Paper 018-2012

Integrating Enterprise Search with Various Clients

Arvind Jagtap, Kaushlendra Rai, and Prasad Kulkarni, SAS Institute Inc., Pune, India

ABSTRACT

In today's world of information technology, the information is exploding through the Internet. Search has become an integral part of life. Enterprise contents are no exception and the enterprise search plays a major role in finding the required information quickly to enhance the productivity of business executives.

A need for searching can arise at any time while working with some popular applications. Hence, integrating search features with these applications can be helpful.

As a part of SaaS, SAS® provides "Search Interface to SAS Content" as a service for searching enterprise contents. This paper demonstrates how this service can be invoked from various clients like Microsoft Office, Internet Explorer, SharePoint, and Google Search Appliance to allow users to search the required contents without leaving the application.

INTRODUCTION

Search Interface to SAS Content is a product provided by SAS to search enterprise contents like SAS reports, BI dashboards, and so on that are useful to business users in order to analyze information. Since a user might need to search these objects from any application, this software is provided as a service for better integration. In a future SAS Business Intelligence release, it will be integrated with some of the SAS BI Clients. But if the user wants to integrate it with any third-party application or any other solution they are developing, then this paper will be useful to know the technical details.

SEARCH INTERFACE TO SAS CONTENTS

This SAS product searches the following types of BI artifacts from the SAS deployment.

- SAS reports
- stored processes
- information maps
- BI dashboards
- visual exploration (a new type that will be supported in an upcoming release)
- second generation SAS reports (a new type that will be supported in an upcoming release)

These artifacts are generally used by business executives to get summarized information in order to make decisions. There can be hundreds of such artifacts created for different combinations of interesting parameters like sales, time, profit, and so on. Whenever a user wants to seek some new information, this product will be handy in fetching the right report for a given set of search keywords.

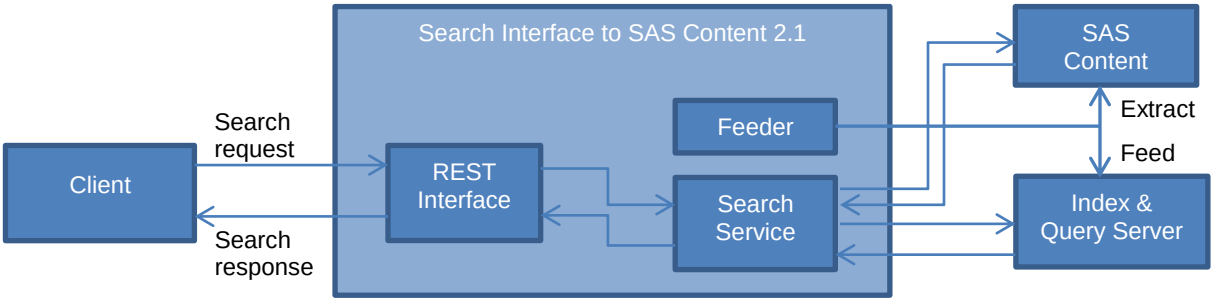

Figure 1. Search Interface to SAS Content Flow

As shown in the above diagram, the feeder extracts the SAS contents and feeds them into the index server for faster recall. Current version 2.1 supports search in the Google Search Appliance index (GSA), Teragram® index (SAS Information Retrieval Studio), and direct search (live) in the SAS Metadata Server.

Search service is exposed through the Representational State Transfer (REST) interface to the clients for easy integration, which is described in detail in this paper. To learn more about the product or back-end part of search, refer the SAS Global Forum paper 331-2010.

REST

To expose the search functionality as a service, REST is used. Since the REST interface works on HTTP and XML messages, the search service can be integrated with any client irrespective of language or platform. It can be integrated with a Java client on Linux as well as a Microsoft Office application on Windows. Unlike CORBA, it can work across firewalls. A downside of it can be the overhead of XML, which is data intensive if there are too many search results.

BUSINESS USE CASES

The integration of the search application with Microsoft Office or any other clients might be useful in situations such as the following:

- If someone is writing a document in Microsoft Word and wants to insert a reference to the relevant BI contents in it, but does not know the path and exact name of report, then he or she can run the predefined macro to open a dialog box to accept the search keyword and show the search results. The user can search the report using a keyword and can then press the insert button to insert a reference to a SAS report from the search result into the document.
- Suppose someone is writing an e-mail about his or her analysis of the business and wants to insert a report as a supporting artifact. He or she can invoke the search functionality from Microsoft Outlook to search a report using the search interface and insert a reference to it into the e-mail.
- While using any SAS BI client or solution, a user might need to refer to a SAS report or SAS BI dashboard. If enterprise search has been integrated, then a user can quickly search the required object and proceed with the work. In this use case, the other approach of using saspfs_sessionid and saspfs_sessionrequest is recommended because the client will have an existing connection to the mid-tier and can leverage the single sign-on using these session variables.

The following sections show how to integrate the product.

INTEGRATION WITH CLIENT APPLICATIONS

In order to integrate search service with your application, you will need the following information:

- How to connect?
- What input information is required?
- What is the output format?
- How to present the search result?

You will find the answers to these questions in the following sections.

INVOKING THE SEARCH SERVICE

Unlike an Internet search, the enterprise search application considers the security of the data and returns only those search results that should be accessible to the user. Therefore, every user that connects to the application must be authenticated first. There are two ways to access the search service. If your application is already connected to the mid-tier and has established a session with the application server, then the single sign-on feature will allow you to access the search service by using SAS PFS Session ID and Session Request. For most of the SAS BI Clients and solutions, this will be the way to access the search service.

But if you want to integrate the search service with third-party software that does not have a connection to the mid-tier, then the authentication of the user will be required in order to gain access to the system. SAS Authentication Service in Web Infrastructure Platform provides a common, standard way for mid-tier clients to authenticate users. This service is useful to do the following:

- support single sign-on across multiple Web applications

- extend that single sign-on to other clients
- simplify procedures that applications need to perform in order to authenticate
- localize actual authentication to a single Web application, which provides a common "front door" for all browser-based access to SAS applications

Initializing the Search Session

Before connecting to the search application, a new session will have to be created as part of the initialization process. To create a new session, make the following request to the authentication service. This request will provide a token that can be used as single key to access all Web applications or services.

URI: http://some-server.com:8080/SASWIPClientAccess/rest/sessions

Formats: XML (JSON)

Content-Type: text/xml; charset=utf-8

HTTP Method: POST

Request body:

```
<?xml version="1.0" encoding="UTF-8"?>
<credentials>
    <userid>your user ID</userid>
    <password>your password</password>
</credentials>
```

Response body:

```
<?xml version="1.0" encoding="UTF-8"?>
<session>
    <token>ZjUzOWVhZTg0ZGUxNTZiMjoxMjQzYzU1YzoxMmYyYmE4YjMzYzotNjg4Mg:qa208waiR3U0c
pqLzxIQXg:lkM6YKJvj4o4XboabSl1JA</token>
</session>
```

The user name and password will be sent in the request body as shown in the XML code above. The response body contains a token that is a combination of saspfs_sessionid, series data, and token data. This token can be extracted and set in the header while invoking the search service.

A token is always valid for one transaction only. With every response, a new token is returned in the response header sas_authentication_token that can be used in the subsequent transactions. If the HTTP session times out or the server restarts, then the sas_authentication_token cookie will allow the communication to be re-established without logging back in.

Making a Search Request

To start searching, the following request can be sent.

URI: http://some-server.com:8080/SASSearchService/rest/content/?q=sales

Authorization: "SASRest " + *auth_token_value*

HTTP Method: GET

Request body: Not required

Response body:

```
<?xml version="1.0" encoding="UTF-8"?>
<rss xmlns:atom="http://www.w3.org/2005/Atom" xmlns:opensearch="http://a9.com/-
/spec/opensearch/1.1/" xmlns:sas="http://www.sas.com/sassearch" version="2.0">
<channel>
<title>SAS Search Results</title>
<sas:facets>
    <!-- facet information --/>
```

```
    …
</sas:facets>
<!-- link information --/>
<item>
    <!-- item contents/>
</item>
…
…
</channel>
</rss>
```

In this search request, the parameter q=temp indicates that the search keyword is “temp.” The value *auth_token_value* that is passed in request header “Authorization” is the token retrieved from the response that is returned by the authentication service. This request with the GET method invokes the search service to search the keyword in the index and returns the search result in the response. This output can be parsed to get the list of matching objects.

Closing the Search Session

Whenever a user finishes searching, he or she can log off from the search service by deleting the session. To delete the session, make the following request to the authentication service.

URI: http://some-server.com:8080/SASWIPClientAccess/rest/sessions/"authentication-token"

Formats: XML (JSON)

Content-Type: text/xml; charset=utf-8

HTTP Method: DELETE

Request body: Not required

Response body:

The user receives one of the following codes:

- 204: indicates a successful request without any content
- 403: indicates that you are trying to destroy a session that does not belong to you
- 404: indicates no such session exists

INPUT PARAMETERS

Only the query parameter is required in which you pass the search keyword. Along with the query parameter, there are some optional request parameters that can be appended to the URL as listed below. They are useful for more control over a search function. The actual parameter name is given in parenthesis.

1. Session identifier (saspfs_sessionid): If your application is already connected to the mid-tier, then the existing session ID can be passed in this parameter. If it is provided, then a token is not required. If both are not provided, then the user will be redirected to the SAS Logon Manager for authentication.
2. Session request (saspfs_sessionrequest): Like saspfs_sessionid, this parameter can be provided in case of an existing connection. This parameter is required along with saspfs_sessionid. If it is missing, then the user will be redirected to SAS Logon Manager.
3. Format (format): This parameter decides the output format. The valid values are RSS and ATOM. This information can also be set into the request header as “Accept=application/rss+xml” or “Accept=application/atom+xml”. If the parameter is not provided, then RSS will be accepted as the default value.
4. Source index (sourceIndex): The search interface can search in live metadata as well as in the index provided by GSA and Teragram. While querying, this parameter can be used to specify where to search. The valid values are teragramIndex and gsaIndex. If the parameter is omitted, then it will search in the live metadata.
5. Language (lang): This parameter is useful if the feed is available in multiple languages. The default language is lang=en-us. It can also be passed through the request header with the name Accept-Language.
6. Start (start): In the case of pagination, the client can pass this parameter to tell the first record number for

the current page. The valid values for this parameter are any positive integer between 0 to the total number of search results. It can be set into the request header as well.

7. Page size (num): This parameter indicates the number of results to be shown on one page. By default, it is set to 10 results per page. It can be set into the request header as well.
8. Facet required (req): This parameter can also be passed in the request header. Categories can be specified along with values in this parameter, for example, req=charttypes:Bar_dataitems:REGION_dataitems:WCOST.
9. Facet exclude (exc): This parameter can be passed in the request header. Categories to be excluded can be specified in this parameter.
10. Start date (startDate): The search result can be filtered based on dates using this field. It works in conjunction with endDate parameter. An example value is startDate=20120422.
11. End date (endDate): This parameter works in conjunction with the startDate parameter. An example value is endDate =20120425. Only number values can be entered for these parameters. If there are invalid values, then the date range will be ignored and will show all the results.

Some of the parameters can also be passed in request headers. Please note that if some information is provided in both places, the parameter value overrides the header information.

OUTPUT FORMAT

The search interface supports two output formats RSS and ATOM. The required format can be set in the request parameter or header "format" as mentioned earlier. Both outputs are the standard feed format. Any format can be selected as per your preference. Since RSS is more popular, it is kept as the default value. The following is the sample output in the RSS format. In case of ATOM, some tags will change. For more information about the differences between RSS and ATOM, see http://www.intertwingly.net/wiki/pie/Rss20AndAtom10Compared#table.

```
<?xml version="1.0" encoding="UTF-8"?>
<rss xmlns:atom="http://www.w3.org/2005/Atom" xmlns:opensearch="http://a9.com/-
/spec/opensearch/1.1/" xmlns:sas="http://www.sas.com/sassearch" version="2.0">
   <channel>
     <title>SAS Search Results</title>
     <sas:facets>
       <sas:facet displayName="SAS Type" id="sastype">
           <sas:bucket value="Dashboard" count="4" />
           <sas:bucket value="InformationMap" count="4" />
           <sas:bucket value="StoredProcess" count="4" />
           <sas:bucket value="Report" count="2" />
       </sas:facet>
       …
       …
     </sas:facets>
<link>http://host:port/SASSearchService?forward=showresult&amp;openType=StoredProce
ss&amp;SBIPURL=%2FShared+Data%2FSASHELP%2FProduct+Sales+Listing</link>
     <item>
<url>http://host:port/SASSearchService?forward=showresult&amp;openType=StoredProces
s&amp;SBIPURL=%2FShared+Data%2FSASHELP%2FProduct+Sales+Listing</url>
     <rel>54.414078</rel>
     <cnt>1</cnt>
     <date>20120131</date>
     <bln></bln>
     <ccd>... &lt;b&gt;Product&lt;/b&gt; Sales Listing ... </ccd>
     <title>Product Sales Listing</title>
     <description></description>

<link>http://host:port/SASSearchService?forward=showresult&amp;openType=StoredProce
ss&amp;SBIPURL=%2FShared+Data%2FSASHELP%2FProduct+Sales+Listing</link>
     <sastype>StoredProcess</sastype>
     <sasmetatype>ClassifierMap</sasmetatype>
     <sasowner></sasowner>
     <sasid>A58MKDZ8.B800001U</sasid>
     <keywords></keywords>
```

```
      <promptlabels></promptlabels>
      <promptdesc></promptdesc>
      <promptkeywords></promptkeywords>
      <dataitems></dataitems>
      <charttypes></charttypes>
      <datalabels></datalabels>
      <commentsubject></commentsubject>
      <commentcontent></commentcontent>
      <authtype>1</authtype>
      <auth>ClassifierMap;A58MKDZ8.B800001U</auth>
<sasimageicon>http://host:port/SASSearchService/images/StoredProcess.gif</sasimagei
con>
      <metadatacreated>31Jan2012:04:35:18</metadatacreated>
      <metadataupdated>31Jan2012:04:35:18</metadataupdated>
      <saspath>/Shared Data/SASHELP/Product Sales Listing</saspath>
      <graphnames></graphnames>
      <graphtitles></graphtitles>
      <datasourcename></datasourcename>
      <datasets></datasets>
      <filters></filters>
      <loaddate>14Mar2012:17:04:36</loaddate>
      <pdate>20120131</pdate>
      </item>
      …
      …
</channel>
</rss>
```

Output 1. RSS Output Format

As shown in the above output XML, facet information is placed in the beginning inside the tag named <sas:facets>. Next is the pagination information in which the total number of search results, starting record number, and page size is given. The client application can display these numbers to the user for easy navigation through search results. Then, an item tag is provided for each search result with a title, description, link, and some more information.

Any XML parser can be used to read this XML output to navigate through these results and extract the required information.

PRESENTING THE SEARCH RESULTS

Once the output XML is parsed, items can be listed by showing their title with a link. A description can be shown if some more information is required. For better performance, pagination can be used to present fewer results at a time. This will be a basic requirement most of the time if we consider the use cases mentioned above. But if the user wants more analysis over search results, then facets can be useful in classifying the results based on some elements.

Faceted Search

It is the dynamic grouping of items or search results into categories that allows the user to drill into the search results by any value in any field. Each facet that is displayed also shows the number of results within the search that match that category. The user can also apply specific constraints (require, exclude, or view) to drill down into the search results. The following figure shows an example of a user interface that uses a faceted search. On the left hand side of the page, categories and the number of results that are classified into each category are visible. The user can filter the results that are visible on right side of the page based on these categories. An active filter is shown at the top.

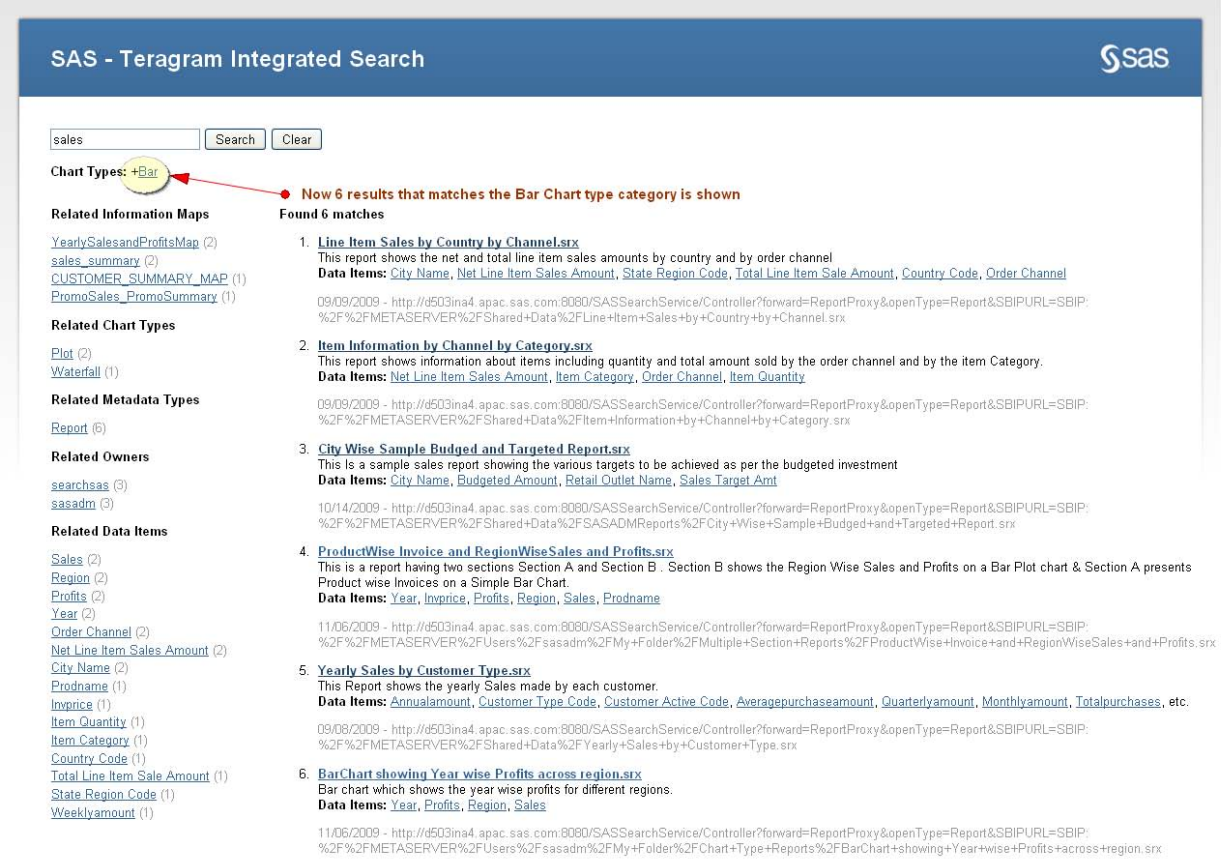

Figure 2. A User Interface That Uses a Faceted Search

CONCLUSION

Search Interface to SAS Content provides a REST interface for the enterprise search feature. The returned search result is also in the standard feed format. It can be used to integrate the enterprise search easily with any type of client. In the case of a third-party client like Microsoft Office, the token authentication method should be used as described above. The search service can be fully controlled through parameters. It also provides information about faceted search and pagination to design and develop rich user interfaces.

REFERENCES

SAS Institute Inc. 2011. *SAS 9.3 Intelligence Platform: Security Administration Guide*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/documentation/cdl/en/bisecag/63082/PDF/default/bisecag.pdf.

SAS Institute Inc. 2011. Search Interface to SAS Content documentation available at
http://support.sas.com/documentation/installcenter/
http://support.sas.com/demosdownloads/setupcat.jsp?cat=Base SAS Software.

RECOMMENDED READING

Rubendall, C., C. Prakash, and S. Prem, 2010. “Integrating the Power of SAS® with the Ease of Search across the Enterprise.” *Proceedings of the SAS Global Forum 2010 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings10/331-2010.pdf.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Arvind Jagtap
SAS Research & Development (India)
Level 2A & Level3, Cybercity, Tower 5, Magarpatta city, Hadapsar
Pune, Maharashtra India – 411013
Work Phone: +91 20 3041-8870
Fax: +91 20 3041-8899
E-mail: Arvind.Jagtap@sas.com

Kaushlendra Rai
SAS Research & Development (India)
Level 2A & Level3, Cybercity, Tower 5, Magarpatta city, Hadapsar
Pune, Maharashtra India – 411013
Work Phone: +91 20 3041-8870
Fax: +91 20 3041-8899
E-mail: Kaushlendra.Rai@sas.com

Prasad Kulkarni
SAS Research & Development (India)
Level 2A & Level3, Cybercity, Tower 5, Magarpatta city, Hadapsar
Pune, Maharashtra India – 411013
Work Phone: +91 20 3041-8870
Fax: +91 20 3041-8899
E-mail: Prasad.Kulkarni@sas.com

APPENDIX A: VBA CODE FOR MICROSOFT OFFICE ADD-IN

An add-in for Microsoft Office can be developed to invoke this search service in any Microsoft Office application like Microsoft Word, Microsoft Outlook, or Microsoft Excel. The following Microsoft Visual Basic for Applications (VBA) code can be referred to know the implementation details of the above mentioned approach. This code assumes that the user has entered a search keyword in cell B1 and it will display the search result in cell B2.

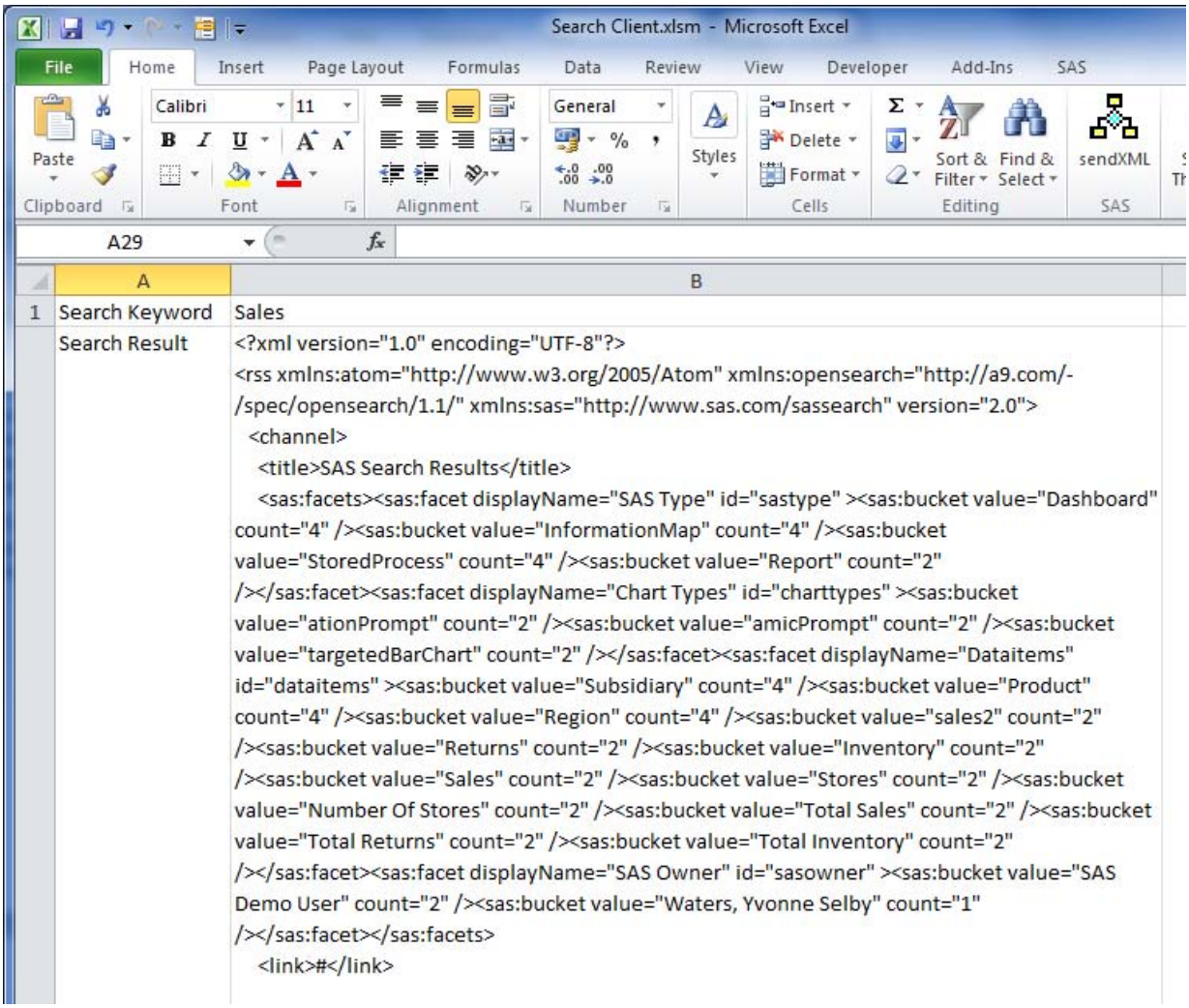

Figure 3. Spreadsheet of Search Keywords

The following code snippet in VBA can be stored in a macro or VBA module. If a shortcut key (for example, Ctrl+r) is assigned, then it can be executed quickly without opening any dialog box. It will execute and search for the entered keyword in the index and write the search result back to spreadsheet as shown in the above screen shot. Replace the <host>:<port> with your host name and port at two places. Also enter the respective user name and password in the XML for the specified server before executing the code.

To enhance this add-in further, a dialog box can be created to accept the search keyword and the output XML can be parsed to increase the readability and format the output. But the focus of this code is only on connecting to the service and fetching the search result.

```
Option Explicit
Dim m_strAuthToken As String

Public Sub invokeSearch()
    Dim strAuthURL As String, strSearchURL As String
    Dim strCredentials As String, strResponse As String
    Dim xmlhtp As New MSXML2.ServerXMLHTTP40

    ' Clear existing entries
    ActiveSheet.Cells(1, 1) = "Search Keyword"
    ActiveSheet.Cells(2, 1) = "Search Result"
    ActiveSheet.Cells(2, 2).Clear

    ' Web Service URL
    strAuthURL = "http://<host>:<port>/SASWIPClientAccess/rest/sessions"
    strSearchURL = "http://<host>:<port>/SASSearchService/rest/content/?q="

    ' Get the search keyword
    strSearchURL = strSearchURL & ActiveSheet.Cells(1, 2)

    If m_strAuthToken = "" Then
        ' XML with user name and password
        strCredentials = "<?xml version=""1.0"" encoding=""UTF-8""?>" & _
                "<credentials>" & _
                "<userid>sasadm@saspw</userid>" & _
                "<password>Password1</password>" & _
                "</credentials>"

        ' Post the xml to the service URL
        With xmlhtp
            .Open "post", strAuthURL, False
            .setRequestHeader "Content-Type", "text/xml; charset=utf-8"
            .send strCredentials

            strResponse = .responseText
        End With

        ' Parse the output XML to get the authentication token
        strResponse = Mid(strResponse, InStr(strResponse, "<token>") + 7)
        m_strAuthToken = Left(strResponse, InStr(strResponse, "</token>") - 1)

    End If


    ' Invoke the search service using token to fetch the search result
    With xmlhtp
        .Open "GET", strSearchURL, False
        .setRequestHeader "Authorization", "SASRest " + m_strAuthToken
        .send ""

        strResponse = .responseText
        m_strAuthToken = .getResponseHeader("sas_authentication_token")
    End With

    ' Write back the RSS response containing the search result
    ActiveSheet.Cells(2, 2) = strResponse

End Sub
```