Google Search Appliance

External Metadata Indexing Guide

Google Search Appliance software version 7.2 and later

Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043
www.google.com

GSA-META_200.01
March 2015

Contents

External Metadata Indexing Guide

This guide is for developers and administrators of the Google Search Appliance who have documents with metadata that is not stored directly in the primary document. A primary document is a record, file, or web page that the search appliance treats as a document to index or serve. The guide explains how to use the external metadata indexing capabilities of the search appliance, either through the use of the Feeds system or the Database Crawler. You should be familiar with the Feeds system and Document Crawler before you read this guide.

Introduction

The Google Search Appliance indexes metadata stored in documents and makes that data available for retrieval at search time. Metadata is data that describes other data. It can provide useful information that can improve the quality of your search results. For example, an HTML document can hold metadata in the `<meta>` tag to describe the author or keywords for the document. Similarly, Microsoft Office files such as Word documents or Excel spreadsheets often contain metadata fields, such as Title, Subject, Author, Date, and many others.

From the perspective of the search appliance, there are two primary types of metadata:

- Metadata that is stored directly in a primary document, as in the example of the HTML document with a `<meta>` tag.

  When the search appliance indexes a document, it automatically indexes the metadata that is stored directly in that document.

- Metadata that is not stored directly in a primary document.

  An example of this is metadata about a document that is stored in a column of a database table or metadata that is pushed in a separate feed.

You can configure the search appliance to index this external metadata and the primary document as a single record.

Because the search appliance automatically indexes metadata that is stored directly in a primary document, this guide describes how to index metadata that is not stored in the primary document. In this guide, **external metadata** refers to metadata that is not stored directly in the primary document. The **primary document** is defined as a record, web page, or any of the over 200 different file types that can be indexed by the search appliance and are acquired through the web crawler, database crawler, file system crawler, or feeder system.

When you index external metadata, it is searchable in the same way that other metadata is searchable. For example, you can use the `partialfields` and `requiredfields` query operators to search for documents with particular metadata. For more information about metadata queries and query operators, see the *Search Protocol Reference*.

Metadata Indexing Configurations

The search appliance has default settings for indexing metadata, including which metadata names are to be indexed, as well as how to handle multivalued metadata fields and date fields. For metadata that is stored in the primary document, you can configure the settings by using the **Index > Index Settings** page in the Admin Console.

The metadata indexing configuration options on this page are also applicable to external metadata. If you make any changes to metadata indexing configurations and the external metadata, you will need to submit the external metadata to the search appliance again.

For detailed information about metadata indexing configuration options, click **Admin Console Help > Index > Index Settings** in the Admin Console.

Implementing External Metadata Indexing

To implement external metadata indexing, you need to identify where your metadata is and describe to the search appliance how it relates to a primary document. Essentially, you need to answer the following questions:

- **Metadata:** Where is the external metadata?
- **Primary Document:** Where is the primary document, and how does the metadata refer to the primary document?

Your answers to these questions will determine which method of external metadata indexing you should use. The methods for external metadata indexing can be grouped into three main categories:

- "External Metadata Stored in a Database" on page 6

  If the metadata is stored in a relational database and references a primary document that is accessible by one of the search appliance's crawlers (web accessible URL, file system accessible URI, BLOB in a relational database), then you can configure the search appliance to automatically index the metadata and primary document, and associate them with one another as a single record in the index.

- "External Metadata Pushed in a Feed" on page 9

  If the metadata is not stored in a relational database or if the reference between the metadata and the primary document is not easily described, you can generate a feed and push it to the search appliance.

- "External Metadata Sent in an HTTP Header" on page 11

  If the metadata is not stored in a relational database or if the reference between the metadata and the primary document is not easily described, you can send the metadata in an HTTP header.

Within each of these categories, there are a variety of indexing scenarios. The scenario that you should use depends on how your primary document is referenced and stored.

External Metadata Stored in a Database

There are three scenarios for indexing external metadata that is stored in a database, depending on how your primary document is referenced and stored. For each of these scenarios, the search appliance indexes a `meta name` for each field in your crawl query and `meta content` for the value in that field.

If you want to use an alias for a field name, you can use the SQL keyword `AS` in the crawl query to give the field a more meaningful name. For example, if your database has an `auth` field, you might prefer to give the field an alias of `author`, because author is a more common search term. Your users could then search for this document by adding `&requiredfields=author` to their search query URL. Creating aliases for obscurely named fields is especially useful if you want to collect values for the `requiredfields` or `partialfields` parameters from your end users.

When using the SQL keyword `AS` to create an alias for a field, use the alias (instead of the original field name) for the following fields on the **Content Sources > Databases** page:

- Document URL Field
- Document ID Field
- Base URL
- Primary Key Fields
- BLOB MIME Type Field
- BLOB Content Field

Scenario 1

Metadata: Stored in a database.

Primary Document: A valid URL stored in a single field in the same database that references the primary document.

If your external metadata is stored in a relational database and one of the fields contains fully qualified URLs that reference primary documents, use the following steps to enable external metadata indexing:

1. In the **Content Sources > Databases** page, create a new database data source.
2. Enter the database properties (type, hostname, port, name, username, password) used to connect to the database that contains the external metadata.
3. Construct the **Crawl Query**, which is a valid SQL statement accepted by the target database that returns all rows of metadata to be indexed. One of the indexed fields must contain the full URI that references the primary document.
4. Under **Data Display / Usage**, select the **Metadata** option.
5. Select the **Document URL Field** option, and enter the name of the database column (field name) that holds valid URLs that reference the primary documents.

In this scenario, the search appliance queries the database for data, then submits a feed with the resulting rows. The search appliance crawls and indexes the set of records that is defined by the crawl query. The URLs extracted from each external metadata record (as defined by the URL field) are added to the crawl queue and crawled by either the web or file system crawler, following the normal crawl policy. When the primary document is crawled, the contents of the primary document and the external metadata are merged into a single record, which is identified by the URL of the primary document in the search appliance index. After the primary document and the external metadata are indexed, the primary document is returned as a search result when search users query for terms in the external metadata or the primary document.

When you re-index your primary document and external metadata, the replacement behavior for this scenario is the same as for a metadata-and-URL feed. For information about metadata-and-URL feeds, see the *Feeds Protocol Developer's Guide*.

Scenario 2

Metadata: Stored in a database.

Primary Document: A pointer to the primary document needs to be constructed from a base URL and a database value.

This scenario is very similar to the first one, except the URL is constructed from a base URL and a document ID.

If your external metadata is stored in a relational database and the URLs that reference primary documents can be constructed by combining a base URL string and a database field, use the following steps to enable external metadata indexing. The database field usually represents a unique document ID number that, when inserted into a base URL string, references a specific document on a web server or file system. For example, suppose that your primary documents are accessible from a URL of the following form:

```
http://cmsystem.acme.corp.com:6502/getdoc?action=get&docid=4662118437
```

In the example, the highlighted number represents a unique document ID stored as a field in the database. You can configure the search appliance to crawl the metadata and construct URLs that reference primary documents by inserting values from one of the database fields into the base URL.

1. In the **Content Sources > Databases** page, create a new database data source.
2. Enter the database properties (type, hostname, port, name, username, password) used to connect to the database that contains the external metadata.
3. Construct the **Crawl Query**, a valid SQL statement accepted by the target database that returns all rows of metadata to be indexed. One of the indexed fields must contain the unique identifier of the primary document to be inserted into a base URL string.
4. Under **Data Display / Usage**, select the **Metadata** option.
5. Select **Document ID Field**, and enter the database column that holds the unique value that is used to construct a primary document URL.
6. In the **Base URL** field, enter the base URL string that is used to construct URLs that reference primary documents. The value of the field that is specified in Document ID Field is inserted into the base URL string, as specified by the `{docid}` tag. If the highlighted document ID in the preceding example was stored in a field called `uniqueID`, the Document ID Field would be `uniqueID` and the base URL string would be: `http://cmsystem.acme.corp.com:6502/getdoc?action=get&id={docid}`

In this scenario, the search appliance queries the database for data, then submits a feed with the resulting rows. The search appliance extracts and indexes the recordset that is defined by the crawl query. The URLs constructed from each external metadata record (as defined in Document ID Field and the Base URL field) are added to the crawl queue and crawled by either the web or file system crawler, following the normal crawl policy. When the primary document is crawled, the contents of the primary document and the external metadata are merged into a single record identified by the URL of the primary document in the search appliance index. After the primary document and the external metadata are indexed, search users can query for terms or keywords in the metadata or the primary document and the primary document will return as a search result.

When you re-index your primary document and external metadata, the replacement behavior for this scenario is the same as for a metadata-and-URL feed. For information about metadata-and-URL feeds, see the *Feeds Protocol Developer's Guide*.

Scenario 3

Metadata: Stored in a database.

Primary Document: The primary document is also stored in the database, as a BLOB.

If your external metadata is stored in a relational database and the primary document is also stored in the database as a BLOB (Binary Large OBject), do the following:

1. In the **Content Sources > Databases** page, create a new database data source.
2. Enter the database properties (type, hostname, port, name, username, password) used to connect to the database that contains the external metadata.
3. Construct the **Crawl Query**, a valid SQL statement accepted by the target database that returns all rows of metadata to be indexed. One of the indexed fields must be the BLOB field that contains the primary document.
4. Under **Data Display / Usage**, select the **BLOB** option.
5. In **BLOB MIME Type Field**, type the database column name that specifies the standard Internet MIME type of the BLOB.
6. In **BLOB Content Field**, type the database column name that contains the BLOB data.
7. Construct a **Serve Query** as a valid SQL statement that is accepted by the target database. The database needs to return a single row of metadata and content to display as the result. Specify each primary key value with a closing question mark. For example, to select metadata for an employee ID and department for a serve query, enter the following statements:

```
SELECT employee_id, dept
FROM employee
WHERE employee_id = ? and dept = ?
```

8. In **Primary Key Fields**, enter the database columns that provide the single row of metadata that you want to serve as the search result. For example:

```
employee_id, dept
```

In Scenario 3, the search appliance queries the database for data, then submits a feed with the resulting rows. The search appliance crawls and indexes the set of records that is defined by the crawl query. The specified BLOBs are pushed in a full content feed and are not crawled.

When you re-index your primary document and external metadata, the replacement behavior for this scenario is the same as for a full feed. For information about full feeds, see the *Feeds Protocol Developer's Guide*.

External Metadata Pushed in a Feed

The remaining scenarios use feeds. Feeds work well when the external metadata is not stored in a relational database, the primary document is not accessible by the search appliance's crawlers, or the reference between the external metadata and the primary document is not easily expressed. You can use a feeds-based solution in any of these cases or any case where you prefer using feeds to implementing the database scenarios.

Feeds are described in general in the *Feeds Protocol Developer's Guide*. You should be familiar with those concepts before you index external metadata by using feeds.

There are two types of feeds:

- **Content feeds**, which include a URL and the contents of the URL (the document itself).
- **Web feeds**, which contain a list of URLs without their contents. The crawler queues the URLs and fetches the contents as normal.

The primary document can be pushed in the feed as a content feed or referenced as a web feed. The following scenarios detail how to index external metadata for primary documents that are pushed as content feeds or web feeds.

Note: Web feeds with a data source name of "web" and a feed type of "incremental" cannot contain external metadata. If external metadata is added to this type of web feed, an error message will display and the URL will not be crawled.

Scenario 4

Metadata: Inserted into the feed XML file.

Primary Document: Inserted into the feed XML file (content feed).

In this scenario, you need to write a script or code that generates the feed XML file and then push the feed XML file to the search appliance. Use the following steps:

1. Create the feed XML file and define the header information, including the data source name, as shown in the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE gsafeed PUBLIC "-//Google//DTD GSA Feeds//EN" "">
<gsafeed>
  <header>
    <datasource>sample</datasource>
    <feedtype>full</feedtype>
  </header>
```

2. Create a <record> element for each primary document. In the <metadata> element, insert one or more <meta> elements, as shown in the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE gsafeed PUBLIC "-//Google//DTD GSA Feeds//EN" "">
<gsafeed>
  <header>
    <datasource>sample</datasource>
    <feedtype>full</feedtype>
  </header>
  <group>
    <record url="http://www.corp.enterprise.com/hello01"
     mimetype="text/plain" last-modified="Tue, 17 Feb 2009 12:45:26 GMT">
      <metadata>
        <meta name="author" content="Jones"/>
        <meta name="project" content="hello01"/>
        <meta name="department" content="engineering"/>
      </metadata>
```

3. Insert the contents of the primary document into the <content> element of the record, as shown in the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE gsafeed PUBLIC "-//Google//DTD GSA Feeds//EN" "">
<gsafeed>
  <header>
    <datasource>sample</datasource>
    <feedtype>full</feedtype>
  </header>
  <group>
    <record url="http://www.corp.enterprise.com/hello01"
     mimetype="text/plain" last-modified="Tue, 17 Feb 2009 12:45:26 GMT">
      <metadata>
        <meta name="author" content="Jones"/>
        <meta name="project" content="hello01"/>
        <meta name="department" content="engineering"/>
      </metadata>
      <content> This is hello02 content. </content>
    </record>
  </group>
</gsafeed>
```

Note: The previous example uses a full feed (<feedtype>full</feedtype>). With full feeds, fed documents are removed from the index within six hours. For more information, see "Removing Feed Content From the Index" in the *Feeds Protocol Developer's Guide*. You can use an incremental feed to avoid fed documents being removed from the index by replacing the <feedtype>full element with the <feedtype>incremental</feedtype> element.

If the content is text-based content, it can be inserted directly into the feed XML file. If it is non-text content (.pdf, .doc, and other file types), you need to base64-encode the content and set the record's encoding attribute to encoding="base64binary", as described in the *Feeds Protocol Developer's Guide*.

You should be aware that when you update a content feed for a primary document, the search appliance does not automatically update the associated metadata feed in its index unless the corresponding record has a <metadata> section. Similarly, if you update a metadata feed, the search appliance does not automatically update the associated primary document feed in its index. If you want to update a content feed and a metadata feed, you should explicitly push both of these feeds to the index.

To update the metadata of a url using metadata-and-url feed, the search appliance must be able to crawl the url specified in the <record> tag. Therefore this process cannot be used for content that was indexed through a content feed (for example, by using the File System connector).

Scenario 5

Metadata: Inserted into the feed XML file.

Primary Document: Referenced by the URL in the feed XML file (web feed).

This scenario is similar to the previous scenario, except that the primary document is referenced by URL only (instead of the contents of the primary document being fed to the search appliance). The feed file therefore contains the `<header>` information and, for each `<record>` element, the URL of the record and the `<metadata>` elements.

1. Create the feed XML file and define the header information, including the data source name, as shown in the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE gsafeed PUBLIC "-//Google//DTD GSA Feeds//EN" "">
<gsafeed>
  <header>
    <datasource>sample2</datasource>
    <feedtype>metadata-and-url</feedtype>
  </header>
```

   Note that the `<feedtype>` element is `metadata-and-url`. This tells the web or file system crawler to pick up the URLs for the primary document and index them accordingly.

2. Create a `<record>` element for each primary document. In the `<metadata>` element, insert one or more `<meta>` elements, as shown in the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE gsafeed PUBLIC "-//Google//DTD GSA Feeds//EN" "">
<gsafeed>
  <header>
    <datasource>sample2</datasource>
    <feedtype>metadata-and-url</feedtype>
  </header>
  <group>
    <record url="http://www.corp.enterprise.com/hello02"
     mimetype="text/plain" last-modified="Tue, 17 Feb 2009 12:45:26 GMT">
      <metadata>
        <meta name="author" content="Stevens"/>
        <meta name="project" content="hello02"/>
        <meta name="department" content="HR"/>
      </metadata>
```

External Metadata Sent in an HTTP Header

At crawl time, the search appliance can accept external metadata, along with documents, through the `X-GSA-External-Metadata` HTTP response header. This is useful for indexing metadata for non-HTML documents, where it is not possible to include metadata. The metadata supplied at crawl time replaces any and all metadata that may have been indexed earlier.

To use this method of indexing external metadata, the web service that stores the content needs to be designed to generate the optional `X-GSA-External-Metadata` HTTP header. The header includes a comma separated list of encoded values, as specified in RFC2616 (http://www.w3.org/Protocols/rfc2616/rfc2616.html), Section 4.2:

```
X-GSA-External-Metadata: value_1, value_2,...
```

Each *value* has the form: *meta-name=meta-value*. Both the *meta-name* and the *meta-value* are encoded according to section 2 of RFC3986 (http://www.ietf.org/rfc/rfc3986.txt) (commonly known as percent-encoding). The search appliance does not transform '+' to space.

The following restrictions apply to this header:

- The meta-value cannot be empty.
- The values for the meta-name and meta-value cannot contain embedded quotation marks.

Document Access Control

The search appliance provides document-level access control to secure the search content. When you index external metadata, the search appliance applies the following access controls:

- The access control of the external metadata in "Scenario 1" on page 6 and "Scenario 2" on page 7 (where the metadata is stored in a database) is the same as the access control for the associated primary document.
- The access control of the external metadata in "Scenario 3" on page 8 (where the metadata and the primary document are stored in the database) is public, which means that it is available to all search users. Scenario 3 is a bad option for data that needs to be secured.
- The access control for external metadata in "Scenario 4" on page 9 and "Scenario 5" on page 11 (where the metadata is inserted into the feed XML file) is specified in the feed, just as with non-metadata feeds.