Extractor Message Bus

Notes about how we handle extractors/RabbitMQ messages.

Currently

- **FILE -** Originally extractors primarily operated on one file. Trigger when file is added to Clowder.
  - *.file.#
  - *.file.image.#

field	description
id	file UUID
intermediateId	file UUID (deprecated)
datasetId	id of dataset file was added to
filename	file name
secretKey	Clowder API key
host	Clowder host URL

- **DATASET -** Later, support for dataset extractors was added. Trigger when file is added to a dataset.
  - *.dataset.file.added
  - *.dataset.file.removed

field	description
id	file UUID
intermediateId	file UUID (deprecated)
datasetId	id of dataset file was added to
secretKey	Clowder API key
host	Clowder host URL

  These trigger when a file is added to a dataset.
  - Because the message contents are otherwise identical, PyClowder currently uses the presence of 'filename' field in message to determine whether to handle as a file or dataset extraction.
  - Max just updated PyClowder2 pull request to include routing_key in the parameters for extractors, so we can check that instead of checking 'filename' field.
- **METADATA -** Later, support for metadata-triggered extractors was added.
  - *.metadata.added
  - *.metadata.removed

field	description
id	file or dataset UUID
metadata	md that was added/removed

  This sends correct messages to RabbitMQ from Clowder but we need to evaluate this one as well because PyClowder has some rough edges in determining how to handle these messages, as they may not require a file OR a dataset to be downloaded or considered if the extractor can work with the metadata alone.
- **COLLECTION** - Eventually, we may want to trigger extractors that process arbitrary collections of datasets.
  - Not quite sure how we're gonna do this yet.

Possible Improvement Ideas

- **Reduce distinction between 'file' and 'dataset' extractors**
  - Clowder has changed so that we no longer present files as separate from datasets - that is, we don't have files outside datasets. So my distinction between the two kinds of extractors is maybe unnecessary now - in both cases, extraction begins when a file is uploaded to a dataset.

  - Once that happens, extractor might want to do several things:
    1. Use the file to generate metadata and attach to the file
    2. Use the file to generate metadata and attach to the dataset
    3. Use the file to convert to a different format and upload to the dataset
    4. Use many files from dataset to generate output files/metadata and add to dataset or files

    ...we can do #1-3 with currently existing file extractors. #4 just requires a way to get list of other files in the dataset (as Rob suggested, ordered by date added)

- With that in mind, I think we could get rid of these messages:
  - *.dataset.file.added
  - *.dataset.file.removed

  ...and instead just use *.file.#. Then, each extractor can have a flag that says whether or not to fetch list of all files in dataset, or just the file that triggered the extractor

- One problem: how do we handle:
  - Dataset-level extractor events (STARTED, PROCESSING, DONE) if these are handled as file events
    - Currently TERRA extractors write 'COMPLETED' as extractor metadata to dataset, and check for that in later extractions
  - Rerunning extractors on a dataset
    - Do we just send the last added file as the 'file' event and trigger that way?

- **Remove intermediateID if no one is using**

- **Revisit what we include in RabbitMQ messages**
  - drop intermediateID
  - other info that could help us make this more efficient?
    - Data that PyClowder fetches on processing to give to extractors
      - List of files in a dataset w/ creation date, file paths
      - List of metadata attached to file/dataset

      (these are likely to make the messages too big, unless we wanted to cap them and only include if under a certain length & let PyClowder fetch otherwise)