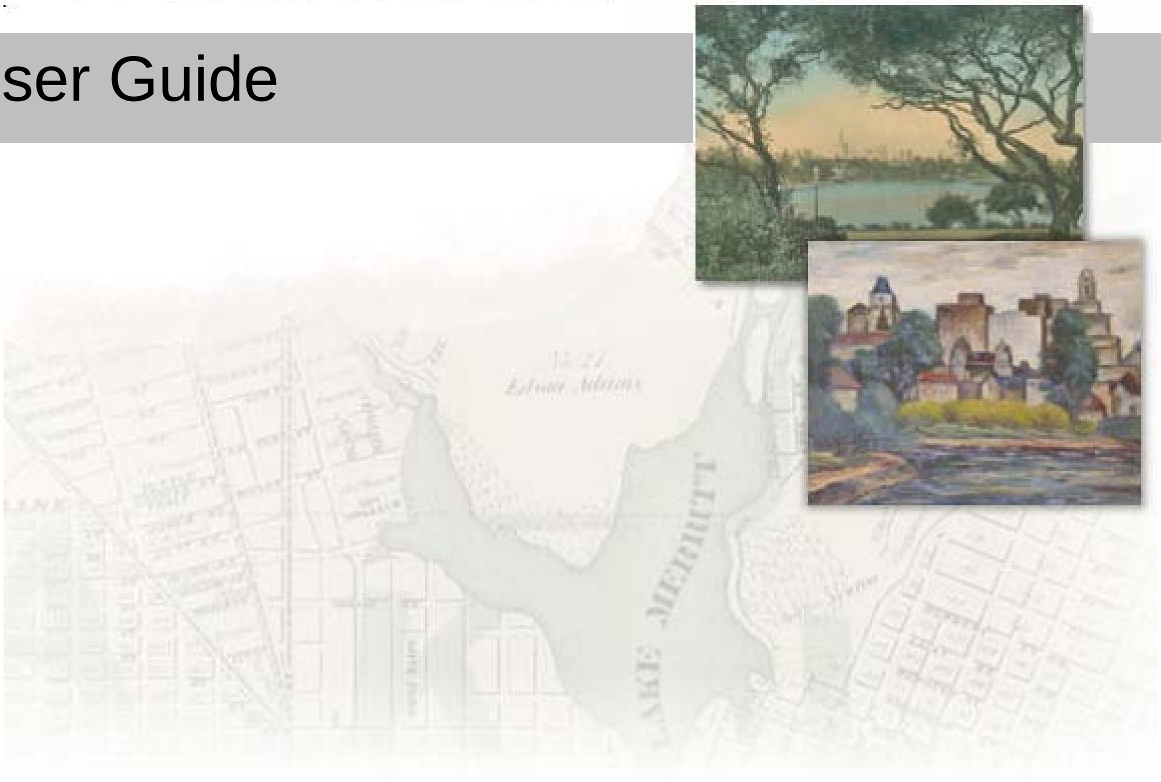
UC3Merritt

# User Guide



Updated June 2012
A service of the University of California Curation Center of the California Digital Library
http://www.cdlib.org/uc3

## Contents

## About Merritt

Merritt is a new cost-effective service that provides the UC community with an easy to use tool to manage, archive, and share their content. Merritt provides significant features for valuable digital objects:

- permanent storage
- access via persistent URLs
- tools for long-term management
- easy-to-use interface for deposit and updates

**Why use Merritt?**

**Take control of your content and provide access how and when you want**
Merritt enables campus constituencies to have direct control over the management, curation, access, and preservation of the information resources underpinning their scholarly activities.

**Share research with others**
UC researchers have datasets that they would like to share within and across disciplinary communities. Merritt users can share content via persistent URLs.

**Meet the data sharing and preservation requirements of a grant-funded project**
Increasingly researchers are required by funding agencies to preserve and share data (such as observational data, experimental results, and code books) as a condition of a grant. Merritt can be used to manage, share, and preserve the research outputs of a grant.

**Provide long-term preservation**
The UC community holds valuable and unique collections in their libraries, museums, and archives (e.g. specimens, oral history recordings, or photographs) that they would like to make available online and preserve for future use. Merritt provides a trustworthy and dependable environment for the long-term management of these valuable materials.

**Frequently Asked Questions**

| | |
|---|---|
| **What kind of content can be put in Merritt?** | Content from all disciplines (sciences, humanities, social sciences) can be deposited in Merritt, including images, videos, datasets, texts, and more. |
| **Who can use Merritt?** | Merritt is available to the UC community. |
| **What are the ways to use Merritt?** | Merritt is available in two ways: 1) via an API (machine to machine programming interface) or 2) a user interface. Either method provides access to all of Merritt's functions. |
| **What does it cost to use Merritt?** | The UC Curation Center provides digital library services to the University of California. UC departments and organizations are charged only for the storage used. |

| | |
|---|---|
| **Who provides support for Merritt?** | Merritt was developed and is supported by the UC Curation Center (UC3). |
| **What is the UC Curation Center?** | UC3 is a creative partnership bringing together the expertise and resources of the California Digital Library (CDL), the ten UC campuses, and the broader international curation community. See http://www.cdlib.org/ uc3. |
| **What is the technical infrastructure behind Merritt?** | The Merritt infrastructure, based on the micro-services concept, was developed by UC3 in consultation with the UC libraries and the international curation community. The infrastructure enables enhanced responsiveness to the ever increasing number, size, and diversity of content, partners, and stakeholder expectations, while simultaneously accommodating an ever changing technical environment. See http://www.cdlib.org/uc3/curation for more information. |
| **Why is it called Merritt?** | Merritt refers to Lake Merritt, near the CDL offices in Oakland. Among other noteworthy attributes, Lake Merritt was the first official wildlife refuge in the United States, and is also a National Historic Landmark. |

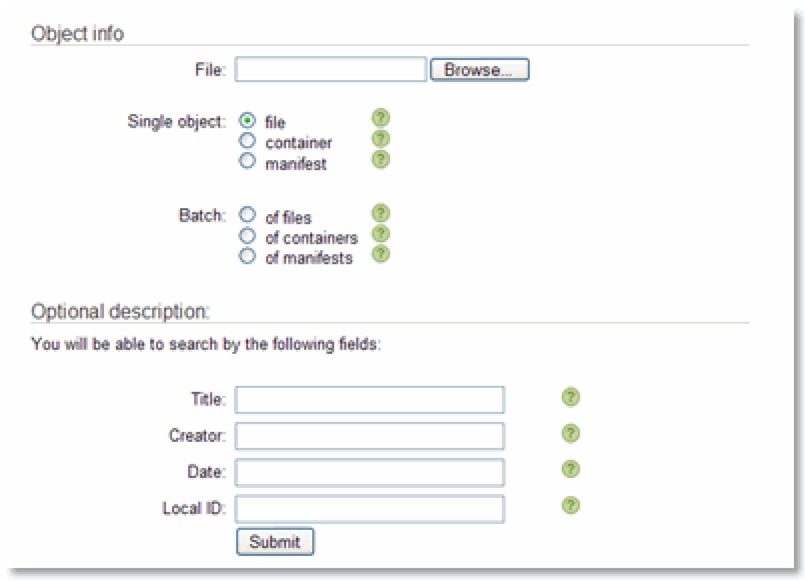
## Adding Objects

You may submit digital objects to Merritt several different ways, based on what's easiest for your local workflow:

- **Merritt User Interface**: directly submit digital objects, either one-by-one or in batches, using the Merritt web interface.
- **Merritt API**: programmatically submit digital objects either one-by-one or in batches using the Merritt API.
- **Merritt METS Feeder**: if your digital objects are formatted using the METS encoding standard, you can easily single or multiple objects using the Merritt METS Feeder.

### Submitting Objects through the Merritt User Interface

The **Add Object** screen allows you to submit digital objects one-by-one or in batches. Single objects can be uploaded directly from your local drive. Batches must be submitted via a manifest file, which you prepare and upload from your local drive.



Object info

File: Browse...

Single object: [✓ file] container manifest

Batch: of files of containers of manifests

Optional description:

You will be able to search by the following fields:

Title:

Creator:

Date:

Local ID:

Submit

The following is a brief description of the options for adding objects. For any option involving a manifest, consult the detailed Using a Manifest guide.

**Single object: file**

Upload a single file from a local drive.

**Single object: container**

Upload a .zip or .tar file from your local drive. This container file may contain any number of component files or associated metadata files for a single object.

**Single object: manifest**

As an alternative to wrapping a multi-file object in a .zip or .tar file, you can also create a .checkm format text manifest to upload the object. You will need to post all component files for the object on a web server, then upload a manifest file that points to the object components. This option allows you to provide and validate checksum information for each file in an object. Details are provided in the Using a Manifest guide.

**Batch: of files**

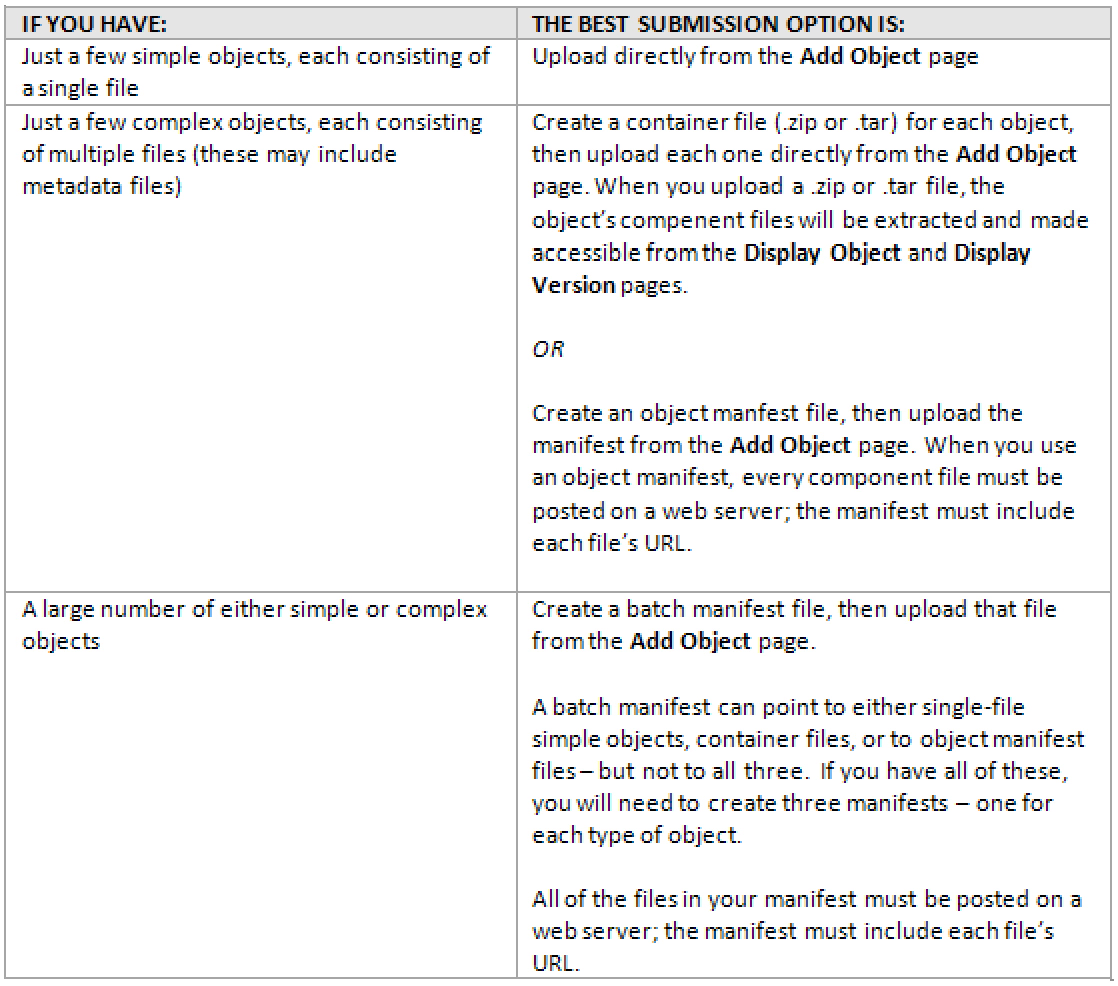
Use this option when you have a long list of simple objects to add. This option requires use of a manifest file. You can provide metadata for each object in the manifest.

**Batch: of containers**

Use this option when you have many complex objects to add and you have enclosed each object in a .zip or .tar file. This option requires use of a manifest file.

**Batch: of object manifests**

Use this option if you have prepared object manifests for a large number of objects.



| IF YOU HAVE: | THE BEST SUBMISSION OPTION IS: |
|---|---|
| Just a few simple objects, each consisting of a single file | Upload directly from the **Add Object** page |
| Just a few complex objects, each consisting of multiple files (these may include metadata files) | Create a container file (.zip or .tar) for each object, then upload each one directly from the **Add Object** page. When you upload a .zip or .tar file, the object's compenent files will be extracted and made accessible from the **Display Object** and **Display Version** pages.<br><br>*OR*<br><br>Create an object manfest file, then upload the manifest from the **Add Object** page. When you use an object manifest, every component file must be posted on a web server; the manifest must include each file's URL. |
| A large number of either simple or complex objects | Create a batch manifest file, then upload that file from the **Add Object** page.<br><br>A batch manifest can point to either single-file simple objects, container files, or to object manifest files – but not to all three. If you have all of these, you will need to create three manifests – one for each type of object.<br><br>All of the files in your manifest must be posted on a web server; the manifest must include each file's URL. |

When you add an object you will receive a confirmation screen with a batch ID number. This screen confirms that the object has been submitted, but does not yet confirm that the object has been successfully ingested. Merritt takes a number of steps (such as fixity checking ) before an object is ingested. In most cases it will take a couple of minutes before the object will be retrieved from a search and displayed.

You will receive an email with a confirmation when the object has either succeeded or failed the ingest process. Any error details will be provided in the email message; you can contact **uc3@ucop.edu** if you have any questions.

Merritt allows you to provide title, creator, date and local identifier metadata either by filling out the **Optional Description** form, by providing metadata in a manifest file, or by submitting an **mrt-erc.txt** file with the object. The metadata you provide becomes searchable in Merritt; we strongly recommend that you provide at least title information. Further details are in the **Merritt and Metadata** guide.

## Submitting Objects through the Merritt API

You can programmatically submit digital objects one-by-one or in batches, using our API. As described in the Ingest specification document, the request arguments are:

| Argument | Value |
|---|---|
| **filename** | (optional) The name of the file |
| **file** | The file itself (e.g., the single file; or the container file, such as a .zip or .tar file). If submitting a manifest, indicate the .checkm format text manifest file here.; consult the detailed Using a Manifest guide for additional information on preparing the manifest |
| **type** | Valid values: <br>• file <br>• container <br>• object-manifest <br>• batch-manifest <br>• container-batch-manifest <br>• single-file-batch-manifest |
| **profile** | The submission profile, which we will provide to the submitter |
| **primaryIdentifier** | (optional) ARK identifier, if known |
| **localIdentifier** | (optional) local identifier, if known |
| **digestType** | (optional) valid values: <br>• adler-32 <br>• crc-32 <br>• md2 <br>• md5 <br>• sha-1 <br>• sha-256 <br>• sha-384 <br>• sha-512 |
| **digestValue** | (optional) digest value, hex-encoded string |
| **creator** | (optional) creator |

| | |
|---|---|
| **title** | (optional) title |
| **date** | (optional) date |
| **note** | (optional) descriptive note |
| **responseForm** | (optional) valid values:<br>• anvl<br>• csv<br>• json<br>• turtle<br>• xhtml<br>• xml |

All of the enumerated values (file type, digest type, response form) are case-insensitive.

**Sample cURL**

```
curl --silent -u user:password \
-F "file=@ucsf_etd_200609.checkm" \
-F "type=container-batch-manifest" \
-F "submitter=username" \
-F "responseForm=xml" \
-F "profile=merritt_demo_content" \
-F "localIdentifier=local-ID-test" \
https: //merritt-stage.cdlib.org/object/ingest
```

## Submitting Objects through the Merritt METS Feeder

If your digital objects are formatted using the METS encoding standard, you may submit them to Merritt using our METS Feeder process. The steps to submitting content using METS:

1) First, prepare your objects in the METS format, using one of the profiles described in the CDL Guidelines for Digital Objects. The METS objects -- and all associated files referenced within the METS objects -- should be on a web-accessible server. If you need to open up a firewall, we can give you the addresses of the machines that need to access the files.

2) Create a "manifest," which is a just simple list that enumerates the URLs to the METS objects. The list should itself be formatted as a text file, with line breaks between each METS object, looking like this:

```
http: //URL/subdir/metsfile1.xml
http: //URL/subdir/metsfile2.xml
```

etc.

A sample manifest is available at http://pwillett.bitbucket.org/METSsample.txt

Next, place the manifest on a web-accessible server.

3) Then send the URL of the manifest to Merritt METS Feeder. The URL should have this format:

http://<feederURL>/?userID=<userID>&authCode=<authCode>&accessGroupID=<accessGroupID>&manifestURL=<manifestURL>
with these elements:

1. feederURL (for Merritt stage, this is “feeder-stage.cdlib.org/feeder-mets/mets/ “.  For Merritt production, this is “feeder.cdlib.org/feeder-mets/mets/”)
2. userID (login)
3. authCode (password)
4. accessGroupID (collection name--we can let you know the collection name)
5. fillin (Optional. if "false", this will suppress fixity check of object components by feeder, speeding up the submission. The implicit default is &fillin=true)
6. manifestURL (with URL encoding replacing ":" (%3a) "/" (%2f) etc)

To fill in some of the variables, it would look like this (no line breaks):

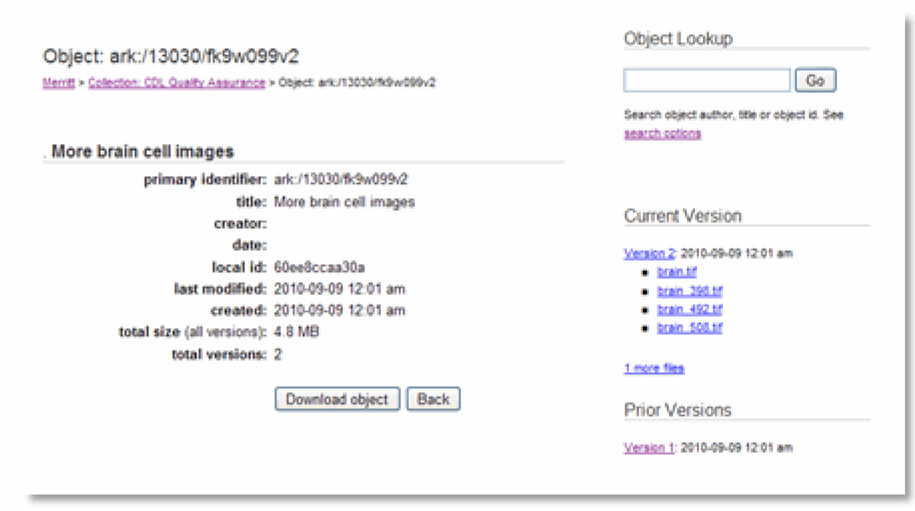
http: //feeder-stage.cdlib.org/feeder-mets/mets/?userID=yourLogin&authCode=yourAuthCode&accessGroupID=ucsd_etd&manifestURL=http%3a%2f%URL%2fsubdirectory%2fFilename

4) We will retrieve the list and ingest the METS objects.

## Retrieving Objects

If you used the Merritt metadata options to provide a title, creator, date or local identifier, you will be able to search within the collection for any of those terms. Terms typed in the search box will be found in any of these fields, as well as in the primary identifier assigned by Merritt.

The **Display Object** screen shows the complete metadata provided by the curator as well as system-generated metadata such as primary identifier, file size, date added (created) and date last modified.



Object: ark:/13030/fk9w099v2

Merritt > Collection: CDL Quality Assurance > Object: ark:/13030/fk9w099v2

. More brain cell images

| | |
|---|---|
| primary identifier: | ark:/13030/fk9w099v2 |
| title: | More brain cell images |
| creator: | |
| date: | |
| local id: | 60ee8ccaa30a |
| last modified: | 2010-09-09 12:01 am |
| created: | 2010-09-09 12:01 am |
| total size (all versions): | 4.8 MB |
| total versions: | 2 |

Object Lookup

Search object author, title or object id. See search options

Current Version

Version 2: 2010-09-09 12:01 am

- brain.tif
- brain_398.tif
- brain_492.tif
- brain_508.tif

1 more files

Prior Versions

Version 1: 2010-09-09 12:01 am

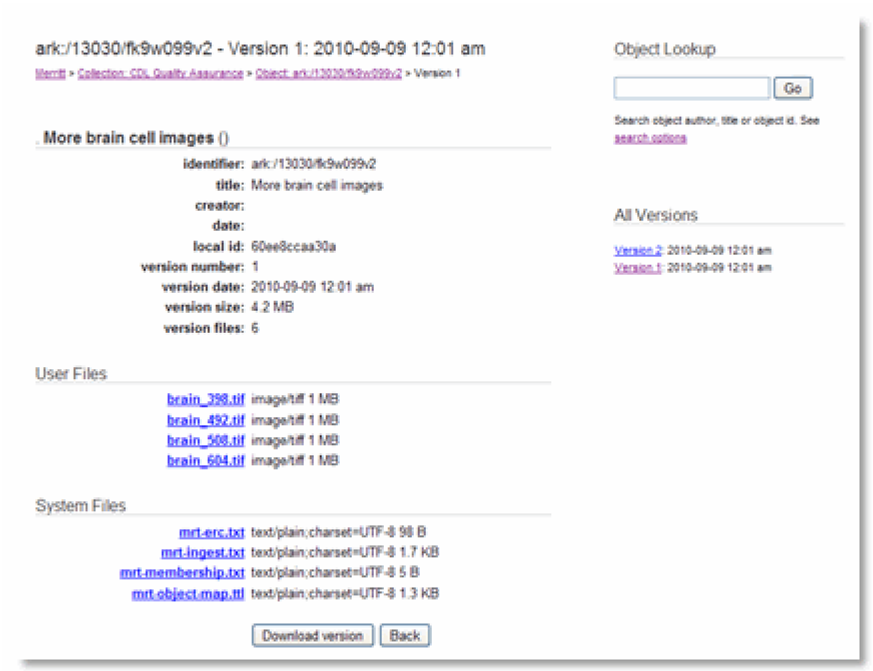
If there are multiple versions of an object, information about the most recent version will display on the **Display Object** page.

If there are multiple component files for the object, up to four of them will be listed in the right sidebar. Clicking on an object file will open the file.

If there are multiple versions of the object, a link to each version will be available in the right sidebar. The **Display Version** page, which will provide links to every component file for the object.

The **Download Object** button on the **Display Object** page will download a .zip file including every version of the object.

The **Display Version** screen provides links to each component file you supplied for the object, along with system-derived metadata.



ark:/13030/fk9w099v2 - Version 1: 2010-09-09 12:01 am

Merritt > Collection: CDL Quality Assurance > Object: ark:/13030/fk9w099v2 > Version 1

More brain cell images ()

| | |
|---|---|
| identifier: | ark:/13030/fk9w099v2 |
| title: | More brain cell images |
| creator: | |
| date: | |
| local id: | 60ee8ccaa30a |
| version number: | 1 |
| version date: | 2010-09-09 12:01 am |
| version size: | 4.2 MB |
| version files: | 6 |

User Files

| | |
|---|---|
| brain_398.tif | image/tiff 1 MB |
| brain_492.tif | image/tiff 1 MB |
| brain_508.tif | image/tiff 1 MB |
| brain_604.tif | image/tiff 1 MB |

System Files

| | |
|---|---|
| mrt-erc.txt | text/plain;charset=UTF-8 98 B |
| mrt-ingest.txt | text/plain;charset=UTF-8 1.7 KB |
| mrt-membership.txt | text/plain;charset=UTF-8 5 B |
| mrt-object-map.ttl | text/plain;charset=UTF-8 1.3 KB |

Search object author, title or object id. See search options

All Versions

Version 2: 2010-09-09 12:01 am
Version 1: 2010-09-09 12:01 am

The **Display Version** screen also provides access to the files that Merritt generates to facilitate object management.

The **Download Version** button will provide you with a .zip file of the version you are viewing. The download will include the system files Merritt generated.

## Merritt and Metadata

We strongly recommend that descriptive metadata accompany each digital object. Metadata are critical to managing digital collections and providing meaningful reports. We are aware that you most likely have descriptive metadata about each object, stored in MARC, MODS, METS, or some other format. If you have metadata records, we will store and preserve them as part of the digital object. To preserve metadata records with a digital object, we recommend that you create a container file (.zip or .tar) include all object component files as well as the metadata files and submit the container file to Merritt.

Merritt also enables you to use metadata to discover and manage content in the repository. You can provide information about title, creator (author), date and local identifier either by typing in metadata on the **Add Object** form, by using a batch manifest, or by submitting a **mrt-erc.txt** file with the object. This guide will help you choose the best approach. Note that elements contained within associated metadata files such as METS or MARC will be preserved but not searchable. If these files contain title or author information, you will also need to provide that data via the **Add Object** form, a manifest or an **mrt-erc.txt** file.

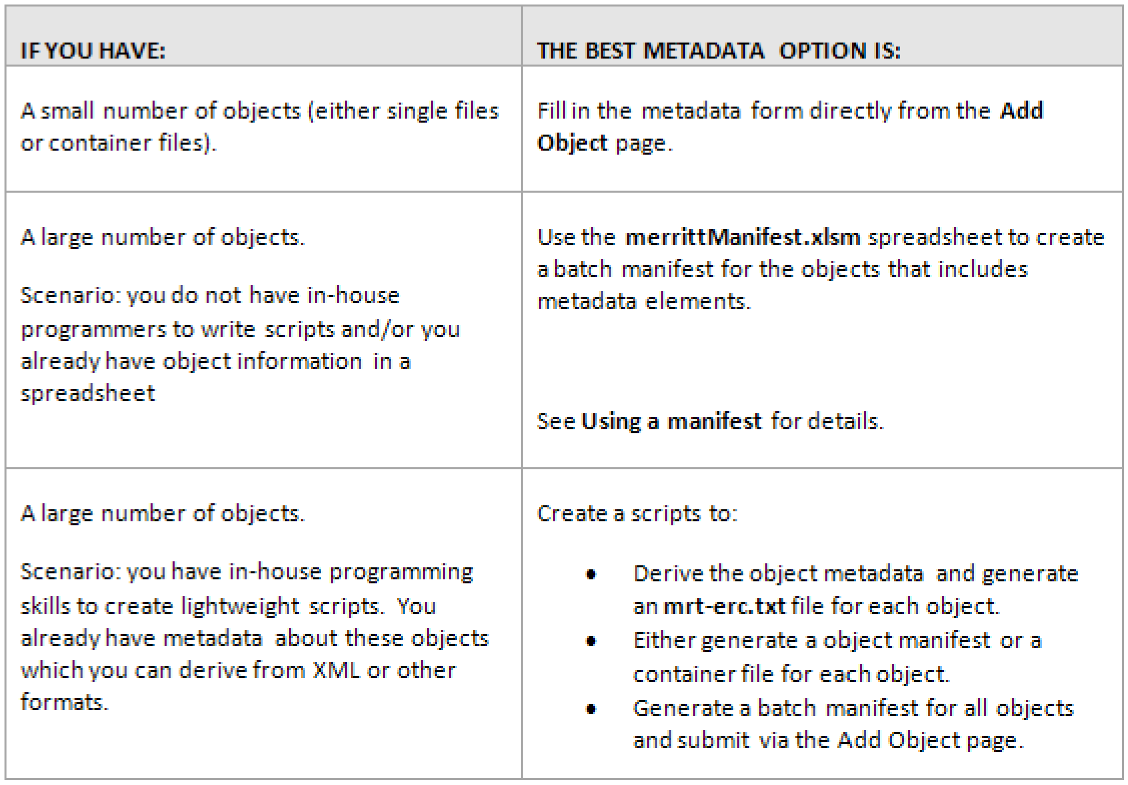
## The Metadata Elements

Merritt uses Dublin Core Kernel Metadata, also expressed as Electronic Resource Citations (ERCs), to provide access to content. These four elements can have different names depending on the means of submission:

| ERC (mrt-erc.txt file) | Dublin Core | Add Object Screen | Manifest File |
|---|---|---|---|
| Who | DC.creator | creator | creator |
| What | DC.title | title | title |
| When | DC.date | date | date |
| Where | identifier | Local Identifier | localIdentifier |

Not only are all of these elements searchable in Merritt, but the **local identifier** element can be used to edit content. (If you have a large number of objects to edit, you can submit a batch manifest that specifies the local identifiers of the objects to be edited along with the revised content - see Editing Objects for details.) At this time these elements are not repeatable.

## Metadata Submission Options



| IF YOU HAVE: | THE BEST METADATA OPTION IS: |
|---|---|
| A small number of objects (either single files or container files). | Fill in the metadata form directly from the **Add Object** page. |
| A large number of objects.<br><br>Scenario: you do not have in-house programmers to write scripts and/or you already have object information in a spreadsheet | Use the **merrittManifest.xlsm** spreadsheet to create a batch manifest for the objects that includes metadata elements.<br><br>See **Using a manifest** for details. |
| A large number of objects.<br><br>Scenario: you have in-house programming skills to create lightweight scripts. You already have metadata about these objects which you can derive from XML or other formats. | Create a scripts to:<br>• Derive the object metadata and generate an **mrt-erc.txt** file for each object.<br>• Either generate a object manifest or a container file for each object.<br>• Generate a batch manifest for all objects and submit via the Add Object page. |

## The ERC format

The Electronic Resource Citation specification is meant to be lightweight and easy-to-supply and provides easy-to-read, meaningful information about the object. We'll use the ERC

metadata you supply in reports about your digital objects and collections. For further details, see the Electronic Resource Citation Specification at:
http://dublincore.org/kernelwiki/FrontPage?action=AttachFile&do=get&target=ercspec.html

Sample ERC file:

```
erc:
who: (:unkn) unknown
what: Jazz for the bears
when: 1920-1932
where: 2001697390
```

If you are using the ERC method, be sure to name the file **mrt-erc.txt** and either include it in a container file with the object or reference it in an object manifest file.

## Using a Manifest to Submit Objects to Merritt

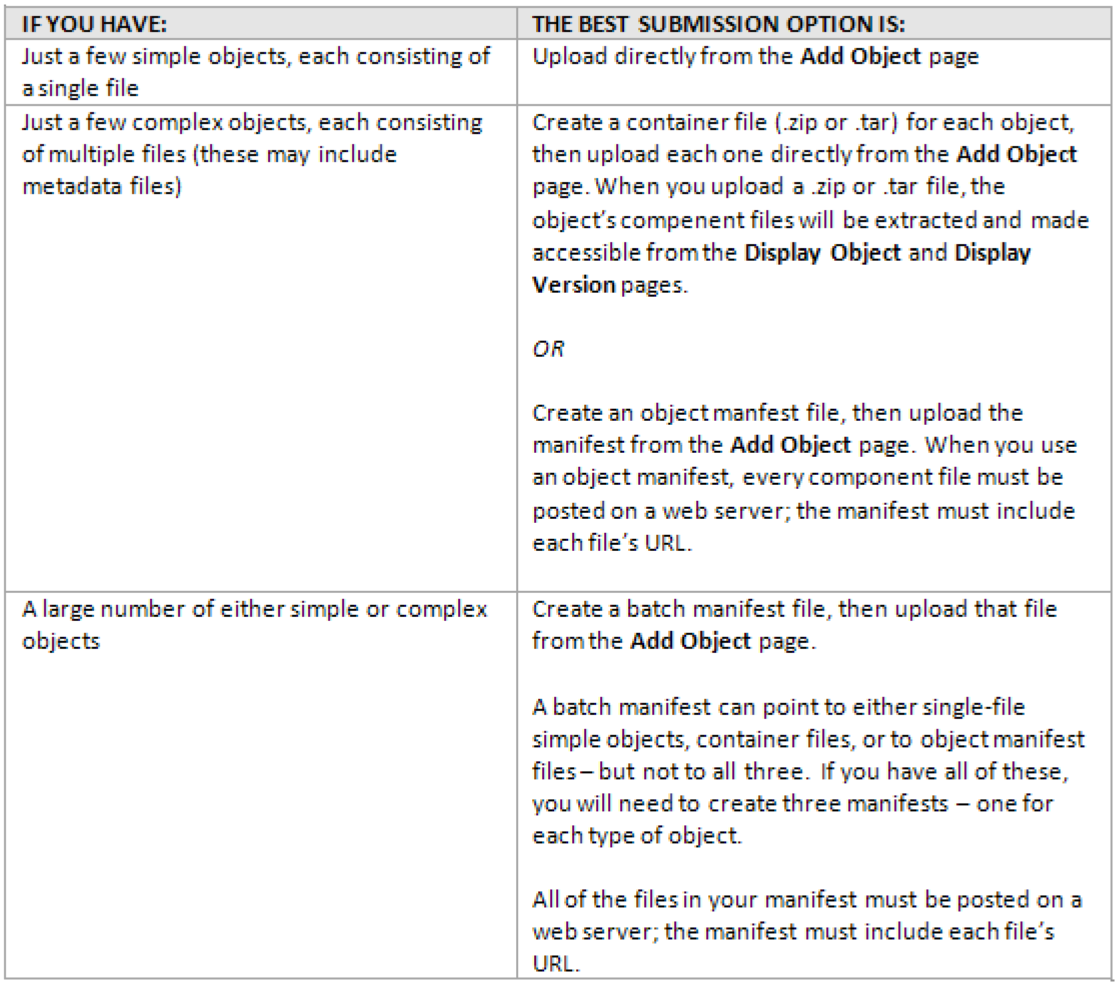
There are a variety of ways to submit digital objects to Merritt; the method you choose will depend on the nature of the digital objects and how many objects you have to submit. One option is to use a *manifest* to add either complex objects consisting of many files or a large batch of objects. A manifest is a simple pipe-delimited text file containing basic information about the files being submitted; just post the objects to a web server and use the **Add Object** page to submit the manifest that points to them.

CDL has created Excel macros that will transform an Excel worksheet into a properly formatted checkm manifest file. The http://merritt.cdlib.org/docs/merrittManifest.xls file contains the macros for generating a manifest as well as worksheets with sample data.

This guide will show you how to prepare manifest files to submit objects to Merritt, either individually or in batches. The guide assumes that you are using MS Excel 2007 to record information about the objects you're submitting, and running a macro to produce a manifest.

You can also use a text editor to create manifest files if you prefer; details are included in the Tips for Creating Text-based Manifests section of this guide.

### Why Use a Manifest?



| IF YOU HAVE: | THE BEST SUBMISSION OPTION IS: |
|---|---|
| Just a few simple objects, each consisting of a single file | Upload directly from the **Add Object** page |
| Just a few complex objects, each consisting of multiple files (these may include metadata files) | Create a container file (.zip or .tar) for each object, then upload each one directly from the **Add Object** page. When you upload a .zip or .tar file, the object's compenent files will be extracted and made accessible from the **Display Object** and **Display Version** pages.<br><br>*OR*<br><br>Create an object manfest file, then upload the manifest from the **Add Object** page. When you use an object manifest, every component file must be posted on a web server; the manifest must include each file's URL. |
| A large number of either simple or complex objects | Create a batch manifest file, then upload that file from the **Add Object** page.<br><br>A batch manifest can point to either single-file simple objects, container files, or to object manifest files – but not to all three. If you have all of these, you will need to create three manifests – one for each type of object.<br><br>All of the files in your manifest must be posted on a web server; the manifest must include each file's URL. |

**For a single object:** In most cases it will be easiest to submit single objects via the Merritt user interface rather than using an object manifest. There are two reasons you might prefer to use an object manifest to submit a single object to Merritt:

1. You have checksum values for every file in a complex object, and you would like each file-level checksum to be verified by Merritt upon ingest.
2. You have a complex object consisting of many files, but do not have access to a utility to create a .zip or .tar container file.

**For a batch of objects:** If you have a large number of objects to submit, it may be more efficient to use a batch manifest. If each object consists of many files, you can create container (.zip or .tar) files for each object, post those container files to a web server, then create a batch manifest with URLs for the container files. You will also be able to supply metadata about each object in the batch manifest.

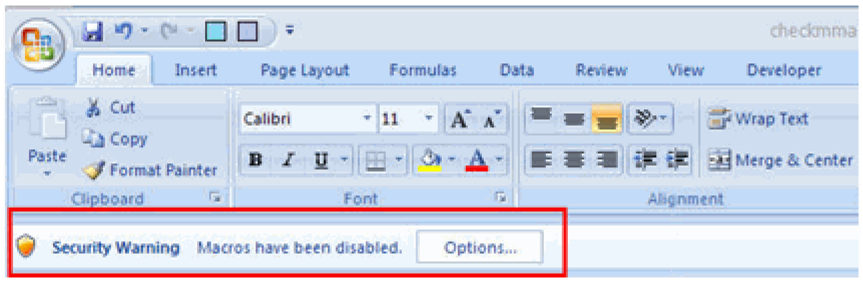
A batch manifest will be an especially good option if you already have information about the objects available in a spreadsheet format, or can easily export a spreadsheet report from another system.

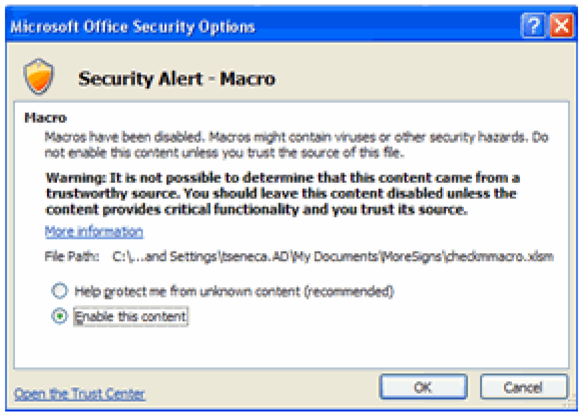
You can also submit a batch manifest that points to object manifests for complex objects. This is a good option if you have in-house scripting expertise and a way to derive object metadata. This is also the option to choose if you have checksum values for every individual component file of every object, and you want Merritt to validate each checksum upon ingest.

**The Excel Macro**

The **merrittManifest.xls** Excel file contains four macros that allow you to create manifests for individual objects and batches of single files, container files or manifest files. This spreadsheet also includes sample worksheets showing the information needed for the batch and object manifests.

When you open the merrittManifest.xls file, you may see a security warning to let you know that the file contains macros, and that they have been disabled. Click **Options**, select **enable this content**, then click **OK**.





Microsoft Office Security Options

Security Alert - Macro

Macro

Macros have been disabled. Macros might contain viruses or other security hazards. Do not enable this content unless you trust the source of this file.

**Warning: It is not possible to determine that this content came from a trustworthy source. You should leave this content disabled unless the content provides critical functionality and you trust its source.**

More information

File Path: C:\...and Settings\tseneca.AD\My Documents\MoreSigns\checkmmacro.xlsm

○ Help protect me from unknown content (recommended)

● Enable this content

Open the Trust Center    OK    Cancel

You can use this excel file directly and simply create a new worksheet for each object or each batch. It's important that each worksheet includes exactly the same columns in exactly the same order, so you may want to copy and paste the column headings from a sample worksheet into your own worksheet. (If you are only entering the bare minimum elements, you will still need headers for the columns you don't use). Note that the macro will ignore any text that is bold, so the header row must be in bold, and no other text should be bold. The headings for required columns are highlighted in orange.

**Workflow: A Single Object**

1. Place the object file(s) on a web server.
2. Download the **merrittManifest.xls** file

3. Create a new worksheet for the object manifest. There can be only one object per object manifest.
4. Provide a URL and file name for each file in the object. You may optionally provide checksum and file size information; **object manifest** section of this guide for details.
5. Choose **View > Macros > View Macros > ThisWorkbook.CreateObjectManifest > Run**
6. Name your object manifest file. A file extension of .checkm will automatically be added to the file.
7. Login to Merritt. If you work with multiple collections, choose the collection this object will be submitted to.
8. Choose **Add Object**.
9. Select the **Single object : Manifest** radio button, and browse to select the object manifest you just created.
10. You will receive an email to acknowledge that the object submission has been received. You will receive another email to tell you whether the object was added successfully or not.

**Workflow: A Batch of Container Files**

1. If your digital objects are composed of many files, create a container (.zip or .tar) file for each object.
2. Place the container files on a web server.
3. Download the **merrittManifest.xls file**.
4. Create a new worksheet for your batch manifest.
5. Provide a URL and file name for each container file in the batch. Provide any optional metadata (details in The Batch Manifest section).
6. Choose **View > Macros > View Macros > ThisWorkbook.BatchOfContainerFiles > Run**.
7. Name your batch manifest file. A file extension of .checkm will automatically be added to the file.
8. Login to Merritt. If you work with multiple collections, choose the collection this batch will be submitted to.
9. Choose **Add Object**.
10. Select the **Batch of containers** radio button, and browse to select the batch manifest you just created.
11. You will receive an email to acknowledge that the batch submission has been received. You will receive another email to tell you whether the object was added successfully or not.

**Workflow: A Batch of Single Files**

1. Place the files on a web server.
2. Download the **merrittManifest.xls** file.
3. Create a new worksheet for your batch manifest.
4. Provide a URL and file name for each file in the batch. Provide any optional metadata (details in The Batch Manifest section).
5. Choose **View > Macros > View Macros > ThisWorkbook.BatchOfSingleFiles > Run**.
6. Name your batch manifest file. A file extension of .checkm will automatically be added to the file.

7. Login to Merritt. If you work with multiple collections, choose the collection this batch will be submitted to.
8. Choose **Add Object**.
9. Select the **Batch of files** radio button, and browse to select the batch manifest you just created.
10. You will receive an email to acknowledge that the batch submission has been received. You will receive another email to tell you whether the object was added successfully or not.

### Workflow: A Batch of Object Manifest Files

1. Place the object files on a web server.
2. Download the **merrittMacro.xls** file.
3. Create a new worksheet for each object manifest. There can be only one object per object manifest.
4. Provide a URL and file name for each file in the object. You may optionally provide checksum and file size information; see the **Object Manifest** section for details.
5. Choose **View > Macros > View Macros > ThisWorkbook.CreateObjectManifest > Run**.
6. Name your object manifest file. A file extension of .checkm will automatically be added to the file.
7. Post each resulting object manifest to a web server.
8. Create a new worksheet for your batch manifest.
9. Provide a URL and file name for each object manifest you created. Provide any optional metadata (details in The **Batch Manifest** section).
10. Choose **View > Macros > View Macros > ThisWorkbook.BatchOfManifestFiles > Run**.
11. Name your batch manifest file. A file extension of .checkm will automatically be added to the file.
12. Login to Merritt. If you work with multiple collections, choose the collection this batch will be submitted to.
13. Choose **Add Object**.
14. Select the **Batch of object manifests** radio button, and browse to select the batch manifest you just created.
15. You will receive an email to acknowledge that the batch submission has been received. You will receive another email when the submission has been processed. You will receive another email to tell you whether the object was added successfully or not.

### The Object Manifest

The object manifest contains a separate row for each file that is considered part of a single object. Each object worksheet or manifest should only include information about one object. Object components can include files of metadata pertaining to the object in any format (METS, marc etc.). The Object Manifest Specification for Merritt is available in a plain text file (to make columns and rows more clear).

The information you can provide in an object manifest is:

fileURL | hashAlgorithm | hashValue | fileSize | filename

Only **fileURL** and **fileName** are required.

The **hashAlgorithm** column specifies what kind of checksum you are providing, if you have a checksum value for a component file in the object. (Accepted checksum algorithms are: Adler-32, CRC-32, MD2, MD5,SHA-1, SHA-256, SHA-384, and SHA-512). If you provide a hashAlgorithm, you must also provide a **hashValue**, and vice-versa. If provided, Merritt will validate any checksum values provided for each file. If the value provided does not match that value that Merritt calculates, the object submission will fail. You will be notified by email that the object was not submitted because the object did not pass a fixity check.

The **fileSize** column is optional and is expressed in bytes.

There are no columns for object-level metadata such as title, creator etc. These can be supplied when you upload the manifest by filling out the form on the Add Object screen, or by also submitting a batch manifest.

**Workflow: The Batch Manifest**

The batch manifest contains a separate row for each object, and can contain only one row for any complex, multi-file object. Rows for complex object should point either to container (.zip or .tar) files or to object manifest files. The Batch Manifest Specification for Merritt is available in a plain text file (to make columns and rows more clear).

The information you can provide in a batch manifest is:

fileUrl | hashAlgorithm | hashValue | fileSize | fileName | primaryIdentifier | localIdentifier | creator | title | date

Only **fileURL** and **fileName** are required.

The **hashAlgorithm** and **hashValue** columns refer to the checksum of whatever file is referenced in the **fileUrl** column. If you provide URLs pointing to object manifest files, the hashValue would be the checksum of the object manifest itself. The hashAlgorithm column specifies what kind of checksum you are providing, if you have a checksum value for a component file in the object. (Accepted checksum algorithms are: Adler-32, CRC-32, MD2, MD5,SHA-1, SHA-256, SHA-384, and SHA-512). If you provide a hashAlgorithm, you must also provide a hashValue, and vice-versa. If provided, Merritt will validate any checksum values provided for each file. If the value provided does not match the value that Merritt calculates, the object submission will fail. You will be notified by email that the object did not pass a fixity check.

The **fileSize** column is optional and is expressed in bytes.

**primaryIdentifier** is the identifier that Merritt uses to track the object. If you are submitting new objects, you will very likely not have a primary identifier. Primary identifiers must be ARK format identifiers. You will generally only use this column if you are using a manifest to edit an existing object. You will be able to see the Merritt-supplied primary identifier for any object in Merritt when you display it.

**localIdentifier** is any identifier you already use to refer to the object. You can provide multiple local identifiers by separating them with a semicolon. The contents of this column will be searchable in Merritt. You will also be able to edit the object by referring to the local identifier in a manifest.

**creator** is the author or creator of the object itself. There are no format requirements for expressing named persons or entities. Merritt will display the creator exactly as entered. This column will be searchable in Merritt.

**title** is the title of the object itself. Merritt will display the title exactly as entered. This column will be searchable in Merritt.

**date** is the publication date of the object itself. If you provide the date in a standard excel format, it will be submitted to in Web UTC datetime format. If you enter a non-standard date format, the date will be submitted as plain text. Merritt will display the date exactly as entered. This column will be searchable in Merritt.

**Tips for Creating Text-based Manifests**

- You do not have to use Excel to create a manifest file; you can also use a text editor or create a script to write manifest files. You can use the sample batch and object manifests and simply edit the area for conveying rows of object information.
- The contents of the manifest are listed on separate lines, with each column delimited by " | " [space] [pipe] [space]. Empty columns are indicated by a space between column delineators: " | | ". (There may be one or two blank spaces in the column).
- The column heading text in the excel spreadsheet is slightly different than in the manifest files. The meaning and order of the columns is the same, but the headings in the text of a manifest file will begin with either "nfo:" or "mrt:" (example: nfo:fileUrl).
- Merritt manifests contain placeholders for two columns that do not appear in the spreadsheet described above: nfo:fileLastModified and mrt:mimetype. These columns are not yet implemented in Merritt, but they do need to be included in any manifest that you type by hand, with the columns left empty.
- Batch manifests must be identified as a batch of containers, batch of single files or a batch of object manifests. The profile line of the manifest identifies the type of batch:

  Batch of container files:

  #%profile | http://uc3.cdlib.org/registry/ingest/manifest/mrt-container-batch-manifest

  Batch of single files:

  #%profile | http://uc3.cdlib.org/registry/ingest/manifest/mrt-single-file-manifest

  Batch of object manifests:

  #%profile | http://uc3.cdlib.org/registry/ingest/manifest/mrt-batch-manifest

- When you've finished editing a sample manifest, be sure to save it with a unique name and a ".checkm" file extension.

- In any batch manifest, empty cells up to the **nfo:fileName** column must be identified with " | | ". After the nfo:fileName column, empty cells should be identified ONLY if they are followed by another cell that has a value. Examples:

**Special Considerations**

- If you are using the Excel spreadsheet, any text in bold will be ignored. Make sure that all of your content is in plain text.
- If you are using the Excel spreadsheet, and if you need to delete a row, use the [right click] [delete] approach. You need to delete the entire row, rather than just the text in the row cells.
- If you are using either the spreadsheet or a text editor to produce a manifest, you cannot use a pipe (vertical bar) character "|" in any of your fields. If you need a pipe character to appear anywhere in an object record, replace it with: %7C

**Optimizing the Macro: Create a Trusted Location (Optional for PC)**

If you plan to create lot of Merritt manifests, you may want to create a trusted location for the merrittManifest.xls file on your computer. This will allow you to run the macro without having to chose enable this content when you open the excel file. It will also enable you to run the macro from other excel files as long as you have the merrittManifest.xls file open. The following instructions are taken from Microsoft documentation for Office 2007 users (http://office.microsoft.com/en-us/excel-help/create-remove-or-change-a-trusted-location-for-your-files-HA010031999.aspx)

1. Open MS Excel.
2. Click the **Microsoft Office Button** , and then click **Excel Options**.
3. Click **Trust Center**, click **Trust Center Settings**, and then click **Trusted Locations**.
4. Click **Add new location**. Important: We recommended that you do not make your entire Documents or My Documents folder a trusted location; doing so increases your security risk. Create a subfolder within Documents or My Documents, and make only that folder a trusted location.
5. In the **Path** box, type the name of the folder that you want to use as a trusted location, or click **Browse** to locate the folder.
6. If you want to include subfolders as trusted locations, select the **Subfolders of this location are also trusted** check box.
7. In the **Description** box, type what you want to describe the purpose of the trusted location.
8. Click OK.
9. Store the merrittManifest.xls spreadsheet in this directory.
10. When you open the merrittManifest.xls file, the macros will be accessible to any other excel file you open in the trusted directory.

## Editing Objects in Merritt

If you have submitted an object to Merritt and need to edit either the object itself or the object metadata, you will need resubmit the revised object and metadata, and specify either the local identifier or the primary identifier.

- *Object metadata* here refers to the four fields Merritt provides on the **Add Object** screen: title, creator, date and local identifier. Changes made here will influence search results in Merritt.
- If you originally provided a local identifier for the object, you will be able to edit it by specifying that local identifier on the **Add Object** screen.
- If you did not supply a local identifier, or if you are trying to edit the local identifier itself, you will need to display the object, make a note of the primary identifier Merritt assigned to the object, and prepare a batch manifest to submit revisions to the ojbect (steps provided below).
- Merritt will create a new object version as a result of any edits you make, including any edits to object metadata.
- If you are updating only one file in a multi-file object, be sure to provide copies of all of the unchanged files and repeat any metadata you provided as well. If you include only the revised file, Merritt will consider that one file to constitute the entirety of the next object version.

**Edit with a Batch Manifest: Steps**

1. Make any changes needed to the object itself.
2. Post the object on a web server. If the object consists of more than one file, we recommend that you post it in a container (.zip or .tar) file. If you choose to post all files of a multi-part object individually, you will also need to create and post an object manifest.
3. Create a batch manifest for the object (even if there is only one object to edit). If the object consists of a single file, you will use the **batch of single files** workflow and format. If you posted the object to a server in a container file, you will use the **batch of container files** workflow and format. If you posted both a multi-part object and an object manifest file, you will use the **batch of object manifests** workflow and format. (See Using a manifest).
4. Include any changes to title, creator or date metadata in the batch manifest. If there are no changes, be sure to include the original correct values.
5. If you did not originally supply a local identifier or are trying to edit the local identifier, display the object in Merritt, make a note of the primary identifier and include it in the batch manifest.
6. If you originally provided a correct local identifier, you will not need to provide a primary identifier, but you will need to provide the correct local identifier again.
7. Go to the **Add Object** screen and add select the option for the type of batch manifest you created.

# Glossary

Batch:

Digital objects can be submitted into the Merritt DPR in groups, called a **batch**. Batch submissions require a manifest. For more information on manifests, see Using a manifest.

Collection:

The digital objects you submit will be part of one or more collections. Depending on your login permissions, the navigation bar will allow you to view and/or add objects to all of your collections.

Completed:

The Submission status is "completed" when all the objects have been processed for ingest. The submission status is independent of job status. Some objects in the submission may not have been successfully deposited in storage (with a job status of "failed") but the entire submission would still have a submission status of "completed."

Container:

Digital objects consisting of multiple files can be packaged into a single container file using tar, gzip or zip. Files submitted in these formats will be unpacked and the components stored (and returned) as separate uncompressed files.

Digital object:

Digital objects may be simple, such as a single JPEG or TIFF image file, or complex with multiple digital files. A digital book, for instance, may consist of a digital image file for each page along with a text file transcription. A digital object could be each file, a page (consisting of a set of image and text files) or the entire book, comprising hundreds or thousands of digital files. The entire digital object would be assigned a single identifier, and stored under that identifier in the Merritt Repository. Metadata about the object would describe the entire object. Decisions about the scope of any digital object are left to the depositor.

Description:

See **Electronic Resource Citation**.

Dublin Kernel metadata:

See Electronic Resource Citation.

Electronic Resource Citation (ERC):

Short, descriptive metadata about a digital object, also known as Dublin Kernel Metadata. (see spec). These metadata will be used to manage digital objects in the repository, and in reports about the collections, so the descriptions should be meaningful. See the Metadata Guide for more information.

Failed:

A submission will receive a status of **failed** if:

- A checksum was provided but could not be validated.
- The object was submitted via a manifest but no object was found at the URL provided in the manifest.

File:

Single files can be submitted to the repository through the user interface directly, or can be submitted in batches using manifests. For more information on the latter option, see the [Manifest Instructions].

ID:

Identifier. See: Submission ID, Job ID, Local ID, Primary ID

Ingest:

The process of submitting a digital object for storage into the Merritt DPR.

Job ID:

Each digital object that is submitted, whether submitted as a single object or part of a batch submission, receives a Job ID. The Job ID is returned in the email notification, and will track the digital object as it is deposited in the Merritt Repository.

Job Status:

Each object that is submitted, whether submitted as a single object or part of a batch submission, will receive a job status that will indicate successful or failed deposit into storage. There are four job statuses that will be reported:

- **Pending:** This status is assigned when the job is first deposited. The digital object is sent to a queue awaiting deposit into storage. While waiting in the queue, the job status will remain "pending."
- **Completed:** Once the object is successfully deposited in storage, the job status is changed to "completed."
- **Failed:** If any problem prevents the digital object from being deposited successfully into the storage, the job status will be "failed." There will be more information in the "status message." Please contact UC3 for more information if you encounter this status.
- **Deleted:** Digital objects will remain in the queue until they are successfully deposited into storage. When they are safely in storage, they will be deleted from the queue.

Kernel Metadata:

see Electronic Resource Citation (ERC).

Local ID:
Digital objects may have multiple identifiers. We will assign an ARK asprimary identifier when we accept a submission to the Merritt repository. Your digital objects may have other identifiers in your systems, and these "local IDs" can be included with the metadata accompanying the digital objects. You can continue to track digital objects using these "local IDs". See Primary ID.

Manifest:
A method for submitting multiple files or objects. See [Manifest Instructions] for more information.

My Profile:
see User Profile

Object Lookup:
This is a basic search feature for all digital objects in the Merritt repository. In addition to the Electronic Resource Citation metadata, this lookup will search the full-text (text, xml, pdf) of any object in the Merritt repository.

Object Manifest:
see Manifest

Primary ID:
We will assign an ARK as primary identifier for any digital object submitted to the Merritt repository. These identifiers are durable, unique and in wide use. We recommend that you update your records to include the primary identifiers for your digital objects. Digital objects will also be retrievable using Local IDs (see Local ID).

Queued:
This status is assigned when digital objects are first submitted. They are put in a queue before being sent to storage to allow for an orderly process.

Single object:
A single digital object may consist of one or more individual digital files. See Digital Object. Single objects may be submitted to the Merritt DPR via the user interface directly, or using manifests. See [Manifest Instructions] for more information.

Status:
See Job Status and Submission Status for more specific information.

Submission ID:
Every submission to Merritt is assigned an identifier called a Submission ID. This ID is returned in response to the initial submission.

Submission Status:
The Submission Status indicates the state of the entire submission, whether a single object or a batch submission.

Version:
Digital objects may change through corrections, improved digitization, or other reasons. These changed versions can be resubmitted to Merritt and stored along with other

versions of the same object. Changed metadata can also be resubmitted, and changing metadata would be considered a new version of the digital object.

## Sample Data

Sample data is available to use as you try out the service. In some cases samples such as manifest and erc files may be useful as examples for formatting your own Merritt files. All of the sample files below are available as a zip file from http://merritt.cdlib.org/samples/samples.zip

### Sample Files

| Sample type | File name |
|---|---|
| Single-file objects | • goldenDragon.jpg<br>• tumbleBug.jpg<br>• generalDrapery.jpg |
| Container file objects | • huskyChicken.zip (object has 3 files)<br>• outdoorStore.zip (object has 8 files)<br>• souvenirs.zip (object has 2 files)<br>• brain_v1.zip (object has 4 files)<br>• brain_v1.tar (object has 4 files)<br>• brain_v1.tar.gz (object has 4 files)<br>• brain_v2.zip (object has 5 files)<br>• brain_v2.tar (object has 5 files)<br>• brain_v2.tar.gz (object has 5 files)<br>• jazzbears.zip (object has 2 files) |
| Object manifests<br>(pointing to multi-file objects) | • 4blocks.checkm<br>• bigHunt.checkm<br>• call911.checkm |
| Batch manifests | • sampleBatchOfContainers.checkm<br>• sampleBatchOfFiles.checkm<br>• sampleBatchOfManifests.checkm |

### Using the Sample Files to Demonstrate Manifests

The http://merritt.cdlib.org/docs/merrittManifest.xls file that contains the macros needed to generate a manifest also contains sample worksheets for objects and each kind of manifest. The objects referenced in those worksheets are all included in the samples.zip file and are posted on Merritt for you to use. (If you run the macro to produce a manifest from the sample worksheets, the sample objects are already posted on a server to add).

## Using the Sample Files to Demonstrate ERC files

The **jazzbears.zip** file contains both a .tif image and a mrt-erc.txt file. The mrt-erc file contains the following metadata:

erc:
who: (:unkn) unknown
what: Jazz for the bears
when: 1920-1932
where: 2001697390

Submit this object as a container file to see that Merritt extracts the mrt-erc-txt metadata.

## Using the Sample Files to Demonstrate Versioning

The "brain_v1" and "brain_v2" sample files will demonstrate both object versioning and will show you how different kinds of container files can be used to submit content to Merritt. The .tar, .zip and .tar.gz file sets are independent of one another; they each include the same files:

- brain_398.tif
- brain_492.tif
- brain_508.tif
- brain_604.tif

v2 of the tar, gz and zip containers add another file: **brain.tif**

To demonstrate container formats, submit any of the files to see that they are unpacked upon ingest.

To test versioning, be sure to provide a **local identifier**, for version 1, then provide the same local identifier for version 2.