NISO TR-06-201X

SUSHI-Lite:
Deploying SUSHI as a lightweight protocol for exchanging usage via web services

A Technical Report of the
National Information Standards Organization

Prepared by
The SUSHI-Lite Technical Working Group
an initiative of the SUSHI (ANSI/NISO Z39.93-2014) Standing Committee

Available for Trial Use and Comments:
July 16 – September 30, 2015

DRAFT FOR TRIAL USE

About NISO Technical Reports

NISO Technical Reports describe the work and conclusions of a working groups chartered by the organization to study a particular problem or issue that is related to NISO's standardization interests. However, unlike NISO standards or recommended practices, they do not have requirements for compliance or recommend "best practices" – although their conclusions may contain some recommendations. Additionally, no consensus process of the NISO voting membership was used in their creation or approval. They are strictly informative in nature.

Published by
National Information Standards Organization (NISO)
3600 Clipper Mill Road
Suite 302
Baltimore, MD 21211
www.niso.org

ISBN: to be added at publication

Contents

Foreword

About this Technical Report

ANSI/NISO Z39.93-2014, also known as the SUSHI (Standardized Usage Statistics Harvesting Initiative) standard, is the key to automating the harvesting of COUNTER (Counting Online Usage of NeTworked Electronic Resources) usage statistics. SUSHI is a critical standard for librarians charged with measuring and monitoring the use of their online collections by eliminating hours of painstaking effort that would otherwise be spent locating, retrieving and loading usage reports. However, environmental requirements evolve and standards like SUSHI need to update to serve these requirements. The introduction of various applications into the marketplace which offer alternative metrics, the development of the COUNTER Journal Usage Factor, the flourishing of institutional repositories and need to capture usage from them, and continued progress towards open and integrated systems in general, have all made an impact on how usage is consumed and exchanged. There is now a need for more lightweight technologies that will allow smaller sets of usage data to be exchanged in real-time.

This technical report describes a method of exchanging COUNTER statistics ranging from usage for a single article to a complete COUNTER report using an easy-to-implement commonly used approach to web services. This report does not replace the SUSHI standard but rather supplements it with an alternative approach for requesting and exchanging usage.

This simplified version of SUSHI will benefit the entire library community by allowing SUSHI and COUNTER to become the standards for delivering content and collection usage data to systems providing analytics and real-time access to usage data. The more forgiving nature of a RESTful interface and JSON data format should minimize compliance issues and help ensure acceptance by the mainstream web development community.

The beneficiaries of this approach includes anyone implementing a web page displaying alternative metrics, in which the user is performing tasks related to the e-resource workflow and access to real-time usage data is desirable. Implementers of the COUNTER Code of Practice for Articles and/or Usage Factor may also benefit by being able to request usage for one article, or all articles from one journal.

Any content provider needing to specify more than just a CustomerID and Requester ID in a SUSHI request will benefit from the new filters. These filters are also required to exchange usage for a single item, a set of items or some other subset of what would normally be in a COUNTER report.

NISO Business Information Topic Committee Members

The Business Information Topic Committee had the following members at the time it approved this Technical Report.

[To be added by NISO after approval]

NISO SUSHI-Lite Technical Report Working Group Members

The following individuals served on the NISO SUSHI-Lite Technical Report Working Group.

Clinton Graham (Observer)
University of Pittsburgh University Library

Brian Gregg (Observer)
University of Pittsburgh University Library

Miriam Lorenz (through March 2015)
Cologne University of Applied Sciences

Guy Marjew
Elsevier

Paul Needham (Co-chair)
Cranfield University

Yogesh Patel (Observer)
Mimas

Oliver Pesch (Co-chair)
EBSCO Information Services

Mike Showalter
Plum Analytics

Alan Stiles
Open University

James Van Mil
University of Cincinnati Libraries

This Working Group is an initiative of the NISO SUSHI Standing Committee.

Trademarks, Services Marks

1. Introduction

1.1 Purpose and Scope

This technical report describes how SUSHI and COUNTER can be adapted to achieve the following three objectives:

- Allow smaller units of usage data to be retrieved with SUSHI. This support will position SUSHI to become the standard for implementing real-time retrieval of usage for single journals or articles as this retrieval is utilized within e-resource workflows and systems offering alternative metric displays.
- Allow for an optional implementation of SUSHI for use with current-day practices for implementing the web services that would be accessing COUNTER usage report snippets. Specifically, specify an optional implementation of SUSHI using a RESTful HTTP interface with COUNTER usage returned formatted in JSON.
- Introduce a generalized filter and report-attribute specification that can be used with the proposed REST style/JSON approach.
    - Filters would allow the client application to limit the scope of the data in the report. For example, a filter could limit the request to a single book, journal or article or a customer account number or department.
    - Report attributes allow the client application to control the format or completeness of data to be returned. Examples of report attributes use would include the exclusion of zero-usage records and the return of minimal item-level metadata.

The result of this alteration is that the SUSHI standard can be transformed to become:

- *The* standard to use when implementing services related to alternative metrics or real-time display of usage within the e-resource workflow.
- Effective in handling more fine-grained SUSHI requests, thus eliminating the current technique of overloading the Requestor ID or Customer ID that some content providers resort to in order to implement their SUSHI service.
- Much easier to implement, through leveraging present-day approaches to the development of web services that use SUSHI to retrieve COUNTER usage.

This report starts by providing a series of pertinent Definitions, before describing the Protocol that SUSHI-Lite employs to simplify the original SOAP/XML service approach. SUSHI-Lite uses a RESTful interface for requests with usage data returned as JSON, JSONP or XML. Security Considerations are reviewed followed by the Request Syntax section which includes the list of parameters available to request small units of usage data. The section on Responses describes the relationship between the SUSHI-Lite responses and COUNTER_SUSHI XML schemas and provides a set of simple rules for interpreting the COUNTER XML to create JSON or JSONP output. The details of the main transaction type between client and server are described in the GetReport Method section which covers the request, processing the request and the server's response. A section on Performance Considerations outlines suggestions for creating a server that will meet performance needs of its clients. In addition to the examples provided throughout the document, more complete examples are included in the Appendices.

1.2 Terms and Definitions

The following terms, as used in this technical report, have the meanings indicated.

Term	Definition
APIKey	An identifier that a service provider has registered for the requesting system. APIKey should be based on the UUID system to ensure uniqueness. UUID (universally unique identifier) is an identifier standard used in software construction, standardized by the Open Software Foundation (OSF) as part of the Distributed Computing Environment (DCE).
Content Provider	The entity that provides the content to which usage statistics relate, typically a scholarly publisher, aggregator or institutional repository. For example, Elsevier provides content via the ScienceDirect platform.
Filter	One or more parameters on the SUSHI request intended to limit the set of usage records to be returned for a specified COUNTER report. Filters can be used to limit the record set by IP range, department, a specific journal or article, and more. Filters are optional.
JSON	JavaScript Object Notation (JSON), is an open standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs. It is used primarily to transmit data between a server and web application, as an alternative to XML. [Wikipedia: http://en.wikipedia.org/wiki/JSON]
JSONP	"JSON with padding" is a communication technique used in JavaScript programs running in web browsers to request data from a server in a different domain, something prohibited by typical web browsers because of security concerns. [Wikipedia: http://en.wikipedia.org/wiki/JSONP]
ReportAttribute	One or more parameters on the SUSHI request intended to specify the format of the results to be returned. ReportAttributes can be used to suppress records with zero-usage, limit the amount of metadata to return, and more. ReportAttributes are optional.
Requestor	For SUSHI requests, the requestor is the organization/entity that is making the request for usage. Note that hosted services for consolidating usage may act as a requestor for many different institutions.
Requestor ID	A unique identifier assigned by a service provider to a requestor. Frequently, a service provider will use the Requestor ID in concert with the Customer ID as a means of access control to the requested usage data.

Term	Definition
REST	Representational State Transfer (REST) is a software architectural style consisting of a coordinated set of architectural constraints applied to components, connectors, and data elements, within a distributed hypermedia system. REST ignores the details of component implementation and protocol syntax in order to focus on the roles of components, the constraints upon their interaction with other components, and their interpretation of significant data elements. [Wikipedia: http://en.wikipedia.org/wiki/Representational_state_transfer]
RESTful	Web service APIs that adhere to the REST architectural constraints are called RESTful APIs. HTTP based RESTful APIs are defined with these aspects: • base URI, such as http://example.com/resources/ • an Internet media type for the data. This is often JSON but can be any other valid Internet media type (e.g., XML, Atom, microformats, images, etc.) • standard HTTP methods (e.g., GET, PUT, POST, or DELETE) • hypertext links to reference state • hypertext links to reference related resources [Wikipedia: http://en.wikipedia.org/wiki/Representational_state_transfer]
Service Provider	The entity that is providing the usage statistics. The end-point that is being called to request usage data.
Scope	As related to ItemIdentifier , the "scope" indicates the level of the identifier (e.g., whether it identifies an article, an issue or a journal).

2. The Protocol

The SUSHI-Lite protocol is implemented using a REST model implemented over HTTP/S with responses returned as JSON or XML. Some general expectations are:

- Requests are submitted using HTTP/S `Get`
- Requests and responses can contain Unicode characters
- Request parameters must be URL-encoded
- Response data returned as XML must be valid XML using standard encoding techniques
- Response data returned as JSON must escape/encode any special characters and entities using standard techniques.

3. Security Considerations

The information returned in the SUSHI-Lite response may be sensitive and considered private. To protect this data:

- SUSHI-Lite implementations are encouraged to use HTTP/S to secure the data channel.
- SUSHI servers may implement authentication challenges using one or more of:
  - Requestor ID
  - Customer ID
  - APIKey
  - IP Address of client
- Other security measures, such as HTTP-level usernames and passwords and private-key-encryption of the data are not supported by this recommended practice and can only be used when client and server both agree to use such methods.

See Appendix D for more information on authentication.

4. Request Syntax

Following the REST model, the SUSHI-Lite request uses the following general syntax:

```
{baseURL}/{Version}/{Method}[?{Report_Parameters}]
```

Where:

- **baseURL** = the end-point of the service. The baseURL alone leads to a human-readable page that describes the available API versions, the methods, and displays Server Registry data.
- **Version** = represents the version of the service as related to the version of SUSHI the service supports (e.g., /v1_8).
- **Method** = required and must be GetReport or a proprietary method offered by the server. (The GetReport method is discussed later in this document.)
- **Report _Parameters** = a list of parameters as dictated by the method. For GetReport, the parameters would identify the report, the requester, the customer, and include filters and report attributes. In the sections below you will find details for the GetReport method.

5. Response Syntax

The server will respond to a SUSHI-Lite request with a response that is formatted as either XML, JSON, or JSONP, as defined by the method and what the user has requested.

5.1 Structure and Contents of the Response

The contents of the response (list of elements) and its structure will vary by method and will be defined using XML schemas. Using the XML schema allows the response structure to be clearly articulated along with the element names and any constraints on those elements.

The SUSHI schema will serve as the envelope for the response and a container for the report. Each method may have its own schema that describes the report being returned with a high-level schema connecting the two. For example, the GetReport method generally returns a COUNTER report; therefore, the response will be governed by the:

COUNTER_SUSHI.xsd

Which connects the

SUSHI.xsd and

COUNTER.xsd

This could be diagrammed as follows:

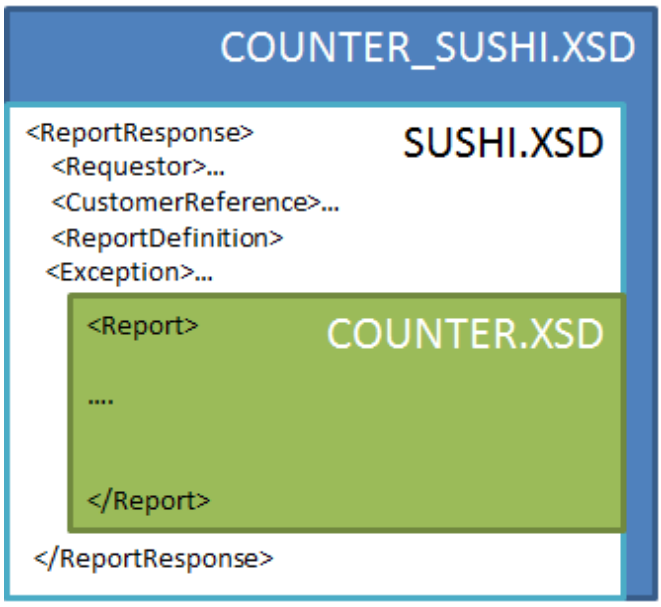

Figure 1: Visual of COUNTER_SUSHI Response

JSON is the most likely format for the response; however, XML is used to describe contents and structure because it offers the formality needed for widespread adoption not offered by JSON. The expectation is the response will be formatted as valid XML transformed to JSON using a standard approach.

5.2 Transforming the XML to JSON

If a request calls for the response to be in JSON, logically the response would be formatted to the specifications of the XML schema(s) governing the method and report and the JSON would be generated by transforming that XML to JSON. See section 8.3 for details.

5.3 Handling Error Conditions

As a general rule, the structure of the SUSHI-Lite response will be governed by the SUSHI schema; therefore, any error conditions that can be reported will be specified within the SUSHI response. The following is a diagram from the SUSHI schema that shows the format of the exception:

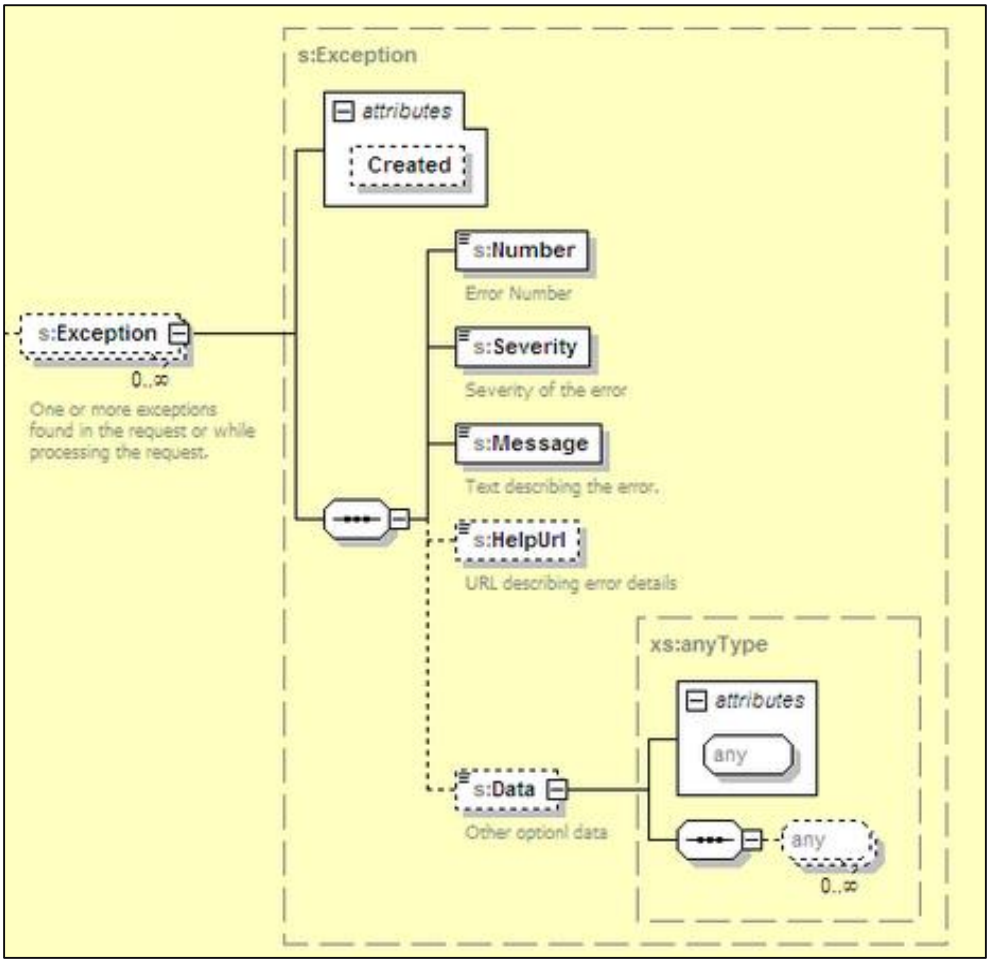

Figure 2: SUSHI Schema Format of the Exception

Specific exceptions are discussed in much more detail in the GetReport Method: Processing Requests section.

6. GetReport Method: Request

Following the REST model, the SUSHI-Lite request for a COUNTER report uses the following general syntax:

```
{baseURL}/{Version}/{Method}?Report={Report}&Release={Release}&RequestorID={RequestorID}&CustomerID={CustomerID}&{Filter_list}&{ReportAttributes_list}
```

Where:

- **baseURL** = the end-point of the service. The baseURL alone (with no added parameters) resolves to a human-readable page that describes the available API versions, the methods, and displays descriptive data about the server as may be found in the SUSHI server registry: http://www.niso.org/workrooms/sushi/registry_server/.
- **Version** = represents the version of the service, as related to the version of SUSHI the service supports (e.g., /v1_8).
- **Method** = GetReport.
- **Report** = the short name of the COUNTER report (e.g., JR1, BR1, BR2, AR1, etc.). This will determine the nature of data to be returned. For example, a JR report will return journal-level usage and an AR report will retrieve article-level usage. View the SUSHI report registry for more information: http://www.niso.org/workrooms/sushi/registry/#reports.
- **Release** = valid release numbers of the Usage Report. If Release is omitted, the server will assume a default equal to the then-prevailing COUNTER release.
- **RequestorID** = the identifier that the service provider has registered for the requestor. Note that some service providers may choose to not control access and thus allow for this element to be eliminated, or they may use the term *anonymous* for uncontrolled access.
- **CustomerID** = the identifier of the institution whose usage is to be obtained (sometimes used in combination with RequestorID to provide access to the correct usage for the institution). Required for requesting usage related to an institution's usage. This should be the same ID used to identify the customer for COUNTER SUSHI reports. If the requested usage is not limited to activity from users of a particular institution, this parameter should be omitted.
- **APIKey** = the identifier that the service provider has registered for the requesting system to identify the project or application requesting the usage.
- **Filter_list** = a list of one or more key-value-pairs that further refine what data is to be returned. Note that Filter_list is not an actual URL parameter but a placeholder for optional filters. See Filter List in this document for options.
- **ReportAttributes_list** = a list of one or more key-value pairs that further qualify the desired report/response. Note that ReportAttributes_list is not an actual URL parameter but a placeholder for optional filters. See Report Attributes in this document for options.

6.1 Filter List

Following is the list of filters that may be imposed on the request.

SUSHI-Lite supports multiple filter elements and multiple values within a single filter instance. The chart below indicates which filters are repeatable.

- Multiple values within a filter instance must all be of the same scope (e.g., a list of article identifiers)
- Multiple values within a filter instance are logically ORed
- Multiple filter instances are ANDed

An example of an excerpt of a request is as follows:

```
…&ItemContributor=orcid:0000-0002-8721-8082&ItemIdentifier=journal:issn:1111-
                              1111|journal:issn:2222-2222…
```

In this example, the usage snippet returned will be limited to contents authored by Smith and published in either of the journals represented by ISSN 1111-1111 or 2222-2222. The following might be the Boolean search logic used by the server:

```
                              Select usage FROM usage_store
WHERE (ORCID="0000-0002-8721-8082") AND ((ISSN=1111-1111) OR (ISSN=2222-2222))
```

Note that for certain reports and certain endpoints, some filters are required; however, if omitted a recommended default will be applied as shown in Table 1.

Table 1: Filter List for SUSHI-Lite Request

Filter name (key)	Description	Comments	Repeat-able	Recommended default value
BeginDate	Usage start date in the form of yyyy-mm-dd or yyyy-mm.	Usage that was transacted on or after this date will be included in the report.	N	First day of the prior month
EndDate	Usage end date in the form of yyyy-mm-dd or yyyy-mm.	Usage that was transacted on or before this date will be included in the report.	N	Last day of the prior month
ItemIdentifier	Identity of an item or items for which usage is requested. The value of an ItemIdentifier is a	The item being identified might be an article, a journal, a book, etc. as identified in the scope or component. The type of identifier might be a	Y	n/a

	three-part colon-separated string that defines the scope, type, and value of the identifier. See *ItemIdentifiers* below.	DOI, ISSN, etc. Examples and detailed explanations are below.		
ItemContributor	Identity of a contributor or contributors for which usage is being requested. This is a three-part, colon-separated string that defines the identifier role, type, and value for the contributor. The role is a well-known role and is optional. Default is *author*.	Filtering to return results for a given contributor (e.g., author) is done by specifying the identifier type (e.g., ORCID) and the value. Examples and more details are below. Including the role is optional, but when included the expectation is that the report will be limited to items where the contributor had the specific role (e.g., where the contributor served as author, illustrator, editor, etc.).	Y	n/a
ResourceType	Identity of the type of resource usage statistics to be returned.	The resource type or document type; this parameter could include values such as Book, Journal or Article. Providing an authoritative list of resource types is beyond the scope of this technical report; therefore, clients and servers should confer before using.	Y	*all*
Publisher	Name of the publisher of the items for which usage is to be retrieved.	Used for service providers that can deliver usage for multiple publishers.	Y	*all*
Platform	Name of the COUNTER platform from which to pull usage.	Used for service providers that can deliver usage for multiple platforms.	N	Service provider's default platform

MetricTypes	List of metric types to return. Must be valid COUNTER_SUSHI metric type.	If multiple metric types requested, implement as pipe-separated list (e.g., ft_total|ft_html). See the SUSHI report registry (http://www.niso.org/workrooms/sushi/registry).	Y	Metric types available for a given report
PubYr	Publication Year in the form of *yyyy*. Special values of *9999* for articles in print or *0001* for unknown are allowed.	This filter is only applicable to requests for COUNTER reports AR1 and JR5. This filter is mutually exclusive to the PubYrFrom and PubYrTo filters.	N	
PubYrFrom	Start year of publication.	This filter is only applicable to requests for COUNTER reports AR1 and JR5.	N	
PubYrTo	End year of publication.	This filter is only applicable to requests for COUNTER reports AR1 and JR5.	N	
IsArchive	Limit usage to just archival content.	This filter is only applicable to requests for COUNTER report JR5.	N	No

6.1.1 ItemIdentifier

The ItemIdentifier is a flexible filter parameter that can be used to limit results to an article, a journal, a book, a publisher, or more. The syntax for use of ItemIdentifier is as follows:

```
ItemIdentifier=[<scope>:]<type>:<value>
```

where:

- ***scope*** defines the level (or scope) of the identifier. For example, since a DOI could apply to an article, a journal, or even an issue, the scope provides the mechanism to specify that fact on the request. If the scope segment of the ItemIdentifier value is omitted, the server will derive the scope from the Report being requested. For example, for Journal reports, the default ItemIdentifier scope is *Journal* and for Article reports the default is *Article*.

- ***type*** is the well-known type of identifier that will be used as the namespace for the scope and value (e.g., doi).
- ***value*** is the actual identifier value.

The following tables show common values for scope, type(s), and a cross-tabulation shows valid types for each scope option.

Table 2: List of Common Options for ItemIdentifier Scope Component

Scope	Description
journal	A serial publication.
issue	An identifiable unit of publication of a journal containing many articles.
article	An individual work found within a journal.
book	Used for print or online books.
chapter	An identifiable section within a book.

Table 3: List of Common Options for ItemIdentifier Type Component

Type(s)	Description	Examples/Comments
issn	Identity of the serial publication for which the usage is being requested.	0277-3732
doi	The digital object identifier that has been registered to identify an article, chapter, book, journal, or other digital object for which usage is to be requested.	10.1080/0361526X.2013.790872 NOTE: Do NOT include the http://dx.doi.org/
isbn	The international standard book number that identified the print of eBook for which usage is to be requested. The ISBN-13 should be used.	The expectation is that the service provider will look for the value in either the PRINT_ISBN or the ONLINE_ISBN fields.
proprietary	A vendor-specific identifier that is being used by prior agreement between the two parties exchanging usage.	

Table 4: Valid Scope and Type Pairings

	journal	issue	article	book	chapter
issn	X				
doi	X	X	X	X	X
isbn				X	
proprietary	X	X	X	X	X

Examples:

1) Requesting usage for a journal, for example, American Journal of Clinical Oncology:
   ItemIdentifier=journal:issn:0277-3732
2) Requesting usage for a specific article by DOI:
   ItemIdentifier=article:doi:10.1080/0361526X.2013.790872

6.1.2 ItemContributor

The ItemContributor parameter is used when requesting usage for items attributed to a contributor. Even though the most common use of this parameter will be to request usage for articles written by a particular author, the ItemContributor has been designed to allow the request to be limited to a particular contributor and role. The syntax for use of ItemContributor is as follows:

```
ItemContributor=[role:]<type>:<value>
```

where:

- ***role*** limits the results to a specific role played by the contributor in the creation of the works for which usage is being requested (e.g., author, editor, etc.).
- ***type*** is the well-known type of identifier that will be used as the namespace for the scope and value (e.g., orcid, isni, name).
- ***value*** is the actual identifier value.

The following table shows suggested values for ItemContributor roles and types. Note that the roles listed in the table are representative of typical roles. This list is not complete or authoritative; consumers should consult individual content providers to determine if roles are supported locally and, if so, which ones.

Table 5: ItemContributor Roles (examples only)

Roles	Description
author	Creator of the work.
editor	Person responsible for editing the work.
illustrator	Person responsible for figures and illustrations.
translator	Person responsible for translating work into another language.
investigator	Person on the investigation team, but who may not actually have a hand in writing the article.
dataanalyst	Person responsible for analyzing the data.

Table 6: ItemContibutor Types

Type(s)	Description	Examples/Comments
orcid	An ORCID ID is a unique identifier for researchers and is used to help ensure a linkage between researchers and their research output and other activities within the research workflow. See http://orcid.org for more information.	0000-0002-8721-8082
isni	International Standard Name Identifier is an ISO standard (ISO 27729:2012) for identifying parties in the creation, production, management and content distribution chains.	0000000056465502 NOTE: Do NOT include the embedded spaces that may be present in the display version of an ISNI.
proprietary	A vendor-specific identifier that is used by prior agreement between the two parties exchanging usage data.	

Examples:

1) Requesting usage for items by an individual where no particular credit or role is specified:
   `ItemContributor=orcid:0000-0002-8721-8082`
2) Requesting usage for items by a contributor where their role was translating the work:
   `ItemContributor=translator:orcid:0000-0002-8721-8082`

Note: If the SUSHI server does not support item contributor roles, it is expected to ignore the roles.

6.2 Report Attributes

The list of attributes that may be included on the request to further refine the response is as follows:

Table 7: Report Attributes That May Appear in a Request

Attribute name (key)	Description	Comments	Repeat-able	Recommended default value
Granularity	Typically COUNTER reports provide usage totals by month for a given report item. The Granularity attribute allows the client to specify the report to be more or less granular.	Values ranging from: Totals, Yearly, Monthly, Daily.	N	Monthly, unless report usually provides different level of granularity.
ExcludeZeroUsage	Indicates if the report should exclude items and metric-type instances where the usage was zero. A value of Y would remove these from the report.	Note that this attribute would only need to be supported for reports where COUNTER specifies that zero usage may be reported (e.g., JR1). It would not be practical to support this attribute for an article-level report where zero usage would always be excluded.	N	Y
Format	Format in which the response is to be returned.	Values: JSON, JSONP, XML. Note that if the format is set to JSONP, the callback parameter should be used to specify a callback function.	N	JSON
Callback	Name of callback function for JSONP. Only used when Format=JSONP.	Provides cross-domain access by wrapping the JSON payload in a function call.	N	callback
Limit	The number of items to return in the response. For COUNTER reports	This option is used on large datasets or reports when only a subset of records is needed, or when the client wants to	N	*all items*

	this refers to the number of ReportItems elements.	retrieve the results in chunks.		
Offset	The row number of the requested dataset at which to begin the retrieval.	When used with the Limit attribute, the client can retrieve the dataset in chunks. This is a one-based value.	N	1
OrderBy	The sort field and order of dataset being retrieved.	<field>[:<order>] order may be asc (ascending) or desc (descending). If item name is omitted, ascending is assumed. The orders available will depend on the type of report being requested. The most common sort orders are Item Name and FT Total. Other sort orders may be offered by the server and used by the client with mutual consent.	N	ItemName:asc ft_total:desc

The following table includes report attributes included in the response but not in the request.

Table 8: Additional Report Attributes That May Appear in a Response But Not the Request

ReportItemCount	The total number of report items found by the server (provided by the server only).	This ReportAttribute is supplied by the server in the response. The client will use the returned value to manage fetching the report in chunks. Note: Since this is a server-provided value, if the client should include this in the request it will be ignored by the server.	N	*n/a*

7. GetReport Method: Processing Requests

This section discusses recommended practices related to a server/service processing a request for a report. Topics include:

- Authentication/authorization
- Validating parameters and exception handling
- Handling server/service load

7.1. Authentication/Authorization

For many institutions, usage data and other information about their collections are considered confidential and are covered by institutional security policies.

The SUSHI-Lite protocol supports several levels and options for protecting the privacy of customers and securely exchanging the usage data including:

- Using HTTP/S to secure the data communication channel.
- Authentication of the requesting organization (the SUSHI-Lite client software) using:
    - the Requestor ID to make sure the client is registered
    - and/or, optionally, checking the IP address of client against a list of authorized clients
- Optionally, an additional layer of security can be obtained by having individual customers authorize individual clients (Requestor IDs) to harvest their usage.
- A further level of security can be implemented using an APIKey to authorize individual users, applications or projects to access specific customer-level data or use specific reports.

An in depth discussion of Authentication and Authorization can be found in Appendix D.

If the client is unable to authenticate the client, the SUSHI response will include the appropriate exception:

Table 9: Error Conditions and Exceptions Related to Authentication

Condition	Exception to return	Exception Number
Unable to find a required RequestorID, CustomerID, or APIKey on the request so not processed.	Insufficient information to process Request.	1030
Server unable to validate the client because of an unknown Requestor ID or a failure to pass IP authentication.	Requestor Not Authorized to Access Service.	2000
The Requester is recognized; however, they are not authorized to access the account in the request.	Requestor is Not Authorized to Access Usage for Institution.	2010

The Requester has submitted an unrecognized account in the request. (The response is the same for both since the server may not want to advertise that the customer ID is invalid to avoid clients systematically phishing for a valid ID.)		
The server expects an APIKey and the one presented is invalid or not valid for the requested customer.	APIKey invalid	2020

See Appendix C for a complete list of exceptions and error codes.

7.2. Validating Parameters and Processing Exceptions

Once the basic authentication step has been completed, the server will then process the remaining parameters of the requests.

When a client omits optional parameters or components of parameter-values, the SUSHI service should process the request if it can do so in an unambiguous way by applying the defaults for the missing information. Sections 6.1 and 6.2 describe if a Filter or ReportAttribute parameter or a component of a parameter value is optional and what default to apply if it is missing.

If there is a problem with a parameter, or if the request cannot be completed as asked, the server should respond with a valid SUSHI response including one or more exceptions to explain the problem.

Following is a list of the most common problems that may be encountered and the exceptions to respond with. A complete list of exceptions can be found in Appendix C.

Table10: Error Conditions and Exceptions Related to Request Parameter Validation

Condition encountered	Exception to return	Exception Number
Report requested is not supported.	Report Not Supported.	3000
Report supported but not the requested version.	Report Version Not Supported.	3010
Begin or End dates not specified or invalid.	Invalid Date Arguments.	3020
There is no usage data for the report requested (after filter criteria has been applied).	No Usage Available for Requested Dates.	3030

Service has not processed the usage for the requested date range.	Usage Not Ready for Requested Dates.	3031
Some months of usage are available but not for all requested months.	Partial Data Returned.	3040
A parameter is not recognized (e.g., the request includes a made-up parameter or a misspelling).	Two options: 1. If unable to process the request, respond with "Parameter Not Recognized in this Context. 2. Otherwise, do not report an exception and process the request (i.e., the server should ignore parameters that are not recognized).	3050
A parameter is included and known as valid; however, it does not make sense for the report/request being made. For example, the caller is requesting report JR1 and they include the PubYr filter, which only applies to JR5.	Parameter Not Recognized in this Context.	3051
Filter value does not meet the standard (e.g., ItemContributor is missing required sub-elements).	Invalid Filter Value.	3060
A filter element includes multiple values in a pipe-delimited list; however, they are not all of the same scope.	Incongruous Filter Value.	3061
ReportAttribute value does not meet the standard (e.g., a SortOrder parameter contains an invalid or unsupported value).	Invalid ReportAttribute Value.	3062
A required filter is missing (e.g., the user is requesting a report but fails to include a Begin and/or End date).	Required Filter Missing.	3070
The filter value is not found in the usage data being processed (e.g., there is no usage for the requested DOI).	No exception is expected related to the filter not matching; however, if the result is that there is no usage to return, then the exception "No usage for requested dates" would be appropriate.	

A required report attribute is missing (e.g., the SUSHI server requires the caller to specify the platform and none was included).	Required ReportAttribute Missing.	3071
The requested value for limit (number of items to return) exceeds the server limit; the server will supply the limit in the Message element of the exception, and then proceed to return its limit for report items.	Limit Requested Greater than Maximum Server Limit.	3080

- If a filter element has multiple values (pipe-delimited-list):
    - All values should be at the same scope or level (e.g., an ItemIdentifier filter could identify multiple articles or multiple journals, but not a combination of both).
    - The assumption is the caller wants the usage for all requested items to be returned, if there is any. Assume a logical "OR" between the values.
- If multiple filter elements are included, the assumption is that multiple filters are intended to narrow the result set; therefore, multiple filters would be assumed to have a logical "AND" between them.
- If it is possible to process the request by ignoring the invalid filters or attributes, or by applying defaults, the processing should go ahead. It should:
    - Include the exceptions found, but also
    - Include the report

8. GetReport: Response

The usage will be returned in a dataset using element structures, organizations and names derived from the COUNTER_SUSHI XML schema. Following are guidelines for formatting the response based on the format requested.

8.1 XML

For COUNTER reports, the response is expected to conform to the COUNTER_SUSHI XML schema with well-formed XML. The following outline represents the logical structure of the major COUNTER-SUSHI elements in a response.

```
ReportResponse
    • Exception (optional)
    • Exception (optional)
    • Requestor
    • CustomerReference
    • ReportDefinition
        ○ ReportName
        ○ ReportRelease
        ○ Filters
            ■ UsageDateRange
            ■ Filter (optional)
            ■ Filter (optional)
            ■ ReportAttribute (optional)
            ■ ReportAttribute (optional)

    • Report (optional)
        ○ Report
            ■ Vendor
            ■ Customer (include the customer ID which allows
              for consortium reports)
                • ReportItems
                • ReportItems
```

For SUSHI-Lite the COUNTER_SUSHI schema will bind the SUSHI1_7.xsd and the COUNTER4_1.xsd.

8.1.1 SUSHI Response

The SUSHI-Lite response is formatted in accordance to version 1.7 of the SUSHI schema. Specifics include:

Table 11: Elements to Include in the SUSHI-Lite Response

Response element path	Request parameter value is derived from	Comment
CustomerReference/ID	CustomerID	
ReportDefinition Name=	Report	
ReportDefinition Release=	Release	
ReportDefinition/Filters/Usage DateRange/Begin	BeginDate	

ReportDefinition/Filters/Usage DateRange/End	EndDate	
ReportDefinition/Filters/Filter	Filter	See Section 6.1.
ReportDefinition/Filters/Report Attribute	ReportAttribute	See Section 6.2.
ReportDefinition/Filters/Report Attribute/Name = ReportItemCount	(The total number of ReportItems found in the COUNTER report that matches the request with the filter criteria applied.)	If there is no usage to report, set the value to 0.
Exceptions	(Zero or more exceptions resulting from analyzing and processing the request.)	Only include exception elements if there is an exception to report.

The SUSHI-Lite protocol is designed to deliver usage in chunks. The client can control the size of the response using the *Limit* ReportAttribute with the *Offset* ReportAttribute controlling the position in the report. For COUNTER reports, it is the ReportItems that are counted. In addition to returning the Filter and ReportAttribute elements from the ReportDefinition in the Request, the server is also expected to return an additional ReportAttribute with the total number of items found that matched the request criteria.

Following is an example of the ReportDefinition element for the response where a client requested a JR1asking for 1,000 items to be returned starting at the first record. In this example, the COUNTER report has 62,435 report items; therefore, the response includes the ReportItemCount as a ReportAttribute (bolded below) to indicate this.

```
RESPONSE snippet
...
       <ReportDefinition Name="JR1" Release="4">
           <Filters>
               <UsageDateRange>
                   <Begin>2014-01-01</Begin>
                   <End>2014-01-01</End>
               </UsageDateRange>
                   <ReportAttribute>
                       <Name>Offset</Name>
                       <Value>1</Value>
                   </ReportAttribute>
                   <ReportAttribute>
                       <Name>Limit</Name>
                           <Value>1000</Value>
                   </ReportAttribute>
                   <ReportAttribute>
                       <Name>ReportItemCount</Name>
                       <Value>62435</Value>
                   </ReportAttribute>
           </Filters>
       </ReportDefinition>
...
End of RESPONSE snippet
```

8.1.2 COUNTER Report Data

The report is formatted in accordance to version 4.1 of the COUNTER XML schema. Use the technique described in section 8.3 to achieve a lossless transformation to JSON.

8.2 JSON and JSONP

JSON responses are informed by the COUNTER_SUSHI XML schema for the response structure, element names, and element values. Use the following basic rules to transform an XML to JSON:

- COUNTER_SUSHI.xsd and the schemas it binds together describe the structure of the response.
- SUSHI.xsd describes the element names for the SUSHI response.
    - Output the Name and Release report attributes in the ReportName and ReportRelease elements.
- COUNTER.xsd describes the element name for the COUNTER report returned by the response.
    - Output the Report and any ItemPerformance attributes as elements, using the elements provided in the schema.
- COUNTERElements.xsd provides the list of allowed values for several of the elements found within the COUNTER schema.
- Transform the XML to JSON following the methodology outlined in section 8.3.

JSONP is simply JSON "padded" or wrapped in a function.

8.3 Rules for Transforming XML to JSON

The SUSHI-Lite protocol provides a simple set of rules for transforming well-formed XML elements into JSON. (The following assumes the reader is knowledgeable about XML and JSON notation.)

- Fields with simple values become key:value pairs
- Multiple instances of similarly named fields at one level (siblings) are turned into key: array_of_values
- Multiple key:values within a parent key are represented in a hash ({} curly brackets bounded).
- Attributes are mapped as { "@attributename" : "attributevalue" }

The following examples show some snippets of XML and JSON to show how the transformation behaves for various scenarios.

This XML snippet includes multiple ItemIdentifiers.

XML Snippet:

```
...
<ItemIdentifier>
  <Type>Print_ISSN</Type>
  <Value>1111-2222</Value>
</ItemIdentifier>
<ItemIdentifier>
  <Type>Online_ISSN</Type>
  <Value>1212-2121</Value>
</ItemIdentifier>
<ItemDataType>Journal</ItemDataType>
...
```

The JSON representation:

```
...
“ItemIdentifier”:[{“Type”:”Print_ISSN”,”Value”:”1111-
2222”},{“Type”: “Online_ISSN”,”Value”:”1212-2121”}],
“ItemDataType”:”Journal”
...
```

In the next example the XML contains attributes.

XML Snippet:

```
<ReportDefinition Name=”JR1” Release=”4”>
     <Filters>
        <UsageDateRange>
                <Begin>2014-01-01</Begin>
                <End>2014-12-31</End>
        </UsageDateRange>
        ...
</ReportDefinition>
```

The JSON representation:

```
“ReportDefinition”: {
        “@Name”:”JR1”,
        “@Release”:”4”,
        “Filters”:{
          “UsageDateRange”:{
                “Begin”:”2014-01-01”,
                “End”:”2014-12-31”
        }
        ...
}
```

Note the ReportDefinition attributes have been transformed by prefixing the attribute name with an “@”.

The following XML shows what happens when returning multiple report exceptions:

XML snippet:

```
<Exception Created=”2015-03-31T13:45:51Z”>
        <Number>3031</Number>
        <Severity>Error</Severity>
        <Message>Usage not ready for requested
        dates</Message>
        <Data>
                “Data not processed for 2014-12”
        </Data>
</Exception>
<Exception Created=”2015-03-31T13:45:51Z”>
        <Number>3040</Number>
        <Severity>Warning</Severity>
        <Message>Request could not be fulfilled in its
        entirety. Data that was available was
        returned</Message>
</Exception>
```

The JSON representation:

```
"Exception": [{
        "@Created":"2015-03-31T13:45:51Z",
        "Number":"3031",
        "Severity":"Error",
        "Message":"Usage not ready for requested dates",
        "Data":{
                "Data not processed for 2014-12"
        },
        {
        "@Created","2015-03-31T13:45:51Z",
        "Number":"3040",
        "Severity":"Warning",
        "Message":"Request Could not be fulfilled in its
entirety. Data that was available was returned",
        }]
```

Note in this instance the attribute named `created` of the <Exception> element is represented as `"@created"` within each instance.

Other notes related to returning COUNTER usage via SUSHI-Lite:

1) The COUNTER XML version 4.1 schema has been expanded to accommodate article level data.
2) The SUSHI-Lite protocol can be used to request any kind of report that can be returned in JSON, JSONP or XML format.
    - COUNTER reports are within the scope of this work.
    - Non-COUNTER reports, such as KBART or transaction log details, are beyond the scope of this project, but SUSHI-Lite could easily support such an exchange provided an agreed-upon schema was produced to represent the report.

9. Performance Consideration

The COUNTER Code of Practice sets the expectation that the server should respond to a request within 120 seconds. This performance expectation will be acceptable to applications using the SUSHI-Lite protocol to request standard COUNTER reports or other large data sets; however, it is anticipated that SUSHI-Lite will also be used by applications that integrate usage snippets into the user experience in real-time, such as requesting the usage for a single journal for a given year. For these applications, the server needs to be responsive and able to respond to reasonable requests in less than 2 seconds.

SUSHI servers should be able to accommodate both types of applications and meet performance standards for response times of:

- < 2 seconds for snippets of usage
- < 120 seconds for full reports

Appendix E offers some suggestions for handling loads.

10. Next Steps

The working group has identified several potential next steps for this initiative. Feedback from the community on this technical report is expected to inform the priority of these steps as well as suggest other directions for this initiative.

1) This technical report will be released for public comment with the expectation that all comments will be responded to and feedback will be added to the report as appropriate.
2) A final report will be published containing the implemented feedback, with the expectation that the protocol and approach described in this technical report will be implemented.
3) Further developments may include the introduction of at least one more method, GetStatus, which will allow a client to programmatically inquire as to the health and capabilities of a given SUSHI-Lite server.
4) The results of implementations of SUSHI-Lite will inform future versions of the SUSHI standard.

The working group believes that SUSHI-Lite solves several problems related to exchanging COUNTER usage data in an efficient and flexible way. Working group members encourage developers to implement the protocol and provide feedback to ensure this work is meeting its intended need.

Appendix A:
Extract of the ReportItem Element from COUNTER Schema Updated to Handle Article-Level Reporting

Below is an extract of the COUNTER XML schema (version 4.1) focusing on the ReportItem element to demonstrate how a ReportItem (e.g., an article) can have a ParentItem element to define the context (e.g., Journal information).

```
<xs:complexType name="ReportItem">
        <xs:sequence>
                <xs:element name="ParentItem" type="c:ParentItem" minOccurs="0" maxOccurs="1">
                </xs:element>
                <xs:element name="ItemIdentifier" type="c:Identifier" minOccurs="0"
maxOccurs="unbounded">
                </xs:element>
                <xs:element name="ItemContributor" type="c:ItemContributor" minOccurs="0"
maxOccurs="unbounded">
                </xs:element>
                </xs:element>
                <xs:element name="ItemAttribute" type="c:ItemAttribute" minOccurs="0"
maxOccurs="unbounded">
                </xs:element>
                <xs:element name="ItemPlatform" type="xs:string" minOccurs="1" maxOccurs="1">
                </xs:element>
                <xs:element name="ItemPublisher" type="xs:string" minOccurs="0" maxOccurs="1">
                </xs:element>
                <xs:element name="ItemName" type="xs:string" minOccurs="1" maxOccurs="1">
                </xs:element>
                <xs:element name="ItemDataType" type="c:DataType" minOccurs="1" maxOccurs="1">
                </xs:element>

                <xs:element name="ItemPerformance" type="c:Metric" minOccurs="1"
maxOccurs="unbounded">
                </xs:element>
        </xs:sequence>
</xs:complexType>

<xs:complexType name="ParentItem">
        <xs:sequence>
                <xs:element name="ParentItem" type="c:ParentItem" minOccurs="0" maxOccurs="1">
                </xs:element>
                <xs:element name="ItemIdentifier" type="c:Identifier" minOccurs="0"
maxOccurs="unbounded">
                </xs:element>
                <xs:element name="ItemDate" type="c:ItemDate" minOccurs="0"
maxOccurs="unbounded">
                </xs:element>
                <xs:element name="ItemPublisher" type="xs:string" minOccurs="0" maxOccurs="1">
                </xs:element>
                <xs:element name="ItemName" type="xs:string" minOccurs="1" maxOccurs="1">
                </xs:element>
                <xs:element name="ItemDataType" type="c:DataType" minOccurs="1" maxOccurs="1">
                </xs:element>
        </xs:sequence>
</xs:complexType>
```

```
<xs:complexType name="Identifier">
	<xs:sequence>
		<xs:element name="Type" type="c:IdentifierType" minOccurs="1" maxOccurs="1">
		</xs:element>
		<xs:element name="Value" type="xs:string" minOccurs="1" maxOccurs="1"/>
	</xs:sequence>
</xs:complexType>

<xs:complexType name="ItemAttribute">
	<xs:sequence>
		<xs:element name="Type" type="c:AttributeType" minOccurs="1" maxOccurs="1">
		</xs:element>
		<xs:element name="Value" type="xs:string" minOccurs="1" maxOccurs="1"/>
	</xs:sequence>
	</xs:complexType>

<xs:complexType name="ItemContributor">
	<xs:sequence>
		<xs:element name="Type" type="c:ContributorType" minOccurs="1" maxOccurs="1">
		</xs:element>
		<xs:element name="Identifier" type="xs:string" minOccurs="0"
maxOccurs="unbounded">
		</xs:element>
		<xs:element name="Value" type="xs:string" minOccurs="1" maxOccurs="1"/>
	</xs:sequence>
</xs:complexType>

<xs:complexType name="ItemDate">
	<xs:sequence>
		<xs:element name="Type" type="c:DateType" minOccurs="1" maxOccurs="1">
		</xs:element>
		<xs:element name="Value" type="xs:string" minOccurs="1" maxOccurs="1"/>
	</xs:sequence>
</xs:complexType>
```

Appendix B: Examples of XML for COUNTER Article Report ReportItems

The following XML snippets (derived from one of the SUSHI-Lite sandboxes) are provided to demonstrate how article-level usage would be returned as a ReportItems element within a COUNTER Article Report. Note that this excerpt only includes a single ReportItems element; therefore, it does not represent a complete SUSHI-Lite response.

```
<ReportItems>
        <ParentItem>
                <ItemIdentifier>
                       <Type>Print_ISSN</Type>
                       <Value>0017-8012</Value>
                </ItemIdentifier>
                <ItemIdentifier>
                       <Type>Proprietary</Type>
                       <Value>32090</Value>
                </ItemIdentifier>
                <ItemIdentifier>
                       <Type>Volume</Type>
                       <Value>92</Value>
               </ItemIdentifier>
                <ItemIdentifier>
                       <Type>Issue</Type>
                       <Value>6</Value>
                </ItemIdentifier>
                <Itemdate>
                        <Type>PubDate</Type>
                       <Value>2014-06</Value>
                </Itemdate>
                <ItemPublisher>Harvard Business School Publication Corp.</ItemPublisher>
               <ItemName>Harvard Business Review</ItemName>
                <ItemDataType>Journal</ItemDataType>
        </ParentItem>
        <ItemContributor>
               <Type>Author</Type>
                <Identifier>orcid:someNumber</Identifier>
                <Value>Margolis, Joshua D</Value>
        </ItemContributor>
        <ItemContributor>
                <Type>Author</Type>
               <Identifier>isni:someOtherNumber</Identifier>
               <Value>Gallo, Amy</Value>
        </ItemContributor>
        <ItemIdentifier>
                <Type>StartPage</Type>
               <Value>123</Value>
       </ItemIdentifier>
       <ItemIdentifier>
               <Type>EndPage</Type>
                <Value>127</Value>
       </ItemIdentifier>
        <ItemIdentifier>
               <Type>DOI</Type>
               <Value>SomeDOI</Value>
        </ItemIdentifier>
       <ItemAttribute>
               <Type>ArticleVersion</Type>
               <Value>VoR</Value>
        </ItemAttribute>
```

```
        <ItemAttribute>
                <Type>ArticleType</Type>
                <Value>Research Article</Value>
        </ItemAttribute>
        <ItemName>Career Choices When Life Is Short</ItemName>
       <ItemPlatform>EBSCOhost</ItemPlatform>
                      <ItemDataType>Article</ItemDataType>
                     <ItemPerformance>
                              <Period>
                                     <Begin>2013-10-01</Begin>
                                     <End>2013-10-31</End>
                              </Period>
                              <Category>Requests</Category>
                              <Instance>
                                      <MetricType>ft_total</MetricType>
                                      <Count>1</Count>
                              </Instance>
                              <Instance>
                                     <MetricType>ft_html</MetricType>
                                     <Count>1</Count>
                               </Instance>
                     </ItemPerformance>
                     <ItemPerformance>
                               <Period>
                                     <Begin>2013-11-01</Begin>
                                     <End>2013-11-30</End>
                              </Period>
                              <Category>Requests</Category>
                              <Instance>
                                     <MetricType>ft_total</MetricType>
                                     <Count>0</Count>
                              </Instance>
                               <Instance>
                                     <MetricType>ft_html</MetricType>
                                      <Count>0</Count>
                              </Instance>
                    </ItemPerformance>
</ReportItems>
<ReportItems>
       <ParentItem>
                <ItemIdentifier>
                       <Type>Print_ISSN</Type>
                       <Value>0018-2753</Value>
                </ItemIdentifier>
               <ItemIdentifier>
                       <Type>Proprietary</Type>
                        <Value>33231</Value>
                </ItemIdentifier>
               <ItemName>History Today</ItemName>
               <ItemPublisher>History Today Ltd.</ItemPublisher>
               <ItemDataType>Journal</ItemDataType>
                     </ParentItem>
                     <ItemIdentifier>
               <Type>StartPage</Type>
               <Value>43</Value>
       </ItemIdentifier>
        <ItemIdentifier>
               <Type>EndPage</Type>
               <Value>51</Value>
       </ItemIdentifier>
       <ItemIdentifier>
               <Type>DOI</Type>
               <Value>SomeOtherDOI</Value>
       </ItemIdentifier>
       <ItemAttribute>
               <Type>ArticleVersion</Type>
               <Value>VoR</Value>
       </ItemAttribute>
```

```
       <ItemPlatform>EBSCOhost</ItemPlatform>
                      <ItemName>Some Article in History Today</ItemName>
                       <ItemDataType>Article</ItemDataType>
                       <ItemPerformance>
                              <Period>
                                     <Begin>2013-10-01</Begin>
                                     <End>2013-10-31</End>
                              </Period>
                              <Category>Requests</Category>
                              <Instance>
                                      <MetricType>ft_total</MetricType>
                                      <Count>0</Count>
                              </Instance>
                              <Instance>
                                     <MetricType>ft_html</MetricType>
                                     <Count>0</Count>
                              </Instance>
                     </ItemPerformance>
                     <ItemPerformance>
                              <Period>
                                     <Begin>2013-11-01</Begin>
                                     <End>2013-11-30</End>
                                      </Period>
                              <Category>Requests</Category>
                              <Instance>
                                     <MetricType>ft_total</MetricType>
                                      <Count>1</Count>
                                      </Instance>
                              <Instance>
                                      <MetricType>ft_html</MetricType>
                                     <Count>1</Count>
                                     </Instance>
                     </ItemPerformance>
</ReportItems>
```

Appendix C:
Handling Errors and Exceptions

The following table is copied from Table 18 of the SUSHI Standard (ANSI/NISO Z39.93-2014) and has been supplemented with additional exception items to address possible exceptions in use of the SUSHI-Lite protocol with filter items and APIKey use. If SUSHI-Lite is adopted as part of the formal SUSHI standard, these additions to the exception list will be incorporated into the standard document.

Table 12: SUSHI Exceptions

Exception	Severity Level	Exception Number	Invocation Conditions
Info or Debug	Info Debug	0	Any. These messages will never be standardized and service providers can design them as they see fit.
Warnings	Warning	1-999	Any. This range is reserved for the use of service providers to supply their own custom warnings.
Service Not Available	Fatal	1000	Service is executing a request, but due to internal errors cannot complete the request. Service must return ReportResponse and no payload.
Service Busy	Fatal	1010	Service is too busy to execute the incoming request. Service must return ReportResponse with this exception and no payload. Client should retry the request after some reasonable time.
Client has made too many requests	Fatal	1020	If the server sets a limit on the number of requests a client can make within a given timeframe, the server will return this error when the client exceeds that limit. The server would provide an explanation of the limit in the Message of the error (e.g., "Client has made too many requests. This server allows only 5 requests per day per RequesterID and CustomerID.").
Insufficient Information to Process	Fatal	1030	There is insufficient data in the request to begin processing (e.g., missing Requestor ID, Report is missing, no Customer ID,

Request			etc.).
Requestor Not Authorized to Access Service	Error	2000	If Requestor ID is not recognized or not authorized by the service.
Requestor is Not Authorized to Access Usage for Institution	Error	2010	If Requestor has not been authorized to harvest usage for the institution identified by the CustomerReference ID, or if the CustomerReference ID is not recognized.
APIKey Invalid	Error	2020	The service being called requires a valid APIKey to access usage data and the key provided was not valid or not authorized for the data being requested.
Report Not Supported	Error	3000	The requested report name, version, or other means of identifying a report that the service can process is not matched against the supported reports.
Report Version Not Supported	Error	3010	Requested version of the data is not supported by the service.
Invalid Date Arguments	Error	3020	Any format or logic errors involving date computations (e.g., end date cannot be less than begin date).
No Usage Available for Requested Dates	Error	3030	Service did not find any data for the date range specified.
Usage Not Ready for Requested Dates	Error, Warning	3031	Service has not yet processed the usage for one or more of the requested months, if some months are available that data should be returned. The exception should include the months not processed in the additional data element.
Partial Data Returned	Warning	3040	Request could not be fulfilled in its entirety. Data that was available was returned.
Parameter Not Recognized in this Context	Warning	3050	Request contained one or more parameters that are not recognized by the Server in the context of the report being serviced. The server should list the Name of unsupported filter in the Message

			element of the Exception. Note: The server is expected to ignore unsupported parameters and continue to process the request, returning data that is available without the parameter being applied.
Invalid Filter Value	Warning, Error	3060	Request contained one or more Filter values in the ReportDefinition that are not supported by the Server. The server should list the Name of unsupported filter values in the Message element of the Exception. Note: The server is expected to ignore unsupported filters and continue to process the request, returning data that is available without the filter being applied.
Incongruous Filter Value	Warning, Error	3061	A filter element includes multiple values in a pipe-delimited list; however, the supplied values are not all of the same scope (e.g., ItemIdentifier filter includes article level DOIs and journal level DOIs or ISSNs).
Invalid ReportAttribute Value	Warning, Error	3062	Request contained one or more ReportAttribute values in the ReportDefinition that are not supported by the Server. The server should list the Name of unsupported report attribute values in the Message element of the Exception. Note: The server is expected to ignore unsupported report attributes and continue to process the request, returning data that is available without the report attribute being applied.
Required Filter Missing	Warning, Error	3070	A required filter was not included in the request. Which filters are required will depend on the report and the service being called. For example, if the service requires that the request define the Platform name and no Platform filter is included, an exception would be returned. In general, the omission of a required filter would be viewed as an *Error*; however, if the service is able to process

			the request using a default value then a *Warning* can be returned. The Message element of the Exception should name the missing filter.
Required ReportAttribute Missing	Warning, Error	3071	A required report attribute was not included in the request. For example, if the service requires that the request define the Platform name and no Platform filter is included, an exception would be returned. In general, the omission of a required filter would be viewed as an *Error*; however, if the service is able to process the request using a default value, then a *Warning* can be returned. The Message element of the Exception should name the missing filter.
Limit Requested Greater than Maximum Server Limit	Warning	3080	The requested value for limit (number of items to return) exceeds the server limit. The server is expected to return data in the response (up to the limit). The Message element of the exception should indicate the server limit.

Note 1: An Error does not interrupt completion of the transaction (in the sense of a programmatic failure), although it may not return the expected report for the reason that is identified. A Fatal exception does not complete the transaction; the problem may be temporary and a retry could be successful.

Note 2: Optional response: Service may respond with the additional exception of Info level and include additional information in the message. For example, if the client is requesting data for a date range where the begin date is before what the service offers, the service might include a HelpURL that can provide more information about supported dates.

Note 3: If multiple exceptions are discovered, each exception should be returned in its own element.

Note 4: Clarifying details about an exception (e.g., the filter that was missing or deemed invalid should be added to the Data element or Message element of the exception so that the caller knows what to correct).

Note 5: If the caller gets the baseURL, the version, or method wrong, the expectation is that they will receive an HTTP 404 error since the specified path is not valid.

Appendix D: Authentication/Authorization

For many institutions, usage data and other information about their collections are considered confidential and are covered by institutional security policies.

Three levels of security that can be implemented by SUSHI-Lite clients and servers are:

1) Securing the data communication channel
2) Authenticating the requesting organization (the SUSHI-Lite client software)
3) Validating the rights of a requesting organization to access specific customer usage data

1. Securing the Communications Channel

The SUSHI-Lite protocol can use either HTTP or HTTP/S for transmission between client and server applications. By using HTTP/S, the communication between client and server is encrypted using SSL (Secure Sockets Layer), thereby preventing any third party from intercepting the transmission and discovering its content.

It is recommended that SUSHI-Lite developers implement their Web services using HTTP/S.

2. Authenticating the Requesting Organization

Using HTTP/S, the communication channel is secure; however, there is still a possibility that an unauthorized software application could access a SUSHI-Lite server and request usage data. To prevent this, the content provider can implement a security layer within the SUSHI-Lite server. Following are three options:

- Requestor ID validation
- IP authentication
- APIKey

The simplest form of authentication would be to validate the Requestor ID. Unless the Requestor is known to the server, the request would not be processed. The Requestor ID can be a simple number or code identifying the client, or, if stronger security is desired, the value for the Requestor ID could include encrypted information, such as the domain of the client so that when the client submits the SUSHI-Lite ReportRequest, the server can decrypt the Requestor ID to verify that the client is legitimate.

Adding IP authentication is another option to create a much stronger level of security. The requesting organization would need to register the IP address of the computer running its SUSHI-Lite client software with the service provider(s). The service provider would only process requests for recognized IP addresses. When utilizing IP authentication for SUSHI-Lite clients, implementers should be aware that many institutions will use a hosted usage harvesting service, which means the same client with the same IP address may be making requests on behalf of many institutions.

The third option is to use an APIKey to control access to the service. An APIKey is a unique identifier that can be assigned to a specific application, project, or service that is using a set of usage. A given APIKey is generally assigned to an application, service, or group of APIs, and is associated with a set of usage (e.g., usage for a particular publisher, customer or group of customers, database, or platform). The service provider may use the APIKey alone or in conjunction with other authentication methods like IP Address or institutional token such as RequestorID for added security. The service provider may also provide the APIKey with a specific service level for a particular database or

platform such as *testing only* or *priority*. With the APIKey approach, multiple research projects or applications from the same institution can harvest that institution's usage data allowing the service provider to control access at the project or application level without affecting other projects or applications.

Examples of APIKey implementations from other disciplines include search (replacing Federated Search), text mining, content view support and content innovation, and enhancement support.

The APIKey can also serve a role that is beyond simple authorization to use a given service. A content provider that issues unique APIKeys per research project or application will be able to create usage reports for the usage. It would be possible to aggregate the historical SUSHI usage per project or application to establish better insight about the operational interaction via SUSHI API between different stakeholders for that project or application. This aggregation would be possible at the various levels, (e.g., the content provider, the service provider, or the end user), and would have to be based on service responses that would have been logged as successful.

In conjunction with the ItemIdentifier and ItemContributor report attributes, it would be possible to establish support for breaking down "usage-on-usage" reports to the particular content or contributor dimensions which are identified as relevant and useful.

In case of more centralized exchange of "usage-on-usage" information, it would also be possible to define a usage-on-usage report template and use SUSHI in the same manner as for more content-oriented report requests.

3. Validating Rights of a Requesting Organization to Access Specific Customer Usage Data

Most content providers will store usage data for a large number of institutions and will also have a large number of requesting organizations who wish to harvest the usage data. The content provider may introduce another security layer to restrict authorized requesting organizations to certain customer data.

The SUSHI-Lite ReportRequest element contains the Requestor ID (identifying the requesting organization) and the CustomerReference ID (identifying the organization whose usage is to be harvested). Service providers can fairly easily set up a system that requires their customers to *authorize* requesting organizations to harvest their data. If the service provider registers the requesting organizations, then it can present its customers with a simple user interface that gives them the option to *activate* SUSHI-Lite harvesting, and then identify the requesting organization(s) permitted to do the harvesting. The result is a mapping between CustomerReference IDs and Requestor IDs, allowing the SUSHI-Lite server to verify that the data harvesting is permitted before processing is continued.

For service providers who are using IP authentication for the requesting organization, a simpler model could be implemented when the requestor and the customer are the same. The SUSHI-Lite server could verify that the IP address of the requestor is included in the IP range registered for the customer and, if so, processing of the request would continue.

Appendix E:
Handling Server/Service Load

In the use of the SUSHI-Lite protocol, the SUSHI service supports harvesting of full COUNTER reports as well as snippets of usage. A common use case for the snippets of usage may be real-time embedding of usage data within another application. Applications which use SUSHI-Lite to embed usage into another application will expect a service to be extremely fast with sub-second response time. An application which retrieves a full COUNTER report should expect that transaction to take longer.

When implementing a SUSHI service, the following are some recommendations that will help achieve the performance expectations of the calling programs:

- Implement a multi-threaded or multiple-queue environment so that multiple requests can be handled simultaneously.
- If report processing is handled asynchronously using a queuing approach, introduce the notion of priorities to the queue so that a snippet request can be assigned a higher priority and jump the queue ahead of requests for lengthy reports.
- If large reports cannot be prepared within the expected 120 seconds (the maximum delay allowed by the COUNTER Code of Practice), return a *server busy* exception immediately and process the request offline. Since most clients will re-request the report every few minutes or hours, the server can continue to return a *server busy* exception until the report processing is complete and then return the report.
- If an APIKey is used and is assigned for a specific project or application, consider assigning a priority for requests made using that key. For example, if an APIKey represents a project that is pulling usage statistics for later analysis, a lower priority could be assigned to all requests using this key. If, on the other hand, the APIKey is associated with real-time integration of usage data in a high-profile application (e.g., showing usage related to a journal in an acquisition module), then all requests using that APIKey could be given high priority.
- Any given service needs to determine what level of activity it is prepared to offer.

Appendix F:
Demonstration Sites Available for SUSHI-Lite Testing

Part of the output of the working group was the creation of working systems and code to serve as demonstrations to the community. Please refer to the up-to-date list of demonstration sites available on the NISO SUSHI web site at: http://www.niso.org/workrooms/sushi/sushi_lite/demo_sites/.

Bibliography

COUNTER (Counting Online Usage of NeTworked Electronic Resources) [website]. Initiative by libraries and publishers. Available at: http://www.projectcounter.org.

SUSHI Report Registry [website]. List of SUSHI-compliant content providers. Available at: http://www.niso.org/workrooms/sushi/registry/.

SUSHI (Standardized Usage Statistics Harvesting Initiative) [website]. Tool for automating the harvesting of COUNTER. Available at: http://www.niso.org/workrooms/sushi/.

JSON (JavaScript Object Notation) [website]. Lightweight data-interchange format based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. Available at: http://www.json.org/.

JSONP (JSON "with padding") [website]. A communication technique used in JavaScript programs running in web browsers to request data from a server in a different domain. Available at: http://en.wikipedia.org/wiki/JSONP.

REST (Representational State Transfer) [website]. *Architecture style* for designing networked applications. Available at: http://rest.elkstein.org/.

HTTP (Hypertext Transfer Protocol) [website]. Application protocol for distributed, collaborative, hypermedia information systems and is the foundation of data communication for the World Wide Web. Available at: http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol.

SOAP (Simple Object Access Protocol) [website]. Messaging protocol that allows programs that run on disparate operating systems (such as Windows and Linux) to communicate using HTTP and XML. Available at: http://www.w3schools.com/webservices/ws_soap_intro.asp.

XML (Extensible Markup Language) [website]. A software- and hardware-independent tool for carrying information. Available at: http://www.w3schools.com/xml/.