Audit Logs to enforce document integrity in Skyline and Panorama

Tobias Rohde[1]; Rita Chupalov[1]; Nicholas Shulman[1]; Vagisha Sharma[1]; Josh Eckels[2]; Brian S Pratt[1]; Michael J MacCoss[1]; Brendan X MacLean[1]

[1]University of Washington, Department of Genome Sciences; [2]LabKey, San Diego, CA

http://skyline.ms

Overview:

Audit logging was implemented in Skyline to facilitate more complex and reproducible proteomics workflows, improve data quality and make the software more suitable to use in regulated environments like drug development. Audit log along with Skyline document can be shared with other workflow participants using Panorama. If multiple versions of the document are uploaded audit log allows Panorama to build complete version tree of the document.

Audit log captures all changes made in the Skyline document including who made the change and when thus creating complete history of the document. Audit log integrity is protected by hashing log data. Encrypting the hash to enable electronic signature protection is planned in the future versions of the software. As such this feature is a step for the Skyline/Panorama suite towards CFR Title 21 Part 11 compliance.

Introduction:

Skyline is a Windows application used for analyzing targeted proteomics mass spectrometry data. Panorama is an open-source proteomics website knowledgebase software for sharing experiments and assays that integrates into a Skyline proteomics workflow. Audit logging was introduced into Skyline with the intention of facilitating collaboration, and improving data quality, integrity and reproducibility. The audit log is built such that one can reproduce the state of a Skyline document given only the log, the data and the original state of the document (if existent). The log data can be uploaded and viewed in Panorama together with the document. The usage of Skyline in regulated environments was also considered to make Skyline comply with the requirements set forth in CFR Title 21 Part 11.10.e, which require there to be "secure, computer-generated, time-stamped audit trails".

Figure 1: Example of an audit log form listing the changes after peak integration boundaries were adjusted.

Methods:

The audit log is built such that one can reproduce the state of a Skyline document given only the log, the data, and the original state of the document.

One audit log entry represents a single GUI transaction and can include multiple modifications made in the document. For example a user can change multiple settings in a dialog window and press OK. All the changes made in the dialog will be recorded as single entry with multiple messages, one for each change.

In Skyline and Panorama this information is presented in a grid, but can be exported into different formats. The audit logs are stored in a human readable invariant XML format, which allows Skyline to display the contents in different languages (see **Figure 3**). Several hashes are computed and stored to verify audit log – document correspondence and ensure audit log integrity on the client side (Skyline) and server side (Panorama). All information shown in **Figure 3** is used to compute the hash of the audit log entry. All entry hashes are combined to compute the root hash of the audit log. Additionally, a hash of the associated Skyline document is stored at the start of the audit log. These hashes can optionally be encrypted to protect the document with electronic signature.
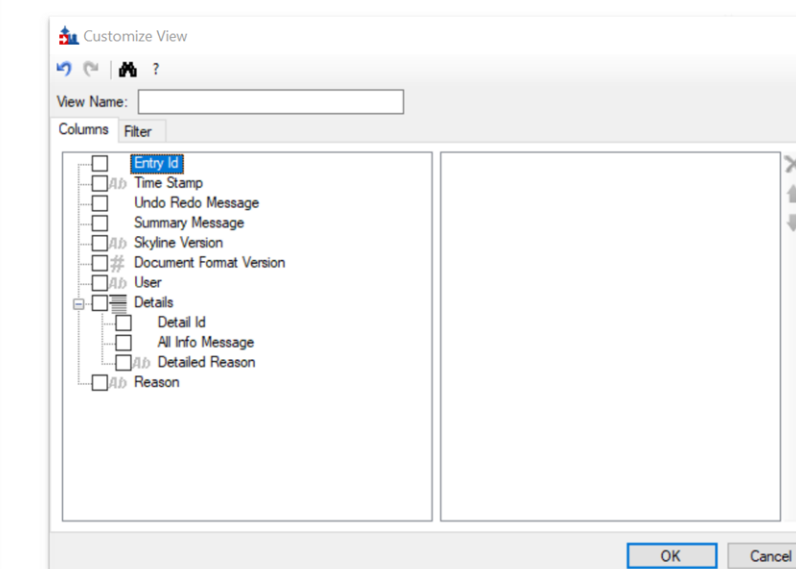
Each audit log entry contains (see **Figure 2**):
- **Time Stamp:** Time at which the change was made.
- **Undo Redo Message:** The shortest and least specific message describing the entire entry. This message will also be displayed in Skyline's toolbar when clicking on the arrows next to the Undo-Redo arrows.
- **Summary Message:** Similar (and often times the same) as the Undo Redo Message, but for certain messages more specific.
- **All Info Message(s):** A list of messages that describe the change in detail.
- **User:** Identity of the user who made the change as authenticated by the local operating system.
- **Reason:** The reason for the change, which can be set by editing the cell in the audit log viewer after the change was made. (Optional)
- **Detailed Reason:** A reason that can be set for each of the detailed "All Info Messages". (Optional)
- **Extra Info:** additional information, usually large amount of data pasted into the document. (Optional)
- **Entry Hash:** Hashed value of all data stored in this entry. The hash is used to verify data integrity as unique identifier of the document version.

Figure 2: The Audit Log viewer in Skyline can be customized to show certain information by creating a new View with specific columns selected.
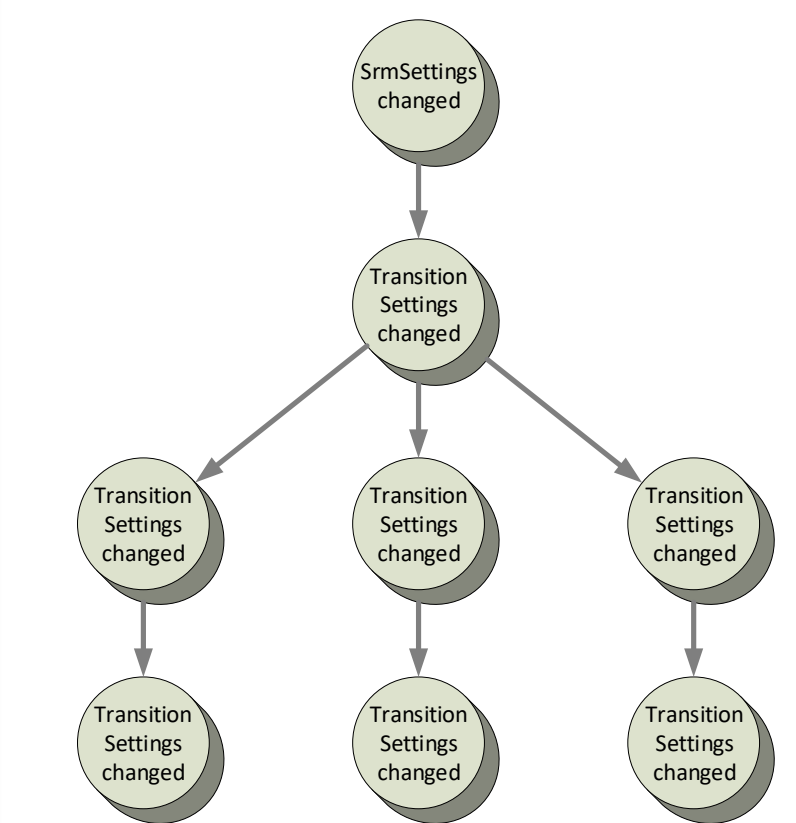

Figure 3: An audit log entry as it appears in the .skyl file. Log message type is a string resource identifier used for localization of the log messages in the Skyline UI. Names are sequences of standard substitution tokens that can be localized and are used to build a human-readable log message. English message is included to make sure log file is readable by a human and to use in Panorama UI that supports English only. It also plays a role in enforcing log integrity.

Audit log implementation in Skyline:
Skyline document has hierarchical structure. To determine changes to be logged application recursively compares old and new document nodes to identify the differences. Differences are arranged in a tree as well and different levels of this tree are logged as different message types under the audit log entry (see **Figure 4**) :
- UndoRedo Level: First node that has >1 children
  - Transition Settings changed
- Summary Level: Same as UndoRedo, but potentially longer:
  - Settings > Transition Settings changed
- All Info: All leaf nodes
  - Settings > Transition Settings > Prediction > Precursor Mass Type changed from "Average" to "Monoisotopic"

By design there is always one UndoRedo message, one Summary message and zero or more All Info messages.

Audit log is stored in its own file separately from the main document.

Audit log in Panorama:
Panorama expects both document and audit log files to be uploaded together. It parses the uploaded XML files and stores them in the database. It also provides log viewing page similar to the one in Skyline.

Panorama folder can be configured to reject a document that does not meet pre-set integrity verification requirements (missing hashes or missing electronic signature).

Figure 4: Audit log diff tree

Results:

There are indications that making Skyline audit log data visible in Panorama will be beneficial for data quality and integrity in public repositories of proteomic data like Panorama Public[1] (https://panoramaweb.org/public.url) as well as in local Panorama installations. While detailed logging is relatively common in command line proteomic tools, GUI-based software usually does not have this information readily available and its implementation in Skyline and sharing it in Panorama is desirable. Since MCP recommends Skyline/Panorama Public as publication data sharing platform[2] understanding the detailed lineage and provenance of the existing data will help researchers to reproduce existing experiments and identify problems.
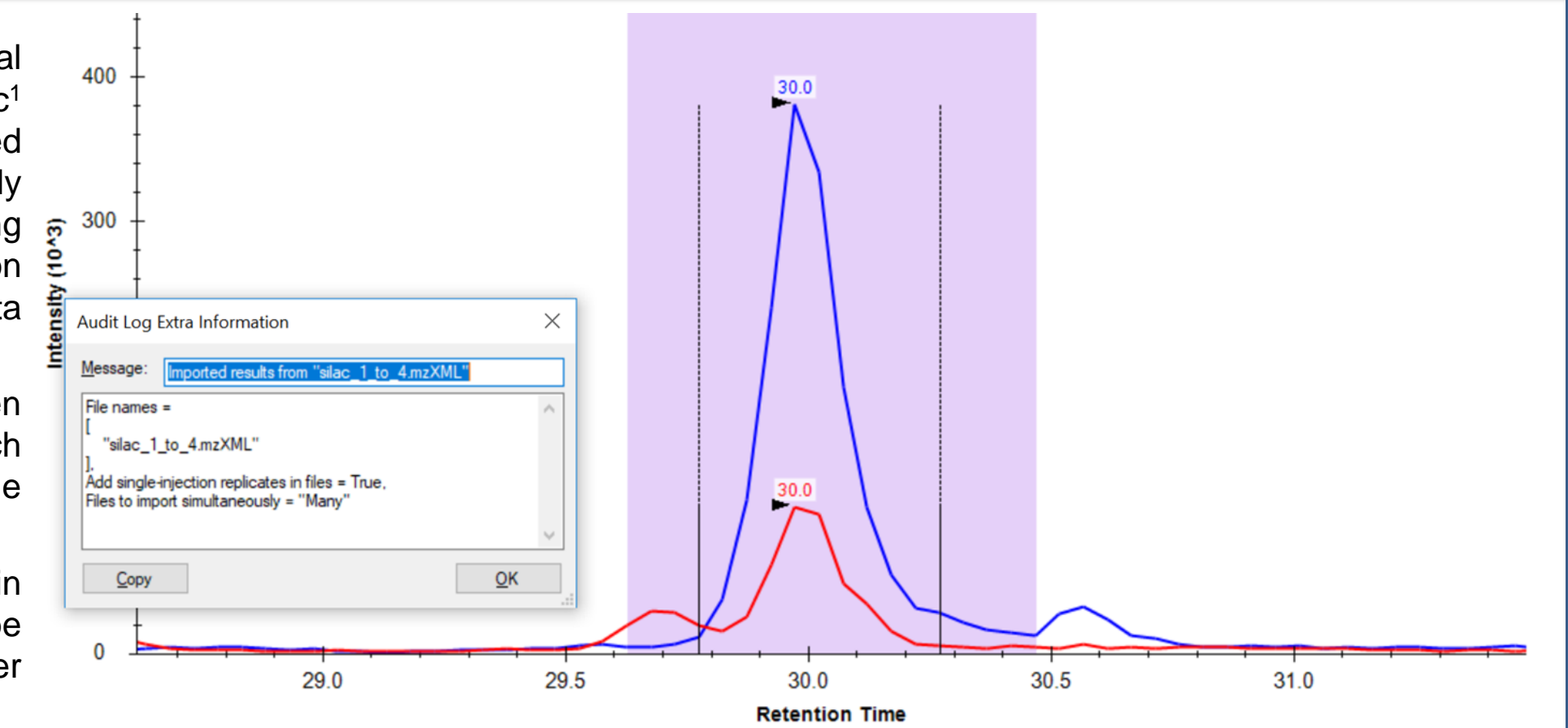
If a company creates a Panorama instance for internal use, sharing audit information between all participants in a project improves communication and enforces project-specific research protocols. The steps that led to generating a specific dataset can be traced and verified at the project level.

Detailed audit logging also allows for more robust and useful versioning of the document in Panorama. Exact relationship between multiple versions of the uploaded document can be identified to build a complete version tree. This enables more complex workflows and better accountability in regulated environments.

Protecting document and audit data with hashes, encryption and electronic signatures in Skyline and enforcing this protection in Panorama will further improve data quality, especially in publicly accessible proteomic repositories because it allows users to prevent low quality (no audit log) or damaged/tampered-with data from entering the repository. It is also very useful in regulated environments because it creates a continuous and tamper-resistant trail of data modification and access on both client and server sides.

Implementation of this feature will move the Skyline/Panorama suite one more step towards CFR Title 21 part 11 compliance, specifically section 11.10, Electronic Records Controls for closed systems

Figure 5: A practical example of audit logging in Skyline. A user imported a result file (A) and modified chromatogram peak integration boundaries (B). The purple area indicates the boundaries initially set by the software, but there is some interference that requires manual adjustment. The audit log panel below the chromatogram shows who and when imported the file and modified the boundaries and provides a place for the user to describe motivation for the change.

Some operations in Skyline can involve large volumes of data (like pasting a whole FASTA file) that would be impractical to show in the table format. If such data is present in the log entry user can click the Extra Info icon to view it in a separate window. The Audit Log Extra Information dialog box to the left demonstrates this functionality.

Figure 6: Audit log viewer page in Panorama. User can select detailed or brief views, select the columns, and export information into multiple formats. Original audit log file can be downloaded as well.

Conclusions:

Implementation of the audit logging feature in the Skyline/Panorama suite is anticipated by many of its users and moves it one step closer to be compliant with the government requirements for use in regulated environment.

Audit logging adds a new level of processing visibility and will help to make processing with Skyline more reproducible.

Complete document history captured in the audit log provides data lineage information and enables document versioning.

Future work includes implementation of the following features:
- stricter document and log integrity protection using encryption and electronic signatures.
- document version viewer to display document version tree.

References
[1] Sharma V, MacLean B, et al. Mol. Cell. Proteomics 17(6):1239-44, (2018)
[2] Abbatiello, S. et al., Mol. Cell. Proteomics 16, 327–328 (2017)