



User Guide

Preservica Active Digital Preservation Guide to OPEX ingest and Monitoring.

Updated to v6.3 13th April 2021

Contents

Glossary	2
Updates.....	2
Addition of SourceID section “Use of SourceID to determine ingest location”	2
Addition of opex.ingest.max.bytes system property	2
Addition of resubmit.....	2
Structure of Guide	3
1. Introduction.....	4
2. OPEX Creation.....	5
2.1. OPEX	5
2.2. Structure of .opex metadata file:	5
Properties Section	6
Transfer Section.....	6
Descriptive Metadata	6
2.3. Structure of OPEX with .opex file	7
2.4. Example .opex.....	8
File .opex.....	8
Folder .opex.....	9
3. Transfer of OPEX.....	11
4. OPEX ingest.....	12
4.1. Introduction.....	12
4.2. Providing access to Bulk Loading functionality	12
4.3. User roles for Bulk Loading.....	12
4.4. Creating Workflows	12
Setting up Inner Ingest Workflow.....	12
Setting up Ingest OPEX (Incremental) Workflow.....	14
4.5. Running an OPEX ingest.....	16
System properties to enable de-duplication	16
Automatically triggered OPEX ingest.....	17
Manually triggered OPEX ingest	17
Use of SourceID to determine ingest location.....	17
Initial ingest to create new Hierarchy	18
Ingest delay property.....	19
Ingest package size	19
5. Monitoring OPEX ingests	20
5.1. Decoding monitor messages	23

Glossary

Bulk Loading: delivered as an integral part of Preservica 6.1 (and above), the Bulk Loader manages ingestion of OPEX(s) alongside .opex through a parent workflow (Incremental Ingest). The expectation is that OPEX contains larger volumes of content in an ordered structure which will be replicated in the Preservica 6.1+ according to the structure and the information in the .opex. The Bulk Load has a monitoring page (monitor) accessible from the Ingest menu.

Incremental Ingest: The overarching Ingest workflow that splits the OPEX into multiple Ingest packages and manages the Ingest process of them all, cross referencing them back to the overall OPEX. When run using folder manifests with transfer manifests in the .opex metadata

OPEX: A structure of Folders and Files which when loaded into Preservica 6.1 whose structure will be maintained. OPEX is also a standard way for exporting DIPs (Dissemination Information Packages) from Preservica.

.opex: metadata associated with the OPEX which provides (a) Descriptive metadata, (b) transfer validation information and (c) Properties e.g. Title and SourceIDs for the Folders or Files

SourceID: A SourceID is a unique ID from the Source system that enables de-duping so that Folders or Files with an existing SourceID do not have to be re-ingested. The SourceID can be used to determine the location of the content on ingest.

Updates

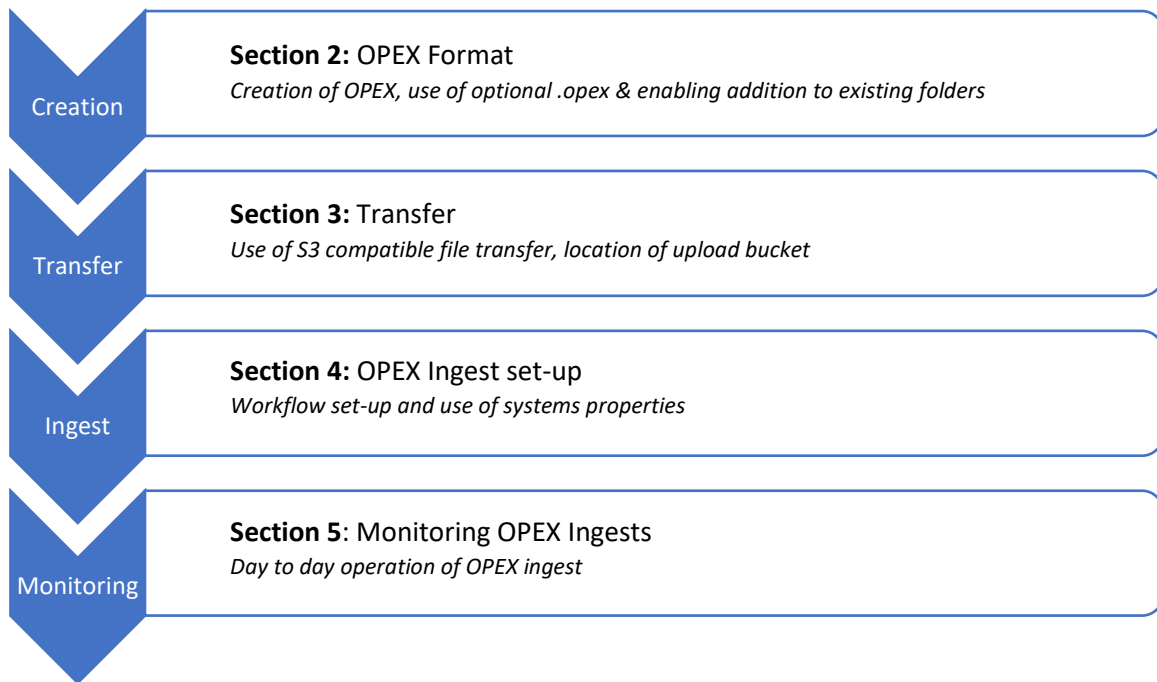
Change	Updated/Added	Valid from
Monitor page now available to SAML users	Updated	v6.2.1
Introduction of PAX to create multi-representation assets	Added	v6.2.1
.opex examples updated	Updated	v6.1
Addition of Resubmit option in Monitor	Added	v6.2.1
Addition of SourceID section "Use of SourceID to determine ingest location"	Added	v6.1
Addition of opex.ingest.max.bytes system property	Added	v.6.2.1
Addition of resubmit	Added	v.6.2.1
Ability to specify opex folder to be used with autowatch	Added	V6.3

Structure of Guide

It is recommended the guide is read in full before users commence ingesting OPEX.

Once a process has been put in place and a design decision made, the regular loading of OPEX into Preservica is intended to be a straightforward process. This can be monitored using the Monitor page under the Ingest menu with checks occurring on a frequency aligned with the OPEX delivery schedule.

The guide is structured to walk the user through the process:



1. Introduction

Preservica has introduced the option to ingest an **OPEX** (Open Preservation Exchange Format¹) of folders and files, to enable users to organise content into an easy to understand format with human readable metadata. Then using the OPEX ingest workflow, to handle the OPEX in a single operation.

Customers may utilise this for a variety of use cases:

- A one-off ingest of a large volume of content when a system is being deprecated to create a structure which mirrors the deprecated system
- To perform a regular, automated ingest into Preservica, with the content being added to existing folders or new folders based on a Source ID or using a title
- To provide a single standard for donors to deliver content to a central Preservica for automated ingest

Addition of optional **.opex metadata** alongside the files and folders, enables for example: the addition of descriptive metadata, instructions as to the destination of the content, comparison with the original fixities or auditing of the content post-ingest.

A **Monitor** page is available to follow the ingest process.

The ingest process allows users to send multiple OPEXs over time and using an ID from the Source (or the folder titles) to enable users to prevent duplicate ingests, whilst adding new files and folder to the initial structure if appropriate.

The OPEX ingest and monitor is available to EPC (Enterprise Private Cloud), EPCP (Enterprise Private Cloud Perform) and On-Premise Customers.

It is expected these ingests may be long running processes but will require less human intervention than using tools such as PUT (Preparation and Upload) or SIP Creator.

To fully utilise the functionality, users are advised to provide .opex metadata at folder and file level, however the process will operate without them.

¹ www.openpreservationexchange.org

2. OPEX Creation



2.1. OPEX

The simplest form of **OPEX** is a structure of files and folders. Each folder will be translated to a Preservica folder, and each file will become an asset when ingested into a Preservica v6.

Note: The file becomes the preservation representation for an asset. From v6.2.1 Preservica will handle PAX to enable handling multi-representation assets. Separate guides are available to explain how a PAX is constructed.

Addition of a **.opex** for a file or folders is optional. It replaces the `.metadata` file used in ingest packages and contains the descriptive metadata.

2.2. Structure of `.opex` metadata file:

The `.opex` metadata elements are all optional.

For a **file** or a **PAX**, the `.opex` metadata file is placed adjacent to the file with the extension `.opex`. For example, for `preservica.jp2`, the OPEX metadata file should be named `preservica.jp2.opex`. In the case of a PAX the naming would be `preservica.pax.zip.opex`

For a **folder**, the `.opex` metadata file is placed within the folder. The `.opex` should be given the name of the folder with the suffix `.opex`, i.e. for a folder named *Images*, the `.opex` file should be: `Images/Images.opex`

The OPEX **container** name should be unique, we suggest a naming convention including the time and date of the extract to avoid duplicates e.g. `opex-2020-09-01`

The `.opex` schema has three elements. They are all optional and can be used as required.

- Transfer
- Properties
- Descriptive Metadata

The tables below show what can be included in each of the Properties and Transfer sections in a `.opex`

For customers who will create their own `.opex` files, please request the schema "*OPEX-Metadata.xsd*" to validate any `.opex` created.

Transfer Section

This section contains information about the provenance and content of the item.

- The **SourceID** element records the id of the object in the source system; this field is also used for **de-duplicating** so existing folders and files with the same SourceID are not ingested.
- For a **folder**, this element can include a **manifest**, which lists all the files and folder which are direct children of the folder. If this is present any missing or extra files or folders will be identified on ingest
- For a **file**, source fixity information can be included, which will be verified by the ingest process.

OPEX Field	Applies to OPEX type		Preservica Field	Notes
	Folder	Asset		
SourceID	✓	✓	Identifier	Creates an identifier in Preservica of type SourceID.
<i>Manifest</i>				
Folders	✓			A listing of all child folders and files including file .opex, but excludes the folder .opex
Files	✓			A listing of all child files content and metadata but excludes the folder .opex
<i>Fixities</i>				
Fixity		✓		The fixity of the content file

Properties Section

- This element can be used to define the **title** and/or **description** of the folder or asset, to override the value in the Preservica XIP which otherwise would be based on the names of the folder or file.
- You may also include a security descriptor (tag) if you do not want the default of 'open'.
- Identifiers can be included which will be recorded in the Folder or Asset properties

OPEX Field	Applies to OPEX type		Preservica Field	Default	Notes
	Folder	Asset			
Title	✓	✓	Title	Folder: Folder Name Asset: File Name without extension	
Description	✓	✓	Description		
SecurityDescription	✓	✓	SecurityTag	Inherited from parent	Security Tag must already exist in Preservica
Identifiers	✓	✓	Identifier	None	e.g. code for catalogue linking, URL for warc seed

Descriptive Metadata

If Descriptive metadata needs to be added to the folder or asset, fragments can be added here. The appropriate XML namespace (with the xmlns attribute) must be included on each fragment.

2.3. Structure of OPEX with .opex file

Below is the structure of a very simple OPEX showing how the File and Folder .opex should be arranged,

The Container Directory is called *opex-2020-09-01*

(*opex-2020-09-01*)

```
Moving Images/  
  RecordGroup1/  
    video1.mp4  
    video1.mp4.opex  
    video2.mp4  
    video2.mp4.opex  
    video3.mp4  
    video3.mp4.opex  
  RecordGroup1.opex  
  RecordGroup2/  
    video4.mp4  
    video4.mp4.opex  
  RecordGroup2.opex  
  Video5.mp4  
  Video5.mp4.opex.  
  MovingImages.opex  
opex-2020-09-01.opex
```

The .opex called *opex-2020-09-01.opex*

- Lists the folder *MovingImages* at the top level of the OPEX

The .opex called *MovingImages.opex*

- lists the folders at *RecordGroup1*, *RecordGroup2* and the file *video5.mp4*

The two .opex called *RecordGroup1.opex* and *RecordGroup2.opex*

- lists the files in each folder

2.4. Example .opex

File .opex

An example of the File .opex for is shown for [Video1.mp4.opex](#)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<opex:OPEXMetadata xmlns:opex="http://www.openpreservationexchange.org/opex/v1.0">
  <opex:Transfer>
    <opex:Fixities>
      <opex:Fixity type="SHA1" value="7692983CA2D0BB268EEA15F894A5743C31550D20"/>
    </opex:Fixities>
  </opex:Transfer>
  <opex:Properties>
    <opex:Description>Video 1 File</opex:Description>
    <opex:SecurityDescriptor >open</opex:SecurityDescriptor>
  </opex:Properties>
  <opex:DescriptiveMetadata>
    <oai_dc:dc xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/ oai_dc.xsd"
xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <dc:title>The name given to the resource.</dc:title>
      <dc:creator>An entity primarily responsible for making the content.</dc:creator>
      <dc:subject>The topic of the content of the resource.</dc:subject>
      <dc:description>An account of the content of the resource.</dc:description>
      <dc:publisher>The entity responsible for making the resource available.</dc:publisher>
      <dc:contributor>An entity responsible for making contributions to the content.</dc:contributor>
      <dc:date>A date associated with an event in the life cycle of the resource.</dc:date>
      <dc:type>The nature or genre of the content of the resource. </dc:type>
      <dc:format>The physical or digital manifestation of the resource.</dc:format>
      <dc:identifier>An unambiguous reference to the resource within a given context.</dc:identifier>
      <dc:source>A Reference to a resource from which the present resource is derived.</dc:source>
      <dc:language>A language of the intellectual content of the resource.</dc:language>
      <dc:relation>A reference to a related resource.</dc:relation>
      <dc:coverage>The extent or scope of the content of the resource.</dc:coverage>
      <dc:rights>Information about rights held in and over the resource.</dc:rights>
    </oai_dc:dc>
  </opex:DescriptiveMetadata>
</opex:OPEXMetadata>
```

Folder .opex

The Folder OPEX for MovingImages Folder ([MovingImages.opex](#)) containing two folders and a mp4 file

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<opex:OPEXMetadata xmlns:opex="http://www.openpreservationexchange.org/opex/v1.0">
  <opex:Transfer>
    <opex:SourceID>ABC-1-1</opex:SourceID>
    <opex:Manifest>
      <opex:Files>
        <opex:File type="content">Video5.mp4</opex:File>
        <opex:File type="metadata">Video5.mp4.opex</opex:File>
      </opex:Files>
      <opex:Folders>
        <opex:Folder>RecordGroup1 </opex:Folder>
        <opex:Folder>RecordGroup2 </opex:Folder>
      </opex:Folders>
    </opex:Manifest>
  </opex:Transfer>
  <opex:Properties>
    <opex>Title>Moving Images</opex>Title>
    <opex>Description>Folder of Moving Images</opex>Description>
    <opex:SecurityDescriptor>open</opex:SecurityDescriptor>
  </opex:Properties>
  <opex:DescriptiveMetadata>
    <oai_dc:dc xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/ oai_dc.xsd"
      xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <dc:title>The name given to the resource.</dc:title>
      <dc:creator>An entity primarily responsible for making the content of the resource.</dc:creator>
      <dc:subject>The topic of the content of the resource.</dc:subject>
      <dc:description>An account of the content of the resource.</dc:description>
      <dc:publisher>The entity responsible for making the resource available.</dc:publisher>
      <dc:contributor>An entity responsible for making contributions to the content.</dc:contributor>
      <dc:date>A date associated with an event in the life cycle of the resource.</dc:date>
      <dc:type>The nature or genre of the content of the resource. </dc:type>
      <dc:format>The physical or digital manifestation of the resource.</dc:format>
      <dc:identifier>An unambiguous reference to the resource within a given context.</dc:identifier>
      <dc:source>A Reference to a resource from which the present resource is derived.</dc:source>
      <dc:language>A language of the intellectual content of the resource.</dc:language>
      <dc:relation>A reference to a related resource.</dc:relation>
      <dc:coverage>The extent or scope of the content of the resource.</dc:coverage>
      <dc:rights>Information about rights held in and over the resource.</dc:rights>
    </oai_dc:dc>
  </opex:DescriptiveMetadata>
</opex:OPEXMetadata>
```

And the Folder OPEX for RecordGroup1 Folder (RecordGroup1.opex) containing three mp4 files

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<opex:OPEXMetadata xmlns:opex="http://www.openpreservationexchange.org/opex/v1.0">
  <opex:Transfer>
    <opex:SourceID>ABC-1-1-1</opex:SourceID>
    <opex:Manifest>
      <opex:Files>
        <opex:File type="content">Video1.mp4</opex:File>
        <opex:File type="metadata">Video1.mp4.opex</opex:File>
        <opex:File type="content">Video2.mp4</opex:File>
        <opex:File type="metadata">Video2.mp4.opex</opex:File>
        <opex:File type="content">Video3.mp4</opex:File>
        <opex:File type="metadata">Video3.mp4.opex</opex:File>
      </opex:Files>
    </opex:Manifest>
  </opex:Transfer>
  <opex:Properties>
    <opex:Title>Record Group 1</opex:Title>
    <opex:Description>Folder of RG1 Videos</opex:Description>
    <opex:SecurityDescriptor>open</opex:SecurityDescriptor>
  </opex:Properties>
  <opex:DescriptiveMetadata>
    <oai_dc:dc xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/ oai_dc.xsd"
xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <dc:title>The name given to the resource.</dc:title>
      <dc:creator>An entity primarily responsible for making the content of the resource.</dc:creator>
      <dc:subject>The topic of the content of the resource.</dc:subject>
      <dc:description>An account of the content of the resource.</dc:description>
      <dc:publisher>The entity responsible for making the resource available.</dc:publisher>
      <dc:contributor>An entity responsible for making contributions to the content.</dc:contributor>
      <dc:date>A date associated with an event in the life cycle of the resource.</dc:date>
      <dc:type>The nature or genre of the content of the resource. </dc:type>
      <dc:format>The physical or digital manifestation of the resource.</dc:format>
      <dc:identifier>An unambiguous reference to the resource within a given context.</dc:identifier>
      <dc:source>A Reference to a resource from which the present resource is derived.</dc:source>
      <dc:language>A language of the intellectual content of the resource.</dc:language>
      <dc:relation>A reference to a related resource.</dc:relation>
      <dc:coverage>The extent or scope of the content of the resource.</dc:coverage>
      <dc:rights>Information about rights held in and over the resource.</dc:rights>
    </oai_dc:dc>
  </opex:DescriptiveMetadata>
</opex:OPEXMetadata>
```

3. Transfer of OPEX



For ingest to occur, the OPEX need to be moved into a named folder in an S3 bucket.

This bucket (usually named `com.preservica.tenant.bulk`) will be flagged as a SIP Upload type; this can be confirmed in the Source Menu.

We advise that the folder is called `opex`. For the remainder of the guide, we will assume this naming.

If more than one Ingest OPEX (Incremental) workflow is operating then the folder can be named to suit the situation, see section xx for more details.

Keys will be provided by Preservica Support.

If the Manifest option in the Transfer element of the Folder `.opex` is **not** present and both content and `.opex` folders are included, whilst the upload to the S3 bucket is in progress, either

(i) the Incremental Ingest workflows should be deactivated until the upload is complete
or (ii) the `opex.ingest.delay` should be set so that the transfer can complete in the time.

If the second option is chosen and the files arrive in an order the workflow cannot handle, the workflow will fail but the user can re-submit the workflow once the files have arrived.

For users of IAE, the order of transfer of the `.opex` and content is controlled so this is not a concern.

Note: If folder manifests are included in the OPEX and expected, as indicated in the workflow, then the workflow will wait for the `opex.ingest.delay` period (see section 4.5). This defines how long the process waits for the file to be transferred. When the files are large and the transfer is long e.g. `.avi` files, the delay period may not be sufficient however the resubmit option is be available and if the transfer does time out then the OPEX can be resubmitted from the monitor.

4. OPEX ingest



4.1. Introduction

The Bulk loading capability consists of a **Monitor** page and an **Ingest OPEX (Incremental)** workflow coupled with a standard **Ingest (v6)** workflow.

4.2. Providing access to Bulk Loading functionality

Access to the Monitor page is dependent on Preservica activating the page in EPC and EPCP Tenancies.

For On-Premise users, the system property is *XX.monitor.feature*, where XX is the tenancy name and the property should be set to true

4.3. User roles for Bulk Loading

As an Ingest process, the functional roles for ingest rely on the usual ingest functional roles.

- Any user with the Manager Role can set up, activate, or adjust either Ingest workflow.
- Any user with the Manager or Ingest Role can manually run either Ingest workflows.

4.4. Creating Workflows

Two workflows are required to ingest an OPEX. The Ingest OPEX (Incremental) is the outer workflow which controls the process and sets of multiple inner or child Ingest (v6) workflows in series to perform the ingest activities.

Setting up Inner Ingest Workflow

The inner or child workflow is a standard Ingest (v6) workflow. An in-use standard ingest workflow can be used as the options selected will be ignored; however, it may be sensible to set up a specific Ingest (v6) workflow for this operation. The set-up is:

Workflow Context Definition

To edit the details of this context, first deselect the 'Active' checkbox.

Workflow Definition: Ingest (v6)

Name: Ingest

Description: Pre-configured Ingest workflow

Email Address: wendy.calm@preservica.com

Send email notification on error

Send email notification on completion

Automatically terminate on unrecoverable error

Allow concurrent workflow instances

Parameters marked with a * are mandatory

Name	Value
Source *	AWS: com.preservica.dev.preview.calm.upload
Folder Ref	null
Delete Source	<input checked="" type="checkbox"/>

Link To Catalogue Workflow: ---

Workflow Trigger

Manual Only

Scheduled

Start Ingest Automatically

Update Cancel

The key fields for the Ingest (v6) that should be completed are:

Name	A short name for the Ingest workflow
Description	A longer description for the Ingest workflow
Email address	The email address for receipt of error and/or success messages depending on the subsequent selection.
Send email notification on error	Select this option so that a user is notified of any error messages. Messages are also available in the workflow details.
Send email notification on completion	We advise you do NOT select this option, every completed ingest workflow would mean a user is notified on completion.
Automatically terminate on unrecoverable error.	Select this option
Allow concurrent workflow instances	Select this option
Source	A source must be selected, but the source used is defined in the Ingest OPEX (Incremental) workflow.
Folder Ref	An existing folder Ref (the UUID of a folder), however the default Folder Ref is defined by the Ingest OPEX (Incremental) workflow.
Delete Source	In normal practice, we advise users to Tick this box, this option is define by the Ingest OPEX (Incremental) workflow.
Link to catalogue	Do not select an option here
Workflow Trigger	This can be left as manual; this workflow is called by the Ingest OPEX (Incremental) workflow.

Activate the Ingest workflow, so it can be selected in the Ingest OPEX (Incremental) workflow.

Setting up Ingest OPEX (Incremental) Workflow

Next set-up of the Ingest OPEX (Incremental) workflow is:

The key fields that should be completed as follows:

Name	A short name for the Ingest OPEX (Incremental) workflow
Description	A longer description for the Ingest OPEX (Incremental) workflow
Email address	The email address for receipt of error and/or success messages depending on the subsequent selection.
Send email notification on error	Select this option so that a user is notified of any error messages. Messages are also available in the workflow details.
Send email notification on completion	We advise you do NOT select this option, every completed ingest workflow would mean a user is notified on completion.
Automatically terminate on unrecoverable error.	Select this option
Allow concurrent workflow instances	Select this option
Folder Ref	Select an existing folder Ref (the UUID of a folder in Preservica). This will be the fall-back location for the top level of the OPEX if it does not have a location for ingest.

Source location	Select the designated S3 bucket for receipt of the OPEX.
Ingest workflow	Select the workflow set up to be the Inner or Child Ingest workflow.
OPEX Container Directory	Enter text e.g. <i>opex</i> This is the folder into which the OPEX in its Container is placed.
Delete Source	Tick this box
Require Folder Manifests	If Folder Manifests are present in all Folder .opex, select this option.
Workflow Trigger	For optimal operation, users should use Start OPEX ingest automatically. However manual and scheduled options are available.

4.5. Running an OPEX ingest

If the Ingest OPEX (Incremental) workflow is set to run automatically, when content is present in the folder named as the OPEX container directory in the S3 bucket, then the user will not need to take any further action. The user should use the Monitor page under the Ingest menu to observe the progress.

Reminder: If there is no .opex metadata or the .opex metadata for the folders does NOT include the Manifest element, the Ingest OPEX (Incremental) workflow should be deactivated whilst the OPEX is uploaded into the opex folder in the S3 bucket if there is a risk that the .opex files will not be loaded in advance of or alongside the content.

System properties to enable de-duplication

For the use case, where the OPEX ingest contains content which may be added to existing folders or during the addition new folders may need to be created and de-duplication options need to be set.

The de-duplication process will use either the SourceID or the Title in the .opex to determine if the folder or file is a duplicate and how to handle the duplicate.

Essentially, if the folder is found to be a duplicate of an existing folder, then the folder is not created again but content in the folder is placed in the existing folder. If the file is found to be a duplicate of an existing file, then the file is not ingested again. There are options in both cases.

For **Folders**, the system property is *ingest.folder.duplicate.check*. The options are:

- 'SourceIDFirst' when this is selected, the SourceID of the folder to be ingested is checked for its presence in Preservica and if found used to match. If this is selected but there is no SourceID the check falls back to Title.
- 'TitleOnly' when this is selected ignores the source ID, even if present and matches on Title.
- The default, if not set, is 'SourceIDFirst'.

The **File** duplicate check can only be carried out if a file SourceID is provided for the asset. The system property is *ingest.asset.duplicate.check*. The options are:

- Options are
 - Global: exclude if found anywhere in the system
 - Local: exclude if found in the location
 - None: do not check
- The default, if not set, is global.

The **default** system properties would:

- For **folders** use the SourceID to see if it is already present, and if it is not to create a new folder. If the SourceID is found on an existing folder, that folder in Preservica will be used and child folders or files added to it. If a SourceID is not found, the Title is used. If the Title is used, the hierarchy is considered to determine if the folder is unique in that location.
- For **files**, the SourceID is checked to see if the file is already present anywhere in Preservica and if it is, it is not ingested again.

The system properties for an EPC or EPCP are accessed and changed by Preservica Support.

Automatically triggered OPEX ingest

For an automatically triggered workflow to commence on ingest, the following should be in place:

- The OPEX in the named OPEX container directory e.g. opex in the designated S3 bucket
- The OPEX is in a Container with a unique name
- Both the Ingest and Incremental Ingest (OPEX) workflows are set-up and activated
- The Workflow Trigger on the Ingest OPEX (Incremental) workflow was set to Start OPEX Ingest Automatically

Review the progress of the progress using the Monitor page.

Manually triggered OPEX ingest

A manual ingest will be required if an ingest has been attempted and terminated, or the OPEX has not been placed in the specified OPEX container directory.

In this case adjust the OPEX container directory in the workflow to reflect the location and name of the top level of the OPEX e.g. opex/opex-2020-09-01/

Go to Ingest, Start, and select Run against the Ingest OPEX (Incremental) workflow.

Review the progress of the progress using the Monitor page.

Use of SourceID to determine ingest location

When using the PUT tool the folder selection enables the user to choose which folder the content should ingested under. In the Ingest OPEX (Incremental) the Folder Ref will provide this instruction. However, if different ingests need to be directed to different folders then SourceIDs can be used to provide the instruction.

If the folders already existing in Preservica, under which the new content is to be ingested, are given a SourceID; and this hierarchy with SourceIDs is included in the OPEX package then the ingest will be located in the required location regardless of the FolderRef in the Ingest OPEX (Incremental) workflow. Duplicated folders will not be created for a second time and the additional folders and assets will be place in the required location.

Example:

To ingest this folder structure (Series1/Box1) under an existing top-level folder called Pictures.

Assuming that the Ingest OPEX (Incremental) has a Folder Ref for Sound as the default.

As Pictures exists in Preservica as a top-level folder and ensuring it has been given a SourceID of Pictures in Identifiers,

Identifiers	
Type	Value
SourceID	Pictures

And with Series1 and Box1 being new at this ingest, this method will:

- put Series1/Box1 under Pictures not Sound
- will give Series1 and Box1 SourceIDs on ingest so they are present in future for later ingests.

If on the next ingest, you want to ingest under a top-level folder called Manuscripts, the same method can be used and both Ingests can use the same Ingest OPEX (Incremental) context.

The package structure for the Pictures/Series1/Box1 ingest would be:

Pictures_Container
 Pictures
 Pictures.opex
 Series1
 Series1.opex
 Box1
 Box1.opex
 Content files

With the, as a minimum, these .opex files

Pictures.opex

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<opex:OPEXMetadata xmlns:opex="http://www.openpreservationexchange.org/opex/v1.0">
  <opex:Transfer>
    <opex:SourceID>Pictures</opex:SourceID>
  </opex:Transfer>
</opex:OPEXMetadata>
```

Series1.opex

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<opex:OPEXMetadata xmlns:opex="http://www.openpreservationexchange.org/opex/v1.0">
  <opex:Transfer>
    <opex:SourceID>Series1</opex:SourceID>
  </opex:Transfer>
</opex:OPEXMetadata>
```

Box1.opex

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<opex:OPEXMetadata xmlns:opex="http://www.openpreservationexchange.org/opex/v1.0">
  <opex:Transfer>
    <opex:SourceID>Box1</opex:SourceID>
  </opex:Transfer>
</opex:OPEXMetadata>
```

For more information, please see *Preservica Tech Note 6.2 - OPEX - Instructions to select parent folder*

Initial ingest to create new Hierarchy

If this is the first OPEX ingest for a collection and the top level SourceID does not exist in Preservica, there are two options:

- (a) manually create the top folder and give it a SourceID that matches the Top folder to be ingested, using de-duplication, the OPEX will be ingested under this folder
- (b) if the top level folder .opex contains metadata and other properties which cannot be added manually in option (a), instead create a folder e.g. called ROOT and make sure the Folder Ref in the Ingest OPEX (Incremental) workflow is set to that Folder Ref, e.g.



When the top-level folder with a SourceID in the OPEX cannot be found in the target Preservica, the ingest will be placed under the specified Folder Ref. Once ingested, use *Change Archival Structure* to move the top-level folder to Root level and after that all subsequent ingests to that folder will be placed in the correct position.

Once you have the top-level folder in place, this ROOT folder is no longer required. The ROOT folder can be deleted; however, a Folder Ref is required in the Ingest OPEX (Incremental) workflow. An existing folder must be selected (not a dash -). This will be a fallback location in case the location given no longer exists,

Ingest delay property

The system property *opex.ingest.delay* controls how long the workflow will wait before it terminates if no new content is delivered to the source.

This is useful when the .opex has a Manifest element in Transfer, but the movement of files into the S3 bucket may take a significant amount of time due to the size. Setting a delay property allows for a significant time to transfer a file and ensures the workflow does not immediately decide all content has been delivered and stop the process. The default is 5 mins (300 seconds).

Ingest package size

The system property *opex.ingest.max.bytes* controls the size of the ingest packages. If this property is not used, then the package size is set to 1000 files regardless of the size of the files. If the property is used, then the ingest packages for the child workflows will be limited to the size set in the property plus one file. Note the property should be set in bytes. 1 GB is 1,073,741,824 bytes.

5. Monitoring OPEX ingests



The Monitor page is designed to give the user a view of the ingest of the OPEX without having to inspect multiple workflow contexts. These workflow contexts are available if a deeper dive is required.

Message	Process	Path	Source ID	Entity	Date
No content is ready now, so waiting for 100s for it to be ready	opex/opex-2020-07-29-08-37-05-630/	/	00a0207-ba8c-478b-9...	HQ_HSB8CPH_0530-0005-	29/07/2020, 19:37:56
Ingested into asset ad12d27a-626b-47cc-a0a7-908a357da363	opex/opex-2020-07-29-08-37-05-630/	8a815a8c-839e-41fc-bdfa-8a7ebbb7e466/32058abd-f8...	00a999d6-bb1f-4079-bf1...	UK_0276-4396_0002.jpg	29/07/2020, 19:37:46
Ingested into asset 9912e8cc-f319-4937-819b-9d53bfcde055	opex/opex-2020-07-29-08-37-05-630/	8a815a8c-839e-41fc-bdfa-8a7ebbb7e466/32058abd-f8...	00a999d6-bb1f-4079-bf1...	UK_0276-4396_0002.jpg	29/07/2020, 19:37:46
Ingested into asset 41f4e498-0522-4963-ab4e-ab56d9564d739	opex/opex-2020-07-29-08-37-05-630/	8a815a8c-839e-41fc-bdfa-8a7ebbb7e466/32058abd-f8...	58c7a673-3f6b-4381-9b...	200208-csg-update-2019-	29/07/2020, 19:37:45
Ingested into asset 1a0ab765-a524-4912-94e8-09d74877f1be	opex/opex-2020-07-29-08-37-05-630/	8a815a8c-839e-41fc-bdfa-8a7ebbb7e466/32058abd-f8...	5c123293-b15b-4c4b-a3...	HQ_HSB8CPH_0130-0011-A...	29/07/2020, 19:37:45
Ingested into asset a755230a-b4da-44d3-b0b9-a372345ae7a5	opex/opex-2020-07-29-08-37-05-630/	8a815a8c-839e-41fc-bdfa-8a7ebbb7e466/32058abd-f8...	3b4f6cc3-a2d9-4082-a2...	Water_Stories_Master_v...	29/07/2020, 19:37:45
Ingested into asset 3040fba-c42b-440f-999d-c897d0370433	opex/opex-2020-07-29-08-37-05-630/	8a815a8c-839e-41fc-bdfa-8a7ebbb7e466/32058abd-f8...	e21f0fc-1769-40d2-a2c...	(p.25)Getting the orders...	29/07/2020, 19:37:45
Ingested into asset 08da329c-7cd8-4702-b3d1-f4cd343781be	opex/opex-2020-07-29-08-37-05-630/	a4575c4d-169d-4a9b-bb34-5710f6d6766b/a3c08a89-L...	601e84d4-6c04-4c13-b0...	shutterstock_11962432.jpg	29/07/2020, 19:37:45
Ingested into asset 88a540c9-2d5e-41ad-acc0-529dd789fb28	opex/opex-2020-07-29-08-37-05-630/	a4575c4d-169d-4a9b-bb34-5710f6d6766b/a3c08a89-L...	00a0207-ba8c-478b-9...	HQ_HSB8CPH_0530-0005-	29/07/2020, 19:37:45

The **Top Panel** of the Monitor shows a “card” per OPEX ingested.

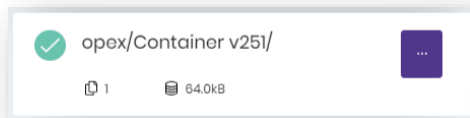
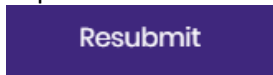
The Top Left Card will be the most recent. The order is left to right and then top to bottom. The card contains

- The name of the OPEX
- The status, in icon form
- The number of files/folders ingested
- The size of the ingest
- The number of errors and warnings for the OPEX if any

The **Bottom Panel** will show a line per message.

Message	Process	Path	Source ID	Entity	Date
No content is ready now, so waiting for 100s for it to be ready	opex/opex-2020-07-29-08-37-05-630/	/	00a0207-ba8c-478b-9...	HQ_HSB8CPH_0530-0005-	29/07/2020, 19:37:56
Ingested into asset ad12d27a-626b-47cc-a0a7-908a357da363	opex/opex-2020-07-29-08-37-05-630/	8a815a8c-839e-41fc-bdfa-8a7ebbb7e466/32058abd-f8...	00a999d6-bb1f-4079-bf1...	UK_0276-4396_0002.jpg	29/07/2020, 19:37:46
Ingested into asset 9912e8cc-f319-4937-819b-9d53bfcde055	opex/opex-2020-07-29-08-37-05-630/	8a815a8c-839e-41fc-bdfa-8a7ebbb7e466/32058abd-f8...	00a999d6-bb1f-4079-bf1...	UK_0276-4396_0002.jpg	29/07/2020, 19:37:46

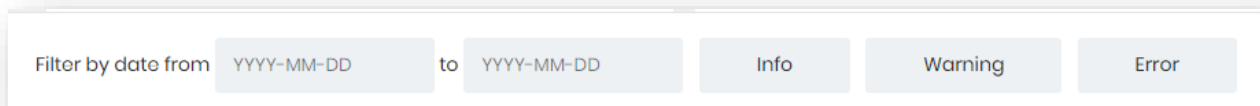
If the Opex Ingest workflow can be **restarted**, a purple menu will appear on the card. Clicking on the three dots (...) will bring up a Resubmit option:



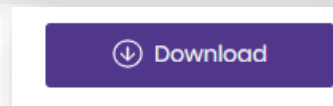
The most recent message will be at the top. The information displayed will be

- Icon to indicate error, warning, info
- The message
- The related process e.g. the name of the OPEX
- The path of the file or folder
- The Source ID (if there is one) taken from the .opex
- The ingested entity (if ingested)
- The date and time of the activity

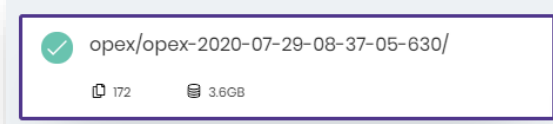
There are options to allow filtering of the messages



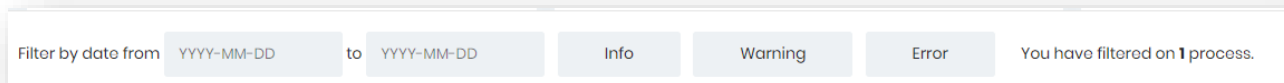
And the ability to download the messages in a csv file.



To see the **messages associated with a single OPEX**, click on the card. The card will be identified by the box around it.

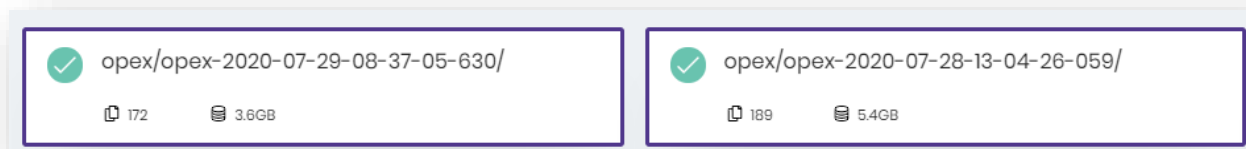


The user will see the message: You have filtered on 1 process



The message list in the bottom panel will only include messages associated with the selected card.

To see the **messages associated with multiple OPEXs**, click on the cards of interest. The cards will remain selected until they are clicked again to unselect them.

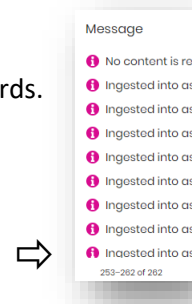


The user will see the message: You have filtered on x processes (where x is 2 in this case).

The message list in the bottom panel will only include messages associated with the selected cards.

At the bottom of the page the user will see the **counts of records** and identify which ones are visible in the form (start) – (end) of (total) in the bottom left hand corner.

If there is no data to show the message “No data” will be displayed.



The **filter options** in the bottom panel can be used to filter the messages

- by date
- by message type e.g. Info, Warning or Error
 - o Click on to select and click again to deactivate.
 - o These options can be multi-selected. The options are on if highlighted in purple.

In the top panel the status of the OPEX ingest overall can be determined from the **Status Icon**

The user sees a Status Icon that indicates success – a green circle with a tick.
The hover-over says **Succeeded**.

The user sees a Status Icon that indicates failure – a red circle with a cross.
The hover-over says **Failed**.

The user sees a Status Icon that indicates Pending – a blue circle with a clock face.
The hover-over says **Pending**.

The user sees a Status Icon that indicates Running – a blue circle with three dots.
The hover-over says **Running**.

The user sees a Status Icon that indicates Suspended – a white circle with amber highlights and a pause image. The hover-over says **Suspended**.

The user sees a Status Icon that indicates Error (Recoverable) an orange circle with a cross and an anticlockwise arrow. The hover-over says **Error (Recoverable)**.



5.1. Decoding monitor messages

The messages listed below cover the messages that are displayed in the process monitor, some further information will be available in the underlying ingest workflows for the batch.

Messages	Cause/Explanation
Error messages	
Failed to ingest asset {ref}	The asset did not ingest. Ingest will continue for other assets.
The manifest references file {path} but this has not been present in the source	A file or folder listed in the manifest part of the Folder .opex was not present
This workflow requires folder manifests, but there is no manifest in the folder opex {path}	In the OPEX (Incremental) workflow, the tickbox for Require Folder Manifests was selected but there was no manifest was in the .opex
The content opex file {path} arrived late, and the associated file was already ingested. This can be avoided by using folder manifests.	A .opex arrived in the S3 bucket after the files had been ingested. The .opex may have included SourceIDs or manifests that may have resulted in some items not being ingested. The process is stopped so the manager can judge if there is an issue.
Warning messages	
Unable to initiate download for metadata file {path}	The metadata file (e.g. .opex) could not be downloaded. Ingest will continue for other assets.
The opex file {path} is present in the source but the corresponding content file is not present	The .opex for the file is present but not the file to which it relates
This workflow requires folder manifests, but there is no folder opex {path}	In the workflow the option is ticked to say there was a folder manifest, but it is not present
Info messages	
Skipping download of {path} because the {newMatchText} is already present (in {localRefs}) in the same SO ({directoryMatchText})	When Local Matching for Assets is in place, this message explains that because Asset name matches an existing Asset in the Folder and will not be ingested.
Skipping download of {path} because the {newMatchText} is already present (globalRef)	When Global Matching for Assets is in place, this message explains that because Asset name matches an existing Asset anywhere in Preservica and will not be ingested.
Matched directory {path} to existing SO {entityRef} via {matchText}	When matching by titles is in place (SourceID is not present or TitleOnly matching is set), a matched Folder by Title has been identified.
No content is ready now, so waiting for {delay} for it to be ready	Because no content is in the S3 bucket, the system will wait for the delay period.
Ingested into asset {entityRef}	Successful ingest of this asset.
There are extra files remaining within the directory	The folder manifest in the folder .opex does not include all the files found in the Folder Manifest, this should be investigated.
The process was suspended because of an unexpected failure in the ingest workflow: code {code}	If the process was suspended because of an unexpected failure in the ingest workflow with a code this is given.
The process was resumed	If the process was stopped with a retry-able error and then resumed, the message given is "The process was resumed".



<http://www.preservica.com/>

Preservica, 32 The Quadrant, Abingdon Science Park, Abingdon, Oxfordshire, OX14 3YS, UK
info@preservica.com - preservica.com