Building capacity in Digital Preservation: Theoretical Issues, Practical Applications

3. Transfer Utilities

Richard Dancy, Staff Archivist
Simon Fraser University Archives and Records Management Department
Study Leave 2018 Project
January 28, 2019

SUMMARY

Transfer utilities support and automate the transfer of digital materials from producers to repositories. This paper looks at the Archives' own custom-built transfer tool, SFU MoveIt, and compares it with two other widely used open-source applications, Exactly and Bagger. It briefly describes the BagIt specification that all three implement and compares features of the utilities. The report recommends that the Archives retain SFU MoveIt as its preferred packaging application, but make substantive changes to it to support the inclusion of transfer metadata in the transfer package itself; and that the Archives adopt Bagger as its in-house transfer validation tool.

TABLE OF CONTENTS

1. INTRODUCTION

Transfer utilities are tools that support and automate the transfer of digital materials from producers to repositories. This paper compares the Archives' own SFU MoveIt utility with Exactly and Bagger, two widely used open-source applications. All three implement the BagIt file packaging format and share the same broad purpose.

The goal of the comparison is to determine if SFU Archives should continue using SFU MoveIt in its current form, replace it, revise it, and / or supplement it with one of the others. A previous paper in this project series (*1. Transfer: Current Approach*) describes the Archives' current workflow and identifies a number of difficulties. The argument here is that some of these can be addressed through changes to the transfer utility.

- Section 2 provides a brief account of the BagIt specification that is implemented by each of the tools under review.
- Section 3 gives an overview of each utility.
- Section 4 offers commentary and assessment.
- Section 5 recommends that Archives should continue to use SFU MoveIt as its preferred packager, but make substantive changes to it; and that it should adopt Bagger as its preferred validation tool.
- Section 6 briefly notes how implementation of the recommendations would address some of the problems flagged earlier in the Study 1 paper.
- Section 7 brings together institution abbreviations and citations of works and websites referenced in the paper.
- The Appendix sets out detailed feature comparisons in a table format.

Note that another paper in this series (*4. Transfer Validation*) looks in more detail at the transfer validation process.

For terminology, see the comments in *Project Overview* report, section 4. I generally follow the OAIS terms, using **producer** for the entity that transfers materials and **archive** (singular) for the entity that receives them. **Institution** is used generically for GLAM organizations (galleries, libraries, museums, archives). **Archives** (capitalized, plural) is usually reserved for the name of a particular institution (SFU Archives, the Archives). **Digital materials** is used for the stuff transferred.

2. THE BAGIT SPECIFICATION

BagIt was developed in 2008 by the US Library of Congress and the California Digital Library (CDL), building on the former's experience with large transfer projects. The Internet Engineering Task-Force (IETF) maintains the specification, and the most recent version of the spec was released in September 2018.

BagIt is a standard for constructing containers ("bags") for the transfer and / or storage of digital files.

- It distinguishes between the **payload** (the objects being transferred / stored) and the **tag** (files recording information about payload objects or the bag itself).

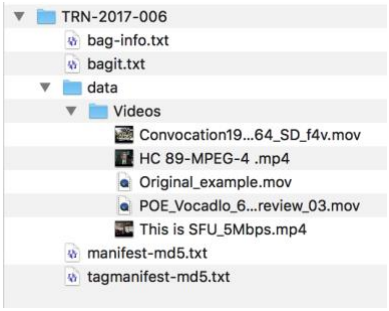
- It prescribes a directory structure for payload and tags within a top-level folder (**base directory**); a bag can have any name.
- The payload can be any arbitrary content in any file formats in any hierarchical organization; but it must be stored in a sub-directory called `/data` immediately below the base directory.
- Two tag files are mandatory and must be stored in the base directory:

  The **bag declaration** (`bagit.txt`) gives the BagIt version number followed and the character encoding of the tag files.

  The **bag manifest** lists each payload file with its checksum; the algorithm used to generate the checksums must be included in the name of the manifest file (`manifest-<algorithm_name>.txt`).
- Other tag files are optional, but two are typically included:

  The **bag metadata** file (`bag-info.txt`) describes the bag and its payload; there are 14 reserved metadata elements; all are optional; most (but not all) are repeatable; users may add any other custom metadata elements as they wish.

  The **tag manifest** (`tagmanifest-<algorithm_name>.txt`) lists each tag file with its checksum.
- Users may add any other custom tag files to the base directory or to any other sub-directory (except for the `/data` sub-directory reserved for the payload).



The main advantage of bags for transfer and storage is that the checksums included in the manifests provide a basis for verifying the integrity of the files over time. They are generated pre-transfer and checked post-transfer to determine whether any files were damaged in transit. For long-term storage, the checksums can be regularly checked to identify files that have suffered bit-rot or undergone some other unintended change.

3. THREE BAGIT UTILITIES

SFU MoveIt, Exactly and Bagger are utilities that serve broadly similar purposes, and all three follow the BagIt specification. Exactly and Bagger are both widely used in the field; interviews with Canadian institutions found a number actively using or experimenting with one or both (e.g. QNS, UTL, DAL, CWA, UWA).

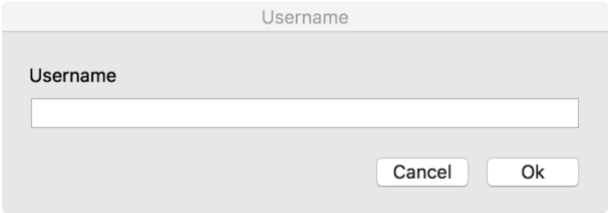
3.1 SFU MoveIt

SFU MoveIt is a free open-source desktop utility for creating transfer packages as BagIt-compliant bags, with certain additional information stored outside the bag. It does not transfer bags, nor does it validate bags post-transfer. These are separate tasks in the Archives' workflow.

SFU MoveIt itself was originally created in 2014 (as CreateBag) by Mark Jordan of SFU Library. The initial goal was an application that would both create a bag and transfer it over a network via FTP. Over the next several years the Library's Alex Garnett took on development, and the utility was refined for the Archives' requirements and rebranded as SFU MoveIt. Transfer functionality was dropped from the application in 2016 when the university launched its OwnCloud file transfer and storage service as SFU Vault.

SFU MoveIt runs a set of Python scripts using the official Python bagit module, with a graphical user interface in both Windows (using Qt) and Mac (using CocoaDialog). There is also a Linux version (using Gtk3), but it is not currently maintained. The latest releases for Windows (June 2017) and Mac (December 2017) are available on the Archives website; the code is on GitHub.

The Windows version is a stand-alone application launched by double-clicking the application icon and navigating via a browser interface to the target transfer folder. The Mac version works as a Mac OS service available by right-clicking the target folder within Finder. User interaction in both versions is managed through a series of dialog boxes.



To create a bag:

- Select the target folder for transfer.
- Enter data through a series of three dialog boxes: *User name*; *Transfer number* (unique ID supplied by the Archives); *Session number* (when there are multiple transfers under same transfer number).
- Click the OK button on the third dialog box to initiate bag creation.

SFU MoveIt creates the bag on the user's desktop. The bag name is based on the user-entered transfer and session numbers, and it contains two files:

- The **bag** – a zip file of a BagIt-compliant bag (using same name as the folder).
- The **meta file** – an SFU-specific `meta.txt` file containing transfer metadata.

The original target folder remains in-place and unchanged on the user's computer; timestamps (date created / modified) of the target files are preserved unchanged in the copies in the bag.

The SFU-specific `meta.txt` is intended both for computer processing and human viewing. It includes:

- **Transfer metadata** – user name, transfer and session numbers, date of packaging.
- A **checksum** of the zipped bag – subsequently used to validate the transfer.
- A **contents list** of all files in the bag (without individual checksums) – can be retained by the producer as a human-readable transfer file list.

Transfer must be done outside the app:

- The producer manually uploads the package to a designated transfer folder on SFU Vault.

Bag validation must be done outside the app:

- The archivist compares the checksum of the post-transfer zipped bag on SFU Vault with the value recorded in the `meta.txt` file.
- The Archives has developed a Mac Automator app (Checksummertime) to support drag-and-drop automated comparison.

Known issues:

- SFU MoveIt will fail in certain edge-cases, e.g. if the user tries to package an empty folder or a folder whose name contains special characters.
- Cancel buttons are not currently working on dialog boxes.
- There are problems uploading MoveIt transfer packages to the web version of SFU Vault (see section 4.5 below).

3.2 Exactly

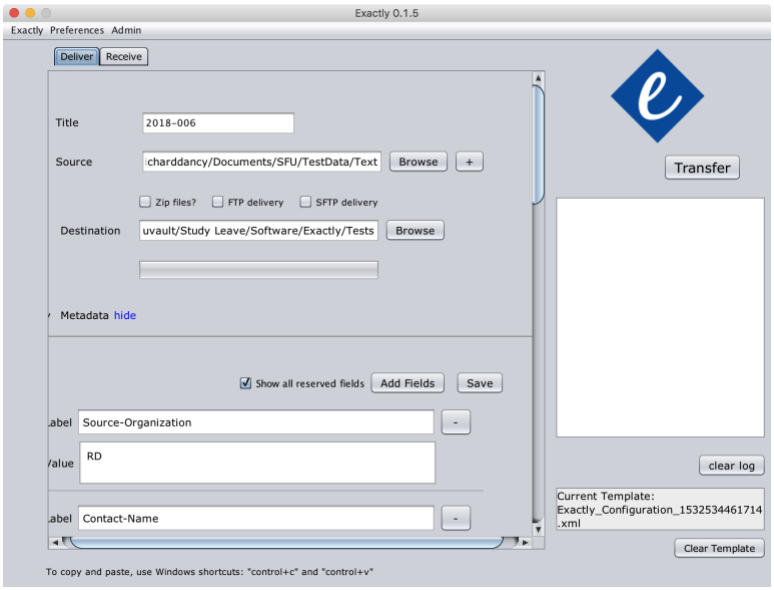
Exactly is a free open-source tool developed by AV Preserve for creating, transferring and validating BagIt-compliant bags. It is a Java-based desktop application, with Mac and Windows versions available for download from the AV Preserve website. The utility was first released in January 2016; the current version is 0.1.5 and was released in September 2017.

The application provides a single interface for all user interaction and data entry, with one tab for bag creation-delivery and another for receipt-validation. Results of all operations (including error messages) are displayed in the **logging quad**. Menus handle administrative settings (import / export of templates, email and FTP settings). Users can save, export and import templates that store admin settings and metadata fields for data entry. It is also possible to use Exactly without the transfer functionality, as simply a bag creator and validator.

To create and transfer a bag:

- Enter the recipient's FTP site settings through the Admin menu.
- Click the Deliver tab and select target folder(s) for transfer via a browser interface; multiple folders from different directories can be added to a single transfer.

- Specify where to store the bag; the original target folders remain in-place and unchanged in the user's system.
- Specify the method of delivery (FTP or SFTP) and can choose to zip files.
- Enter a bag *Title*; this will be the name of the highest-level folder of the bag.
- Enter metadata; all BagIt reserved metadata fields are supported; some are generated automatically (e.g. *Bagging-Date*); others are entered by the user; all user-enter metadata fields (except *Title*) are optional.
- Add custom metadata fields if needed (i.e. fields additional to the BagIt reserved fields) and can repeat any field.
- Click the Transfer button to initiate bag creation and delivery.



To validate a bag:

- Click the Receipt tab and select the bag via a browser interface.
- Click the Validate button; the results appear in the logging quad.
- The user can also "unpack" a bag, i.e. extract a copy of the payload and copy it to a specified destination; tag files are put into a separate `metadata` sub-directory.

Known issues / issues that arose during testing:

- Payload files in the bag do not preserve the timestamps (date created / modified) of the originals; timestamps are overwritten. This is a known issue (see the *Exactly User Guide*, p. 2), and a workaround has been developed (see section 4.3 below).
- When repeating BagIt reserved fields, the user must manually enter the field name; this could be a source of error.

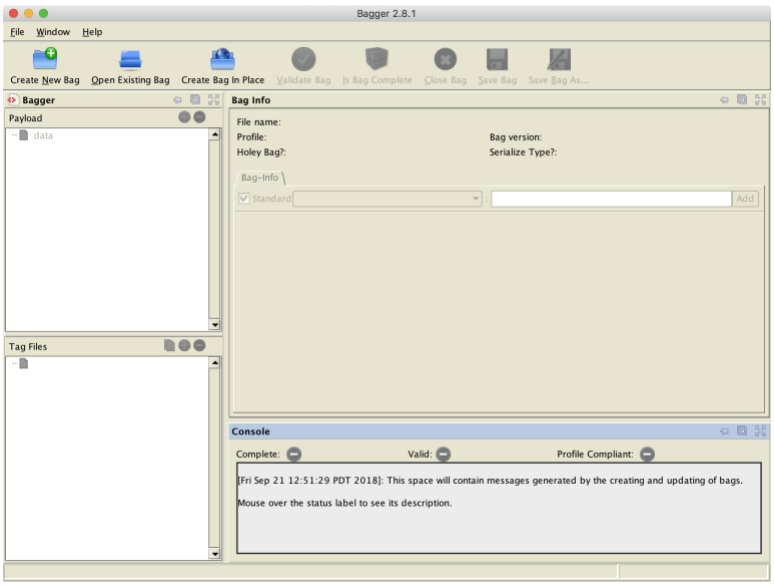
- During testing, Exactly did not unpack the bag as described in the *User Guide* (p. 12); rather than extracting the payload and creating a `metadata` sub-directory for tag files, Exactly simply copied the entire bag to the new destination.

3.3 Bagger

Bagger is a free open-source utility developed by the US Library of Congress as a graphical user interface for the BagIt specification to create and validate bags. Development began in 2008, with version 1.0 appearing in November 2010. The latest version (2.8.1) was released in April 2018 and is available on GitHub.

Bagger is a Java-based desktop application. It downloads as a single zip file that contains a top-level directory holding all the necessary sub-directories, scripts and configuration files; users can move and run it from any location on their computer. On running the app for the first time, it will create a `bagger` directory in the user's Home folder to store data entry templates.

Bagger provides a single interface for all user interaction and data entry. It has separate panes for viewing bag contents (payload and tags) and for entering and editing metadata. Results of all operations (including error messages) are displayed in the **console** pane. Action buttons along the top of the display allow the user to initiate various actions (create, open, close, validate, save bag). There are no admin settings to configure. Users can add / remove data entry profiles (sets of metadata fields with default values) as JSON files in the `bagger` directory in their Home folder.



To create a bag:

- Click the Create New Bag button and select a data-entry profile from the list.
- Click the Add button to add files or folders to the bag; multiple selections can be made from different path hierarchies.

- Enter metadata; all BagIt reserve fields are available for selection; fields can be added or removed; the user may add custom fields.
- Click Save As to initiate bag creation.
- On the pop-up dialog box enter a bag name and specify whether to zip files or not, whether to create manifests or not and which algorithm to use to generate checksums.

Bagger also provides options to:

- Create a "holey" bag, i.e. files are listed in the bag manifest but are not physically included in the bag; they must be fetched during the bag validation stage.
- Create a bag "in place," i.e. the target folders and files are transformed into a bag rather than copied.

Transfer must be done outside the app.

To validate a bag:

- Click the Open Existing Bag button and navigate to the bag via a browser interface.
- There are separate buttons for checking the completeness of a bag (i.e. it has all the necessary elements according to the BagIt specification) and for validating it (i.e. checking the checksum).
- Bagger displays validation results in the console pane; it will also validate the bag against the profile requirements (e.g. required fields or field order as specified in the JSON profile).

Known issues / issues that arose during testing:

- When launching Bagger on my laptop (2017 MacBook Pro running High Sierra OS 10.13.6), Terminal throws a warning message: "An illegal reflective access operation has occurred." The problem appears to be a conflict between Gradle (Bagger's build system) and Java 10 (see discussion here). It does not appear to affect the program's operation and will presumably be resolved at some point.
- Bagger lists all tag files included in a bag and provides a viewer to display their content. This works for standard BagIt tags, but does not appear to work with custom tag files; the user can see that the files exists, but cannot view their content in-app.

4. COMPARISON AND ASSESSMENT

For detailed feature comparisons in table format, see the Appendix.

4.1 Scope

Exactly has the broadest scope, supporting bag creation, transfer and validation. SFU MoveIt has the narrowest, covering bag creation only. Bagger handles creation and validation, but not transfer.

For SFU Archives, the transfer functionality of Exactly is not an major advantage, because the Archives has settled on SFU Vault (OwnCloud) rather than FTP as its main transfer platform (see the comments below on transfer).

On the other hand, the ability of both Exactly and Bagger to validate bags is relevant, as the lack of proper bag validation was noted in paper 1 (*Transfer: Current Approach*) as a problem the Archives should resolve.

4.2 User experience

All of the applications are easy to download and install (though Bagger throws the warning message noted above). Exactly requires some configuration, but only when using it to transfer bags – then the user must enter FTP site settings (for the destination) and email server settings (for email notifications). Neither SFU MoveIt nor Bagger requires admin configuration.

Each of the applications comes with a *User Guide*, and producers using any of them to package their own materials for transfer will likely need some initial training.

SFU MoveIt is the most lightweight with the simplest interface. It requires user to only enter three pieces of information (user name, transfer number, and session number). The difference between transfer and session number may not be immediately self-evident, but the user is neither exposed to nor required to understand the BagIt specification itself. This simplicity is one of the app's strengths. On the other hand, the need to click through multiple dialog boxes may be off-putting for some users, and the final dialog box offers users a chance to review the data they entered but not to correct it. Fatal errors (e.g. attempting to bag an empty folder) cause the program to fail, but error messages are not parsed in user-friendly language, and the user needs to start the operation again.

Exactly and Bagger offer more complex interfaces. In both, all user interaction takes place through a single window which is clean and generally intuitive. They display the results of all operations in a logging pane, errors are generally explained and users can correct and continue without starting from scratch. Both applications do however, expose and require users to understand something of the BagIt specification; for example, metadata field names use the BagIt titles whose meaning will not be self-evident to many producers.

Exactly and Bagger are designed as generic applications that can be used by any institution in any workflow, and that flexibility is a strength. But in the context of SFU Archives' workflows, they may contain more than is needed for purposes of producer packaging.

4.3 Bag creation

Exactly and Bagger are more flexible than SFU MoveIt in terms of how users select bag content. Users of Exactly and Bagger can select folders or individual files and they can select multiple folders / files from different directory hierarchies on their systems. In SFU MoveIt, the user must select a folder (not a file) and can only select one folder (rather than make multiple selections). To transfer multiple folders from different hierarchies on their systems, users of SFU MoveIt must either first move all the folders into a single directory; or process each as separate sessions under a single transfer number.

Both Exactly and Bagger allow users to specify where on their computer to create the local copy of the bag; in SFU MoveIt it is always created to the user's desktop. With each utility, the bag contains a copy of the target files, the originals remaining unchanged in their original locations. In both SFU MoveIt and Bagger the bag payload files

preserve the timestamps (date created / modified) of the original target files; but in Exactly these are lost, overwritten with the current timestamp of the bag creation event. Exactly addresses this problem by creating a custom tag file (`FileSystemData.txt`) that captures for each item certain file system information, including the original timestamps; but the files themselves carry new timestamps.

Of the three, Bagger alone offers the option of creating a bag "in place" – this transforms the originals into a bag rather than just copying them to a new one. Bagger is also the only utility to support BagIt's "holey" bag, where payload files are listed in the manifest but not actually included in the `/data` directory; rather a standard `fetch.txt` file specifies the url location from which the files must be retrieved on bag validation.

Bag names are set by users in each application. In SFU MoveIt bag names derive from the transfer and session numbers the user entered. Exactly users enter the bag name in a field, while Bagger users specify a name on first saving the bag. Bagger alone allows special characters (e.g. `#@.`) in bag names; SFU MoveIt will fail if the user enters special characters in the dialog boxes, while Exactly gives an error (but does not explain what the error is).

The three utilities were tested for bag creation against a common set of sample folders. The largest folder size tested was a set of video files totaling 3.23 GB. The three were comparable in terms of speed on my MacBook Pro 2017 laptop (running High Sierra 10.13.6). Bagger was perhaps slightly faster, but both it and Exactly handled the largest folder in around a minute; SFU MoveIt was slightly slower, packaging the same materials in just under 3 minutes. For the typical transfers SFU Archives is likely to receive, all three applications work at acceptable speeds.

The bags themselves created by each application successfully validated in both Exactly and Bagger.

In general, the greater flexibility of Exactly and Bagger offer certain advantages over SFU MoveIt, especially with the ability to add multiple selections to the bag. But none seem decisive in the context of SFU Archives' transfer workflows, and it is hard to think of any immediate SFU use cases for Bagger's ability to create bags in place or holey bags. The loss of the original timestamps in Exactly, on the other hand, is a major drawback.

4.4 Metadata

Exactly and Bagger provide full support for all 14 BagIt reserved metadata fields. Both programs automatically capture bag creation and size information (*Bagging-Date*, *Bag-Size, Payload-Oxum*) in the bag metadata tag file (`bag-info.txt`). Users can repeat BagIt fields, though this is easier in Bagger than Exactly. In Bagger one simply selects the field from the drop-down list and clicks the Add button as needed, thus controlling the number of repetitions and the order of fields. In Exactly, the user clicks a button to display all reserved fields, but then must remove the ones not being used and manually add and enter the field titles of elements being repeated. In both applications the user can add an unlimited number of custom fields. All metadata fields are included in the `bag-info` tag file.

Both applications also support the use of data entry templates. In Exactly the user exports the current configuration (admin settings + metadata fields + field values) as an xml file, which can then be imported by another user. Templates are not saved across sessions. But on starting Exactly, the application loads the set of settings / fields / values that were last used. If the user changed these during the last session, the template must be imported fresh.

Bagger handles templates quite differently, storing **profiles** as modifiable JSON files in the `bagger` folder in the user's Home directory (created on first running the utility). Each profile lists a set of fields and their properties. The sort order of fields can be specified; individual fields can be flagged as mandatory ("required"); any field can be assigned a required or default value; and a field can be associated with a value list that will appear as a drop-down menu in the app. Bagger comes pre-loaded with a number of profiles (e.g. the accession forms used by the Indiana State Archives and the State Archives of North Carolina); these can be removed by deleting the associated JSON files and new ones simply added. On creating a new bag, the user is required to select a profile from the list; select "no profile" to start from scratch.

By comparison with the others, SFU MoveIt's metadata support is extremely sparse. Two reserved BagIt fields are captured automatically for the `bag-info` file: *Bagging-Date* and *Payload-Oxum*. Users enter only three pieces of information; these are not mapped to BagIt reserved metadata fields, and they are not included in the `bag-info` file. SFU MoveIt always zips the bag, and it creates a custom file outside of the bag itself (`<bag_name>-meta.txt` file). This `meta` file is used in the Archives' workflow to validate the transfer checksum (of the zipped bag as a whole), and it also provides producers with a human-readable file transfer list.

In itself, the thinness of SFU MoveIt's metadata is not a problem. In the Archives' transfer workflow (see paper 1 in this series), the producer has already previously entered 17 pieces of information into the online Transfer Request form; repeating this in SFU MoveIt would be redundant. On the other hand, the Archives now captures this information in a database but not in the transfer package itself or in the eventual AIP. As noted in paper 1, this is a gap in the Archives' current practices. Most if not all of the data elements from the Transfer Request form could be mapped to BagIt reserved fields. By moving producer data entry to the packaging utility itself, SFU Archives would ensure this data is included in the transfer package and AIP. It would also eliminate the need for the online webform, resolving the problem (noted in paper 1) of indefinite retention of transfer data; in addition it would simplify the transfer procedure.

With respect to metadata, then, there is a strong case for either revising SFU MoveIt or replacing it by Exactly or Bagger. Of the latter two, Bagger's handling of metadata seems to me stronger. It is easier to add, repeat and order fields in Bagger; Bagger alone supports drop-down lists for controlled data entry; and Bagger can store multiple (and better) templates in the app without needing to reload them at the start of a new session.

4.5 Transfer

Of the three utilities, only Exactly is able to transfer bags over a network, doing this via FTP or SFTP. I did not test this functionality. In interviews, one institution did report problems using Exactly to process large video files (the operation did not complete over many days, no error message). In my own testing (as noted above), Exactly had no problem packaging large files quickly; the problem reported here likely relates to transfer failure. Exactly's own documentation notes that it "intermittently fails to transfer files into a DropBox managed folder" as a known issue (*Exactly User Guide*, p. 2).

Transfer functionality was originally envisioned for SFU MoveIt, but dropped when the university launched SFU Vault, a local implementation of OwnCloud software. The Archives prefers SFU Vault as its primary transfer platform for several reasons:

- SFU Vault can handle large file sizes at high speeds.
- It is less susceptible to network interruptions and transfer failures.
- If interruptions do occur, transfers can simply be resumed rather requiring a full restart.
- SFU Vault provides an easy drag-and-drop interface that will be familiar to most producers; they do not need to worry about installing and configuring FTP clients.
- The Archives can easily set up SFU Vault folder shares and links for producers inside and outside the university.

Given this, the main question for SFU Archives is how to integrate a purely packaging utility for producers with SFU Vault. As such, Exactly's transfer capabilities would not represent a significant advantage for the Archives.

As noted in paper 1 in this report series (*Transfer: Current Approach*), there is a limitation on transfer via the web version of SFU Vault. With the desktop version, the user can upload folders with sub-directories and files; but the online version can handle upload of single files only. SFU MoveIt generates the transfer package as a folder containing two files, and so it cannot be uploaded directly via the web interface.

This is not an issue for internal SFU transfers. Installation of the desktop version of OwnCloud is a simple and supported process with SFU Vault. But non-SFU donors who have not installed OwnCloud must rely on the web interface to access the shared transfer folder. As such they will have to upload the two MoveIt files separately, making the transfer process more cumbersome. To date, all SFU MoveIt transfers have been from internal SFU departments.

Addressing this issue and providing a common transfer experience for internal and external transfers should be priority.

4.6 Bag validation

Exactly and Bagger are designed to validate bags as well as package them. They both check the structure of the bag to verify its compliance with the BagIt specification, and they generate checksums for the payload and tag files to compare these against the checksums recorded in the manifests. Bagger in addition validates a bag against its profile requirements in terms of expected or required fields, data and sort order (assuming the validator instance of Bagger recognizes the profile).

In testing, both applications validate packages quickly (less than a minute for a large-ish 3.2 GB bag). Both caught any errors introduced for testing purposes (checksums changed or deleted; required tag files deleted; new files added to the payload directory). Both clearly flag the fail result, describe the nature of the error, and list the problematic files in their respective log panes.

SFU MoveIt does not validate bags. The Archives' current workflow requires the archivist to compare the checksum of the zipped bag against the value stored in the custom `meta.txt` file. The Archives has developed a tool (Checksummertime) to automate the through drop-and-drag. An archivist drags the `meta.txt` files onto the app,

it generates a checksum of the zipped bag, compares it against the value in the `meta.txt` file, and returns the test results (match or does not match).

This works fine as is, but it is not a full bag validation, as it checks neither the structure of the bag nor the individual files' checksums. Where failures occur, there is no way to pinpoint the problem file(s). It is also a Mac-specific tool that runs only on the Mac OS. Given that the Archives has a Linux workstation (mainly used right now for digitization), there may be some advantage in having an OS-independent utility for bag validation.

In sum, using either Exactly or Bagger as a bag validator would represent an upgrade on current practice.

4.7 Bag edit and revision

Of the three utilities only Bagger allows a user to edit a bag after it has been created. Users can open an existing bag and add or remove payload files and tag files, and add or revise metadata, then resave the bag.

The Archives' transfer procedure provides good use cases for this functionality. The Archives could use Bagger to add validation information (e.g. date the transfer was validated, by whom) and accession data (accession number, other descriptive elements from the accession record). This would ensure that post-transfer metadata created by the Archives could be added to the transfer package prior to ingest to Archivematica and in this way be included in the eventual AIP. This would address a gap in the Archives' current practices that was noted in paper 1.

4.8 Reporting

One advantage of including richer metadata in the bag itself is that it makes the transfer package available to other reporting tools. Exactly includes a nice feature that creates the `bag-info` metadata file in multiple file formats (text, csv, xml) to facilitate import of the data into other programs. Mark Jordan of SFU Library has developed a proof-of-concept tool (BagIt Indexer) for indexing and searching multiple bags across multiple storage locations.

Currently the Archives relies on its own custom FileMaker AIS database to manage transfer and accessions metadata and track transfers, backlog and processed AIPs. Getting more of this metadata into the transfer package in a standardized way via the packaging utility opens the possibility of eventually moving to other reporting and management tools as they are developed and mature.

5. RECOMMENDATIONS

The strength of SFU MoveIt is its simplicity. It lowers barriers and allows producers to easily transfer their materials to SFU Archives. It minimizes what the producer needs to know and includes only what is required for the specific purposes of the Archives' workflow. On the other hand, that workflow has several gaps (identified in paper 1) that could be addressed by improving the transfer utility.

It would be possible to simply swap out SFU MoveIt for Exactly or Bagger. Of the two, Bagger better suits the Archives' needs. The main advantage of Exactly (its ability to transfer bags via FTP) does match one of the Archives'

use cases (private donors who are not running a local installation of OwnCloud). But it should be possible to resolve the current problem with SFU Vault in other ways (see Recommendation 3 below).

Bagger in my opinion is both a better packager and validator. As a packager, it does not overwrite timestamps and it manages metadata better (easier to add, repeat and order fields, supports drop-down lists, more powerful templates). As a validator, it can check not only the structure of the bag and the checksums, but also the metadata against an expected profile; and its ability to edit an existing bag supports the inclusion of post-transfer accessioning data in the bag.

The downside of adopting Bagger is that it would present a greater learning curve for producers. It exposes them to BagIt terminology which would need to be explained. it includes more than is needed for the Archives' transfer procedures and these additional features would also need to be explained. The flexibility that is an advantage for a generic software tool (allowing different institutions to adapt the tool to their own processes) becomes a disadvantage in the hands of the producer. The Archives wants producers to package their materials in a certain standardized way; with Bagger they are free to depart from that as they see fit.

A "best of both worlds" solution is to retain SFU MoveIt as the primary packaging tool for producers and use Bagger in-house for validation and internal purposes.

The goal of the re-design would be to keep SFU MoveIt focused solely on what is needed for the Archives' transfer procedure, but revise it to support the creation of richer metadata in the bag; and change the way it outputs the transfer package to support users of the web-based version of SFU Vault.

At the same time, adopting Bagger as its in-house validation tool would provide the Archives with an OS-independent utility to perform full bag validation and allow archivists to add post-transfer accession data to the bag prior to Archivematica ingest.

Finally, the Archives could also use Bagger to package the output of its own digitization projects, taking advantage of the utility's support for customized templates to create one or more standardized digitization profiles. While this is not really a transfer issue (and so outside the scope of the present paper), the Archives currently manages digitization metadata in a largely ad hoc way. Using Bagger to package digitized output would enable the Archives to move to a more systematic approach and ensure that the metadata accompanies the objects in the AIP.

Recommendation 1: Data entry

Discontinue using the online Transfer Request form and move all producer data entry into SFU MoveIt.

- Incorporate the current data elements on the Transfer Request form into SFU MoveIt as the single point of producer data entry.
- Review the data elements on the Transfer Request form and map them to BagIt fields.
- Wherever possible try to use a BagIt reserved field and minimize the number of custom fields.
- Eliminating the Transfer Request form simplifies the process, but it does remove one of the validation points (accept or reject the request) and the Archives would need to re-think the workflow.

Recommendation 2: Interface

Build an interface for SFU MoveIt that includes all data entry fields in a single window.

- There will be too many data elements to manage user interaction via a series of dialog boxes.
- Use SFU-specific terminology in field labels, but map these to reserved BagIt fields in the `bag-info` file.
- Beyond the data entry fields, the interface should need only an Add button to allow users to select the payload folders / files and a Create button to initiate bag creation.
- Ideal (but less critical) would be the ability to select multiple folders from different file paths in the directory hierarchy.

Recommendation 3: Output

Design SFU MoveIt to output the transfer package as a single zipped bag.

- This would support drag-and-drop to SFU Vault via the web interface that many private donors will have to use.
- The use of Bagger as the validation tool (recommendation 4) would eliminate the need for the extra SFU-specific `meta.txt` file currently required for verifying checksums.

Recommendation 4: Validation

Adopt Bagger as the Archives' transfer validation tool.

- Archives' staff should use Bagger to validate transfers received and to add accession metadata to the bag prior to ingest to Archivematica.
- The Archives should review the data elements in the current Accession record and determine which would be appropriate for inclusion in the bag.
- On that basis, the Archives could create an **SFU Archives Accession** Bagger profile.
- In designing the workflow, the Archives should minimize redundant data entry between the Bagger profile and the AIS Accession record; scripts may be developed to move common data between the utilities.

Recommendation 5: Digitization packages

Integrate Bagger into the workflow for digitization projects to package output as bags prior to ingest to Archivematica.

- The Archives should identify metadata relating to the digitization process that should be regularly captured.
- On that basis, the Archives could create standard digitization Bagger profile templates; there may be a need for different templates for different media formats (e.g. sound recordings vs video vs film).

6. IMPLEMENTATION

The recommendations outlined in section 5 provide a basis to resolve several of the areas of concern and requirements flagged in paper 1 in this report series (*Transfer: Current Approach*). Taken together they would allow the Archives to:

- Eliminate the online Transfer Request form so there is no indefinite retention of transfer data in Adobe Experience Manager (requirement 2).
- Capture both transfer metadata and accession metadata in the transfer package (requirements 1 and 6).
- Simplify the transfer process via the SFU Vault web interface for non-SFU producers that do not have access to a desktop version of OwnCloud (requirement 3).
- Ensure a full BagIt validation of the transfer package's structure, checksums, and metadata profile (requirement 4).

In addition, they provide a better basis for managing digitization metadata.

** **Note** (January 2019): The Archives will revise SFU MoveIt in the spring / summer 2019 to ensure that the code can run on the latest release of Mac OS. The recommendations in this report will be reviewed as part of that project.

7. REFERENCES

7.1 Institution abbreviations

DAL Dalhousie University Archives
CWA City of Winnipeg Archives
QNS Queens University Archives
UTL University of Toronto Libraries
UWA University of Winnipeg Archives

7.2 Websites

Simon Fraser University Archives and Records Management Department. 2017. "Digital Preservation at SFU." Last modified April 7, 2017. http://www.sfu.ca/archives/digital-repository.html (accessed September 26, 2018).

7.3 Software

Archivematica. Artefactual Systems Inc. https://www.archivematica.org/en/ (accessed September 26, 2018).

AtoM (Access-to-Memory). Artefactual Systems Inc. https://www.accesstomemory.org/en/ (accessed September 26, 2018).

Bagger. LibraryofCongress GitHub repository. Last commit April 30, 2018. https://github.com/LibraryOfCongress/bagger

BagIt Indexer. mjordan GitHub repository. Last commit December 4, 2017. https://github.com/mjordan/bagit_indexer.

Checksummertime. SFU Archives and Records Management Department. Available on request.

Exactly. AV Preserve. Last accessed September 26, 2018. https://www.weareavp.com/products/exactly/.

OwnCloud. OwnCloud Foundation. Last accessed September 26, 2018. https://owncloud.org.

SFU MoveIt. SFU Archives and Records Management Department. Last modified December 7, 2017. http://www.sfu.ca/archives/digital-repository/sfu-moveit.html. Code on axfelix GitHub repository. Last commit December 14, 2017. https://github.com/axfelix/moveit.

7.4 Implementations

SFU AtoM. Last accessed September 26, 2018. https://atom.archives.sfu.ca.

SFU Vault. Last accessed September 26, 2018. https://www.sfu.ca/itservices/collaboration/sfu-vault.html.

7.5 Documents

AV Preserve. 2017. *Exactly User Guide: Version 0.1.5*. New York: AV Preserve, September 20, 2017. https://www.weareavp.com/wp-content/uploads/2018/06/Exactly-User-Guide_v.0.1.5.pdf (accessed September 26, 2018).

Kunze, J. et al. 2018. *The BagIt File Packaging Format (V1.0).* Internet Engineering Task-Force (IETF), Internet draft (v17), September 17, 2018. https://tools.ietf.org/html/draft-kunze-bagit-17 (accessed September 26, 2018).

Library of Congress. 2008. "Library Develops Specification for Transferring Digital Content." *Library of Congress News Archive,* June 2, 2008. http://www.digitalpreservation.gov//news/2008/20080602news_article_bagit.html (accessed September 26, 2018).

Library of Congress. [2018]. *Bagger User Guide: Version 2.1.3, Supporting BagIt Version 4.4.* Washington: Library of Congress, n.d. Ships with application in the `/doc` directory.

Simon Fraser University Archives and Records Management Department. 2017. *SFU MoveIt User Guide v2*. Burnaby: Simon Fraser University, August 18, 2017.

http://www.sfu.ca/content/dam/sfu/archives/DigitalPreservation/MoveItUserGuide.pdf (accessed September 26, 2018).

APPENDIX: FEATURE COMPARISON

The following table provide a detailed feature comparison between SFU MoveIt, Exactly and Bagger. For analysis, see section 4 above. Points that significantly differentiate one utility from the others (for good or bad) are highlighted.

	SFU MoveIt	Exactly	Bagger
Scope			
Bag creation	Yes	Yes	Yes
Bag transfer	No	**Yes**	No
Bag validation	**No**	Yes	Yes
User experience			
Installation	Download and install separate files for Windows (`exe`) and Mac OS (`dmg`)	Download and install separate files for Windows (`exe`) and Mac OS (`dmg`)	Download and install single zip file; unzip, move top-level folder to any location on computer
Check for updates	No	No	No
Admin settings	None	Email notifications; import / export templates; recipient FTP settings	None; but metadata templates can be created outside the app and stored in the appropriate directory
Uninstall	Uninstaller available for Mac version; uninstall Windows version by simply deleting the .exe file	No uninstaller available to delete associated config files	Delete the top-level folder and sub-directories; no other associated files

User interface	3 separate pop-up dialog boxes for data entry; 1 box = 1 piece of metadata; 4th dialog box launches bag creation Known issue: cancel buttons not working properly on dialog boxes	Single window for all user interaction Separate tabs for bag creation-delivery and receipt-validation Operation results displayed in "logging quad" Menus for admin settings	Single window for user all interaction Separate panes for viewing bag content (payload and tag files) Separate pane for entering / editing metadata Operation results displayed in "console" pane
Error handling	Fatal errors result in fail; error messages displayed but not parsed in user-friendly language; user must start again	Error messages display in logging quad, allowing user to correct and continue	Error messages display in console, allowing user to correct and continue
Email notifications	No	**Yes; can be configured to email transfer and error notices**	No
Learning curve	User needs to understand difference between Transfer and Session numbers	User exposed to/needs to understand BagIt terminology (field labels)	User exposed to/needs to understand BagIt terminology (field labels)
Progress indicators	None in app; Mac menu bar shows script running (spinning gear)	Logging quad updates results; progress bar for bag creation/transfer and validation	Console updates results; progress bar for bag creation and validation
Bag creation			
Objects that can packaged	Must be a folder; cannot select individual files	Folder or individual file	Folder or individual file
Selection of target objects	Navigate to and select folder through a directory browser (Windows version); right-click target folder in Finder (Mac version)	Navigate to and select folder through the app's directory browser	Navigate to and select folder through the app's directory browser

Allow multiple selections?	No; user can only select one folder, but folder can have sub-directories For multiple folders from different hierarchies, the user must either first move all folders into a single directory; or process the transfer in multiple sessions	Yes; no limit to number of selections; folders and files can be added from different path hierarchies	Yes; no limit to number of selections; folders and files can be added from different path hierarchies
Edge case: user selects an empty folder	Fails: no validation	Validation check: error message in logging window, but no explanation of error	Validation check; error message in pop-up window, but no explanation of error
Edge case: folder name contains special characters (e.g. `#@.`)	Fails: no sanitization	Creates bag with special characters in name	Creates bag with special characters in name
Bag location	Bag created to desktop; originals left unchanged in-place	User specifies where to create local copy of bag; originals left unchanged in-place; copy of the bag can be transferred to designated recipient	User can create a bag in place (transform originals into a bag) or create a local copy to any location, leaving the originals unchanged in their original locations
Bag name	Generated from user-entered transfer and session numbers	Supplied by user in designated field	Supplied by user on saving the bag
Timestamps (date created / modified) of originals	Preserved	**Overwritten**	Preserved
Testing: speed (target folder size = 3.23 GB)	Less than 3 minutes.	Less than 1 minute	Less than 1 minute.

Metadata			
Bag-it reserved metadata fields	**Captures 2 of 14 BagIt reserved fields in `bag-info` file; user-enter metadata not mapped to BagIt fields or captured in bag**	Supports all BagIt reserved fields; allows addition of custom fields; all metadata recorded in `bag-info` file	Supports all BagIt reserved fields; allows addition of custom fields; all metadata recorded in `bag-info` file
Custom metadata fields	No	Yes; add a new field, enter field label and value	Yes; uncheck "Standard field" box, enter field label and value, click Add
Repeatable fields	No	Yes; but the second time a standard field is added the user must manually enter the field label	Yes; select field from the drop-down list and click Add as often as needed
Drop-down lists for data entry	No	No	Yes; controlled terms in value lists managed in the JSON profile
Sort field order	No	No	Yes; order determined by sequence in which fields added or by profile But on opening an existing bag, field order may be different than as originally entered
Support customized data entry templates	No	Yes; particular configurations (admin settings + metadata fields) can be saved, exported, imported	Yes; profiles stored as JSON files in the bagger folder in the user's Home directory. Profiles can specify field names, default data, required data, associated value list, sort order
Support standard BagIt reserved tag files	Yes; automatically creates bag declaration, bag and tag manifests, bag metadata files	Yes; automatically creates bag declaration, bag and tag manifests, bag metadata files	Yes; automatically creates bag declaration, bag and tag manifests, bag metadata files

Allow custom tag files	None created in the bag itself; but `meta.txt` file created outside bag for subsequent bag validation purposes	Yes, but only those created automatically by the application. Bag metadata included in multiple file formats (txt, csv, xml) `FileSystemData` file captures original timestamps (overwritten on the bag objects) `TransferComplete` file summarizes operation User cannot add other customized tags	None create automatically, but user can add any custom tag through a browse and select interface
Edge case: bag name duplicates an existing file path on user's system	Fails; operation cancelled with unparsed error messages	Validation check: error message shown in logging quad; user required to change title	Not allowed; saves the bag in the existing folder with the same name. But existing bags can be edited and saved; will overwrite existing bag with new data
Edge case: user enter no data in fields	Creates folder named –	Validation check: user required to supply *Title* (= bag name); all other fields are optional	Allowed; all fields optional
Edge case: user enters special characters (e.g. `#@.`) into fields	Fails; operation cancelled with unparsed error messages	Allowed	Allowed
Test (validate) bags created by one utility in another	SFU MoveIt bags successfully validated by Exactly and Bagger	Exactly bag successfully validated by Bagger.	Bagger bag successfully validated by Exactly
Transfer			
Transfer bags over a network	No	**Yes: via FTP or SFTP**	No

Specify recipient address	n/a	**Yes; add recipient FTP settings via admin interface**	n/a
Email notice of completion or errors	n/a	**Yes; email notices can be configured**	n/a
Testing: performance	n/a	Did not test	n/a
Bag validation			
Validate zipped bags	No	No; bag must first be manually unzipped before validation	No; bag must first be manually unzipped before validation
Validate bag compliance (i.e. structure of the bag)	No	Yes	Yes
Validate bag checksums	No; but use `meta.txt` file to validate checksum of zipped bag through a separate process	Yes	Yes
Validate template / profile compliance	No	No	**Yes**
Support in-tool inspection of bag contents	No	No; "unpack" bag to copy to a new location to inspect contents outside of the app	User can view payload file names, tag file names, and the content of standard tag files; content of payload files must be viewed outside of the app
Testing: speed (target bag size = 3.23 GB)	n/a	Fast (< 10 seconds)	Fast (< 10 seconds)
Testing: alter checksums in manifest to trigger error	n/a	Validation fails: bag location highlighted in red; nature of error explained in logging quad; files with failed checksums identified	Validation fails; fail result flagged in pop-up warning window; files with failed checksums listed in pop-up window and console pane

Testing: delete checksums in manifest to trigger error	n/a	Validation fails: bag location highlighted in red; nature of error explained in logging quad; file with missing checksums identified	Validation fails; fail result flagged in pop-up warning window; files with missing checksums listed in pop-up window and console pane
Testing: delete payload manifest file to trigger error	n/a	Validation fails: bag location highlighted in red; nature of error explained in logging quad; missing payload files identified	Validation fails; fail result flagged in pop-up warning window; payload files not listed in manifest listed in pop-up window and console pane
Testing: add tag file to /data directory reserved for payload to trigger error	n/a	Validation fails: bag location highlighted in red; nature of error explained in logging quad; files included in payload missing from manifest identified	Validation fails; fail result flagged in pop-up warning window; file(s) included in payload missing from manifest listed in pop-up window and console pane
Bag edit / revision			
Add / delete payload files	No	No	**Yes; bag manifest(s) revised**
Add / delete tag files	No	No	**Yes; tag manifest(s) revised**
Add / delete / edit metadata	No	No	**Yes; bag-info file revised**
Reporting			
Shows process start / end times?	Timestamp for completion only in `meta` file	Timestamp for completion only in `TransferComplete` file	No
Include custom tag files for reporting?	No	**Yes; bag metadata file created in csv and xml to facilitate import into other programs**	No