Google Search Appliance

Administrative API Developer's Guide: .NET

Google Search Appliance software version 7.2 and later

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

GSA-CSAPI_200.01
March 2015

Contents

Administrative API Developer's Guide: .NET 5
Introduction 5
Getting Started 6
Authenticating Your Google Search Appliance Account 7
Content Sources 7
Crawl URLs 8
Data Source Feeds 9
Crawl Schedule 12
Crawler Access Rules 12
Host Load Schedule 15
Freshness Tuning 16
Connector Managers 17
OneBox Modules Settings 18
OneBox Modules Retrieve and Delete 19
Pause or Resume Crawl 20
Document Status 21
Index 21
Collections 22
Index Diagnostics 24
Content Statistics 29
Reset Index 30
Search 31
Front Ends: Remove URLs and a Relative OneBox 31
Output Format XSLT Stylesheet 33
KeyMatch Settings 35
Related Queries 37
Query Suggestion 40
Search Status 41
Reports 41
Search Reports 42
Search Logs 45
GSA Unification 47
Configuring a GSA Unification Network 48
Adding a GSA Unification Node 49
Retrieving a Node Configuration 49
Retrieving All Node Configurations 49
Updating a Node Configuration 50
Deleting a Node 50

Administration 50
License Information 50
Import and Export 51
Event Log 52
System Status 53
Shutdown or Reboot 54

Index 55

Administrative API Developer's Guide: .NET

Introduction

This guide provides .NET programming information about how to use the Google Data API to create, retrieve, update, and delete information for one or more Google Search Appliance devices.

Use the information in this guide to create or learn about coding .NET applications that programmatically set the administrative functions for the Admin Console of a search appliance. This document uses C# examples, but you can use any .NET programming language with the .NET Google Data API client library.

The audience for this guide is .NET programmers who want to programmatically configure a Google Search Appliance. Before using the Google Search Appliance Administrative API, you need an Admin Console user name and password for the search appliance to which you direct your commands.

Before using the Google Search Appliance Administrative API, read "Getting Started" on page 6 to download and configure required software.

Note: This guide uses *data source feed* to indicate the search appliance's Feeds API (described in the *Feeds Protocol Developer's Guide*).

The organization of this guide corresponds to the navigation features of the Google Search Appliance:

- "Content Sources" on page 7
- "Index" on page 21
- "Search" on page 31
- "Reports" on page 41
- "GSA Unification" on page 47
- "Administration" on page 50

Getting Started

The google-enterprise-gdata-api open source site (https://code.google.com/p/google-enterprise-gdata-api) provides ZIP files that contain sample C#.NET example files, the .NET client library (DLLs), source code, and a sample application for your reference.

The information in this section helps you understand how to write your own applications based on the C#.NET client library and how to run the provided open source sample applications.

Before starting, you need the following:

- Microsoft Visual C# 2008 Express Edition (http://www.microsoft.com/exPress/download/#webInstall), which includes a free version of Visual Studio so that you can work with the .NET client library.
- The latest version of the API software, which contains the client library and the sample application files. The API software is available at http://code.google.com/p/google-enterprise-gdata-api/downloads/list.
- Admin Console user name and password for the search appliance to which you direct your commands.

Getting Samples

After you download the software and acquire search appliance credentials, get started as follows:

1. Unzip the API ZIP file and navigate to the `cs` folder. The client library DLL files are present in the `lib` folder and the sample application is present in the `sample` folder.
2. Start Microsoft Visual C# 2008 Express Edition and click **File** > **Open Project**, browse to the location where you stored the `gsa.sln` solution file, and open the solution file, which appears in the Solution Explorer.
3. Click **Build** > **Build Solution** to build the project. Ensure that the build runs without errors. The binaries and DLL files are put in the `cs\sample\bin \Release` folder. The output binary is the `GsaCommandLine.exe` executable file in the `Release` folder.
4. Open a command prompt and run the command to view its options:

```
C:\GoogleDataAdministrativeAPI\cs\sample\bin\Release>GsaCommandLine.exe
Usage: GsaCommandLine <command> <options> <query_parameters> feed entry
commands:
    retrieve
    update
    insert
    delete
options:
  --protocol:
  --hostname:
  --port:
  --username:
  --password:
  --input: The input entry file for insert/update
<query_parameters>:
  All the query parameters can be specified by --<query>=<value>
Example:
  GsaCommandLine retrieve --protocol=http --hostname=gsa1 --port=8000
    --username=user --password=password config crawlURLs
C:\GoogleDataAdministrativeAPI\API-Gdata\cs\sample\bin\Release>
```

You can run a command using the parameters that are shown. Be sure for the host name to use just the search appliance name and not a URL.

Building Your Applications

This section explains how to build your own applications using the client library outside the solution file provided by the ZIP archive.

To build an application:

1. Copy the following client library DLL files from the `cs\lib` folder to your development folder and add them in the reference path:
   - `Google.GData.Apps.dll`
   - `Google.GData.Client.dll`
   - `Google.GData.Extensions.dll`
   - `Google.GData.Gsa.dll`
2. In Visual Studio, create or open a new project.
3. Click **Project** > **Add Reference**.
4. Click the **Browse** tab and navigate to the folder where you copied the DLL files.
5. Select the DLLs to use in your project.

Authenticating Your Google Search Appliance Account

Before making API calls with the .NET client library, you must construct a new `GsaService` object.

In the constructor that follows, replace *`myUserId`* and *`myPassword`* with your Admin Console authentication information:

```
using Google.GData.Gsa;
GsaService myService = new GsaService(gsaAddr, gsaPort, "myUserId",
  "myPassword"));
```

Content Sources

The sections that follow describe how to configure the **Content Sources** features of the Admin Console:

- "Crawl URLs" on page 8
- "Data Source Feeds" on page 9
- "Crawl Schedule" on page 12
- "Crawler Access Rules" on page 12
- "Host Load Schedule" on page 15
- "Freshness Tuning" on page 16
- "Connector Managers" on page 17

- "OneBox Modules Settings" on page 18
- "OneBox Modules Retrieve and Delete" on page 19
- "Pause or Resume Crawl" on page 20
- "Document Status" on page 21

Crawl URLs

Retrieve and update crawl URL patterns on a search appliance using the `crawlURLs` entry of the `config` feed.

Property	Description
`doNotCrawlURLs`	Do Not Crawl URLs with the following patterns, separate multiple URL patterns with new line delimiters.
`followURLs`	Follow and crawl only URLs with the following URL patterns, separate multiple URL patterns with new line delimiters.
`startURLs`	Start crawling from the following URLs, separate multiple URL patterns with new line delimiters.

Retrieving Crawl URLs

Retrieve information about the URL patterns that the search appliance is crawling as follows:

```
// Send a request and print the response
GsaEntry myEntry = myService.GetEntry("config", "crawlURLs");
Console.WriteLine("Start URLs: " + myEntry.GetGsaContent("startURLs"));
Console.WriteLine("Follow URLs: " + myEntry.GetGsaContent("followURLs"));
Console.WriteLine("Do Not Crawl URLs: " + myEntry.GetGsaContent
("doNotCrawlURLs"));
```

Updating Crawl URLs

Update the crawl URL settings on a search appliance as follows—in the example that follows, `example.com` is requested for crawling, and spreadsheets are requested to not be crawled.

```
// Create an entry to hold properties to update
GsaEntry updateEntry = new GsaEntry();

// Add a property for adding crawl URLs to updateEntry
updateEntry.AddGsaContent("startURLs", "http://www.example.com/");
updateEntry.AddGsaContent("followURLs", "http://www.example.com/");
updateEntry.AddGsaContent("doNotCrawlURLs", ".xls$");

// Send the request
myService.UpdateEntry("config", "crawlURLs", updateEntry);
```

Data Source Feeds

Retrieve, delete, and destroy data source feed information for the search appliance using the `feed` feed. The following parameters let you search for a string and retrieve source statements.

Parameter	Description
`query`	The query string. When used to retrieve all feed information, the query parameter is overloaded to mean the `feedDataStore`. When getting information about a single feed, the parameter is a query. Each log statement contains a query string to be retrieved.
`startLine`	The starting log statement to retrieve. The default value is line `1`.
`maxLines`	The maximum number of log statements to retrieve. The default value is `50` statements.

Use the following properties to view data source feed records and content.

Property	Description
`errorRecords`	The number of documents that had errors and were not added to the data source feed.
`feedDataSource`	The name of the data source.
`feedState`	Feed state: `ACCEPTED:0, IN_PROGRESS:1, COMPLETED:2, COMPLETED_WITH_ERROR:3, FAILED_IN_ERROR:4`
`feedTime`	The system's time stamp at the start of each stage, in milliseconds.
`feedType`	Feed type: `FULL_FEED:0, INCREMENTAL:1, DELETED:2, METADATA_AND_URL:3`
`fromLine`	The starting line of a log.
`logContent`	The log's content.
`successRecords`	The number of documents in the search appliance index (the number of documents that were successfully indexed).
`toLine`	The ending line of a log.
`totalLines`	Total lines in a log.
`updateMethod`	Indicate to delete a data source feed. This value can only be `delete`.

Note: You can only get information about data source feeds, and delete or destroy data source feeds. Inserting new data source feeds is not provided by this API. You can create new feeds using the Admin Console user interface.

Retrieving Data Source Feed Information

Retrieve all data source feed information from a search appliance using the `feed` feed:

```
// Send a request and print the response
Dictionary<string, string> queries = new Dictionary<string, string>();
queries.Add("query",feedDataSource);
GsaFeed myFeed = myService.QueryFeed("feed", queries);

foreach(GsaEntry myEntry in myFeed.Entries) {
  //get information on each myEntry
  Console.WriteLine("Feed Name: " + myEntry.GetGsaContent("entryID"));
  Console.WriteLine("Feed Data Source: " + myEntry.GetGsaContent
    ("feedDataSource"));
  Console.WriteLine("Feed Type: " + myEntry.GetGsaContent("feedType"));
  Console.WriteLine("Feed State: " + myEntry.GetGsaContent("feedState"));
  Console.WriteLine("Feed Time: " + myEntry.GetGsaContent("feedTime"));
  Console.WriteLine("Error Records: " + myEntry.GetGsaContent("errorRecords"));
  Console.WriteLine("Success Records: " + myEntry.GetGsaContent
    ("successRecords"));
  Console.WriteLine("Log Content: " + myEntry.GetGsaContent("logContent"));
}
```

You can get the individual feed information from a search appliance as follows:

```
GsaEntry myEntry = myService.GetEntry("feed", FEED_NAME);
Console.WriteLine("Feed Data Source: " + myEntry.GetGsaContent
("feedDataSource"));
Console.WriteLine("Feed Type: " + myEntry.GetGsaContent("feedType"));
Console.WriteLine("Feed State: " + myEntry.GetGsaContent("feedState"));
Console.WriteLine("Feed Time: " + myEntry.GetGsaContent("feedTime"));
Console.WriteLine("Error Records: " + myEntry.GetGsaContent("errorRecords"));
Console.WriteLine("Success Records: " + myEntry.GetGsaContent("successRecords"));
Console.WriteLine("Log Content: " + myEntry.GetGsaContent("logContent"));
```

Note: A feed log of all data source feeds can be retrieved only by getting individual feeds.

Deleting Data Source Feeds

Delete a data source feed to remove all documents for a feed from the index on the search appliance. In the Admin Console, deleted feed sources are marked with `Delete`.

Delete data source feeds as follows:

```
// Create an entry to hold properties to update
GsaEntry updateEntry = new GsaEntry();

// Add a property to updateEntry
updateEntry.AddGsaContent("updateMethod", "delete");
myService.UpdateEntry("feed", FEED_NAME, updateEntry);
```

Note: Delete data source feeds only of the type `FULL_FEED` or `incremental`. After deleting a data source feed, the deleted feed still exists, and the feed type changes to `DELETED`.

Destroying Data Source Feeds

After deleting a data source feed, you can destroy the feed so that the feed no longer exists on the search appliance:

```
myService.DeleteEntry("feed", FEED_NAME);
```

Trusted Feed IP Addresses

Retrieve and update trusted feed IP addresses using the `feedTrustedIP` entry of the `config` feed.

Retrieve the IP addresses of trusted feeds using the `trustedIPs` property.

Property	Description
`trustedIPs`	Trusted IP addresses. This value is a list of one or more IP addresses. Specify `all` to indicate that the search appliance trust all IP addresses. If the value is a list of IP addresses, separate each IP address with white space.

Retrieving Trusted Feed IP Addresses

Retrieve the trusted feed IP addresses as follows:

```
// Send the request and print the response
GsaEntry myEntry = myService.GetEntry("config ", "feedTrustedIP");
Console.WriteLine("Trusted IP Addresses: " +
  myEntry.GetGsaContent("trustedIPs"));
```

Updating Trusted Feed IP Addresses

Update trusted feed IP addresses as follows:

```
// Create an entry to hold properties to update
GsaEntry updateEntry = new GsaEntry();

// Add a property for the feeds trusted IP addresses
// to updateEntry
updateEntry.AddGsaContent("trustedIPs", "127.0.0.1");

// Send the request
myService.UpdateEntry("config", "feedTrustedIP", updateEntry);
```

Crawl Schedule

Retrieve and update the crawl schedule for a search appliance.

Property	Description
`crawlSchedule`	The crawl schedule is only available in scheduled crawl mode. The value of `crawlSchedule` has the format: *`Day`*`,`*`Time`*`,`*`Duration`* Where: • *`Day`* is a number representing the days of a week: `0` means Sunday and `1` means Monday. • *`Time`* is a 24-hour representation of time. The time pertains to the search appliance and not the computer running the application to set the value. • *`Duration`* is the representation for the time period in minutes. The duration cannot be greater than `1440`, which means 24 hours. A scheduled crawl begins on the values in *`Day`* and *`Time`* and continues for the *`Duration`*.
`isScheduledCrawl`	Set to `1` if the search appliance is in scheduled crawl mode or set to `0` if the search appliance is in continuous crawl mode.

Retrieving the Crawl Schedule

Retrieve the crawl mode and get the crawl schedule as follows:

```
GsaEntry myEntry = myService.GetEntry("config", "crawlSchedule");
Console.WriteLine("Is Scheduled Crawl: " + myEntry.GetGsaContent
  ("isScheduledCrawl"));
Console.WriteLine("Crawl Schedule: " + myEntry.GetGsaContent("crawlSchedule"));
```

Updating the Crawl Schedule

Update the crawl schedule or crawl mode as follows:

```
// Create an entry to hold properties to update
GsaEntry updateEntry = new GsaEntry();

// Add a property to updateEntry
updateEntry.AddGsaContent("isScheduledCrawl", "1");
updateEntry.AddGsaContent("crawlSchedule", "0,0300,360\n2,0000,1200");

// Send the request
myService.UpdateEntry("config", "crawlSchedule", updateEntry);
```

Crawler Access Rules

Create, retrieve, update, and delete crawler access rules for a search appliance.

Crawler access rules instruct the crawler how to authenticate when crawling the protected content.

Property	Description
`domain`	Windows domain for NTLM, or empty for HTTP Basic authorization.
`isPublic`	Indicates whether to allow users to view results of both the public content (normally available to everyone) and the secure (confidential) content. The value can be `1` to enable users to view content as public, or `0` to require users to authenticate to view secure content.
`order`	Indicates that the crawler access rules are sequential. The order indicates the sequence. The order is an integer value starting from `1`.
`password`	Password for authentication.
`urlPattern`	URL pattern that matches the protected files.
`username`	User name for authentication.

Inserting a Crawler Access Rule

Insert a new crawler access rule as follows:

```
// Create an entry to hold properties to insert
GsaEntry insertEntry = new GsaEntry();

// Add properties to insertEntry
insertEntry.AddGsaContent("entryID", "#URL pattern for the new crawler access
  rule");
insertEntry.AddGsaContent("domain", "domainone");
insertEntry.AddGsaContent("isPublic", "1");
insertEntry.AddGsaContent("username", "username");
insertEntry.AddGsaContent("password", "password");

// Send the request
myService.InsertEntry("crawlAccessNTLM", insertEntry);
```

Retrieving Crawler Access Rules

Retrieve a list of crawler access rules as follows:

```
// Send a request and print the response
GsaFeed myFeed = myService.GetFeed("crawlAccessNTLM");

foreach(GsaEntry myEntry in myFeed.Entries) {
  Console.WriteLine("URL Pattern: " + myEntry.GetGsaContent("urlPattern"));
  Console.WriteLine("User Name: " + myEntry.GetGsaContent("username"));
  Console.WriteLine("Order: " + myEntry.GetGsaContent("order"));
  Console.WriteLine("Domain: " + myEntry.GetGsaContent("domain"));
  Console.WriteLine("Is Public: " + myEntry.GetGsaContent("isPublic"));
}
```

Retrieve an individual crawler access rule as follows:

```
// Send the request and print the response
GsaEntry myEntry = myService.GetEntry("crawlAccessNTLM", "urlPattern");
Console.WriteLine("URL Pattern: " + myEntry.GetGsaContent("urlPattern"));
Console.WriteLine("User Name: " + myEntry.GetGsaContent("username"));
Console.WriteLine("Order: " + myEntry.GetGsaContent("order"));
Console.WriteLine("Domain: " + myEntry.GetGsaContent("domain"));
Console.WriteLine("Is Public: " + myEntry.GetGsaContent("isPublic"));
```

Note: The `password` property doesn't appear when retrieving crawler access rules.

Updating a Crawler Access Rule

Update a crawler access rule as follows:

```
// Create an entry to hold properties to update
GsaEntry updateEntry = new GsaEntry();

// Add properties to updateEntry
updateEntry.AddGsaContent("urlPattern", "#new URL pattern");
updateEntry.AddGsaContent("domain", "newdomain");
updateEntry.AddGsaContent("isPublic", "0");
updateEntry.AddGsaContent("order", "2");
updateEntry.AddGsaContent("username", "newuser");
updateEntry.AddGsaContent("password", "newpass");

// Send the request
myService.UpdateEntry("crawlAccessNTLM", "urlPattern", updateEntry);
```

Deleting a Crawler Access Rule

Delete a crawler access rule as follows:

```
myService.DeleteEntry("crawlAccessNTLM", "urlPattern");
```

Host Load Schedule

Retrieve and update host load schedule information from the search appliance using the `hostLoad` entry of the `config` feed.

Property	Description
`defaultHostLoad`	The default web server host load, a float value. This value measures the relative load on the search appliance based on the number of connections that a search appliance can handle. You can set the `defaultHostLoad` to a decimal value starting at `0`, where `0` indicates to not crawl the content from the number of URL patterns that you specify. Any decimal value of `1` or more sets the average number of connections per minute. A decimal value under `1` sets the percentage of time during which the search appliance opens connections. For more information, see the online help for **Content Sources > Web Crawl > Host Load Schedule**.
`exceptionHostLoad`	Exceptions to the default web server host load are listed as multiple lines of text where each line is in the format: *`hostName startTime endTime loadFactor`* Where: • *`hostName`* is a URL or asterisk (`*`) to represents all hosts. If a *`hostName`* line contains multiple load data values, separate the host line into multiple lines with each line containing one load data value, without overlap. • *`startTime`* and *`endTime`* are integer value between `0` and `23` (`0` = 12 midnight, `23` = 11 pm). • *`loadFactor`* is a float value. See `defaultHostLoad` for an explanation of the `loadFactor` value.
`maxURLs`	Maximum number of URLs to crawl, an integer value.

Retrieving the Host Load Schedule

Retrieve information about the host load schedule from a search appliance as follows:

```
// Send the request and print the response
GsaEntry myEntry = myService.GetEntry("config", "hostLoad");
Console.WriteLine("URL Pattern: " + myEntry.GetGsaContent("defaultHostLoad"));
Console.WriteLine("User Name: " + myEntry.GetGsaContent("exceptionHostLoad"));
Console.WriteLine("Order: " + myEntry.GetGsaContent("maxURLs"));
```

Updating the Host Load Schedule

Update the host load schedule setting in a search appliance as follows:

```
// Create an entry to hold properties to update
GsaEntry updateEntry = new GsaEntry();

// Add a property for the Host Load Schedule to updateEntry
updateEntry.AddGsaContent("defaultHostLoad", "2.4");
updateEntry.AddGsaContent("exceptionHostLoad",
  "* 3 5 1.2 \n www.example.com 1 6 3.6");
updateEntry.AddGsaContent("maxURLs", "3000");

// Send the request
myService.UpdateEntry("config", "hostLoad", updateEntry);
```

Freshness Tuning

Increase or decrease the crawling frequency by specifying URL patterns.

Property	Description
`archiveURLs`	URL patterns for pages that contain archival or rarely changing content.
`forceURLs`	URL patterns for pages to recrawl regardless of their response to `If-Modified-Since` request headers.
`frequentURLs`	URL patterns for pages that change often (typically more than once a day).

Retrieving the Freshness Configuration

Retrieve the freshness tuning configuration as follows:

```
GsaEntry myEntry = myService.GetEntry("config", "freshness");
Console.WriteLine("Archive URLs: " + myEntry.GetGsaContent("archiveURLs"));
Console.WriteLine("Frequent URLs: " + myEntry.GetGsaContent("frequentURLs"));
Console.WriteLine("Force URLs: " + myEntry.GetGsaContent("forceURLs"));
```

Updating the Freshness Configuration

Update the settings for freshness tuning as follows:

```
// Create an entry to hold properties to update
GsaEntry updateEntry = new GsaEntry();

// Add a property for updateEntry
updateEntry.AddGsaContent("archiveURLs", "http://good/");
updateEntry.AddGsaContent("frequentURLs", "http://frequent/");
updateEntry.AddGsaContent("forceURLs", "http://force/");

// Send the request
myService.UpdateEntry("config", "freshness", updateEntry);
```

Recrawling URL Patterns

If you discover that a set of URLs that you want to have in the search index are not being crawled you can inject a URL pattern into the queue of URLs that the search appliance is crawling. URLs may not appear in the index because changes were made to the web pages, or because a temporary error or misconfiguration was present when the crawler last tried to crawl the URL.

Property	Description
`recrawlURLs`	URL patterns to be recrawled.

Recrawling URL Patterns Example

Recrawl URL patterns as follows:

```
// Create an entry to hold properties to update
GsaEntry updateEntry = new GsaEntry();

// Add a property to updateEntry
updateEntry.AddGsaContent("recrawlURLs", "http://recrawl/page.html");

// Send the request
myService.UpdateEntry("command", "recrawlNow", updateEntry);
```

Connector Managers

Add, retrieve, update, and delete a connector manager associated with a search appliance.

Property	Description
`description`	A description of the connector manager.
`url`	The URL of the application server where the connector manager is installed.
`status`	The status of the connection between the search appliance and the connector manager that is deployed on an application server. The value can be `Connected` or `Disconnected`. The `Disconnected` value may occur if the application server is down or there are problems on the network.

Adding a Connector Manager

Add a connector manager to a search appliance as follows:

```
// Create an entry to hold properties to insert
GsaEntry insertEntry = new GsaEntry();

// Add property for insertEntry
insertEntry.AddGsaContent("entryID", "ConnectorManagerOne");
insertEntry.AddGsaContent("description", "Connector Manager One Description");
insertEntry.AddGsaContent("url", "http://example.com:port/");

// Send the request
myService.InsertEntry("connectorManager", insertEntry);
```

Retrieving a List of Connector Managers

Retrieve a list of connector managers as follows:

```
// Send the request and print the response
GsaFeed myFeed = myService.GetFeed("connectorManager");

foreach(GsaEntry myEntry in myFeed.Entries) {
  Console.WriteLine("Status: " + myEntry.GetGsaContent("status"));
  Console.WriteLine("Description: " + myEntry.GetGsaContent("description"));
  Console.WriteLine("URL: " + myEntry.GetGsaContent("url"));
}
```

Retrieve an individual connector manager as follows:

```
// Send the request and print the response
GsaEntry myEntry = myService.GetEntry("connectorManager", "ConnectorManagerOne");
Console.WriteLine("Status: " + myEntry.GetGsaContent("status"));
Console.WriteLine("Description: " + myEntry.GetGsaContent("description"));
Console.WriteLine("URL: " + myEntry.GetGsaContent("url"));
```

Updating a Connector Manager

Update the description and URL for a connector manager as follows:

```
// Create an entry to hold the properties to update
GsaEntry updateEntry = new GsaEntry();

// Add properties to updateEntry
updateEntry.AddGsaContent("description", "new description");
updateEntry.AddGsaContent("url", "#new URL");

// Send the request
myService.UpdateEntry("connectorManager", "ConnectorManagerOne", updateEntry);
```

Deleting a Connector Manager

Delete a connector manager as follows:

```
myService.DeleteEntry("connectorManager", "ConnectorManagerOne");
```

OneBox Modules Settings

Retrieve and update OneBox module settings for the search appliance using the `config` feed.

Retrieving OneBox Module Settings

Retrieve OneBox information for a search appliance as follows:

```
// Send the request and print the response
GsaEntry myEntry = myService.GetEntry("config", "oneboxSetting");
Console.WriteLine("Max Results: " + myEntry.GetGsaContent("maxResults"));
Console.WriteLine("Timeout: " + myEntry.GetGsaContent("timeout"));
```

The properties for retrieving a OneBox are as follows:

Property	Description
`maxResults`	Maximum number of results.
`timeout`	OneBox response timeout in milliseconds.

Updating OneBox Module Settings

Update the OneBox settings for a search appliance as follows—in this example three results are requested and the timeout is set to 2000 milliseconds.

```
// Create an entry to hold properties to update
GsaEntry updateEntry = new GsaEntry();

// Add properties for the OneBox settings to updateEntry
updateEntry.AddGsaContent("maxResults", "3");
updateEntry.AddGsaContent("timeout", "2000");

// Send the request
myService.UpdateEntry("config", "oneboxSetting", updateEntry);
```

OneBox Modules Retrieve and Delete

Retrieve and delete OneBox modules from the search appliance using the `onebox` feed.

Property	Description
`logContent`	The logs content of OneBox logs.

Note: Inserting a new OneBox module, updating an existing OneBox module, and retrieving a detailed configuration of a OneBox module are not supported by this API.

Retrieving OneBox Module Information

Retrieve information about all the OneBox modules from a search appliance using the `onebox` feed:

```
// Send the request and print the response
GsaFeed myFeed = myService.GetFeed("onebox");

foreach(GsaEntry myEntry in myFeed.Entries) {
  // Get information on each myEntry
  Console.WriteLine("OneBox Name: " + myEntry.GetGsaContent("entryID"));
}
```

Note: Because detailed information about the OneBox configuration not supported by this API, the `onebox` feed provides only OneBox module names.

Retrieve an individual OneBox module's log information from a search appliance as follows:

```
// Send the request and print the response
GsaEntry myEntry = myService.GetEntry("onebox",ONEBOX_NAME);
Console.WriteLine("OneBox Log: " + myEntry.GetGsaContent("logContent"));
```

Note: You can only retrieve OneBox log entries individually.

Deleting a OneBox Module

Delete a OneBox module from a search appliance as follows:

```
myService.DeleteEntry("onebox", ONEBOX_NAME);
```

Pause or Resume Crawl

Check crawl status, pause the crawl, or resume the crawl.

Property	Description
`pauseCrawl`	Indicates: `1` if crawl is paused, `0` if crawling is occurring.

Retrieving Crawl Status

Retrieve the status of crawl as follows:

```
// Send the request and print the response
GsaEntry myEntry = myService.GetEntry("command", "pauseCrawl");
Console.WriteLine("Pause Crawl: " + myEntry.GetGsaContent("pauseCrawl"));
```

Pausing or Resuming Crawl

Pause or resume a crawl as follows:

```
// Create an entry to hold properties to update
GsaEntry updateEntry = new GsaEntry();

// Add properties to updateEntry
updateEntry.AddGsaContent("pauseCrawl", "0");
updateEntry.AddGsaContent("doNotCrawlURLs", "http://frequent/");

// Send the request
myService.UpdateEntry("command", "pauseCrawl", updateEntry);
```

Document Status

Retrieve document status using the properties that follow.

Property	Description
`crawledURLsToday`	The number of documents crawled since yesterday. (Note that the time pertains to the search appliance, not the computer sending this request.)
`crawlPagePerSecond`	Current crawling rate.
`errorURLsToday`	The document errors since yesterday.
`filteredBytes`	The document bytes that have been filtered.
`foundURLs`	The number of URLs found that match crawl patterns.
`servedURLs`	The total number of documents that have been served.

Retrieving Document Status

Retrieve the document status as follows:

```
GsaEntry myEntry = myService.GetEntry("status", "documentStatus");
Console.WriteLine("Served URLs: " + myEntry.GetGsaContent("servedURLs"));
Console.WriteLine("Crawled Pages Per Second: " + myEntry.GetGsaContent
  ("crawlPagePerSecond"));
Console.WriteLine("Crawled URLs Today: " + myEntry.GetGsaContent
  ("crawledURLsToday"));
Console.WriteLine("Found URLs: " + myEntry.GetGsaContent("foundURLs"));
Console.WriteLine("Filtered Bytes: " + myEntry.GetGsaContent("filteredBytes"));
Console.WriteLine("Error URLs Today: " + myEntry.GetGsaContent
  ("errorURLsToday"));
```

Index

The sections that follow describe how to configure the **Index** features of the Admin Console:

- “Collections” on page 22
- “Index Diagnostics” on page 24
- “Content Statistics” on page 29
- “Reset Index” on page 30

Collections

Retrieve, update, create, or delete the collections of documents on the search appliance.

Property	Description
`collectionName`	The name of the collection to create, which is only required when creating a new collection.
`doNotCrawlURLs`	The URL patterns of content that you want to exclude from this collection.
`followURLs`	The URL patterns of content that you want to include in this collection.
`importData`	Indicates that the collection settings exported from the Admin Console are only required when creating a new collection from an import.
`insertMethod`	The method of creating a new method, which is only required when creating a new collection. Possible values: `default`, `customize`, `import`.

Creating a Collection

Create a new collection as follows:

```
// Create an entry to hold properties to insert
GsaEntry insertEntry = new GsaEntry();

// Add a property to insertEntry
insertEntry.AddGsaContent("collectionName", "new_collection");
insertEntry.AddGsaContent("insertMethod", "default");

// Send the request
myService.InsertEntry("collection", insertEntry);
```

Create a new collection with a default setting as follows:

```
insertEntry.AddGsaContent("insertMethod", "default");
```

Specify the settings for a new collection as follows:

```
// Add property for insertEntry
insertEntry.AddGsaContent("collectionName", "new_collection");
insertEntry.AddGsaContent("insertMethod", "customize");
insertEntry.AddGsaContent("followURLs", "#url to follow");
insertEntry.AddGsaContent("doNotCrawlURLs", "#url to not follow");
```

Retrieving All Collections

Retrieve a list of collections as follows:

```
// Send the request and print the response
GsaFeed myFeed = myService.GetFeed("collection");

foreach(GsaEntry myEntry in myFeed.Entries) {
  Console.WriteLine("Collection Name: " + myEntry.GetGsaContent("entryID"));
  Console.WriteLine("Follow URLs: " + myEntry.GetGsaContent("followURLs"));
  Console.WriteLine("Do Not Crawl URLs: " + myEntry.GetGsaContent
("doNotCrawlURLs"));
}
```

Retrieving a Collection

Retrieve the attributes of a single collection as follows:

```
// Send the request and print the response
GsaEntry myEntry = myService.GetEntry("frontend", "default_collection");
Console.WriteLine("Follow URLs: " + myEntry.GetGsaContent("followURLs"));
Console.WriteLine("Do Not Crawl URLs: " + myEntry.GetGsaContent
  ("doNotCrawlURLs"));
```

Updating a Collection

Update the attributes of a collection as follows:

```
// Create an entry to hold properties to update
GsaEntry updateEntry = new GsaEntry();

// Add properties to updateEntry
updateEntry.AddGsaContent("followURLs", "http://good/");
updateEntry.AddGsaContent("doNotCrawlURLs", "http://frequent/");

// Send the request
myService.UpdateEntry("collection", "default_collection", updateEntry);
```

Deleting a Collection

Delete a collection as follows:

```
myService.DeleteEntry("collection", "new_collection");
```

Index Diagnostics

List crawled documents and retrieve the status of documents in a search appliance using the `diagnostics` feed.

Document Status Values

The following tables list the document status values.

Note: Use the `all` to indicate any status value.

Successful Crawl:

Success Value	Crawl Description
1	Crawled from remote server
2	Crawled from cache

Crawl Errors:

Errors	Retrieval Error
7	Redirect without a location header
11	Document not found (404)
12	Other HTTP 400 errors
14	HTTP 0 error
15	Permanent DNS failure
16	Empty document
17	Image conversion failed
22	Authentication failed
25	Conversion error
32	HTTP 500 error
33	The robots.txt file is unreachable
35	Temporary DNS failure
36	Connection failed
37	Connection timeout
38	Connection closed
40	Connection refused
41	Connection reset
43	No route to host
50	Other error

Crawl Exclusions:

Excluded	Description
3	Not in the URLs to crawl
4	In the URLs to not crawl
5	Off domain redirect
6	Long redirect chain
8	Infinite URL space
9	Unhandled protocol
10	URL is too long
13	The robots.txt file indicates to not index
18	Rejected by rewrite rules
19	Unknown extension
20	Disallowed by a meta tag
24	Disallowed by the robots.txt file
26	Unhandled content type
27	No filter for this content type
34	`robots.txt` forbidden

Listing Documents

Query parameters:

Value	Description
`collectionName`	Name of a collection that you want to list. The default value is the last used collection.
`flatList`	Indicates: `false`: (Default) List the files and directories specified by the URL. `true`: List all files specified by a URL as a flat list.
`negativeState`	Indicates: `false`: (Default) Return documents with a status equal to `view`. `true`: Return documents with a status equal to `view`.
`pageNum`	The page you want to view. The files from a URL may be separated into several pages to return. Page numbers start from 1. The default value is `1`.
`sort`	The key field on which to sort: • `""`: The default value. • `crawled`: Sort by the number or crawled documents. • `errors`: Sort by the number of errors. • `excluded`: Sort by the number of excluded documents. • `file`: sort by file name. • `host`: Sort by host name.
`uriAt`	The prefix of URL of a document that you want to list. If not blank, it must contain at least `http://hostname.domain.com/`. The default value is `""`.
`view`	The filter of document status. The values of `view` are described in “Document Status Values” on page 24. The default value is `all`.

List documents by sending an authenticated `GET` request to the `root` entry of the `diagnostics` feed.

A `description` entry, a set of document status entries and a set of directory status entries are returned.

Description entry properties:

Property	Description
Entry Name	`description`
`numPages`	The total number of pages to return.
`uriAt`	URL prefix taken from the query parameters.

Directory status entry properties:

Property	Description
Entry Name	The URL of the directory.
`numCrawledURLs`	The number of crawled documents in this directory,
`numExcludedURLs`	The number of excluded URLs in this directory.
`numRetrievalErrors`	The number of retrieval error documents in this directory.
`type`	`DirectoryContentData` or `HostContentData`.

Document status entry properties:

Property	Description
Entry Name	The URL of the document.
`docState`	The status of this document. See "Document Status Values" on page 24 for possible `docState` values.
`isCookieServerError`	Indicates if a cookie server error occurred.
`timeStamp`	The last time the search appliance processed this document.
`type`	`FileContentData`

Example:

```
Dictionary<string, string> queries = new Dictionary<string, string>();
queries.Add("uriAt","http://server.com/secured/test1/");
GsaFeed myFeed = myService.QueryFeed("diagnostics", queries);

foreach(GsaEntry entry in myFeed.Entries) {
    Console.WriteLine(entry.GetGsaContent("entryID"));
  if (entry.GetGsaContent("entryID").Equals("description")) {
    Console.WriteLine("Number of Pages: " + entry.GetGsaContent("numPages"));
    Console.WriteLine("URI At: " + entry.GetGsaContent("uriAt"));
  } else if (entry.GetGsaContent("type").Equals("DirectoryContentData" )||
      entry.GetGsaContent("type").Equals("HostContentData")) {
      Console.WriteLine("Type: " + entry.GetGsaContent("type"));
      Console.WriteLine("Number of Crawled URLs: " +
        entry.GetGsaContent("numCrawledURLs"));
      Console.WriteLine("Number of Retrieval Errors: " +
        entry.GetGsaContent("numRetrievalErrors"));
      Console.WriteLine("Number of Excluded URLs: " +
        entry.GetGsaContent("numExcludedURLs"));
  } else if (entry.GetGsaContent("type").Equals("FileContentData")) {
      Console.WriteLine("Type: " + entry.GetGsaContent("type"));
      Console.WriteLine("Time Stamp: " + entry.GetGsaContent("timeStamp"));
      Console.WriteLine("Document State: " + entry.GetGsaContent("docState"));
      Console.WriteLine("Is Cookie Server Error: " +
        entry.GetGsaContent("isCookieServerError"));
  }
}
```

Viewing Index Diagnostics for a Document

Retrieve detailed information about a document by sending an authenticated `GET` request to a document status entry of the `diagnostics` feed. The parameter is as follows.

Parameter	Description
`collectionName`	Name of the collection for which you want to view crawl diagnostics.

A detailed document status entry is returned.

Detailed document status entry properties:

Property	Description
Entry Name	The URL of the document.
`backwardLinks`	The number of backward links to this document.
`collectionList`	A list of collections that contain this document.
`contentSize`	The size of the document content.
`contentType`	The type of the document.
`crawlFrequency`	The frequency at which the document is being scheduled to crawl, with possible values of `seldom`, `normal`, and `frequent`.
`crawlHistory`	A multi-line history of the document crawl including the timestamp when the document was crawled, the document status code and description in the following format: *`timestamp status_code status_description`* *`timestamp status_code status_description`* For status code values, see "Document Status Values" on page 24.
`currentlyInflight`	If the document is currently in process.
`date`	The date of this document.
`forwardLinks`	The number of forward links for this document.
`isCached`	Indicates if the cached page for this document is ready.
`lastModifiedDate`	The last modified date of this document.
`latestOnDisk`	The timestamp of the version being served.

```
GsaEntry entry = myService.GetEntry("diagnostics",
  "http://server.com/secured/test1/doc_0_2.html");
Console.WriteLine("Collection List: " + entry.GetGsaContent("collectionList"));
Console.WriteLine("Forward Links: " + entry.GetGsaContent("forwardLinks"));
Console.WriteLine("Backward Links: " + entry.GetGsaContent("backwardLinks"));
Console.WriteLine("Is Cached: " + entry.GetGsaContent("isCached"));
Console.WriteLine("Document Date: " + entry.GetGsaContent("date"));
Console.WriteLine("Last Modified Date: " + entry.GetGsaContent
  ("lastModifiedDate"));
Console.WriteLine("Latest Serving Version Timestamp: " +
  entry.getGsaContent("latestOnDisk"));
Console.WriteLine("Currently In Process: " + entry.getGsaContent
  ("currentlyInflight"));
Console.WriteLine("Content Size: " + entry.getGsaContent("contentSize"));
Console.WriteLine("Content Type: " + entry.getGsaContent("contentType"));
Console.WriteLine("Crawl Frequency: " + entry.getGsaContent("crawlFrequency"));
Console.WriteLine("Crawl History: " + entry.getGsaContent("crawlHistory"));
```

Content Statistics

Retrieve content statistics for each kind of document using the `contentStatistics` feed.

Common query parameters for all requests:

Parameter	Description
`collectionName`	Name of the collection for which you want to view content statistics.

Content statistics entry properties:

Property	Description
`avgSize`	The average document size for this content type.
Entry Name	The MIME type of the documents, such as, `plain/text`.
`maxSize`	The maximum document size for the crawled files with this MIME type.
`minSize`	The minimum document size for the crawled files with this MIME type.
`numFiles`	The total number of crawled files for this MIME type.
`totalSize`	The total size of all crawled files for this MIME type.

Retrieving Content Statistics For All Crawled Files

Retrieve content statistics for all crawled files in a search appliance by sending an authenticated `GET` request to the `root` entry of the `contentStatistics` feed.

A list of content statistics entries are returned.

```
GsaFeed myFeed = myService.GetFeed("contentStatistics");
foreach(GsaEntry entry in myFeed.Entries) {
  Console.WriteLine("Entry Name: " + entry.GetGsaContent("entryID"));
  Console.WriteLine("Maximum Size: " + entry.GetGsaContent("maxSize"));
  Console.WriteLine("Minimum Size: " + entry.GetGsaContent("minSize"));
  Console.WriteLine("Total Size: " + entry.GetGsaContent("totalSize"));
  Console.WriteLine("Average Size: " + entry.GetGsaContent("avgSize"));
  Console.WriteLine("Number of Files: " + entry.GetGsaContent("numFiles"));
}
```

Retrieving Content Statistics For a Crawled File

Retrieve content statistics for a single crawled file by sending an authenticated `GET` request to a content statistics entry of the `contentStatistics` feed.

The following content statistics for a crawled file are returned:

```
GsaEntry entry = myService.GetEntry("contentStatistics", "text/html");
Console.WriteLine("Maximum Size: " + entry.GetGsaContent("maxSize"));
Console.WriteLine("Minimum Size: " + entry.GetGsaContent("minSize"));
Console.WriteLine("Total Size: " + entry.GetGsaContent("totalSize"));
Console.WriteLine("Average Size: " + entry.GetGsaContent("avgSize"));
Console.WriteLine("Number of Files: " + entry.GetGsaContent("numFiles"));
```

Reset Index

Reset the index for a search appliance using the following properties.

WARNING: Resetting an index deletes all the documents in the index. Depending on the number of documents to crawl, crawling an index can take many days to complete.

Property	Description
`resetIndex`	1 if index is reset, 0 if index is not reset.
`resetStatusCode`	Status code for resetting index.
`resetStatusMessage`	Status message: `ERROR`, `PROGRESS`, `READY`.

Retrieving Status of a Reset Index

Retrieve the status of a reset index as follows:

```
// Send the request and print the response
GsaEntry myEntry = myService.GetEntry("command", "resetIndex");
Console.WriteLine("Reset Index: " + myEntry.GetGsaContent("resetIndex"));
Console.WriteLine("Reset Status Code: " + myEntry.GetGsaContent
  ("resetStatusCode"));
Console.WriteLine("Reset Status Message: " + myEntry.GetGsaContent
  ("resetStatusMessage"));
```

Resetting the Index

Reset the index as follows:

```
// Create an entry to hold properties to update
GsaEntry updateEntry = new GsaEntry();

// Add a property to updateEntry
updateEntry.AddGsaContent("resetIndex", "1");
myService.UpdateEntry("command", "resetIndex", updateEntry);
```

Search

The sections that follow describe how to configure the **Search** features of the Admin Console:

- "Front Ends: Remove URLs and a Relative OneBox" on page 31
- "Output Format XSLT Stylesheet" on page 33
- "KeyMatch Settings" on page 35
- "Related Queries" on page 37
- "Query Suggestion" on page 40
- "Search Status" on page 41

Front Ends: Remove URLs and a Relative OneBox

Retrieve, update, insert, or delete front ends to remove URLs or a relative OneBox for the search appliance using the `frontend` feed. Retrieve a front end using the following properties.

Property	Description
`frontendOnebox`	OneBox Modules that are relative to this front end. This value is a comma-separated list of OneBox names. The OneBox modules are triggered for this front end in the order that you specify.
`removeUrls`	Remove URLs that are relative to this front end. These are URL patterns that you do not want to appear in the search results for this front end.

Retrieving Front Ends, Remove URLs, and a Relative OneBox

Retrieve all the front end information for a search appliance as follows:

```
// Send a request and print the response
GsaFeed myFeed = myService.GetFeed("frontend");

foreach(GsaEntry myEntry in myFeed.Entries) {
  //get information for each myEntry
}
```

Get information about a front end as follows:

```
// Send a request and print the response
GsaEntry myEntry = myService.GetEntry("frontend", FRONTEND_NAME);
Console.WriteLine("Front End OneBox: " + myEntry.GetGsaContent
  ("frontendOnebox"));
Console.WriteLine("Remove URLs: " + myEntry.GetGsaContent("removeUrls"));
```

Updating Remove URLs and a Relative OneBox

Update the URLs to remove from the search results, and update a OneBox module in a front end as follows:

```
// Create an entry to hold properties to update
GsaEntry updateEntry = new GsaEntry();

// Add properties to updateEntry
updateEntry.AddGsaContent("frontendOnebox", "oneboxtwo");
updateEntry.AddGsaContent("removeUrls", "http://www.example.com/");

// Send the request
myService.UpdateEntry("frontend", FRONTEND_NAME, updateEntry);
```

Inserting Front Ends and Remove URLs

Insert a front end and remove a URL from the search results as follows:

```
// Create an entry to hold properties to insert
GsaEntry insertEntry = new GsaEntry();

// Add properties to insertEntry
insertEntry.AddGsaContent("entryID", FRONTEND_NAME);
insertEntry.AddGsaContent("removeUrls", "http://www.example3.com/");

// Send the request
myService.InsertEntry("frontend", insertEntry);
```

Deleting a Front End

Delete a front end from the search appliance as follows:

```
myService.DeleteEntry("frontend", FRONTEND_NAME);
```

Output Format XSLT Stylesheet

Retrieve and update the XSLT template and other output format-related properties for each language of each front end using the `frontend` entry of the `outputFormat` feed.

Parameter	Description
`language`	Specify a language for the output format properties that you want to retrieve. Each front end can contain multiple languages, and each language has its own output format properties. Each front end + language can have its own XSLT stylesheet. The `language` parameter enables you to retrieve and update a stylesheet for a front end associated with a language. Administrators who use the Admin Console set the language in their browser and the Admin Console then displays in that language (if the Admin Console has been translated into that language). Hence the `language` parameter for the `outputFormat` feed is limited to the values to which the Admin Console is translated.

Use the following properties to access the XSLT template information.

Property	Description
`isDefaultLanguage`	Set to `1` if the designated language is the default language for the specified front end, set to `0` if not.
`isStyleSheetEdited`	Set to `0` if the style sheet is the default stylesheet that has not been previously edited. Set to `1` if the style sheet has been edited.
`language`	When retrieving, the `language` is determined by the language that is specified by the `query` parameter. When updating, the `language` is passed as an entry property to specify the language of the output stylesheet.
`restoreDefaultFormat`	Set to `1` to restore custom-edited XSLT code back to the default values. A `0` value has no effect.
`styleSheetContent`	The output format XSLT code content.

Note: For the update action, the `restoreDefaultFormat` content is mutually exclusive from the `styleSheetContent`. For each update action, you should either restore the output format XSLT stylesheet back to the original default values, set the XSLT stylesheet to a custom format, or neither, but not both.

Retrieving the Output Format XSLT Stylesheet

Retrieve the output format stylesheet information from a search appliance as follows:

```
Dictionary<string, string> queryMap = new Dictionary<string, string>();

// Initialize the query map
queryMap.Add("language", "en");
GsaEntry myEntry = myService.QueryEntry("outputFormat", "default_frontend",
  queryMap);
Console.WriteLine("Language: " + myEntry.GetGsaContent("language"));
Console.WriteLine("Default Language: " + myEntry.GetGsaContent
  ("isDefaultLanguage"));
Console.WriteLine("Is the Style Sheet Edited: " + myEntry.GetGsaContent
  ("isStyleSheetEdited"));
Console.WriteLine("XSLT Stylesheet Content: " + myEntry.GetGsaContent
  ("styleSheetContent"));
```

Updating the Output Format XSLT Stylesheet

Update the output format stylesheet information in a search appliance as follows:

```
// Create an entry to hold properties to update
GsaEntry updateEntry = new GsaEntry();

// The language parameter is passed as part of
// the entry because we cannot use a query parameter
updateEntry.AddGsaContent("language", "en");

// Indicate that the XSLT stylesheet has default values
updateEntry.AddGsaContent("isDefaultLanguage, "1");

// Add this line to update the style sheet content
updateEntry.AddGsaContent("styleSheetContent", "{my new style sheet XSLT}");

// Or add this line to restore the stylesheet content to
// the default, which is mutually exclusive from the previous line
updateEntry.AddGsaContent("restoreDefaultFormat", "1");

// Send the request and print the response
myService.UpdateEntry("outputFormat", "default_frontend", updateEntry);
Console.WriteLine("Output Format: " + updateEntry.GetGsaContent("outputFormat"));
Console.WriteLine("Default Front End: " + updateEntry.GetGsaContent
  ("default_frontend"));
```

KeyMatch Settings

KeyMatch settings let you promote specific web pages on your site. The following parameters let you find KeyMatches by search, and specify a starting line number and the number of lines to access.

Parameter	Description
`query`	A query string to perform a full text search. For example, if you set `computer` in the `query` parameter, then you get all KeyMatch settings that contain the word `computer`.
`startLine`	The starting line number of a result, default value is 0.
`maxLines`	The number of result lines in a response, default value is 50 lines.

Use the following properties to set KeyMatch configurations.

Property	Description
`line_number`	The *`line_number`* of the KeyMatch configuration rule.
`newLines`	The new KeyMatch configuration to update. This value may include multiple KeyMatch statements. The line delimiter is `\n`.
`numLines`	The number of total result lines.
`originalLines`	The original KeyMatch configurations to change. The value may include multiple KeyMatch statements. The line delimiter is `\n`.
`startLine`	The starting line number of a KeyMatch configuration to change. The minimum value is 0.
`updateMethod`	The method to change a KeyMatch configuration. Possible values are: • `update`. Updates part of the KeyMatch configuration table to the new configurations. Delete KeyMatch configurations using the `update` method. See the example that follows. • `append`. Adds a new KeyMatch configuration to the end of the KeyMatch configuration table. • `replace`. Deletes all rules in the KeyMatch configuration table and then appends the new rules that you provide.

Note: The format for a KeyMatch configuration rule is as follows:

`Search_Term``,`*`KeyMatch_Type`*`,`*`URL`*`,`*`Title`*

`KeyMatch_Type` can be: `KeywordMatch`, `PhraseMatch`, or `ExactMatch`.

`Search_Term` and *`URL`* fields cannot be empty. The KeyMatch configuration conforms to the CSV format, which uses commas to separate values.

Retrieving KeyMatch Settings

Retrieve KeyMatch settings as follows:

```
Dictionary<string, string> queryMap = new Dictionary<string, string>();

// Initialize the query map
queryMap.Add("query", "myQuery");
queryMap.Add("startLine", "0");
queryMap.Add("maxLines", "50");

// Send the request and print the response
GsaEntry myEntry = myService.QueryEntry("keymatch", "myFrontend", queryMap);
foreach(KeyValuePair<string, string> kvp in myEntry.GetAllGsaContents()) {
  if (Regex.IsMatch(kvp.Key, @"^\d+$")) {
    Console.WriteLine("The lines for " + kvp.Key + " are: " + kvp.Value);
  }
}
Console.WriteLine("The number of lines are: " + myEntry.GetGsaContent
("numLines"));
```

Changing KeyMatch Settings

The following example appends KeyMatch settings:

```
// Create an entry to hold properties to append
GsaEntry appendEntry = new GsaEntry();
appendEntry.AddGsaContent("updateMethod", "append");

// Prepare new content
string newLines =
  "image,KeywordMatch,http://images.google.com/,Google Image Search\n" +
  "video,KeywordMatch,http://www.youtube.com/,Youtube\n" +
  "rss feed,PhraseMatch,http://www.google.com/reader,Reader";
appendEntry.AddGsaContent("newLines", newLines);

// Send the request to the search appliance
myService.UpdateEntry("keymatch", "myFrontend", appendEntry);
```

The following example updates KeyMatch settings:

```
// Create an entry to hold properties to update
GsaEntry updateEntry = new GsaEntry();
updateEntry.AddGsaContent("updateMethod", "update");

// Set the start line number
updateEntry.AddGsaContent("startLine", 0);

// Provide the original content
string originalLines =
  "image,KeywordMatch,http://images.google.com/,Google Image Search\n" +
  "video,KeywordMatch,http://www.youtube.com/,Youtube\n" +
  "rss feed,PhraseMatch,http://www.google.com/reader,Reader";
updateEntry.AddGsaContent("originalLines", originalLines);

// Prepare new content
string newLines =
  ",,,\n" +
  "video,KeywordMatch,http://video.google.com/,Video Search\n" +
  "rss feed,PhraseMatch,http://www.example.com/,RSS example";
updateEntry.AddGsaContent("newLines", newLines);

// Send the request to the search appliance
myService.UpdateEntry("keymatch", "myFrontend", updateEntry);
```

Note: Delete a setting by changing the statement to three commas (, , ,).

The following example replaces KeyMatch settings:

```
// Create an entry to hold properties to replace
GsaEntry replaceEntry = new GsaEntry();
replaceEntry.AddGsaContent("updateMethod", "replace");

// Prepare new content
string newLines =
  "image,KeywordMatch,http://images.google.com/,Google Image Search\n" +
  "video,KeywordMatch,http://www.youtube.com/,Youtube\n" +
  "rss feed,PhraseMatch,http://www.google.com/reader,Reader";
replaceEntry.AddGsaContent("newLines", newLines);

// Send the request to the search appliance
myService.UpdateEntry("keymatch", "myFrontend", replaceEntry);
```

Related Queries

Use related queries to associate alternative words or phrases with specified search terms. Related queries are also known as synonyms. Use the following parameters to search for a query and access lines from a starting line number and for a maximum amount of lines.

Parameter	Description
`query`	A query string to perform a full-text search.
`startLine`	The starting line number of the result, the default value is to start at line 0.
`maxLines`	The number of result lines in the response, the default value is 50 lines.

Use the following properties to access related queries.

Property	Description
line number	The *line number* of the related query configuration rule (in all the rules).
`newLines`	The new related query configuration to add. This value may include multiple lines of related query statements. The delimiter is `\n`.
`numLines`	The total number of result lines.
`originalLines`	The original related query configuration to change. This value may include multiple lines of related query statements. The delimiter is `\n`.
`startLine`	The starting line number of the related query configuration to change. The minimum value is `0`.
`updateMethod`	The method to use to change related query configurations. Possible values are: • `update`. Updates part of the related query configuration table to the new configuration. Deletes related query configurations using the `update` method. See the example that follows. • `append`. Adds a new related query configuration to the end of the synonym configuration table. • `replace`. Deletes all rules in the related query configuration table and then appends the new rules that you provide.

Note: A related query configuration rule is in the following format:

```
Search_Terms,Related_Queries
```

The *`Search_Terms`* and the *`Related_Queries`* values cannot be empty. The related query configuration rules conform to the CSV format, which uses commas to separate values.

Retrieving Related Queries

Retrieve related queries as follows:

```
Dictionary<string, string> queryMap = new Dictionary<string, string>();

// Initialize the query map
queryMap.Add("query", "myQuery");
queryMap.Add("startLine", "0");
queryMap.Add("maxLines", "50");

// Send the request and print the response
GsaEntry myEntry = myService.QueryEntry("synonym", "myFrontend", queryMap);

foreach(KeyValuePair<string, string> kvp in myEntry.GetAllGsaContents()) {
  if (Regex.IsMatch(kvp.Key, @"^\d+$")) {
    Console.WriteLine("The line " + kvp.Key + " is: " + kvp.Value);
  }
}
```

Changing Related Queries

The following example appends related queries:

```
// Create an entry to hold properties to append
GsaEntry appendEntry = new GsaEntry();
appendEntry.AddGsaContent("updateMethod", "append");

// Prepare new content
string newLines = "airplane,aircraft\n" + "google,googol\n" + "stock,security";
appendEntry.AddGsaContent("newLines", newLines);

// Send the request to the search appliance
myService.UpdateEntry("synonym", "myFrontend", appendEntry);
```

The following example updates related queries:

```
// Create an entry to hold properties to update
GsaEntry updateEntry = new GsaEntry();
updateEntry.AddGsaContent("updateMethod", "update");

// Set the starting line number
updateEntry.AddGsaContent("startLine", 0);

// Provide the original content
string originalLines = "airplane,aircraft\ngoogle,googol";
updateEntry.AddGsaContent("originalLines", originalLines);

// Prepare new content
string newLines = "airplane,helicopter\n,";
updateEntry.AddGsaContent("newLines", newLines);

// Send the request to the search appliance
myService.UpdateEntry("synonym", "myFrontend", updateEntry);
```

Note: Delete a setting by changing the statement to a comma (,) value.

The following example replaces related queries:

```
// Create an entry to hold properties to replace
GsaEntry replaceEntry = new GsaEntry();
replaceEntry.AddGsaContent("updateMethod", "replace");

// Prepare new content
string newLines = "airplane,aircraft\n" + "google,googol\n" + "stock,security";
replaceEntry.AddGsaContent("newLines", newLines);

// Send the request to the search appliance
myService.UpdateEntry("synonym", "myFrontend", replaceEntry);
```

Query Suggestion

There are two features for working with query suggestions:

- "Query Suggestion Blacklist" on page 40
- "Query Suggestion Refresh" on page 41

Query Suggestion Blacklist

The query suggestion blacklist supports the /suggest feature described in the "Query Suggestion Service /suggest Protocol" chapter of the *Search Protocol Reference*. This feature uses the `suggest` feed to retrieve and update the query suggestion blacklist entries.

Property	Description
`suggestBlacklist`	Content of the suggest blacklist file.

The query suggestion blacklist supports the regular expressions in the re2 library (http://code.google.com/p/re2/wiki/Syntax). If you want specify an exact match, you need to use the following syntax:

```
^the_word_to_match$
```

Retrieving Query Suggestion Blacklist Information

Retrieve query suggestion blacklist information as follows:

```
// Create a GsaClient
GsaClient client = new GsaClient("SearchAppliance", 8000, "username",
  "password");
// Get and print the current content of the blacklist file
GsaEntry entry = client.GetEntry("suggest", "suggestBlacklist");
Console.WriteLine("Current content: " + entry.GetGsaContent("suggestBlacklist"));
```

Updating Query Suggestion Blacklist Entries

Update query suggestion blacklist entries as follows:

```
// Update the content
entry = new GsaEntry();
entry.AddGsaContent ("suggestBlacklist",
  "bad_word_3\n^bad_word_1$\ncar[0-9]{4}.*\n");
client.UpdateEntry("suggest", "suggestBlacklist", entry);
```

Query Suggestion Refresh

The query suggestion refresh supports the /suggest feature described in the "Query Suggestion Service /suggest Protocol" chapter of the *Search Protocol Reference*. This feature uses the `suggest` feed to refresh the query suggestion database.

Property	Description
`suggestRefresh`	Triggers a query suggestion refresh.

Refresh query suggestions as follows:

```
GsaClient client = new GsaClient("SearchAppliance", 8000, "username",
  "password");
entry = new GsaEntry();
entry.AddGsaContent ("suggestRefresh", "1");
client.UpdateEntry("suggest", "suggestRefresh", entry);
```

Search Status

Retrieve the search status for the search appliance using the `servingStatus` entry of the `status` feed.

Property	Description
`queriesPerMinute`	Average queries per minute served recently on the search appliance.
`searchLatency`	Recent search latency in seconds.

Retrieving Search Status

Retrieve the current search appliance serving status as follows:

```
GsaEntry myEntry = myService.GetEntry("status", "servingStatus");
Console.WriteLine("Queries Per Minute: " + myEntry.GetGsaContent
  ("queriesPerMinute"));
```

Reports

The sections that follow describe how to configure the **Reports** features of the Admin Console:

- "Search Reports" on page 42
- "Search Logs" on page 45

Search Reports

Generate, update, and delete search reports using the `searchReport` feed.

Search report entry properties:

Property	Description
`collectionName`	(Write only) The collection name—only use to create a search report.
`diagnosticTerms`	Terms to exclude when running scripts that create diagnostic data from test queries. All the specified terms in a search query are removed from the report. Use commas to separate multiple terms.
Entry Name	*`Search_Report_Name`*@*`Collection_Name`*
`isFinal`	(Read only) Indicates if a search report contains a final result. If so, it means the last update date is later than the `reportDate`.
`reportContent`	(Read only) The search report content. Only use for requests to get search report content when the content is ready.
`reportCreationDate`	(Read only) The creation date of a search report.
`reportDate`	The dates of each query that is collected in the search report.
`reportName`	(Write only) The report name—only use to create a search report.
`reportState`	(Read only) The status of a search report: • `0`: The search report is initializing. • `1`: The search report is generating. • `2`: The search report is complete. • `3`: A non-final complete report is generating. • `4`: The last report generation failed.
`topCount`	The number of top queries to generate.
`withResults`	Indicates if a query should only count searches that have results. The default value is `false`.

Listing a Search Report

List search report entries by sending an authenticated `GET` request to the `root` entry of the `searchReport` feed. Query parameter:

Parameter	Description
`collectionName`	Collection Name of search report. The default value is `all.collections`.

A list of search report entries returns:

```
GsaFeed myFeed = myService.GetFeed("searchReport");
foreach(GsaEntry entry in myFeed.Entries) {
  Console.WriteLine("Entry Name: " + entry.GetGsaContent("entryID"));
  Console.WriteLine("Report State: " + entry.GetGsaContent("reportState"));
  Console.WriteLine("Report Creation Date: " +
    entry.GetGsaContent("reportCreationDate"));
  Console.WriteLine("Report Date: " + entry.GetGsaContent("reportDate"));
  Console.WriteLine("Is Final: " + entry.GetGsaContent("isFinal"));
  Console.WriteLine("With Results: " + entry.GetGsaContent("withResults"));
  Console.WriteLine("Top Count: " + entry.GetGsaContent("topCount"));
  Console.WriteLine("Diagnostic Terms: " +
    entry.GetGsaContent("diagnosticTerms"));
}
```

Creating a Search Report

Create a new search report entry by sending an authenticated `POST` request to the `root` entry of the `searchReport` feed.

The possible date formats for reports are as follows.

Purpose	Format
Date	**date**_*month_day_year*
Month	**month**_*month_year*
Year	**year**_*year*
Date range	**range**_*month_day_year_month_day_year*

For example, to specify the range of dates from 2 January 2009 to 23 September 2009, use this statement:

```
insertEntry.addGsaContent("reportDate", "range_1_2_2009_9_23_2009);
```

A new search report entry will be generated and returned as follows.

```
GsaEntry insertEntry = new GsaEntry();
insertEntry.AddGsaContent("reportName", "bbb");
insertEntry.AddGsaContent("collectionName", "default_collection");
insertEntry.AddGsaContent("reportDate", "month_5_2009");
insertEntry.AddGsaContent("withResults", "true");
insertEntry.AddGsaContent("topCount", "100");
myService.InsertEntry("searchReport", insertEntry);
```

Retrieving a Search Report

Retrieve the search report status and get search log content by sending an authenticated `GET` request to a search report entry of the `searchReport` feed.

A search report entry with log content (if content is ready) is returned:

```
GsaEntry entry = myService.GetEntry("searchReport", "bbb@default_collection");
Console.WriteLine("Entry Name: " + entry.GetGsaContent("entryID"));
Console.WriteLine("Report State: " + entry.GetGsaContent("reportState"));
Console.WriteLine("Report Creation Date: " +
  entry.GetGsaContent("reportCreationDate"));
Console.WriteLine("Report Date: " + entry.GetGsaContent("reportDate"));
Console.WriteLine("Is Final: " + entry.GetGsaContent("isFinal"));
Console.WriteLine("With Results: " + entry.GetGsaContent("withResults"));
Console.WriteLine("Top Count: " + entry.GetGsaContent("topCount"));
Console.WriteLine("Diagnostic Terms: " + entry.GetGsaContent("diagnosticTerms"));

string status = entry.GetGsaContent("reportState");
if (status.Equals("2") || status.Equals("3")) {
  Console.WriteLine("Report Content: " + entry.GetGsaContent("reportContent"));
}
```

Updating a Search Report

Update the search report status and get search report content by sending an authenticated `PUT` request to a search report entry of the `searchReport` feed. There are no properties.

A search log entry is returned:

```
GsaEntry updateEntry = new GsaEntry();
myService.UpdateEntry("searchReport", "bbb@default_collection");
```

Deleting a Search Report

Delete a search report by sending an authenticated `DELETE` request to a search report entry of the `searchReport` feed.

The search report entry will be deleted:

```
myService.DeleteEntry("searchReport", "bbb@default_collection");
```

Search Logs

Generate, update, and delete a search log using the `searchLog` feed. A search log lists all search queries for a specified time frame in a format similar to a common log format (CLF).

Search log entry properties:

Property	Description
`collectionName`	(Write-only) The collection name—use only to create a search log.
Entry Name	*`Search_Log_Name`*@*`Collection_Name`*
`fromLine`	(Read only) The first line of a search log that is returned in the log content—only returned when getting search log content and the content is ready.
`isFinal`	(Read only) If the search log contains the final result. If so, it means the last update date is later than the `reportDate`.
`logContent`	(Read only) A part of the content of the search log—only returned when getting search log content and the content is ready.
`reportCreationDate`	(Read only) The creation date of a search log.
`reportDate`	The dates of the queries in the search log.
`reportName`	(Write-only) The report name—use only to create a search log.
`reportState`	(Read only) Search log status: • `0`: Initialized. • `1`: Report is generating. • `2`: Report completed. • `3`: Non-final complete report is generating. • `4`: Last report generation failed.
`toLine`	(Read only) The last line of a search log that is returned in the log content—only returned when getting search log content and the content is ready.
`totalLines`	(Read only) The number of lines of a search log that are returned in the log content—only returned when getting search log content and the content is ready.

Listing a Search Log

List search log entries by sending an authenticated `GET` request to the `root` entry of the `searchLog` feed.

Parameter	Description
`collectionName`	Collection name of a search log. The default value is `all.collections`.

A list of search log entries will be returned.

```
GsaFeed myFeed = myService.GetFeed("searchLog");
foreach(GsaEntry entry in myFeed.Entries) {
  Console.WriteLine("Entry Name: " + entry.GetGsaContent("entryID"));
  Console.WriteLine("Report State: " + entry.GetGsaContent("reportState"));
  Console.WriteLine("Report Creation Date: " +
    entry.GetGsaContent("reportCreationDate"));
  Console.WriteLine("Report Date: " + entry.GetGsaContent("reportDate"));
  Console.WriteLine("Is Final: " + entry.GetGsaContent("isFinal"));
}
```

Creating a Search Log

Create a new search log entry by sending an authenticated `POST` request to the `root` entry of the `searchLog` feed.

A new search log entry will be generated and returned.

```
GsaEntry insertEntry = new GsaEntry();
insertEntry.AddGsaContent("reportName", "bbb");
insertEntry.AddGsaContent("collectionName", "default_collection");
insertEntry.AddGsaContent("reportDate", "date_3_25_2009");

myService.InsertEntry("searchLog", insertEntry);
```

Retrieving a Search Log

Check the search log status and get search log content by sending an authenticated `GET` request to the search log entry of the `searchLog` feed using the following query parameters.

Parameter	Description
`query`	Query string for the `logContent`. The `logContent` contains many lines of logs. The query string applies to each line, and only lines that contain the query string are returned.
`maxLines`	The maximum `logContent` lines to retrieve. The default value is 50 lines.
`startLine`	The first `logContent` lines to retrieve. The default value is 1 line.

A search log entry with `logContent`, if content is ready, is returned.

```
Dictionary<string, string> queries = new Dictionary<string, string>();
queries.Add("query","User");
queries.Add("startLine","1");
queries.Add("maxLine","10");

GsaEntry entry = myService.QueryEntry("searchLog", "bbb@default_collection",
  queries);
Console.WriteLine("Entry Name: " + entry.GetGsaContent("entryID"));
Console.WriteLine("Report State: " + entry.GetGsaContent("reportState"));
Console.WriteLine("Report Creation Date: " +
entry.GetGsaContent("reportCreationDate"));
Console.WriteLine("Report Date: " + entry.GetGsaContent("reportDate"));
Console.WriteLine("Is Final: " + entry.GetGsaContent("isFinal"));
string status = entry.GetGsaContent("reportState");
if (status.Equals("2") || status.Equals("3")) {
  Console.WriteLine("Log Content: " + entry.GetGsaContent("logContent"));
  Console.WriteLine("To Line: " + entry.GetGsaContent("toLine"));
  Console.WriteLine("From Line: " + entry.GetGsaContent("fromLine"));
  Console.WriteLine("Total Lines: " + entry.GetGsaContent("totalLines"));
 }
```

Updating a Search Log

Update the search log status and get search log content by sending an authenticated `PUT` request to the search log entry of the `searchLog` feed. No properties are required.

```
GsaEntry updateEntry = new GsaEntry();
myService.UpdateEntry("searchLog","bbb@default_collection");
```

Deleting a Search Log

Update the search log status and get search log content by sending an authenticated `DELETE` request to a search log entry of the `searchLog` feed.

The search log entry will be deleted.

```
myService.DeleteEntry("searchLog", "bbb@default_collection");
```

GSA Unification

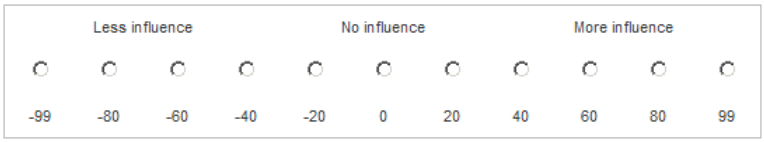
The sections that follow describe how to configure the GSA Unification features of the Admin Console:

- "Configuring a GSA Unification Network" on page 48
- "Adding a GSA Unification Node" on page 49
- "Retrieving a Node Configuration" on page 49
- "Retrieving All Node Configurations" on page 49
- "Updating a Node Configuration" on page 50
- "Deleting a Node" on page 50

GSA Unification is also known as dynamic scalability. This section describes use of the `federation` feed.

Configuring a GSA Unification Network

Retrieve, update, create, or delete the GSA Unification node configuration and retrieve the node configuration of all nodes in the network on the Google Search Appliance.

Property	Description
`applianceId`	The ID of the search appliance, required to identify the node during node operations.
`federationNetworkIP`	The private tunnel IP address (virtual address) for the node. This address must be an RFC 1918 address. **Note:** A GSA Unification works best when the IP addresses of the nodes are numerically near, such as 10.1.1.1, 10.1.1.2, 10.1.1.3, and so on. The search appliance disallows a GSA Unification for nodes that are not in the same /16 subnet. This is a problem only if there are more than 65534 nodes in a GSA Unification network. GSA Unification nodes communicate on TCP port 10999.
`hostname`	The host name of the search appliance.
`nodeType`	The type of search appliance. Possible values: • `PRIMARY`: The node merges results from other nodes. • `SECONDARY`: The node serves results to the other nodes. • `PRIMARY_AND_SECONDARY`: The node acts as both a primary and secondary node.
`scoringBias`	The scoring bias value for this node. Valid values are integers between -99 and 99. The scoring bias value reflects the weighting to be given to results from this node. A higher value means a higher weighting. The values and their equivalent in the Admin Console are: 
`secretToken`	The secret token that you use to establish a connection to this node. This token can be any non-empty string. The remote search appliance needs this token for the connection handshake.

Adding a GSA Unification Node

Add a GSA Unification node as follows:

```
// Create an entry to hold properties to insert
GsaEntry insertEntry = new GsaEntry();

// In the following example code, add a secondary
// node with arbitrary values for the various settings.
// Add properties to insertEntry
insertEntry.AddGsaContent("entryID", "node_appliance_id");
insertEntry.AddGsaContent("nodeType", "SECONDARY");
insertEntry.AddGsaContent("federationNetworkIP", "10.0.0.2");
insertEntry.AddGsaContent("secretToken", "token");
insertEntry.AddGsaContent("hostname", "corp.domain.x.com");
insertEntry.AddGsaContent("scoringBias", "20");

// Send the request
myService.InsertEntry("federation", insertEntry);
```

Retrieving a Node Configuration

Retrieve the configuration information about a GSA Unification node as follows:

```
// Send a request and print the response
GsaEntry myEntry = myService.GetEntry("federation", "applianceId");
string type = myEntry.GetGsaContent("nodeType");
Console.WriteLine("Node Type: " + type);
Console.WriteLine("GSA Unification Network IP: " + myEntry.GetGsaContent
  ("federationNetworkIP"));
Console.WriteLine("Host Name: " + myEntry.GetGsaContent("hostname"));
Console.WriteLine("Secret Token: " + myEntry.GetGsaContent("secretToken"));
Console.WriteLine("Scoring Bias: " + myEntry.GetGsaContent("scoringBias"));

if (type.Equals("SECONDARY")) {
  Console.WriteLine("Remote Front End: " + myEntry.GetGsaContent
    ("remoteFrontend"));
  Console.WriteLine("Node Timeout: " + myEntry.GetGsaContent("slaveTimeout"));
}

if (type.Equals("PRIMARY") || type.Equals("PRIMARY_AND_SECONDARY")) {
  Console.WriteLine("Secondary Nodes: " + myEntry.GetGsaContent
    ("secondaryNodes"));
}
```

Retrieving All Node Configurations

Retrieve information on all GSA Unification nodes as follows:

```
// Send the request and print the response
GsaFeed myFeed = myService.GetFeed("federation");

foreach(GsaEntry gsaEntry in myFeed.Entries) {
  // Process each entry
}
```

Updating a Node Configuration

Update the configuration of a node as follows:

```
// Create an entry to hold properties to update
GsaEntry updateEntry = new GsaEntry();

// Add properties to updateEntry
updateEntry.AddGsaContent("entryID", "applianceId");
updateEntry.AddGsaContent("nodeType", "PRIMARY");
updateEntry.AddGsaContent("federationNetworkIP", "10.0.0.3");
updateEntry.AddGsaContent("secretToken", "new_secret_token");
updateEntry.AddGsaContent("hostname", "new_hostname");
updateEntry.AddGsaContent("scoringBias", "20");

// Send the request
myService.UpdateEntry("federation", "applianceId", updateEntry);
```

Deleting a Node

Delete a node as follows:

```
myService.DeleteEntry("federation", "applianceId");
```

Administration

The sections that follow describe how to configure the **Administration** features of the Admin Console:

- "License Information" on page 50
- "Import and Export" on page 51
- "Event Log" on page 52
- "System Status" on page 53
- "Shutdown or Reboot" on page 54

License Information

Retrieve license Information from the search appliance using the `licenseInfo` entry of the `info` feed.

Note: You can only license information, but not update or install a new license through this API.

Retrieving License Information

Retrieve license information using the following properties.

Property	Description
`applianceID`	Provides the identification value for the Google Search Appliance software. This value is also known as the serial number for the search appliance.
`licenseID`	Provides the unique license identification value.
`licenseValidUntil`	Identifies when the search appliance software license expires.
`maxCollections`	Indicates the maximum number of collections. You can configure collections at the **ndex** > **Collections** page.
`maxFrontends`	Indicates the maximum number of front ends. You can configure front ends at the **Search > Search Features > Front Ends** page.
`maxPages`	Maximum number of content items that you can index with this product. Content items include documents, images, and content from the feeds interface.

Retrieving License Information Example

Retrieve the license Information from a search appliance as follows:

```
// Send the request and print the response
GsaEntry myEntry = myService.GetEntry("info", "licenseInfo");
Console.WriteLine("Appliance ID: " + myEntry.GetGsaContent("applianceID"));
Console.WriteLine("License ID: " + myEntry.GetGsaContent("licenseID"));
Console.WriteLine("License Valid Until: " + myEntry.GetGsaContent
  ("licenseValidUntil"));
Console.WriteLine("Maximum Front Ends: " + myEntry.GetGsaContent
  ("maxFrontends"));
Console.WriteLine("Maximum Pages: " + myEntry.GetGsaContent("maxPages"));
Console.WriteLine("Maximum Collections: " + myEntry.GetGsaContent
  ("maxCollections"));
```

Import and Export

Import or export a search appliance configuration using the `importExport` entry of the `config` feed.

The following is the common query parameter for all requests.

Parameter	Description
`password`	The password of the exported configuration.

Specify `importExport` entry properties.

Property	Description
`password`	The password of the configuration file.
`xmlData`	The content of an exported configuration saved as XML data.

Exporting a Configuration

Export a search appliance configuration by sending an authenticated `GET` request to the `importExport` entry of the `config` feed.

The following `importExport` entry is returned:

```
Dictionary<string, string> queries = new Dictionary<string, string>();
queries.Add("password", "12345678");

GsaEntry entry = myService.QueryEntry("config", "importExport", queries);
Console.WriteLine("XML Data: " + entry.GetGsaContent("xmlData"));
```

Importing a Configuration

Import a search appliance configuration sending an authenticated `PUT` request to the `importExport` entry of the `config` feed.

```
GsaEntry updateEntry = new GsaEntry();
updateEntry.AddGsaContent("xmlData", "<config data>");
updateEntry.AddGsaContent("password", "12345678");

myService.UpdateEntry("config", "importExport", updateEntry);
```

Event Log

Retrieve lines from the event log for a search appliance by using the `eventLog` entry of the `logs` feed. The following parameters let you make a query, specify a starting line, and specify the number of event log statements to retrieve.

Parameter	Description
`query`	Query string for the `logContent`. The `logContent` contains many lines of logs. The query string applies to each line, only lines that contain the query string are returned.
`startLine`	The starting line number to retrieve from the event log. The default value is `1`.
`maxLines`	The maximum number of lines in the event log to retrieve. The default value is `50` lines.

Use the following properties to retrieve event log lines and event log content.

Property	Description
`fromLine`	The starting line of a log.
`logContent`	The log's content.
`toLine`	The ending line of a log.
`totalLines`	Total lines of the log.

Retrieving an Event Log

Retrieve the event log information from a search appliance as follows:

```
Dictionary<string, string> queries = new Dictionary<string, string>();
queries.Add("query","User");
queries.Add("startLine","10");
queries.Add("maxLine","2");
GsaEntry myEntry = myService.QueryEntry("logs", "eventLog", queries);
Console.WriteLine("Log Content: " + myEntry.GetGsaContent("logContent"));
Console.WriteLine("Total Lines: " + myEntry.GetGsaContent("totalLines"));
Console.WriteLine("From Line: " + myEntry.GetGsaContent("fromLine"));
Console.WriteLine("To Line: " + myEntry.GetGsaContent("toLine"));
```

System Status

System status for the search appliance can be retrieved through the `systemStatus` entry of the `status` feed.

Property	Description
`cpuTemperature`	Temperature of the CPU. Set to `0` if okay, `1` if caution, `2` if critical.
`diskCapacity`	Remaining disk capacity of a search appliance. Set to `0` if okay, `1` if caution, `2` if critical.
`machineHealth`	Health of the motherboard. Set to `0` if okay, `1` if caution, `2` if critical.
`overallHealth`	Overall health of the a search appliance. Set to `0` if okay, `1` if caution, `2` if critical.
`raidHealth`	Health of the RAID array. Set to `0` if okay, `1` if caution, `2` if critical.

Note: Some health properties may not exist in certain versions of the search appliance.

Retrieving System Status

Retrieve a the current search appliance system status as follows:

```
GsaEntry myEntry = myService.GetEntry("status", "systemStatus");
Console.WriteLine("Overall Health: " + myEntry.GetGsaContent("overallHealth"));
Console.WriteLine("Disk Capacity: " + myEntry.GetGsaContent("diskCapacity"));
Console.WriteLine("RAID Health: " + myEntry.GetGsaContent("raidHealth"));
Console.WriteLine("CPU Temperature: " + myEntry.GetGsaContent("cpuTemperature"));
Console.WriteLine("Machine Health: " + myEntry.GetGsaContent("machineHealth"));
```

Shutdown or Reboot

Shut down or reboot the search appliance.

Property	Description
`command`	Command sent to the search appliance. The command can be `shutdown` or `reboot`.
`runningStatus`	Indicates the search appliance status: • `shuttingDown`: If you sent the `shutdown` command. • `rebooting`: If you sent the `reboot` command. • `running`: If the search appliance is operating normally.

Shutting Down or Rebooting

Shut down or reboot the search appliance as follows:

```
// Create an entry to hold properties to update
GsaEntry updateEntry = new GsaEntry();

// Add a property to updateEntry
updateEntry.AddGsaContent("command", "reboot");

myService.UpdateEntry("command", "shutdown", updateEntry);
```

Index

Symbols

.NET client library (DLLs) 6
.NET Google Data API client library 5

A

Admin Console 6, 10, 31
Administration 50–54
API software 6
authentication 7

C

C#.NET example files 6
collections
  create 22
  delete 23
  retrieve 23
  update 23
config feed 8, 11, 15, 18, 51
connector managers
  add 17
  delete 18
  retrieve 18
  update 18
content statistics, retrieve 29
contentStatistics feed 29, 30
crawl and index 7–23
crawl diagnostics
  query parameters 26–27
  retrieve document information 28
  status values 24–25
crawl mode, update 12
crawl schedule
  retrieve 12
  update 12
crawler access rules
  delete 14
  insert 13
  retrieve 13
  update 14
crawlURLs
  retrieve 8
  update 8

D

data source feed
  delete 10
  destroy 11
  retrieve 10
diagnostics feed 24, 26, 28
document status, retrieve 21

E

event log, retrieve 53
export configuration 52

F

federation feed 48
feed feed 9, 10
feeds
  API 5
  config 8, 11, 15, 18, 51
  contentStatistics 29, 30
  data source 5
  diagnostics 24, 26, 28
  federation 48
  feed 9, 10
  frontend 31
  info 50
  logs 52
  onebox 19
  outputFormat 33
  searchLog 45, 46, 47
  searchReport 42, 44
  status 41, 53
  suggest 40, 41
freshness tuning configuration
  retrieve 16
  update 16
front end
  insert 32
front ends
  delete 32
  retrieve 31
frontend feed 31

G

google-enterprise-gdata-api open source site 6
GSA Unification
  add nodes 49
  configure 47–50
  delete nodes 50
  retrieve nodes 49
  update nodes 50
GsaService object 7

H

host load schedule
  retrieve 15
  update 16

I

import configuration 52
info feed 50

K

KeyMatch settings
  retrieve 36
  update 36

L

license information, retrieve 51
logs feed 52

M

Microsoft Visual C# 2008 Express Edition 6

O

OneBox
  delete module 20
  retrieve modules 19
  retrieve settings 18
  update settings 19
onebox feed 19
outputFormat feed 33

P

pause crawl 20

Q

query suggestion
  refresh 41
  retrieve blacklist 40
  update blacklist 40

R

reboot a search appliance 54
recrawl URL patterns 17
related queries
  retrieve 38
  update 39
remove URLs, update 32
reset index
  reset 30
  retrieve status 30
resume crawl 20

S

sample applications 6
search logs
  create 46
  delete 47
  entry properties 45
  list entries 45
  retrieve 46
  update 47
search reports
  create 43
  delete 44
  entry properties 42
  list entries 42
  retrieve 44
  update 44
searchLog feed 45, 46, 47
searchReport feed 42, 44
serving 31–41
serving status, retrieve 41
shut down a search appliance 54
status and reports 41–53
status feed 41, 53
suggest feed 40, 41
system status, retrieve 53

T

trusted IP addresses
  retrieve 11
  update 11

U

URL patterns
  crawl 8
  recrawl 17

X

XSLT stylesheet
  retrieve 34
  update 34