #20

__gt__(*other*)
Sort by *message_id* i.e. ss > ro or sorted([ro, ss])

__lt__(*other*)
Sort by *message_id* i.e. ro < ss or sorted([ro, ss])

__str__()
The XML string of the MOS file

classmethod from_file(*mos_file_path*)
Construct from a path to a MOS file

Parameters **mos_file_path** (*str*) – The MOS file path

classmethod from_s3(*bucket_name*, *mos_file_key*)
Construct from a MOS file in an S3 bucket

Parameters

- **bucket_name** (*str*) – The name of the S3 bucket
- **mos_file_key** (*str*) – A MOS file key within the S3 bucket

classmethod from_string(*mos_xml_string*)
Construct from an XML string of a MOS document

Parameters **mos_xml_string** (*str*) – The XML string of the MOS document

inspect()
Print an outline of the key file contents

merge(*ro*)
Merge into the *RunningOrder* object provided.

Replaces the story tag in the running order with the one in the roStorySend message

property base_tag
The base tag (xml.etree.ElementTree.Element) within the *xml*, as determined by *base_tag_name*

property base_tag_name
The name of the base XML tag for this file type (str)

property dict
Convert XML to dictionary using xmltodict library. Useful for testing. (*dict*)

property message_id
The MOS file's message ID (int)

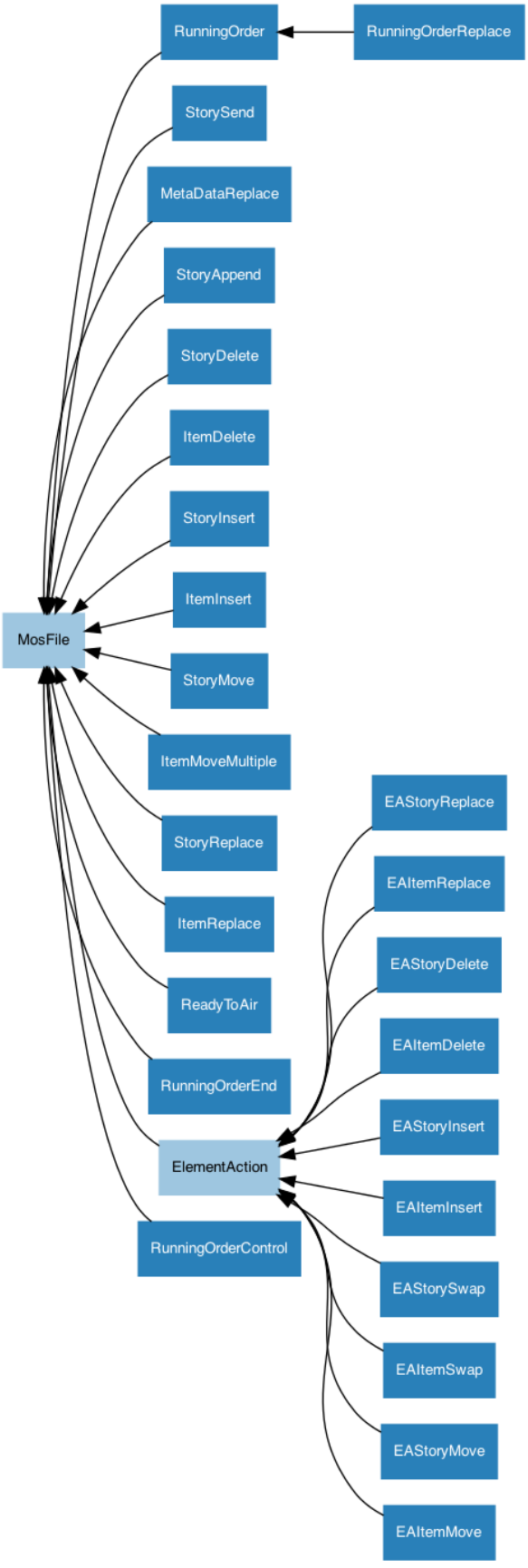
property ro_id
The running order ID (str)

property story
The story to which this roCtrl message relates

property xml
The XML element of the MOS file (xml.etree.ElementTree.Element)

2.4.2 Base classes

Since some logic is shared between MOS file management, some inheritance is used in the implementation:


RunningOrder
RunningOrderReplace
StorySend
MetaDataReplace
StoryAppend
StoryDelete
ItemDelete
StoryInsert
ItemInsert
MosFile
StoryMove
ItemMoveMultiple
StoryReplace
ItemReplace
ReadyToAir
RunningOrderEnd
ElementAction
RunningOrderControl
EAStoryReplace
EAItemReplace
EAStoryDelete
EAItemDelete
EAStoryInsert
EAItemInsert
EAStorySwap
EAItemSwap
EAStoryMove
EAItemMove

MosFile

class mosromgr.mostypes.**MosFile**
Base class for all MOS files

classmethod from_file(*mos_file_path*)
Construct from a path to a MOS file

Parameters **mos_file_path** (*str*) – The MOS file path

classmethod from_s3(*bucket_name*, *mos_file_key*)
Construct from a MOS file in an S3 bucket

Parameters

- **bucket_name** (*str*) – The name of the S3 bucket
- **mos_file_key** (*str*) – A MOS file key within the S3 bucket

classmethod from_string(*mos_xml_string*)
Construct from an XML string of a MOS document

Parameters **mos_xml_string** (*str*) – The XML string of the MOS document

property base_tag
The base tag (xml.etree.ElementTree.Element) within the *xml*, as determined by *base_tag_name*

property base_tag_name
The base tag (xml.etree.ElementTree.Element) within the *xml*, as determined by *base_tag_name*

property dict
Convert XML to dictionary using xmltodict library. Useful for testing. (*dict*)

property message_id
The MOS file's message ID (int)

property ro_id
The running order ID (str)

property xml
The XML element of the MOS file (xml.etree.ElementTree.Element)

ElementAction

class mosromgr.mostypes.**ElementAction**
Base class for various roElementAction MOS files

classmethod from_file(*mos_file_path*)
Construct from a path to a MOS file

Parameters **mos_file_path** (*str*) – The MOS file path

classmethod from_s3(*bucket_name*, *mos_file_key*)
Construct from a MOS file in an S3 bucket

Parameters

- **bucket_name** (*str*) – The name of the S3 bucket
- **mos_file_key** (*str*) – A MOS file key within the S3 bucket

classmethod from_string(*mos_xml_string*)
Construct from an XML string of a MOS document

Parameters **mos_xml_string** (*str*) – The XML string of the MOS document

property base_tag
The base tag (xml.etree.ElementTree.Element) within the *xml*, as determined by *base_tag_name*

property base_tag_name
The name of the base XML tag for this file type (str)

property dict
Convert XML to dictionary using xmltodict library. Useful for testing. (*dict*)

property message_id
The MOS file's message ID (int)

property ro_id
The running order ID (str)

property xml
The XML element of the MOS file (xml.etree.ElementTree.Element)

2.5 API - MOS Elements

This part of the module provides a collection of classes used to provide easy access to certain elements within a *MosFile* object, such as a list of stories within a running order, and the items within a story.

Although usually not required directly, the MOS Element classes can be imported as follows:

```
from mosromgr.moselements import Story
```

2.5.1 Element classes

Story

class mosromgr.moselements.**Story**
Bases: *mosromgr.moselements.MosElement*

This class represents a Story element within any *MosFile* object, providing data relating to the story. The Story ID, Story slug, duration and more are exposed as properties, and the parent XML element is provided for further introspection.

__str__()
The XML string

property body
A list of elements found in the story body. Each item in the list is either a string (representing a <p> tag) or an *Item* object (representing an <item> tag). Unlike *script*, this does not exclude empty paragraph tags.

property duration
The story duration (the sum of the text time and media time found within mosExternalMetadata->mosPayload), in seconds (float)

property end_time
The transmission end time of the story (datetime.datetime or None if not available in the XML)

property id
The Story ID (str)

property items
List of *Item* elements found within the story (can be None if not available in the XML)

property offset
The time offset of the story in seconds (float or None if not available in the XML)

property script
A list of strings found in paragraph tags within the story body, excluding any empty paragraphs or technical notes in brackets.

property slug
The Story slug (str or None if not available in the XML)

property start_time
The transmission start time of the story (datetime.datetime or None if not available in the XML)

property xml
The parent XML element (xml.etree.ElementTree.Element)

Item

class mosromgr.moselements.**Item**
Bases: *mosromgr.moselements.MosElement*

This class represents an Item element within any *MosFile* object, providing data relating to the item within a *Story*. The Item ID and Item slug are exposed as properties, and the parent XML element is provided for further introspection.

__str__()
The XML string

property id
The Item ID (str)

property note
The item note text (str or None if not found)

property slug
The Item slug (str or None if not available in the XML)

property xml
The parent XML element (xml.etree.ElementTree.Element)

2.5.2 Base classes

MosElement

class mosromgr.moselements.**MosElement**
Abstract base class for MOS elements

__str__()
The XML string

property id
The element ID (str)

property slug
The element slug (str or None if not available in the XML)

property xml
The parent XML element (xml.etree.ElementTree.Element)

2.6 API - MOS Collection

This part of the module provides a wrapper class *MosCollection* which stores references to specified MOS files, strings or S3 object keys so the *MosFile* objects can be recreated when needed rather than kept in memory.

Note: Note that creating a *MosCollection* from strings does not benefit from memory efficiency as the strings would still be held in memory.

The *MosCollection* is typically imported like so:

```
from mosromgr.moscollection import MosCollection
```

MOS collections are constructed using one of three classmethods. Either from a list of file paths:

```
mos_files = ['roCreate.mos.xml', 'roStorySend.mos.xml', 'roDelete.mos.xml']
mc = MosCollection.from_files(mos_files)
```

from a list of strings:

```
mos_strings = [roCreate, roStorySend, roDelete]
mc = MosCollection.from_strings(mos_files)
```

or from an S3 bucket:

```
mc = MosCollection.from_s3(bucket_name=bucket_name, prefix=prefix)
```

2.6.1 MosCollection

class mosromgr.moscollection.**MosCollection**
Wrapper for a collection of MOS files representing a partial or complete programme

__str__()
The XML string of the collection's running order

classmethod from_files(*mos_file_paths*, *, *allow_incomplete=False*)
Construct from a list of MOS file paths

Parameters

- **mos_file_paths** (*list*) – A list of paths to MOS files
- **allow_incomplete** (*bool*) – If `False` (the default), the collection is permitted to be constructed without a `roDelete`. If `True`, a *InvalidMosCollection* will be raised if one is not present. (keyword-only argument)

classmethod from_s3(*, *bucket_name*, *prefix*, *suffix='.mos.xml'*, *allow_incomplete=False*)
Construct from a list of MOS files in an S3 bucket

Parameters

- **bucket_name** (*str*) – The name of the S3 bucket (keyword-only argument)

- **prefix** (*str*) – The prefix of the file keys in the S3 bucket (keyword-only argument)
- **suffix** (*str*) – The suffix of the file keys in the S3 bucket (keyword-only argument). Defaults to '.mos.xml'.
- **allow_incomplete** (*bool*) – If `True`, the collection is permitted to be constructed without a `roDelete`. If `False` (the default), a *InvalidMosCollection* will be raised if one is not present. (keyword-only argument)

classmethod from_strings(*mos_file_strings*, *, *allow_incomplete=False*)

Construct from a list of MOS document XML strings

Parameters

- **mos_file_paths** (*list*) – A list of paths to MOS files
- **allow_incomplete** (*bool*) – If `False` (the default), the collection is permitted to be constructed without a `roDelete`. If `True`, a *InvalidMosCollection* will be raised if one is not present. (keyword-only argument)

merge(*, *strict=True*)

Merge all MOS files into the collection's running order (*ro*). If *strict* is `True` (the default), then merge errors will be fatal. If `False`, then merge errors will be downgraded to warnings.

property completed

Whether or not the running order has had a *RunningOrderEnd* merged (`bool`)

property mos_readers

A list of *MosReader* objects representing all MOS files in the collection, except the *RunningOrder* (`roCreate`) which is held in *ro*

property ro

The collection's *RunningOrder* object

property ro_id

The running order ID

property ro_slug

The running order slug

2.6.2 MosReader

The *MosReader* class is internal and is not intended to be constructed by the user. A *MosCollection* object will contain a list of *MosReader* instances, so users may find it useful to refer to its members.

class mosromgr.moscollection.**MosReader**

Internal construct for opening and inspecting a MOS file for the purposes of classifying, sorting and validating a *MosCollection*. Provides the means to reconstruct the *MosFile* instance when needed in order to preserve memory usage.

property message_id

The message ID of the MOS file (`str`)

property mos_object

Restore the MOS object and return it (*MosFile*)

property mos_type

The *MosFile* subclass this object was classified as (returns the class object, not an instance or a string)

property ro_id

The MOS file's running order ID (`str`)

2.7 API - Utilities

This part of the module provides a collection of generic utilities which are largely for internal use.

The various utilities are typically imported like so:

```
from mosromgr.utils import s3
```

Warning: This part of the module should not be considered part of the stable API and is subject to backwards-incompatible changes.

2.7.1 S3

AWS S3 utilities

get_mos_files

mosromgr.utils.s3.**get_mos_files**(*bucket_name*, *prefix=None*, ***, *suffix='.mos.xml'*)
Retrieve MOS files from given S3 bucket in location defined by *prefix*. Returns a list of file keys.

get_file_contents

mosromgr.utils.s3.**get_file_contents**(*bucket_name*, *file_key*)
Open the S3 file and return its contents as a string

2.7.2 XML

XML helper functions

remove_node

mosromgr.utils.xml.**remove_node**(*parent*, *node*)
Remove *node* from *parent*.

replace_node

mosromgr.utils.xml.**replace_node**(*parent*, *old_node*, *new_node*, *index*)
Replace *old_node* with *new_node* in *parent* at *index*.

insert_node

mosromgr.utils.xml.**insert_node**(*parent*, *node*, *index*)
Insert *node* in *parent* at *index*.

find_child

mosromgr.utils.xml.**find_child**(*parent*, *child_tag*, *id=None*)
Find an element with *child_tag* in *parent* and return `(child, index)` or `(None, None)` if not found. If *id* is provided, it will be searched for, otherwise the first child will be returned.

2.8 API - Exceptions

The module's exceptions and warnings are typically imported like so:

```
from mosromgr.exc import MosRoMgrException
```

The library's base warning is *MosRoMgrWarning* and others are detailed below.

2.8.1 Errors

MosRoMgrException

exception mosromgr.exc.**MosRoMgrException**
Bases: Exception

Base class for all `mosromgr` exceptions

UnknownMosFileType

exception mosromgr.exc.**UnknownMosFileType**
Bases: *mosromgr.exc.MosRoMgrException*

Exception raised when a MOS file type cannot be determined

MosMergeError

exception mosromgr.exc.**MosMergeError**
Bases: *mosromgr.exc.MosRoMgrException*

Exception raised when MOS merge fails

MosCompletedMergeError

exception mosromgr.exc.**MosCompletedMergeError**
Bases: *mosromgr.exc.MosMergeError*

Exception raised when MOS merge is attempted on a completed *RunningOrder*

InvalidMosCollection

exception mosromgr.exc.**InvalidMosCollection**
Bases: *mosromgr.exc.MosRoMgrException*

Exception raised when MosCollection fails validation

MosInvalidXML

exception mosromgr.exc.**MosInvalidXML**
Bases: *mosromgr.exc.MosRoMgrException*

Exception raised when *MosFile* cannot parse given XML

2.8.2 Warnings

MosRoMgrWarning

exception mosromgr.exc.**MosRoMgrWarning**
Bases: Warning

Base class for all warnings in mosromgr

MosMergeNonStrictWarning

exception mosromgr.exc.**MosMergeNonStrictWarning**
Bases: *mosromgr.exc.MosRoMgrWarning*

Warning raised when a merge error occurs in non-strict mode

ItemNotFoundWarning

exception mosromgr.exc.**ItemNotFoundWarning**
Bases: *mosromgr.exc.MosRoMgrWarning*

Warning raised when an item cannot be found during a *MosFile* merge

StoryNotFoundWarning

exception mosromgr.exc.**StoryNotFoundWarning**
Bases: *mosromgr.exc.MosRoMgrWarning*

Warning raised when a story cannot be found during a *MosFile* merge

DuplicateStoryWarning

exception mosromgr.exc.**DuplicateStoryWarning**
Bases: *mosromgr.exc.MosRoMgrWarning*

Warning raised when a story being added is already found during a *EAStoryInsert* merge

2.9 Command line interface

This section lists the module's command line commands and provides a reference to their arguments. For examples, see the *Using the command line interface* section.

2.9.1 mosromgr

```
usage: mosromgr [-h] [--version] {help,detect,inspect,merge} ...

mosromgr is a tool for managing MOS running orders

optional arguments:
-h, --help            show this help message and exit
--version             show program's version number and exit

commands:
{help,detect,inspect,merge}
help                  Displays help about the specified command
detect                Detect the MOS type of one or more files
inspect               Inspect the contents of a MOS file
merge                 Merge the provided MOS files
```

2.9.2 mosromgr detect

```
usage: mosromgr detect [-h] [-f [files [files ...]]] [-b bucket] [-p prefix] [-s suffix]␣
↪[-k key]

Detect the MOS type of one or more files

optional arguments:
  -h, --help            show this help message and exit
  -f [files [files ...]], --files [files [files ...]]
                        The MOS files to detect
  -b bucket, --bucket-name bucket
                        S3 bucket name containing the MOS files
```

(continues on next page)

(continued from previous page)

```
  -p prefix, --prefix prefix
                        The prefix for MOS files in the S3 bucket
  -s suffix, --suffix suffix
                        The suffix for MOS files in the S3 bucket
  -k key, --key key     The file key for a MOS file in the S3 bucket
```

2.9.3 mosromgr inspect

```
usage: mosromgr inspect [-h] [-f [files [files ...]]] [-b bucket] [-p prefix] [-s␣
↪suffix] [-k key]

Inspect the contents of a MOS file

optional arguments:
  -h, --help            show this help message and exit
  -f [files [files ...]], --files [files [files ...]]
                        The MOS files to inspect
  -b bucket, --bucket-name bucket
                        name of the S3 bucket containing the MOS files
  -p prefix, --prefix prefix
                        The prefix for MOS files in the S3 bucket
  -s suffix, --suffix suffix
                        The suffix for MOS files in the S3 bucket
  -k key, --key key     The file key for a MOS file in the S3 bucket
```

2.9.4 mosromgr merge

```
usage: mosromgr merge [-h] [-f [files [files ...]]] [-b bucket] [-p prefix] [-s suffix]
                      [-o outfile] [-i]

Merge the provided MOS files

optional arguments:
  -h, --help            show this help message and exit
  -f [files [files ...]], --files [files [files ...]]
                        The MOS files to merge
  -b bucket, --bucket-name bucket
                        S3 bucket name containing MOS files
  -p prefix, --prefix prefix
                        The file prefix for MOS files in the S3 bucket
  -s suffix, --suffix suffix
                        The file suffix for MOS files in the S3 bucket
  -o outfile, --outfile outfile
                        Output to a file
  -i, --incomplete      Allow an incomplete collection
```

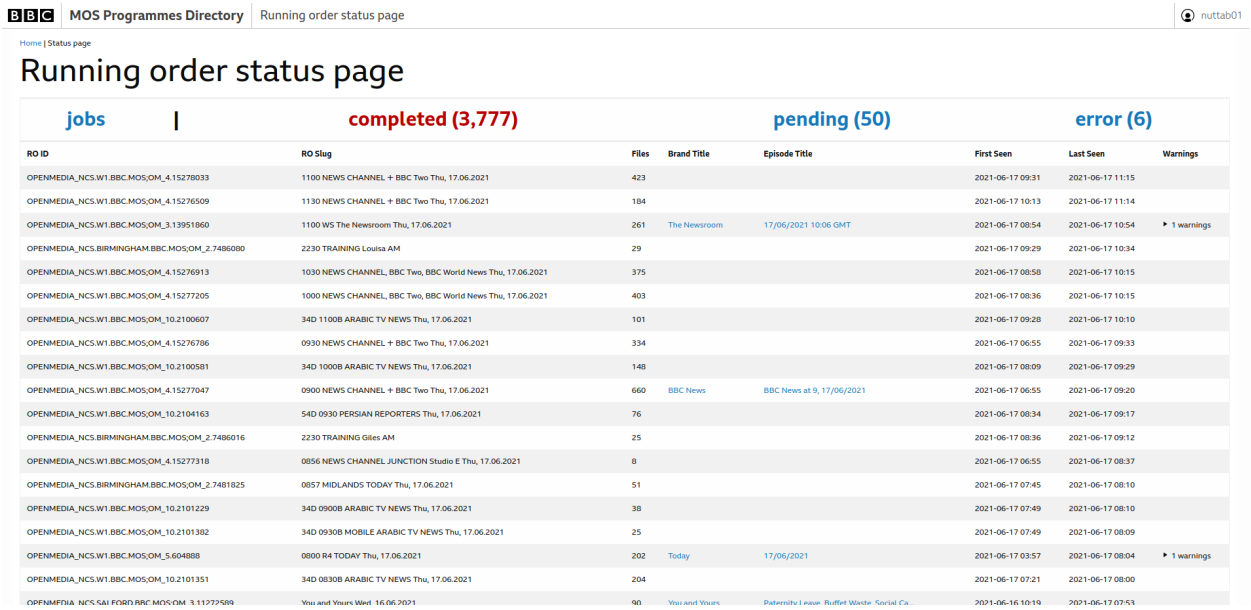
2.10 Uses of mosromgr

This section lists projects which have been known to use the *mosromgr* module. If you have used *mosromgr* in a project and would like to add it to the list, please edit this file and open a pull request, open an issue, or send an email to bbcnewslabsteam@bbc.co.uk.

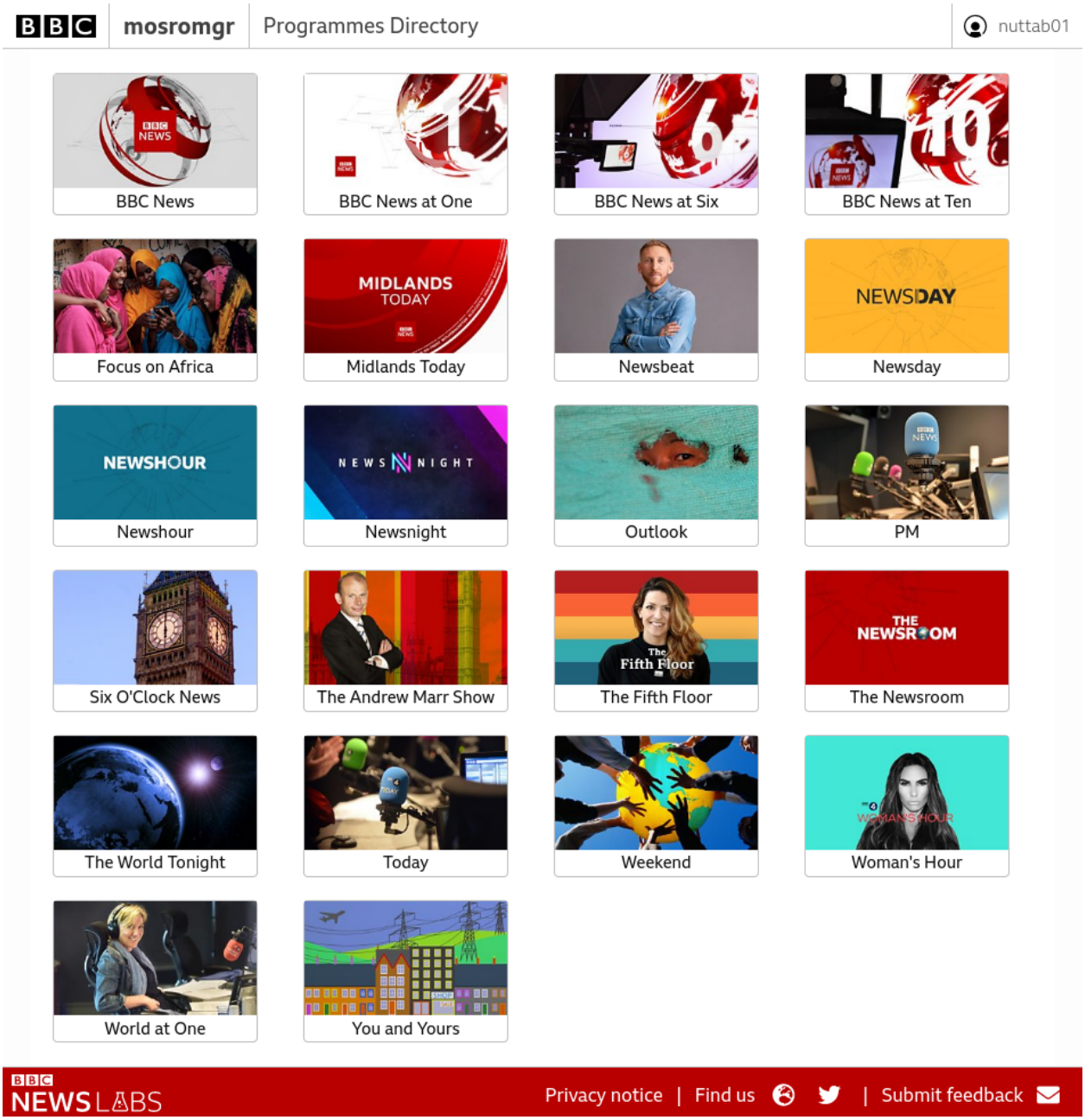
2.10.1 BBC News Labs - MOS pipeline

We have a collection of AWS services making up a pipeline which processes MOS messages in real time, updates a status dashboard, publishes completed MOS running orders and JSON summaries to an internal document store, and populates a directory of programmes with new episodes and lists of stories (complete with timing information) as they become available.

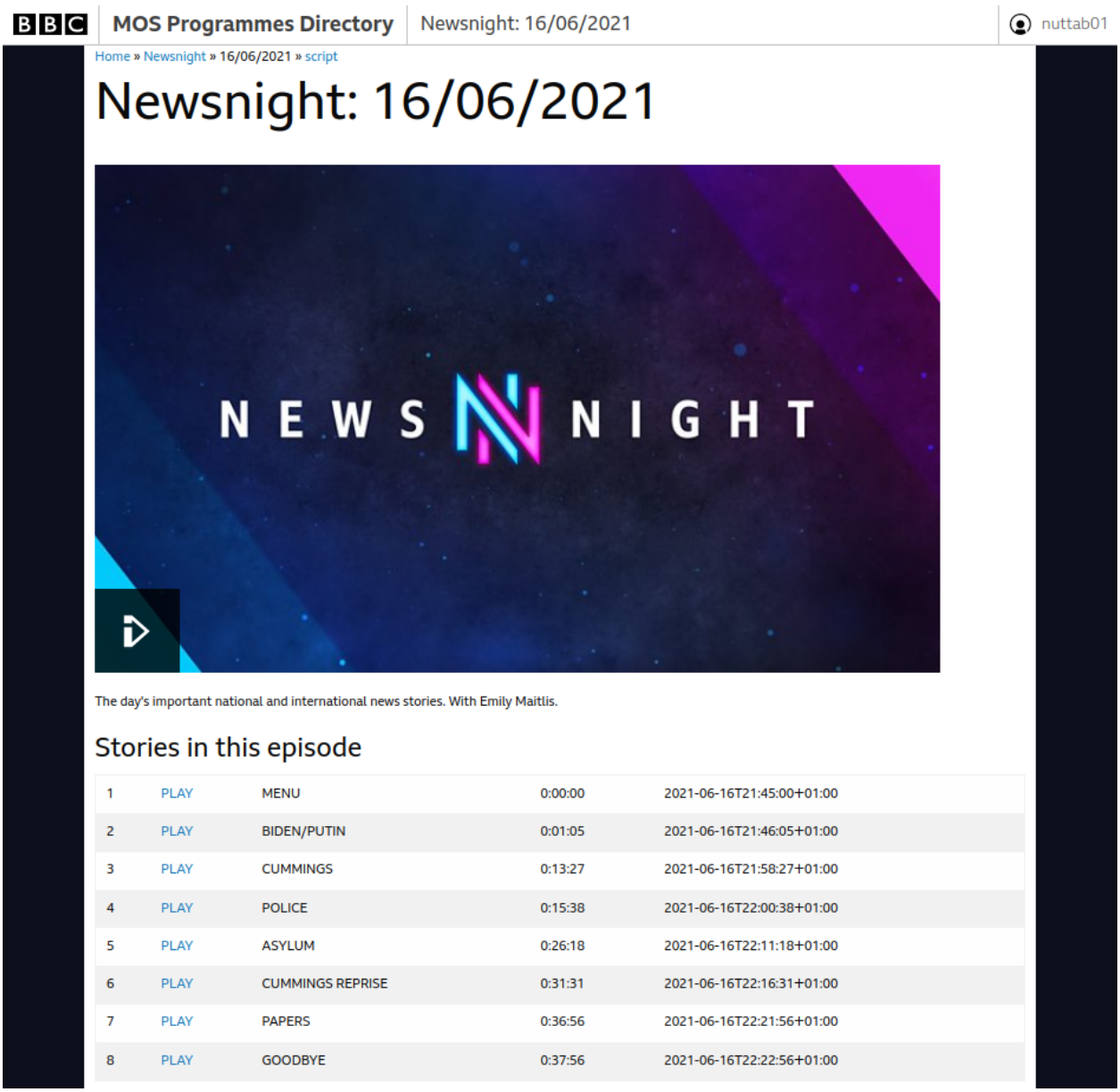
Status dashboard:



Programmes directory:

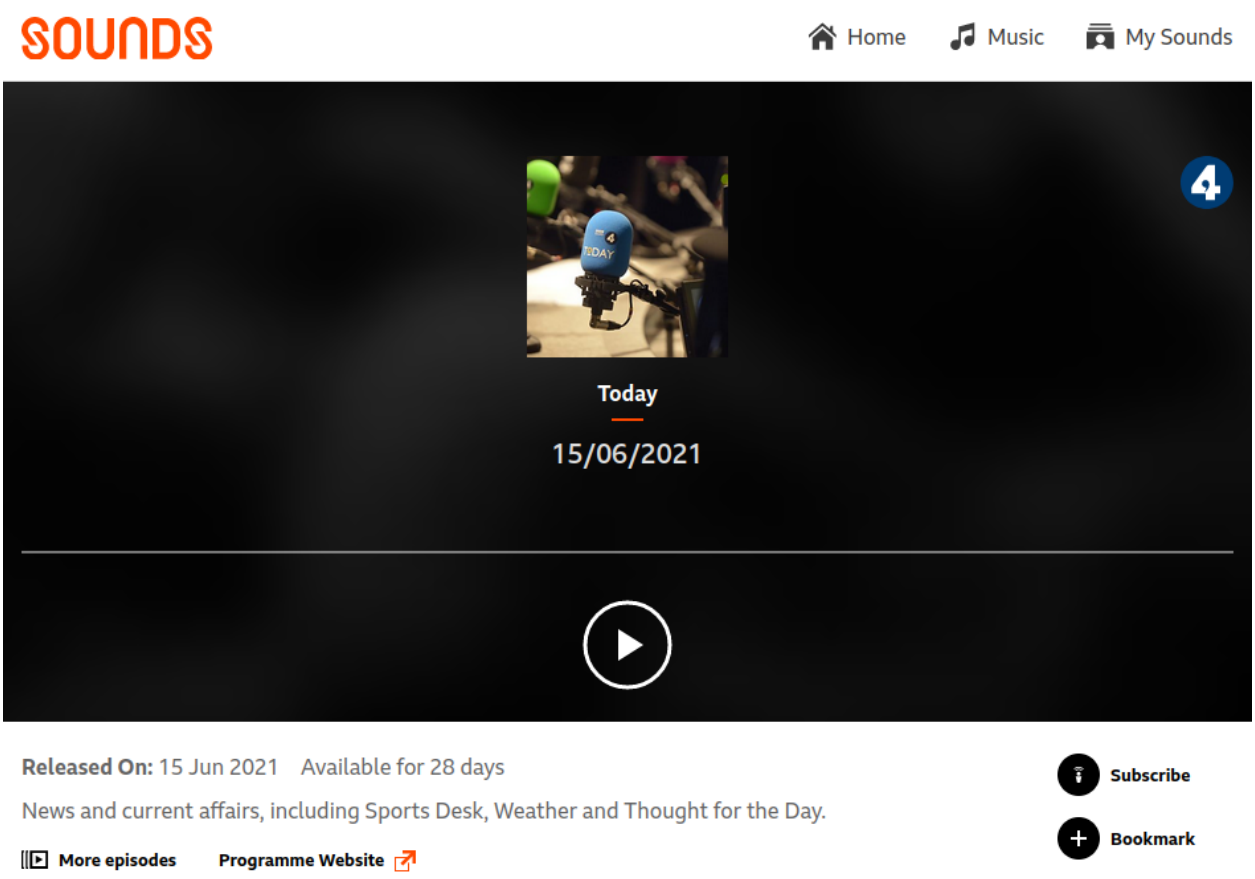


Example chapterised breakdown of an episode of Newsnight:



2.10.2 BBC News Labs - Auto chapterisation

We were able to decorate the player timeline with chapter points in certain BBC TV and radio programmes:



We used the script and story timing information extracted from the running order and aligned it against the transcript.

BBC MOS Programmes Directory | Newsnight: 16/06/2021 | nuttab01

Home » Newsnight » 16/06/2021 » script

Newsnight: 16/06/2021

MENU

MENU-PRE TITLE TEASE
The US President meets the man he called a killer face to face. Does Russia Or America need this more?

MENU-POST TITLE "ALSO TONIGHT"
Biden and Putin discuss keeping critical infrastructure - like energy and water - off-limits to cyberattack. How does that deal work if you don't trust the guy who's said it?

ALSO TONIGHT

An exclusive investigation into sexual misconduct by police officers has found that there have been almost 1,500 allegations of wrongdoing across Britain in the last 5 years. So what happens when things go wrong?

AND

Aleem Maqbool is in Kent where the Council is facing a desperate effort to find the resources to shelter and house hundreds of unaccompanied child migrants.

BIDEN/PUTIN

BIDEN/PUTIN-INTRO
Good Evening,Last time a US president and A Russian presidetn met it was in Helsinki. It was bizarre. And it was unforgettable. Donald Trump revealing to the press he had no reason to suspect Russia of election meddling and that he believed Vladimir Putin more readily than America's own security and intelligence agencies. So in some ways, the bar for Joe Biden was set pretty low. And this president was in no hurry to raise expectations. He warned that this was no meeting of friends. There is little trust between America and Russia right now. But it was a meeting of strategic minds. There are areas on which both can benefit from agreement. And so they talked -together - and to the press -separately. Joe Biden raised the case of jailed opposition leader Navalny - warning of terrible consequneces if he died in prison. And mocked Putin for comparing that to the uprising on the Capital of January 6th. So who needs more out of this relationship. And Can they find one? Here's Mark Urban. (pkg) .

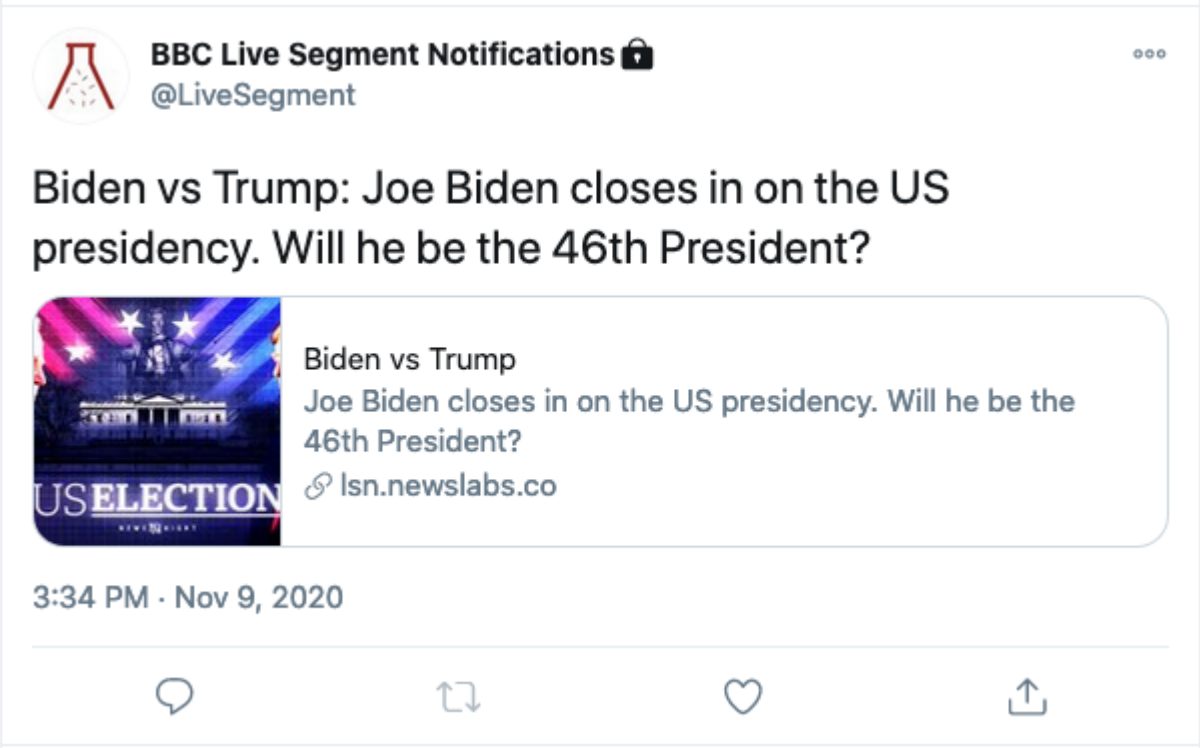
BIDEN/PUTIN-INTRO (2)
(pres).

And joining us now we have Nina Yanco-wicz, a cyber-security and disinformation specialist and a fellow at the Wilson center and we have Thomas Graham a former white house advisor on Russia and a senior director for Russia on the National Security Council.

<1. Nina Yancowicz who needs who more from this- Putin or BIden?

2.10.3 BBC News Labs - Live Segment Notifications

We developed a proof-of-concept in which a note within a story in a running order could trigger a tweet to alert people of an upcoming story in time to watch live, or link to the clip of the story on-demand:



2.11 Changelog

Warning: Note that the library is currently in beta. The API and CLI are not yet stable and may change. Once the library reaches v1.0, it will be considered stable. Please consider giving *Feedback* to help stabilise the API.

2.11.1 Release 0.9.0 (2021-06-21)

- Updated *mosromgr inspect* CLI command to work for all file types
- Corrected some singular *`MosFile` MOS element* properties that should have been lists (e.g. `source_story` should have been `source_stories`)
- Improved validation and error handling when merging various *`MosFile`* objects
- Added *`script`* and *`body`* to *`Story`*
- Added *`script`* and *`body`* to *`RunningOrder`*
- Added non-strict mode to *`mosromgr.moscollection.MosCollection.merge()`* method and CLI
- Corrected some edge cases in *`MosFile`* subclass merge implementations (e.g. empty `storyID` tag means move to bottom)

2.11.2 Release 0.8.1 (2021-04-14)

- Fixup release

2.11.3 Release 0.8.0 (2021-04-13)

- Improved validation and error handling when merging various *MosFile* objects
- Added more arguments to CLI commands
- Corrected some singular *MosFile API - MOS Elements* properties that should have been lists (e.g. `source_story` should have been `source_stories`)

2.11.4 Release 0.7.0 (2021-01-08)

- Ensured exceptions are raised when story IDs are not found when merging
- Ensured tags aren't overwritten when they are empty in *MetaDataReplace*
- Ensured target story is found when merging *StoryInsert* and *StoryReplace*
- Added *RunningOrderControl* class (for `roCtrl` messages)
- Changed `tx_time` to *start_time*

2.11.5 Release 0.6.0 (2020-12-01)

- Added support for <StoryDuration> as an alternative to <MediaTime> and <TextTime>

2.11.6 Release 0.5.0 (2020-11-30)

- Added *ReadyToAir* MOS Type
- Improved error message on invalid *MosCollection*

2.11.7 Release 0.4.0 (2020-11-30)

- Changed `closed` property to *completed*
- Added transmission time and offset to *Story* class
- New *Command line interface* with separate commands for `detect`, `inspect` and `merge`
- Make MosCollection raise exceptions on failure, not just warnings

2.11.8 Release 0.3.0 (2020-11-24)

- Switched from complicated `__init__` constructors to multiple `from_` classmethods e.g. *`from_file()`*
- Replaced `get_mos_object` function with detection logic in the *`MosFile`* and *`ElementAction`* base classes
- Replaced `MosContainer` class with *`MosCollection`*

2.11.9 Release 0.2.0 (2020-11-24)

- Added *API - MOS Elements* - a collection of classes used to provide easy access to certain elements within a *`MosFile`* object

2.11.10 Release 0.1.0 (2020-11-24)

- Implemented most standard MOS message types as *`MosFile`* subclasses, supporting merging subsequent messages into the original running order
- Implemented a MOS file detection function (`get_mos_object`)
- Added a `MOSContainer` class as a wrapper for a complete programme
- Added a CLI for merging MOS files

2.12 Development

This page contains reference material for those interested in developing and contributing to the **mosromgr** module.

The project source code is hosted on GitHub at https://github.com/bbc/mosromgr which also includes the issue tracker.

2.12.1 Setting up for Development

1. Clone the repository and enter the directory:

```
$ git clone https://github.com/bbc/mosromgr
$ cd mosromgr
```

2. Create a virtual environment e.g. using virtualenvwrapper:

```
$ mkvirtualenv mosromgr
```

3. Install the project for development:

```
$ make develop
```

After completing these steps, the library and command line interface will be available to use within your environment. Any modifications made to the source code will be automatically reflected within the environment.

2.12.2 Tests

The test suite uses pytest. Tests are organised mirroring the source code.

Running the tests

To run the linter, test suite and coverage analysis, activate the environment and run:

```
$ make test
```

For more control when running tests, run `pytest` directly, for example `pytest -vvxk story` will run tests with `story` in the name (`-k story`) with verbose output (`-vv`), and stop at the first failure (`-x`).

2.12.3 Documentation

The documentation is built using sphinx using the diataxis framework.

Building the documentation

To build the documentation, activate the environment and run:

```
$ make doc
```

This will generate the required diagrams and build the HTML docs which will be located in `docs/build/html`. Serve them with the command:

```
$ make doc-serve
```

You'll now be able to open the docs on your browser at `http://localhost:8000/`.

2.13 Feedback

Before we release v1.0 and stabilise the API, we are seeking other organisations using the MOS protocol to test *mosromgr* on their own MOS files and provide feedback so we can integrate any necessary changes to make sure it works effectively beyond the BBC's use.

If you can help, please test the module on your own MOS files and report back to us using our discussion board or issue tracker on GitHub, or email us at bbcnewslabsteam@bbc.co.uk.

2.14 Indices and tables

- genindex
- modindex
- search

CHAPTER

THREE

ISSUES AND QUESTIONS

Questions can be asked on the discussion board, and issues can be raised on the issue tracker.

CHAPTER

FOUR

CONTRIBUTING

Source code can be found on GitHub at github.com/bbc/mosromgr.

Contributions are welcome. Please refer to the contributing guidelines.

CHAPTER

FIVE

CONTRIBUTORS

- Ben Nuttall
- Owen Tourlamain
- Rob French
- Lucy MacGlashan
- Dave Bevan
- Matthew Sim

CHAPTER

SIX

LICENCE

Licensed under the Apache License, Version 2.0.

CHAPTER

SEVEN

CONTACT

To get in touch with the maintainers, please contact the BBC News Labs team: bbcnewslabsteam@bbc.co.uk

PYTHON MODULE INDEX

m

INDEX

Symbols

B

C

D

E

F

G

I

M

N

O

R

S

T

U

X