# Creating a Local Usage Collection System: PySUSHI

**Christopher Hergert, Karen Harker, Sephra Byrne**

*University of North Texas*

**Introduction**

In this paper, we will discuss the design and implementation of a built-in-house usage collection system for electronic resource usage via release 4 and release 5 of the SUSHI protocol, as well as the design and integration of additional modules for the loading of manually downloaded COUNTER reports. The first part will focus on the history and development of the COUNTER standard and SUSHI protocol to motivate the design of an efficient collection mechanism, the second will outline design specifications and needs, then the third and fourth parts of the paper will focus on the design of a usage collection system for release 4 and release 5 of the SUSHI protocol, respectively. The fifth and final part of this paper will focus on the design of a cohesive system for incorporating distinct data files, or "flat files," that have been obtained without the SUSHI protocol, into the existing usage collection system. References will be made to the design of the PySUSHI system and software library, made available by the University of North Texas Libraries (Hergert, C. and Harker, K., 2019) on Github.

**History**

Since the early days of electronic resource availability in libraries, librarians have sought to measure and quantify patrons' usage of those electronic resources, similar to checkout counts for physical materials, but those electronic resources were made available through provider platforms that varied widely in implementation. As a function of these difference implementations, the platforms' methods of counting resource utilization varied widely as well, making the direct comparison of electronic resources from different platforms an extremely tenuous endeavour.

In 2003, Project COUNTER was formed to standardize a series of metrics, as well as the schema for a finite series of reports that compiled these metrics, for the major categories of digital resources (books, journals, and later multimedia databases). The first iteration of the COUNTER Code of Practice was issued in 2003, with wide adoption by electronic resource vendors across the industry, and then subsequent versions of the standard followed (Pesch, 2007).

The SUSHI protocol was initially released in 2007, and it defines the transmission parameters and schema for automatic transmission of COUNTER-compliant data. With the release of the third iteration of the COUNTER protocol in 2009, SUSHI availability became a requirement for a vendor to be certified as COUNTER-compliant [i]. SUSHI has gone through several methodological changes over time, and most active platforms that currently provide SUSHI access do so via the SUSHI protocols accompanying the COUNTER COP4 and COP5, commonly referred to respectively as "SUSHI 4" and "SUSHI 5".

**Design Requirements and Specifications**

Several commercial providers offer subscription-based services that will make the requests to electronic resource providers, parse the server responses, and translate these responses into flat files that can easily be opened with products like Microsoft Excel, and the intent of this home-built system is to emulate the functionality of these services. To that end, the system should have four distinct necessary features, any of which another designer may choose to disregard or exclude based on their own needs: encoding request parameters and making the request to a provider's SUSHI server, receiving and parsing the server's response, storing that response data, and then logging a success/failure of the operation.

In the PySUSHI implementation, the Python programming language is used because of its versatile series of libraries that are available and easy to use, including the Requests, PyODBC, Pandas, and LXML libraries. These libraries provide easy HTTP requesting, tabular data manipulation, SQL database accessibility and manipulation, and XML-parsing functionalities, respectively. A SQL database is the natural selection for long-term data storage, based on the easily tabularized nature of time-series data like resource usage.

**COUNTER COP4**

COUNTER COP4 (Project COUNTER, 2012) was superseded as the COUNTER Foundation's active code of practice in January of 2019, but the transition to full COUNTER COP5 compliance remains ongoing, and some electronic resource platforms have not yet transitioned their SUSHI servers to the SUSHI specifications in the COP5 from COP4. For this reason, it remains necessary to have a mechanism to interact with SUSHI servers that still operate under the COUNTER COP4 specifications.

The SUSHI specifications that accompany the COUNTER COP4 is built on the WSDL SOAP messaging protocol (Pesch, 2007), meaning that the message format will be XML and the transmission will be via HTTP (Mitra and LaFon, 2007). Authentication is obtained using some combination of two login parameters obtained from the provider: the Requestor ID and the Customer ID, which some providers may refer to as a "Customer Reference". The COUNTER COP4 includes that the IP address of the requesting computer may also be used for authentication, but that is beyond the scope of this paper except to say that this is best done via correspondence with the provider platform's support team. the authentication parameters for the SUSHI request must be encoded into an XML tree, which will then be transmitted to the SUSHI server via an HTTP request, and the server's response will consist of either an error/exception message or a data load consisting of the requested report in an XML format. Both responses are necessarily encoded as UTF-8.

The other required parameters for a SUSHI request the report type, start_date, and end_date. Start_date and end_date must both be formatted as "YYYY-MM-DD" and then encoded in the datetime format, with the two start_date prior to the end_date and neither date later than the current date of the request. Other request parameters are available, such as requestor_name, requestor_email, and customer_name, but these are required to be non-mandatory (NISO, 2014).

To request a COUNTER report, the five mandatory request parameters should be written as text fields to nodes titled "End", "Begin", "Name", "CustomerReference", and "ReportDefinition". The root node of the tree should be tagged "Envelope", with one empty child node that is tagged "ReportRequest", which is a parent to each of the nodes housing a request parameter. This entire tree should be serialized into a string variable starting at the root node, which can then be sent to the SUSHI server's URL as an HTTP GET or POST request. To make the SUSHI server aware of how to interpret the data in this request, the following headers should also be included with the HTTP request: the content type should be defined as "text/xml", the charset should be defined as "UTF-8" [ii], and the provided content length should be the byte-length of the XML tree with the authentication and report date and type information.

When a response is received from the SUSHI server, it can be easily converted to a format that can be parsed using any XML-formatting software library such as the LXML library for python, which will then allow assignation of the root locations to individual variables as well as making the nodes' metadata available via accessor methods. The COUNTER COP4 specifications require that the SUSHI server reply to the client within 120 seconds (Project COUNTER, 2014), and that a "Server Busy" exception will be returned as the response payload when the required computing will require more time. For this reason, the first check that should be run when parsing COUNTER 4 reports in SUSHI format is whether there is a node tagged "exception" or "error" that is a child of the main root node, and either of these is present then the error should be examined, and no further parsing need take place. If neither of these nodes is present, then an array of strings should be created that will house the metadata to be harvested for each resource; sample metadata by their node tags include the resource name, tagged "ItemName", the platform name tagged "ItemPlatform", a print ISSN number tagged "Print_ISSN", etc. The complete list of these identifiers can be found in Section 3.4 of the SUSHI specifications (NISO, 2014). The server response has several layers of metadata that can be easily traversed by searching the tree for a node with the "ReportItems" tag, which should then be assigned to the root of the search tree. Each child of the new root will be a single resource, with each child of each resource node being either a month of usage or one of several metadata identifiers for that resource such as the print ISSN or DOI.

Once the response has been converted from an XML tree into a tabular format, it can be inserted into a table in the long-term storage database. For this purpose, PySUSHI is designed to house report data based on the type of report, e.g., one table for JR1 reports, one table for DB1 reports, etc.

Logging for COUNTER COP4 reports via SUSHI can achieve the goal of a pass/fail determination largely based on whether either of the exception or error field are populated, because these are both well-defined situations that almost universally indicate that the report request has failed or been queued for processing. If the goal for the application being developed is speed, then full-scope logging can be achieved with an independent logging process that takes the beginning and ending dates of the requested report, as well as the requested report type and the platform name, from the request parameters and create the log entries while the primary process is parsing the SUSHI server response.

**COUNTER COP5**

The COUNTER COP5 and the accompanying SUSHI protocol specifications were released in 2019 and are designed to streamline the process of usage-gathering by addressing the growing number of distinct reports that were added to the COP4 specification over time, as well as to update the SUSHI protocol to incorporate newer technologies (Project COUNTER, 2021). The goal of streamlining the entire protocol was achieved by doing away with the 20+ reports that were available in COP4 and replacing them with just four master reports that can be customized with a series of "attributes", which are parameters that are URL-encoded into the HTTP request. The second goal of updating the SUSHI protocol include replacing the SOAP/XML messaging protocol with a RESTful SUSHI API that uses JSON for the messaging protocol (Project COUNTER, 2021).

Making requests to a SUSHI server is far easier in COP5, although it can also be more uncertain because more authentication parameters are permitted. The customer reference ID and requestor ID have been retained as the "customer_id" and "requestor_id", and many providers have kept the values previously assigned to each client for these values from their COP4 SUSHI implementation when they transitioned to COP5, but the COP5 comes with the added "apiKey" authentication parameter. Not all of these authentication parameters are required for all platforms, but the parameters required for any given provider will be noted on that provider's website where the parameter values are provided. The begin_date and end_date parameters continue to be required for every report requested, and the required format is "YYYY-MM-DD". The report type is the final required parameter, and it is one of the four master reports ( "TR", "DR", "PR", or "IR"), but the report type is included differently from the dates and authentication parameters; the report type is directly appended to the end of the SUSHI server's endpoint address to complete the URI (COUNTER Foundation, 2021a). With the URI completed, the authentication parameters and attribute values are URL-encoded and appended to complete the URL, which is then sent to the SUSHI server via an HTTP GET request. Because no headers are required for SUSHI requests under COP5, a request URL can be composed and submitted in the URL bar of a web browser, which is very useful for testing and troubleshooting.

Because the report is encoded as JSON, it can be converted in Python to a dictionary-array structure that accurately emulates JSON hierarchy and indexability, and it does not include the envelope and extraneous metadata overhead that the XML trees in COP4 included. To begin parsing COP5 reports, the same checks for exceptions/errors need to be performed, and these have been combined into just one field called "exceptions" in COP5. The exception messages are no longer well-standardized in COP5, but the a two-part check can be done to determine a report's suitability for parsing by checking for the presence of an exception field and checking for any usage records in the "Report_Items" sub-dictionary; if there is an exception with no usage records, then the report can be logged definitively as a failure, and if there are usage records with no exception message, then the report can be logged definitively as a success, but when both are present then some attention may be required from an operator. In PySUSHI, this situation is addressed by logging the contents of the exception message to the user's console and continuing with the usage-gathering, allowing the operator to decide whether to remove that usage from long-term storage based on their interpretation of any data validity concerns.

The JSON tree in COP5 reports can be directly converted to a tabular format using any one of several Python libraries that serve this purpose, but this is not ideal for long-term storage of usage. A more appropriate table design is such that every row

includes usage for one unique resource-metric-month combination, for example, the *Journal of Modern Psychiatry* has a single month for all usage categorized under the unique_item_investigation metric in December of 2020. This design schema removes the issue of expanding the column set of a table, which is not optimal for time-series data collection and makes data collation difficult if a resource switches from one provider platform to another. In the PySUSHI system, one table is used for all reports of a given master report type and its derivative reports, but this causes the tables to increase in size very quickly. For designers who have greater constraints on their available storage space, this may prevent indexing on any more than a single row ID field from being viable; in this case, it is recommended that the specific reports needed be assessed, and that individual tables be created for each derivative report.

Logging successes and failures in COP5 SUSHI is more difficult than logging COP4 SUSHI because there is not a simple binary choice based on the presence of exceptions or errors, and the non-standardized exception messages in COP5 necessitate that the exception message be logged in the case that one is received. This does not preclude the possibility of pre-logging based on metadata used in the request and then populating the success/failure value when the request returns a value, but it can make the task more difficult if the size of the designated log field for potential exception messages is dynamically allocated. For this reason, if storage permits, it is ideal to statically allocate the logging table's maximum field size to the logged message field.

**Incorporating Flat Files**

When SUSHI is not available, resource providers may still seek to offer COUNTER-compliant usage statistics as downloadable files for a requested date range, most often in a tabular file that can be easily edited in Microsoft Excel. The number of metadata header rows above the column headers in the report definitions of each COUNTER iteration are a deterministic consequence of the other specifications in each COUNTER iteration, therefore a certain number of rows must be ignored when reading in the tabular usage data from the data file [iii].

If the long-term storage database utilizes shortened identifiers for particular resource providers, such as Sierra's platform codes, it is recommended that these be incorporated into the title of each usage file formulaically, along with the date range, because the identifier may otherwise be very difficult to match with resource names that have spelling errors or differences between the record and the value in the report. The date range is recommended to be included in the title of the report because these reports are not always encoded in standard UTF-8, and some providers have formatting artifacts from their report-generating software that will leave dates with extra characters that may not appear in spreadsheet software, e.g. the date 8/2020 may actually be encoded as "=8/2020". These extra characters will not bother spreadsheet software, but it they will prevent correct conversion to the datetime format when parsing the report, so it is recommended to begin from the right-side of the header column and overwrite the column titles moving left with the dates in "MM/YYYY" format so that the last month is on the far right and the earlier month is on the far left.

With this formatting corrected, the tabular data imported from the data file can be modified in-memory with a SQL query [iv] that will transform the data from a horizontal structure to a vertical structure that is more appropriate to time-series data, and which will match the schema of the tables used for data acquired via SUSHI.

Logging for data files is more difficult due to the larger breadth of formatting issues can arise and the lack of request metadata that benefits SUSHI request logging, but pass/fail logging can be achieved by running the entire data import in a try-pass block, where logging is contingent on whether data can be correctly imported *and* reformatted in-memory to match the necessary table schemas. If any step in that process fails, the default outcome is that no usage is placed in long-term storage and a failure is logged. This failure can be overwritten after the issues in the report's formatting have been manually addressed, and when the report is successfully uploaded.

**Conclusion**

While many services are available that will request and parse usage data from a list of provider platforms, it is very possible to build software that will serve these purposes in-house, and for a considerably lower cost. Additional benefits include the

flexibility to add data analysis and transformation modules that may never be available on the commercial usage collection platforms, and the Python language offers a host of options for adding this flexibility through libraries like SciKit, Pandas, and PyODBC.

**References**


1. Hergert, C. and Harker, K. (2019), “PySUSHI”, available at: https://github.com/unt-libraries/PySUSHI/ (accessed 17 October 2021)
2. Mitra, N. and LaFon, Y. (2007), “SOAP Version 1.2 Part 0”, available at: https://www.w3.org/TR/2007/REC-soap12-part0-20070427/ (accessed 17 October 2021)

3. NISO (2014), “NISO SUSHI Protocol: COUNTER-SUSHI Implementation Profile”, available at https://groups.niso.org/apps/group_public/download.php/13635/RP-14-2014_COUNTER_SUSHI_IP.pdf (accessed 14 October 2021)
4. Pesch, O. (2007), “Usage statistics: About COUNTER and SUSHI”, *Information Services and Use*, Vol. 27, p207-213
5. COUNTER Foundation (2012), "The COUNTER Code of Practice for e-Resources: Release 4", available at https://www.projectcounter.org/wp-content/uploads/2016/01/COPR4.pdf (accessed 17 October 2021)
6. COUNTER Foundation (2021), "Changes from COUNTER Release 4", available at https://cop5.projectcounter.org/en/5.0.2/01-introduction/02-changes-from-counter-release-4.html (accessed 18 October 2021)
7. COUNTER Foundation (2021a), "COUNTER_SUSHI_0 API", available at https://app.swaggerhub.com/apis/COUNTER/counter-sushi_5_0_api/5.0.2 (accessed 17 October 2021)
8.

**Notes**

i. “SUSHI Protocol – History/Origins”, available at https://www.niso.org/standards-committees/sushi/historyorigins (accessed 14 October 2021)

ii. While the SUSHI documentation specifies that the encoding be UTF-8, this is a detail that does not seem to have been regularly checked when provider platforms were being audited for COUNTER compliance. Some providers have been encountered that do not correctly interpret UTF-8 request data without this being specified.

iii. The number of ignorable metadata header rows is 7 for COUNTER COP4 and 13 for COP5, which can be observed in the report examples at https://www.projectcounter.org/code-of-practice-sections/usage-reports and https://www.projectcounter.org/code-of-practice-five-sections/4-1-usage-reports respectively.

iv. Examples of SQL queries that will appropriately transform these horizontally-structured base reports into vertically-structured tables are available in the parsing functions at https://github.com/unt-libraries/PySUSHI/blob/master/Release%202/Counter5pybr.py