Google Search Appliance

Search Protocol Reference

Google Search Appliance software version 7.2 and later

Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043

www.google.com

GSA-XML_200.02
March 2015

Contents

Chapter 1

Introduction

The Google Search Appliance uses a simple HTTP-based protocol for serving search results. This enables you to control how search results are requested and how they are presented to end users. This guide describes the technical details of search requests and results. This guide assumes that you have a basic understanding of the HTTP protocol and the HTML document format.

For terminology definitions, see the *Google for Work Glossary*.

The Google Search Appliance accepts search requests as input, and returns search results as output.

Search requests, the input, are simple HTTP requests to the Google Search Appliance. Search users typically use HTML forms displayed in a web browser to make these requests, but other applications can also send search requests by making appropriate HTTP requests. For information on the search request format and options, see “Request Format” on page 6.

Search results, the output, are returned in either HTML or XML formats, as specified in the search request.

HTML-formatted results can be displayed directly in a web browser. The search appliance generates HTML results by applying an XSL stylesheet to the XML results. You can customize the appearance of the HTML results by modifying this stylesheet. For more information, see “Custom HTML” on page 53.

XML-formatted output makes it possible to process the search results in web applications or other environments. For information on the XML results format, see “XML Output” on page 54.

Note: In this guide, long URLs may appear as multiple lines for better readability. In a browser, all URLs are continuous strings.

Chapter 2

Request Format

The information in this section helps you create custom searches for your web site. By using search parameters, special query terms and filters in your search requests, you can refine and enhance searches to serve your needs.

This section contains:

Request Overview

Using the Google search protocol is as simple as requesting a page from a web server. The Google search request is a standard HTTP GET or POST command, which returns results in either XML or HTML format, as specified in the search request.

The search request is a URL that combines the following:

- Your Google Search Appliance host name or IP address, which were assigned when the search appliance was set up
- Search interface port (default HTTP serving port: 80 for HTTP and 443 for HTTP over SSL/TLS)
- A path describing the search query. The path starts with “/search?”, and is followed by one or more name-value pairs (input parameters) separated by the ampersand (&) character.

The GET command has a 2KB limit on query strings. To submit longer query strings, use the POST command, as described in “Using the POST Command” on page 7.

Using the POST Command

In some instances, your query strings might exceed the 2KB URL length limit of GET requests and be truncated. This might happen when you submit dynamic navigation queries containing a large number of metadata filters. You can avoid this limitation by submitting POST requests instead, which have a much larger body limit (10KB).

POST Limitations

POST support is only available for:

- Requests for search service (/search)
- Public search
- Secure search, but only for cookie and basic authentication, and only when the Trusted Applications feature is used (see "Using Trusted Applications" in *Managing Search for Controlled-Access Content*)

POST support is not available for other Universal Login Auth Mechanisms. You must use the GET command for these.

If you are sending non UTF-8 data, you must include the ie parameter (described on page 16) in the POST body. This parameter sets the character encoding that is used to interpret the query string. You should also specify the access parameter (as shown in Search Request Examples (POST command)) in the POST body when sending POST requests.

The following search parameters are not included by default in a POST request:

- entqr--Sets the query expansion policy.
- entqrm--Controls query expansions for meta tags.
- entsp--Controls the use of the advanced relevance scoring parameters.
- filter-Activates or deactivates automatic results filtering.
- ip--Indicates the IP address of the user who submitted the search query.
- tlen--Specifies the number of bytes that would be used to return the search results title.
- ulang--Indicates the language of the user who submitted the search query
- wc--Specifies the number of wildcard expansions for a wildcard expression.
- wc_mc--Specifies whether or not the search appliance considers all words with * as wildcard terms.

If you want to include any of these parameters in a POST request, you must add them. For more information about these parameters, see "Search Parameters."

Structure of the POST Body

The structure of the POST body is a URL-encoded query string. It is like the URL of a GET request, after the question mark.

Submitting a Search Request

Typically, search users make search requests by entering search parameters in a HTML form rendered in a web browser (like the following):

```
<form method="GET" action="http://search.mycompany.com/search">
   <input type="text" name="q" size="32" maxlength="256" value="query string">
   <input type="submit" name="btnG" value="Google Search">
   <input type="hidden" name="site" value="default_collection">
   <input type="hidden" name="client" value="default_frontend">
   <input type="hidden" name="output" value="xml_no_dtd">
   <input type="hidden" name="proxystylesheet" value="default_frontend">
</form>
```

Such forms are the most recognizable methods for generating GET requests, but there are numerous other ways. For example, a web page may include a direct link that brings users to a page of search results:

```
http://search.mycompany.com/search?q=query+string
                           &site=default_collection
                           &client=default_frontend
                           &output=xml_no_dtd
                           &proxystylesheet=default_frontend HTTP/1.0
```

Alternatively, a web application may make a HTTP GET request directly:

```
GET /search?q=query+string&site=default_collection
                           &client=default_frontend
                           &output=xml_no_dtd
                           &proxystylesheet=default_frontend HTTP/1.0
```

Each of these examples results in the same GET request. The HTTP response to this request contains the first page of search results for the query "query string", restricted to URLs in the collection named "default_collection." The results are rendered into HTML format using the XSL stylesheet associated with the front end named "default_frontend".

You can search multiple collections by separating collection names with the OR character (|) or the AND character (.), for example: `&site=col1.col2` or `&site=col1|col2`.

The rest of the examples that follow use the raw HTTP GET format (as in the last example).

Search Request Examples (GET Command)

Example 1. This request returns the first 10 results that match the search query terms **"bill"** and "material":

```
GET /search?q=bill+material&output=xml&client=test&site=operations HTTP/1.0
```

Explanation:

The search query is "bill material".

```
GET /search?q=bill+material&output=xml&client=test&site=operations HTTP/1.0
```

Search is limited to the documents in the "operations" collection.

```
GET /search?q=bill+material&output=xml&client=test&site=operations HTTP/1.0
```

Results are returned in the Google XML output format.

```
GET /search?q=bill+material&output=xml&client=test&site=operations HTTP/1.0
```

Example 2. This request returns results numbered 11-15 that match the same query terms and collection as example 1. As specified by the `proxystylesheet` parameter, the results are rendered in the custom HTML output format defined by the front end named "test."

```
GET /search?q=bill+material&start=10&num=5&output=xml_no_dtd&proxystylesheet=
test&client=test&site=operations HTTP/1.0
```

Explanation:

This search request uses the same search query terms and collection as in Example 1.

```
GET /search?q=bill+material&start=10&num=5&output=xml_no_dtd&proxystylesheet=
test&client=test&site=operations HTTP/1.0
```

Results numbered 11–15 are returned.

```
GET /search?q=bill+material&start=10&num=5&output=xml_no_dtd&proxystylesheet=
test&client=test&site=operations HTTP/1.0
```

Results are returned in custom HTML output format, which is created by applying the XSL stylesheet associated with the "test" front end to the standard XML results. See "proxystylesheet" on page 18.

```
GET /search?q=bill+material&start=10&num=5&output=xml_no_dtd&proxystylesheet=
test&client=test&site=operations HTTP/1.0
```

Example 3. This request returns the first 10 German results that match the search query **"**Star Wars Episode +I":

```
GET /search?q=Star+Wars+Episode+%2BI&output=xml_no_dtd&lr=lang_de&ie=
latin1&oe=latin1&client=test&site=movies&proxystylesheet=test HTTP/1.0
```

Explanation:

The search query term is "Star Wars Episode +I". Search is limited to documents in the "movies" collection.

```
GET /search?q=Star+Wars+Episode+%2BI&output=xml_no_dtd&lr=lang_de&ie=
latin1&oe=latin1&client=test&site=movies&proxystylesheet=test HTTP/1.0
```

Results show the first 10 German results.

```
GET /search?q=Star+Wars+Episode+%2BI&output=xml_no_dtd&lr=lang_de&ie=latin1&oe=
latin1&client=test&site=movies&proxystylesheet=test HTTP/1.0
```

Results are returned in Google custom HTML output format, which is created by applying the XSL stylesheet associated with the "test" front end to the standard XML results.

```
GET /search?q=Star+Wars+Episode+%2BI&output=xml_no_dtd&lr=lang_de&ie=latin1&oe=
latin1&client=test&site=movies&proxystylesheet=test HTTP/1.0
```

Search Request Examples (POST command)

The following examples show search requests that use the POST command for public search only. The POST command should have a target to search:

```
POST /search HTTP/1.0
```

The query string payload is not part of the header; it appears in the body of the request. The following examples show query strings. Take note that line breaks are used for readability only and should not be present in actual code.

This request returns the first 10 results that match the search query terms **"bill"** and "material":

```
q=bill+material&output=xml&client=test&site=operations&access=p
```

This request returns results numbered 11-15 that match the same query terms and collection as example 1. As specified by the `proxystylesheet` parameter, the results are rendered in the custom HTML output format defined by the front end named "test."

```
q=bill+material&start=10&num=5&output=xml_no_dtd&proxystylesheet=test
&client=test&site=operations&access=p
```

Search Parameters

This section lists the valid name-value pairs that can be used in a search request and describes how these parameters modify the search results.

All search requests must include the parameters `site`, `client`, **`q`**, and `output`. All parameter values must be URL-encoded (see "Appendix B: URL Encoding" on page 108), except where otherwise noted.

access

Specifies whether to search public content, secure content, or both.

Possible values for the `access` parameter are:

Value	Description
p	search only public content
s	search only secure content
a	search all content, both public and secure

Default value: p

as_dt

Modifies the `as_sitesearch` parameter as follows:

Value	Modification
i	Include only results in the web directory specified by **`as_sitesearch`**
e	Exclude all results in the web directory specified by `as_sitesearch`

For example, to exclude results, use `as_dt=e`.

Default value: i

as_epq

Adds the specified phrase to the search query in parameter `q`.

For example, to add the terms "hello there" use `as_epq=hello there`

This parameter has the same effect as using the `phrase` special query term (see "Phrase Search" on page 28).

Default value: Empty string

as_eq

Excludes the specified terms from the search results.

For example, to filter out results that contain the term "deprecated," use `as_eq=deprecated`

This parameter has the same effect as using the exclusion (-) special query term (see "Exclusion" on page 26).

Default value: Empty string

as_filetype

Specifies a file format to include or exclude in the search results. Modified by the `as_ft` parameter. For a list of possible values, see "File Type Filtering" on page 27.

For example, to include only pdf files in results, use `as_filetype=pdf`

Default value: Empty string

as_ft

Modifies the `as_filetype` parameter to specify filetype inclusion and exclusion options. The values for `as_ft` are:

Value	Description
i	Adds the special query term `filetype:` to the query followed by the value of `as_filetype.`
e	Adds the special query term `-filetype:` to the query followed by the value of `as_filetype.`

For example, to add the special query term `filetype:, use as_ft=i`

Query is the string that is included in the response's q element. Both `as_filetype` and `as_ft` are also returned in the response's PARAM elements.

Default value: Empty string

as_lq

Specifies a URL, and causes search results to show pages that link to the that URL. This parameter has the same effect as the `link` special query term (see "Back Links" on page 24). No other query terms can be used when using this parameter.

For example, to return results that have links to http://myUrl.com/Page, use `as_lq=http://myUrl.com/Page`

Default value: Empty string

as_occt

Specifies where the search engine is to look for the query terms on the page: anywhere on the page, in the title, or in the URL.

Value	Meaning
any	anywhere on the page
title	in the title of the page
url	in the URL for the page

For example to specify that the search engine should only look in titles, use `as_occt=title`

Default value: any

as_oq

Combines the specified terms to the search query in parameter `q`, with an `OR` operation.

For example to search for documents that contain the terms"London" or "Paris," use: `as_oq=London Paris`, `as_oq=London%20Paris`, or `as_oq=London+Paris`

This parameter has the same effect as the `OR` special query term and is used only for single words (see "Boolean OR Search" on page 24).

Default value: Empty string

as_q

Adds the specified query terms to the query terms in parameter `q`.

For example, to add the terms "enterprise" and "large" use `as_q=enterprise large`

Default value: Empty string

as_sitesearch

Limits search results to documents in the specified domain, host or web directory, or excludes results from the specified location, depending on the value of `as_dt`. This parameter has the same effect as the `site` or `-site` special query terms. It has no effect if the `q` parameter is empty.

When the Google Search Appliance receives a search request that includes the `as_sitesearch` parameter, it converts the value of the parameter into an argument to the `site` special query term and appends it to the value of `q` in the search results. For example, suppose that a search contains these parameters:

```
q=mycompany&as_sitesearch=www.mycompany.com
```

The raw XML of the search results contains the following:

```
<q>mycompany site:www.mycompany.com</q>
```

The default XSLT stylesheet displays the value of the `q` tag in the search box on the results page. Consequently, using an `as_sitesearch` parameter changes the user's search query by modifying the contents of the search box.

The specified value for `as_sitesearch` must contain fewer than 125 characters. See also the site parameter (see "site" on page 19).

Default value: Empty string

client

Required parameter. If this parameter does not have a valid value, other parameters in the query string do not work as expected.

A string that indicates a valid front end and the policies defined for it, including KeyMatches, related queries, filters, remove URLs, and OneBox Modules. Notice that the rendering of the front end is determined by the proxystylesheet parameter. Example: **`client=myfrontend`**

dnavs

Used when the dynamic navigation feature is enabled and applied to a front end.

This parameter stores the current dynamic navigation filters applied in the search results. It does not affect the search results in any way and is used only in the XSLT rendering logic. Dynamic navigation uses the `q` parameter for affecting search results by appending the selected filters as `inmeta:` query terms.

entqr

This parameter sets the query expansion policy according to the following valid values:

Value	Description
0	None
1	Standard (`entqr=1`)—Uses only the search appliance's synonym file.
2	Local (`entqr=2`)—Uses all displayed and activated synonym files.
3	Full (`entqr=3`)—Uses both standard and local synonym files.

Standard terms use only the search appliance's internal contextual (synonym) files for query expansion. Local terms use all displayed and activated synonym files, including any uploaded files. After you configure and enable the appropriate query expansion files, set the query expansion policy for a front end. Each front end has a policy that specifies whether it uses the search appliance's built-in logic (the "standard" set of terms), your own list of synonyms (the "local" set), or both (the "full" set). Query expansion files are used only if the query expansion policy for a front end is set to Local or Full.

If this parameter is omitted, the query expansion value specified for the front end is used.

Default value: 0

entqrm

The `entqrm` parameter controls query expansions for meta tags according to the following valid values::

Value	Description
0	None
1	Names (`entqrm=1`) Enables query expansion only for meta-tag names.
2	Values (`entqrm=2`) Enables query expansion only for meta-tag values.
3	Both (`entqrm=3)` Enables query expansion for both meta-tag names and values.

Default value: 0

entsp

The `entsp` parameter controls the use of the advanced relevance scoring parameters that you set under **Result Biasing** on the Admin Console. The parameter accepts the following valid values:

Value	Description
No value	If you do not specify a value for the `entsp` parameter in the search request, the scoring policy specified for the current front end is used. For example, if the search appliance uses a front end called `my_frontend` in which the scoring policy `my_scorepolicy` is configured, omitting the `entsp` parameter means that the scoring policy `my_scorepolicy` is used.
0	Do not use any scoring policy.
a	Specifies that the default scoring policy for the search appliance is used. It should be named as `default_policy`.
a__xxx	Specifies a particular advanced scoring policy. For example, for a source biasing policy called `mypolicy`, the parameter is set with the following syntax: `entsp=a__mypolicy` Note that the above syntax uses two underscores between the `a` and the name of the source biasing policy.

Default value: 0

exclude_apps

Controls whether Google Apps content from the user's Google Apps domain displays in search results, according to the following values:

Value	Description
No value	If you omit the `exclude_apps` parameter in the search request, Google Apps content will not display in search results.
0	(exclude_apps=0). Google Apps content will display in search results, as determined by the **Google Apps results** sidebar element in the front end. See the table below.

The following table lists how Google Apps content will display in search results based on the values of the `exclude_apps` and `only_apps` search parameters, and the setting of the **Google Apps results** sidebar option in the front end. See also "only_apps" on page 17.

exclude_apps=	only_apps=	Google Apps Sidebar Element	Result
0	—	Disabled (default)	Google Apps content and normal GSA organic results both display in the main body of search results.
0	—	Enabled	Google Apps content displays in the sidebar of search results, while normal GSA organic results display in the main body of search results.
—	1	Disabled (default)	Only Google Apps content will display, in the main body of search results.
—	1	Enabled	Only Google Apps content will display, in the sidebar of search results.
—	—	Disabled (default)	Only normal GSA organic results will display, in the main body of search results.
—	—	Enabled	Google Apps content will display in the sidebar of search results, while normal GSA organic results display in the main body of search results.

filter

Activates or deactivates automatic results filtering. By default, filtering is applied to Google search results to improve results quality. See "Automatic Filtering" on page 31 for more information.

Default value: 1

getfields

Indicates that the names and values of the specified meta tags should be returned with each search result, when available. See "Meta Tags" on page 42 for more information.

Meta tag names or values must be double URL-encoded (see "Appendix B: URL Encoding" on page 108).

Default value: Empty string

gsaRequestID

A GSA-generated ID that is set at the start of a query session and that exists only for the length of a query. Serving logs use this value, which is sent back to the search appliance for each subsequent request during the query session.

Default value: None.

ie

Sets the character encoding that is used to interpret the query string. See “Internationalization” on page 35 for more information.

Default value: `latin1`

ip

When queries are made using the HTTP or HTTPS protocol, the `ip` parameter contains the IP address of the user who submitted the search query. You do not supply this parameter with the search request. The `ip` parameter is returned in the XML search results. For example:

```
<PARAM name="ip" value="172.24.96.29" original_value="172.24.96.29"/>
```

Default value: Value is not set in the search request; the value is automatically returned in the search results.

lr

Restricts searches to pages in the specified language. If there are no results in the specified language, the search appliance displays results in all languages. The search appliance may use the language parameter to segment search queries in some Asian languages that do not normally have spaces between words. As a result, you might see different results to your search depending on the value of the lr parameter. See “Language Filters” on page 32 for more information.

Default value: Empty string

num

Maximum number of results to include in the search results. The maximum value of this parameter is 1000. Taken together, the values of the `start` (see “start” on page 20) and `num` parameters determine the range of the results that are returned.

The initial index point of the search results is the value of the `start` parameter (see “start” on page 20). The ending index point of the search results is the value of the `start` parameter (see “start” on page 20) plus the value of the `num` parameter minus 1. All index points are zero based, meaning the first result has the value 0.

The actual number of results may be smaller than the requested value.

Default value: 10

numgm

Number of KeyMatch results to return with the results. A value between 0 to 50 can be specified for this option.

Default value: 3

oe

Sets the character encoding that is used to encode the results. See “Internationalization” on page 35 for more information.

Default value: UTF8

only_apps

Restricts search results to only Google Apps content from the user’s Google Apps domain, according to the following values:

Value	Description
No value	If you omit the `only_apps` parameter in the search request, search results will not be restricted to only Google Apps content.
1	(only_apps=1). Only Google Apps content will display in search results, as determined by the **Google Apps results** sidebar element in the front end (see “exclude_apps” on page 15).

output

Required parameter. If this parameter does not have a valid value, other parameters in the query string do not work as expected.

Selects the format of the search results. Example: `output=xml`

Value	Output Format
`xml_no_dtd`	XML results or custom HTML (See `proxystylesheet` parameter for details.)
`xml`	XML results with Google DTD reference. When you use this value, omit `proxystylesheet`.

partialfields

Restricts the search results to documents with meta tags whose values contain the specified words or phrases.

(See “Meta Tags” on page 42 for more information.)

Meta tag names or values must be double URL-encoded (see “Appendix B: URL Encoding” on page 108).

Default value: Empty string

proxycustom

Specifies custom XML tags to be included in the XML results. The default XSLT stylesheet uses these values for this parameter: `<HOME/>`, `<ADVANCED/>`. The `proxycustom` parameter can be used in custom XSLT applications. See “Custom HTML” on page 53 for more information.

This parameter is disabled if the search request does not contain the `proxystylesheet` tag. If custom XML is specified, search results are not returned with the search request.

Meta tag names or values must be double URL-encoded (see “Appendix B: URL Encoding” on page 108).

Default value: Empty string

proxyreload

Instructs the Google Search Appliance when to refresh the XSL stylesheet cache. A value of 1 indicates that the Google Search Appliance should update the XSL stylesheet cache to refresh the stylesheet currently being requested. This parameter is optional. By default, the XSL stylesheet cache is updated approximately every 15 minutes. (See “Custom HTML” on page 53 for more information.) Take note that updating the XSL stylesheet cache increases latency for the search request and should not be used in production environment with high load or during performance testing.

Default value: 0

proxystylesheet

If the value of the `output` parameter is `xml_no_dtd`, the output format is modified by the `proxystylesheet` value as follows:

Proxystylesheet Value	Output Format
Omitted	Results are in XML format.
Front End Name	Results are in Custom HTML format. The XSL stylesheet associated with the specified Front End is used to transform the output.

See “Custom HTML” on page 53 for more details. Notice that a valid front end and the policies defined for it are determined by the client parameter. If the `proxystylesheet` value is an empty string (`""`), an error is returned.

Default value: N/A

q

Search query as entered by the user.

See “Query Terms” on page 22 for additional query features.

Default value: N/A

rc

Request an accurate result count for up to 1M documents. When `rc = 1`, the user will get accurate result count. This might introduce high latency. `rc=0` works like current default search estimates, as described in “Appendix A: Estimated vs. Actual Number of Results” on page 106.

Default value: 0

requiredfields

Restricts the search results to documents that contain the exact meta tag names or name-value pairs. See “Meta Tags” on page 42 for more information.

Meta tag names or values must be double URL-encoded (see “Appendix B: URL Encoding” on page 108).

Default value: Empty string

secure_estimates

Retrieves estimates for secure searches if **Show Per-Query Estimates** is enabled on the **Search > Search Features > Query Settings** page in the Admin Console and the `secure_estimates` search parameter is set to 1 in the request:

```
&secure_estimates=1
```

Default value: 0

site

Required parameter. Limits search results to the contents of the specified collection.

If this parameter does not have a valid value, other parameters in the query string do not work as expected. Omitting this parameter from a search query causes the entire search index to be queried instead of limiting search results.

If this parameter contains characters that are not allowed, the search appliance does not return any results for the query. This parameter allows `.` `_` `-` and `|`.

You can search multiple collections by separating collection names with the OR character, which is notated as the pipe symbol, or the AND character, which is notated as a period.

The following example uses the AND character:

```
    &site=col1.col2
```

The following example uses the OR character:

```
    &site=col1|col2
```

Query terms `info`, `link` and `cache` ignore collection restrictions that are specified by the `site` query parameter.

The `site` parameter is required for Advanced Search Reporting.

sitesearch

Limits search results to documents in the specified domain, host, or web directory. Has no effect if the `q` parameter is empty. This parameter has the same effect as the `site` special query term.

Unlike the `as_sitesearch` parameter, the `sitesearch` parameter is not affected by the `as_dt` parameter. The `sitesearch` and `as_sitesearch` parameters are handled differently in the XML results. The `sitesearch` parameter's value is not appended to the search query in the results. The original query term is not modified when you use the `sitesearch` parameter. The specified value for this parameter must contain fewer than 125 characters.

Default value: Empty string

sort

Specifies a sorting method. Results can be sorted by date. (See "Sorting" on page 37 for sort parameter format and details.)

Default value: Empty string

start

Specifies the index number of the first entry in the result set that is to be returned. Use this parameter and the `num` parameter (see "num" on page 16) to implement page navigation for search results. The index number of the results is 0-based. For example:

- `start`=0, `num`=10, returns the first 10 results. These are returned by default if you do not specify values for `start` or `num`.
- `start`=10, `num`=10, returns the next 10 results.

The maximum number of results available for a query is 1,000, i.e., the value of the start parameter added to the value of the `num` parameter cannot exceed 1,000.

Default value: 0

tlen

Specifies the number of bytes that would be used to return the search results title. If titles contain characters that need more bytes per character, for example in utf-8, this parameter can be used to specify a higher number of bytes to get more characters for titles in the search results.

Default value: 70 bytes

ud

Specifies whether results include `ud` tags. A `ud` tag contains internationalized domain name (IDN) encoding for a result URL. IDN encoding is a mechanism for including non-ASCII characters. When a `ud` tag is present, the search appliance uses its value to display the result URL, including non-ASCII characters.

The value of the `ud` parameter can be zero (0) or one (1):

- A value of 0 excludes `ud` tags from the results.
- A value of 1 includes `ud` tags in the results.

As an example, if the result URLs contain files whose names are in Chinese characters and the `ud` parameter is set to 1, the Chinese characters appear. If the `ud` parameter is set to 0, the Chinese characters are escaped.

Default value:

- When a search request includes the `proxystylesheet` parameter, the default value for `ud` is 1 and cannot be modified.
- When the search request does not include the `proxystylesheet` parameter, the default value for `ud` is 0 and the value can be modified.

ulang

Gets the user's browser language. The user can specify this search parameter. If it is not specified, it takes the value from HTTP headers in the received search request. XSLT uses this parameter to translate titles and snippets into the user's browser language.

Note: A similar parameter, inlang, is for GSA internal use only.

wc

Specifies the number of wildcard expansions for the wildcard expression. Takes values in the range of 0-1000, where 0 disables wildcard search.

For example, the wildcard term `go*` expands into any word that begins with the pattern "go." If `wc=3`, then the search expands to include at most 3 expanded terms.

Default value: 200

wc_mc

Specifies whether or not the search appliance considers all words with * as wildcard terms. Valid values are:

- 1--Consider all words with * as wildcard terms
- 0--To use a wildcard term, the user must type the full wildcard expression: `wildcard:pattern*`

Default value: 1

For more information, see "Wildcard Search."

Custom Parameters

In addition to the “Search Parameters” on page 10, you can also define custom parameters in a search request. The search appliance returns custom parameters and their values in the search results.

For security reasons, all space characters in a custom parameter are replaced by an underscore (_). For example:

```
http://search.customer.com/search?q=customer+query
 &site=collection
 &client=collection
 &output=xml_no_dtd
 &myparam=test+this
```

This search request includes the custom parameter myparam with a value of test+this . The space character (represented as "+") in the custom parameter myparam is replaced by the underscore character (_) in the XML output.

The resulting XML output looks like this:

```
<param name="q" value="customer query" original_value="customer+query"/>
<param name="myparam" value="test_this" original_value="test+this" />
```

The unmodified value can be retrieved from the original_value attribute.

Query Terms

By default, the Google Search Appliance returns only pages that include all of your search terms. You do not need to include “AND” between terms. The order of search terms affects the search results. To further restrict a search, just include more terms. To use keywords such as AND as regular search terms instead of as special keywords, enclose them in quotes.

The search appliance may ignore common words and characters such as where and how and other digits and letters that slow down a search without improving the results.

If a common word is essential to getting the results you want, you can include the word by putting double quotes around it. For example, to ensure that Google includes the “I” in a search for “Star Wars Episode I”, enter the search query as follows:

```
Star Wars Episode “I”
```

Special Characters: Query Term Separators

By default, non-alphanumeric characters in a search query separate the query terms in the same way as space characters. For example, the following search term is not one query term, but six query terms:

3,6-DICHLORO-2-PYRIDINECARBOXYLIC ACID

The terms are:

3
6
DICHLORO
2
PYRIDINECARBOXLYIC
ACID

The following characters are exceptions:

Character	Description
Double quote mark (")	Used as a special query term for phrase searches. Note that using double quotation marks for phrase search does not reduce the number of query terms. For example, the search term 3,6-DICHLORO-2-PYRIDINECARBOXYLIC ACID is six query terms whether or not it is enclosed in quotation marks.
Forward slash (/)	Used as a special query term for phrase searches.
Plus sign (+)	Treated as a Boolean `AND`.
Minus sign or hyphen (-)	Treated as part of a query term if there is no space preceding it. A hyphen that is preceded by a space is the Exclude Query Term operator. A hyphen after a parenthesis is treated as the Exclude Query Term operator. For example, the query `Fmoc-Cys(Trt)-OH` returns documents that contain `Fmoc-Cys(Trt)` and excludes documents that contain `OH` in addition to `Fmoc-Cys(Trt)`.
Decimal point (`.`)	Treated as a query term separator unless it is part of a number (for example, `250.01`). For example, `dancing.parrot` is equivalent to `"dancing parrot"` with quotes in the query. The term `dancing.parrot` is not equivalent to `dancing parrot` (without quotes).
Ampersand (`&`)	Treated as another character in the query term in which it is included.

If a document contains a number, with or without a decimal point, that has letters immediately before or after it, the letters are treated as a separate word or words. For example, the string `802.11a` is indexed as two separate words, `802.11` and `a`.

Note: An underscore (or under bar) is not a query term separator. For example, if you search for `taino_the_parrot`, the only valid search result is a document that contains the exact phrase, `taino_the_parrot`. A search for `taino` or `parrot` does not return the `taino_the_parrot` result.

Special Query Terms

Google search supports the following special query terms. A user or search administrator can use these terms to access additional search features.

Note: All query terms must be correctly URL-encoded in a search request (see "Appendix B: URL Encoding" on page 108).

Anchor text search

Restricts the search to pages that contain all the search terms that are specified in the anchor text in links to the page. For example, `allinanchor:best museums sydney` returns only pages in which the anchor text in links to the pages contain the words "best," "museums," and "sydney." The following example shows an anchor tag:

```
<a href="http://foo.com"> museums </a>
```

`allinanchor:` evaluates the text between `>` and `</a>`. `allinanchor:` evaluates only `<a href` anchor tags. It does not evaluate `<a name` anchor tags.

An anchor is a marker inserted at a specific section of a page. It lets the writer of the document create links to these anchors, which, when clicked, quickly take the reader to the specified section of the same page or another page. The table of contents at the top of this document, for example, uses hyperlinks to anchors embedded throughout this document.

Do not include any other search operators with the `allinanchor:` operator.

Sample usage:

```
allinanchor:membership directory
```

Back Links

The query prefix `link:` lists web pages that have links to the specified web page. No spaces can come between `link` and the web page URL.

The URL pattern for the linked-to web page must appear in Follow and Crawl URL patterns on the **Content Sources > Web Crawl > Start and Block URLs** page in the Admin Console. Otherwise, the link query does not produce any search results. For example, consider the following the query `link:http//www.example.com/child.html`. For this query to return any results, `www.example.com/` must appear in Follow and Crawl URL patterns.

No other query terms can be specified when using this special query term. Query terms `info`, `link` and `cache` ignore collection restrictions that are specified by the `site` parameter. The search request parameter `as_lq` (see "as_lq" on page 12) can also be used to submit a `link` request.

The query term `link:` returns 25 results as default but you can configure this number by using the **Search > Search Features > Query Settings** page in the Admin Console.

Sample usage:

```
link:www.google.com
```

Boolean OR Search

Google search supports the Boolean `OR` operator. To retrieve pages that include either word A or word B, use an uppercase OR between terms. The search request parameter `as_oq` (see "as_oq" on page 12) can also be used to submit a search for any term in a set of terms.

For additional information on the use of `OR`, see "Usage Notes" in "Using inmeta to Filter by Meta Tags" on page 48.

Sample usage:

```
vacation london OR paris
```

The OR operator takes precedence over the AND operator, so this example will be treated as:

```
vacation AND (london OR paris)
```

Cached Results Page

The query prefix `cache:` returns the cached HTML version of the specified web document that the Google search crawled. Note there can be no space between `cache:` and the web page URL. Words that appear in the query are highlighted in the cached document.

To use Google's default cached result display, omit the `output` parameter in the cache request. To customize the display of cached results, request XML or Custom HTML `output` as part of the cache request and ensure that your parser or stylesheet handles the incoming cache data. Query terms `info`, `link` and `cache` ignore collection restrictions that are specified by the `site` parameter. See also the site parameter (see "site" on page 19).

Sample usage:

```
cache:www.google.com web
```

Date Range Search

Restrict search to documents with modification dates that fall within a time frame. You can search any dates between `1900-01-01` and `2079-06-06`. For a complete list of date formats, see "Acceptable Date Formats" on page 110.

Date range searches by themselves do not return results and must be accompanied by a search term.

Only documents that have a modification date are returned for a daterange query. Documents that do not have modification dates are excluded from the results.

To specify dates in ISO 8601 format (such as YYYY-MM-DD), use two dots (`..`) to separate dates in the date range. For example, to search for documents that contain the word parrot and were modified between August 1, 2008 and December 24, 2008, enter the following statement:

```
parrot daterange:2008-08-01..2008-12-24
```

You can specify that a search be for all modification dates before a date by preceding the date with the two dots. For example, to search for all documents containing parrot that were modified before August 8, 2008, specify the date range with the following statement:

```
parrot daterange:..2008-08-08
```

You can specify that a search be for all documents that were modified after a specific date by specifying a date followed by two dots. For example, to search for all documents that were modified after January 1, 2009 that contain parrot, specify the date range with the following statement:

```
parrot daterange:2009-01-01..
```

To specify how a search appliance sorts search results by document dates, use **Index** > **Document Dates** in the Admin Console. You can sort search results by the dates found in a document's URL, a meta tag, the title, the body, or when the document was last modified. If you choose to sort by a meta tag, the meta tag that you specify can contain only a date.

Dates in Julian format can be treated as a date range only with the `daterange` keyword. Without the `daterange` keyword, Julian dates are considered a number range search. (A Julian date is an integer number of days that have elapsed since noon on January 1, 4713 BC. For example, August 1, 2008 at noon has a Julian date of 2454680.)

For further options for searching dates in meta tags, see "Using inmeta to Filter by Meta Tags" on page 48.

Sample usage:

```
election daterange:2008-01-20..2009-01-20
election daterange:2008-01-20..
election daterange:..2009-01-20
parrot daterange:2452122-2452234
```

Directory Restricted Search

Restrict search to documents within a domain or directory. Enter the query followed by `site:` followed by the host name and path of the web directory. To limit the search to a domain, specify a string that matches a complete name-segment of the canonical host name.

To search a particular directory on a web server (including the root directory), specify a string that is the complete canonical name of the host server followed by the path of the directory. If the forward slash character (/) is at the end of the web directory path specified, then search is limited to the files within that directory. Files in sub-directories are not considered.

The URLs used with `site` must contain fewer than 119 characters. The exclusion operator (-) can be applied to this to remove a web directory from consideration in the search. You can specify one `site` term per search request or multiple site terms using the Boolean OR operator.

The search request parameters `as_sitesearch` (see "as_sitesearch" on page 13) and `as_dt` (see "as_dt" on page 10) can also be used to submit directory restricted searches. See also the site parameter (see "site" on page 19).

Sample usage:

- Domain search examples:

  ```
  site:www.google.com
  site:google.com
  site:com
  ```

- Directory search examples:

  ```
  admission site:www.stanford.edu/group/uga
  site:www.google.com/enterprise/
  site:www.google.com/about
  gxp site:www.corp.google.com/eng/howto OR site:www.corp.google.com/eng/doc
  ```

Exclusion

Sometimes what you're searching for has more than one meaning. For example, the term "bass" can refer to either fishing or music. You can exclude a word from your search by putting a minus sign (-) immediately in front of the term you want to exclude from the search results. Be sure to include a space before the minus character.

The search request parameter `as_eq` (see "as_eq" on page 11) can also be used to submit terms to exclude.

Sample usage:

```
bass -music
```

File Extension Filtering

The query prefix `ext:` filters the results to include only documents with the specified file extension. No spaces can come between `ext:` and the type. For example, `ext:pdf`, which retrieves all documents with the `pdf` extension.

You can combine this prefix with the `filetype` prefix to construct the following types of query : `filetype:pdf AND ext:pdf`, which retrieves all documents with the Mime type `pdf` and with the `pdf` extension.

You can exclude file types by putting a minus sign before `ext`, such as `-ext:pdf`. For more information, see "File Extension Exclusion" on page 27.

Sample usage:

```
whitepaper ext:doc OR ext:pdf
```

File Extension Exclusion

The query prefix`-ext:` filters the results to exclude documents with the specified file extension. No spaces can come between `-ext:` and the specified extension.

You can exclude multiple file types by adding more `-ext:` terms to the search query.

Sample usage:

```
whitepaper -ext:pdf -ext:doc
```

File Type Filtering

The query prefix `filetype:` filters the results to include only documents with the specified MIME content type. No spaces can come between `filetype:` and the type.

You can exclude file types by putting a minus sign before filetype, such as `-filetype:pdf`. For more information, see "File Type Exclusion" on page 27.

See also "as_filetype" and "as_ft" on page 11 for including and excluding documents from the search results.

You can specify multiple file types by adding `filetype:` terms to the search query, combined with the Boolean `OR`.

Sample usage:

```
whitepaper filetype:doc OR filetype:pdf
```

File Type Exclusion

The query prefix`-filetype:` filters the results to exclude documents with the specified file extension. No spaces can come between `-filetype:` and the specified extension.

You can exclude multiple file types by adding more `-filetype` terms to the search query.

Sample usage:

```
whitepaper -filetype:doc
-filetype:pdf
```

Meta Tag Search

You can filter results by meta tags and their values using `inmeta`. Used with the operators ~ or =, `inmeta` restricts results to required or partial meta tag values in the same way as the `requiredfields` and `partialfields` search parameters.

Sample usage:

```
    inmeta:department=Human Resources
```

There is a 128 character limit for inmeta queries.

The 128 characters includes the inmeta term and metatag name/value:

`inmeta:<name>=<value>` => 128 characters in total.

This limit also applies to dynamic navigation. That is, the attribute values displayed in the sidebar cannot exceed 128 characters.

See “Meta Tags” on page 42 for more details.

Number Range Search

To search for documents or items that contain numbers within a range, type your search term and the range of numbers separated by two periods (`..`). You can set ranges for weights, dimensions, prices (dollar currencies only), and so on. Be sure to specify a unit of measurement or some other indicator of what the number range represents.

Sample usage:

```
    pencils $1.50..$2.50
```

Phrase Search

Search for complete phrases by enclosing them in quotation marks or by connecting them with hyphens. Words marked in this way appear together in all results, exactly as you enter them. Phrase searches are especially useful when searching for famous sayings or proper names.

You can also use the `as_epq` search request parameter (see “as_epq” on page 11) to submit a phrase search.

Sample usage:

```
    "yellow pages"
    yellow-pages
```

Text Search (one term)

If you precede a query term with `intext:`, the search appliance restricts the search to documents that contain the search word in the titles or body text of the documents. The search appliance does not search for the query word in the metadata, anchors, or urls.

Sample usage:

```
    intext:google
```

Text Search (all terms)

If you precede a query term with `allintext:`, the search appliance restricts the search to documents whose titles or body text contains the search terms. The search appliance does not search for the query words in the metadata, anchors, or urls. Returns only documents that have the search terms in the title or body text of the document.

Sample usage:

```
allintext:google search
```

Title Search (one term)

If you precede a query term with `intitle:`, Google search restricts the results to documents containing that word in the title.

Putting `intitle:` in front of every word in your query is equivalent to putting `allintitle:` at the front of your query.

For plain text files, the search appliance displays results using the first 70 KB of the file as the title. Because the document does not have a title, the `intitle` special query term does not work for plain text files.

Sample usage:

```
intitle:google
```

Title Search (all terms)

If you precede a query with `allintitle:` Google search restricts the results to those with all of the query words in the result title.

For plain text files, the search appliance displays results using the first 70 KB of the file as the title. Because the document does not have a title, the `allintitle` special query term does not work for plain text files.

Sample usage:

```
allintitle:google search
```

URL Search (one term)

If you precede a query term with `inurl:`, Google search restricts the results to documents containing that word in the result URL. No spaces can come between the `inurl:` and the following word.

The term `inurl` works only on words, not on URL components. In particular, it ignores punctuation and uses only the first word following the `inurl:` operator. To find multiple words in a result URL, use the `inurl:` operator for each word. Preceding every word in your query with `inurl:` is equivalent to putting `allinurl:` at the front of your query.

Sample usage:

```
inurl:Google search
```

URL Search (all terms)

If you precede a query with `allinurl:` Google search restricts the results to those with all of the query words in the result URL.

The term `allinurl` works only on words, not URL components. In particular, it ignores punctuation. Thus, `allinurl: foo/bar` restricts the results to page with the words “foo” and “bar” in the URL, but doesn’t require that they be separated by a slash within that URL, that they be adjacent, or that they be in that particular word order. There is currently no way to enforce these constraints.

Sample usage:

```
allinurl: Google search
```

Web Document Info

The query prefix `info:` returns a single result for the specified URL if the URL exists in the index. No other query terms can be specified when using this special query term. Query terms `info`, `link` and `cache` ignore collection restrictions that are specified by the `site` parameter.

Sample usage:

```
info:www.google.com
```

Wildcard Search

If you precede a query with `wildcard:`, you can search by entering a wildcard pattern instead of the exact spelling of a term. By default, wildcard search is enabled for each front end of the search appliance. However, to use wildcard search, you must ensure that wildcard indexing is also enabled for your search appliance by using the **Index > Index Settings** page in the Admin Console.

Note: If wildcard indexing is disabled, users will experience search issues in which case the search appliance administrator can disable implicit wildcard search on each front-end.

The search appliance supports two `wildcard:` operators:

- *--Matches zero or more characters
- ?--Matches exactly 1 character

The search appliance is able to consider all words with * as wildcard terms, so users don't need to prepend the wildcard: special operator to a pattern that contains this operator. To enable the search appliance to do this, click the **Consider words with * as wildcards by default** checkbox on the **Search > Search Features > Front Ends > Filters** page.

Take note that words that have special characters, such as apostrophes, in them are not matched by wildcard search.

Using wildcards can simplify queries for long names, technical data, pharmaceutical information, or strings where the exact spelling varies or is unknown. A user can search for all words starting with a particular pattern, ending with a particular pattern, or having a particular substring pattern. A wildcard query term must satisfy at least one of the following conditions:

- A sequence to at least 2 characters at the start of a word, for example: go*
- A sequence to at least 2 characters at the end of a word, for example: *le
- A sequence of at least 3 characters anywhere in the word, for example: *ear*

Sample usage:

```
wildcard:test*
wildcard:?nter
```

Wildcard search is also supported for metadata queries, but the `wildcard:` special operator is omitted. For example: `inmeta:name*`. Also, metadata queries are %-encoded, which affects the form of an `inmeta:` wildcard query.

Wildcard search is not supported for other common queries, including `filetype`, `inurl`, `intext`, and so on. Also, wildcard search is not supported with Chinese, Japanese, Korean, or Thai.

Filtering

Google search provides many ways for you to filter the results that are returned from your search query. In addition to the automatic filtering and language filtering described in this section, the search appliance provides filtering by query parameters (see "Search Parameters" on page 10), query terms (see "Query Terms" on page 22) and meta tags (see "Meta Tags" on page 42), which are documented in their respective sections.

Automatic Filtering

Google uses automatic filtering to ensure the highest quality search results.

Google search uses two types of automatic filters:

- **Duplicate Snippet Filter**—If multiple documents contain identical titles as well as the same information in their snippets in response to a query, only the most relevant document of that set is displayed in the results.
- **Duplicate Directory Filter**—If there are many results in a single web directory, then only the two most relevant results for that directory are displayed. An output flag indicates that more results are available from that directory.

By default, both of these filters are enabled. You can disable or enable the filters by using the `filter` parameter settings as shown in the table.

Filter value	Duplicate Snippet Filter	Duplicate Directory Filter
`filter=1`	Enabled (ON)	Enabled (ON)
`filter=0`	Disabled (OFF)	Disabled (OFF)
`filter=s`	Disabled (OFF)	Enabled (ON)
`filter=p`	Enabled (ON)	Disabled (OFF)

When a search filter is enabled and removes some results, the search results output indicates that results were filtered. See "Appendix A: Estimated vs. Actual Number of Results" on page 106 for more information about how a filtered result set is identified and for recommendations for displaying the results.

Although the `filter=0` option exists, Google recommends against setting `filter=0` for typical search requests, because filtering significantly enhances the quality of most search results.

For queries that contain the `site` special query term or the `as_sitesearch` query parameter, automatic filtering does not take place.

When the Google Search Appliance filters results, the top 1000 most relevant URLs are found before the filters are applied. A URL that is beyond the top 1000 most relevant results is not affected if you change the filter settings.

Language Filters

Language filters limit a search to pages in the specified languages. The Google Search Appliance has built-in language filters that detect the language of a query and return appropriate results. You can combine language filters to further restrict search results.

Note: When the search appliance receives a language-restricted search request for which there are no results in the languages specified by a filter, it displays search results in all languages.

This section covers:

- "Automatic Language Filters" on page 32
- "Combining Language Filters" on page 34

Automatic Language Filters

The Google Search Appliance automatically detects the language of each search query and returns results in that language. For example, if a user submits a search query in Hungarian (lang_hu), results are automatically returned in Hungarian.

The algorithm for automatically determining the language of a web document is not customizable. The language of a document is determined primarily by the language used for the majority of the text in the body of the document.

Note: Encoding schemes for the input and output of search requests are also important when you provide international search. For more information on encoding, see "Internationalization" on page 35. For more information on how language filtering works with Simplified Chinese and Traditional Chinese, see "Language Filtering for Traditional and Simplified Chinese" on page 34.

The automatic language filters are:

Language	Automatic Language Filter Name
Arabic	lang_ar
Chinese (Simplified)	lang_zh-CN
Chinese (Traditional)	lang_zh-TW
Czech	lang_cs
Danish	lang_da
Dutch	lang_nl
English	lang_en
Estonian	lang_et
Finnish	lang_fi
French	lang_fr
German	lang_de
Greek	lang_el
Hebrew	lang_iw
Hungarian	lang_hu
Icelandic	lang_is
Italian	lang_it
Japanese	lang_ja
Korean	lang_ko
Latvian	lang_lv
Lithuanian	lang_lt
Norwegian	lang_no
Portuguese	lang_pt
Polish	lang_pl
Romanian	lang_ro
Russian	lang_ru
Spanish	lang_es
Swedish	lang_sv
Turkish	lang_tr

If you want to filter languages other than the above, obtain the language code from ISO 639 (see http://www.loc.gov/standards/iso639-2/php/code_list.php), index a document corpus containing the desired languages, and run tests to determine that the search results are as expected.

Language Filtering for Traditional and Simplified Chinese

The search appliance determines the encoding of a search query and uses that encoding to return search results. For example, if a user enters a search query using Traditional Chinese, the search results are returned in Traditional Chinese. If a query is entered using Simplified Chinese, the results are also in Simplified Chinese. The original encoding of the documents does not affect what is returned. If documents encoded in Traditional Chinese are crawled and a Simplified Chinese query is entered, the documents returned are encoded in Simplified Chinese.

However, if a search query uses characters that are common to both Simplified and Traditional Chinese, the search appliance's behavior is indeterminate. In some cases, the search appliance detects such queries as Simplified Chinese, but in other cases, the language is detected as Traditional Chinese. One example of a query that returns indeterminate results is the term Hong Kong. To resolve this issue, use the lr parameter to specify whether you want to enforce Traditional Chinese (lang_zh-TW) or Simplified Chinese (lang_zh-CN).

Combining Language Filters

Search requests that use the `lr` parameter support the Boolean operators identified in the following table in order of precedence.

Boolean Operator	Sample Usage	Description
Boolean NOT [-]	`-lang_fr`	Removes all results that are defined as part of the Language Filter immediately following the – operator. The example `lr` value would remove all results in French.
Boolean AND [.]	`gloves.hats`	Returns results that are in the intersection of the results returned by the collection to either side of the dot operator. The example `restrict` value returns results which are in both the "hats" and "gloves" custom collections.
Boolean OR [|]	`lang_en|lang_fr`	Returns results that are in either of the results returned by the collection to either side of the pipe operator (|). The example `lr` value returns results matching the query that are in either French or English.
Parentheses [( )]	`(gloves).(-(lang_hu|lang_cs))`	All terms within the innermost set of parentheses are evaluated before terms outside the parentheses are evaluated. Use parentheses to adjust the order of term evaluation. The example `lr` value returns all results in the "gloves" custom collection that are not in either the Hungarian or Czech collections.

Note: Spaces are not valid characters in the collection string.

Internationalization

To support searching documents in multiple languages and character encodings, Google provides the **ie** and **oe** parameters. The **ie** parameter indicates how to interpret characters in the search request. The **oe** parameter indicates how to encode characters in the search results. To appropriately decode the search query and correctly encode the search results, supply the correct **ie** and **oe** parameters, respectively, in the search request.

Note: When you are providing search for multiple languages, Google recommends using `utf8` encoding value for the **ie** and **oe** parameters.

Examples

Example 1. The following search request interprets the search query “gloves” using `latin1` encoding, searches for English or French results, and returns results using `latin1` encoding:

```
GET /
search?q=gloves&client=test&site=test&lr=lang_en|lang_fr&ie=latin1&oe=latin1
```

Example 2. This request interprets the search query “gloves” using `latin2` encoding, searches for results which are not in Hungarian or Czech, and returns results using `latin2` encoding:

```
GET /search?q=gloves&client=test&site=test&lr=(-lang_hu).(-
lang_cs)&ie=latin2&oe=latin2
```

Example 3. This request interprets the search query “gloves” using `utf8` encoding, searches for results which are in Simplified or Traditional Chinese, and returns results using `utf8` encoding:

```
GET /search?q=gloves&client=test&site=test&lr=lang_zh-CN|lang_zh-
TW&ie=utf8&oe=utf8
```

Note: For information on language-specific searches that use the lr parameter, see “Language Filters” on page 32.

Character Encoding Values

Here is a list of encoding values that can be used with the parameters **ie** and **oe**:

Language	Encoding Value	Alternate Encoding Value
Chinese (Simplified)	gb	GB2312
Chinese (Traditional)	big5	Big5
Czech	latin2	ISO-8859-2
Danish	latin1	ISO-8859-1
Dutch	latin1	ISO-8859-1
English	latin1	ISO-8859-1
Estonian	latin4	ISO-8859-4
Finnish	latin1	ISO-8859-1
French	latin1	ISO-8859-1
German	latin1	ISO-8859-1
Greek	greek	ISO-8859-7
Hebrew	hebrew	ISO-8859-8
Hungarian	latin2	ISO-8859-2
Icelandic	latin1	ISO-8859-1
Italian	latin1	ISO-8859-1
Japanese	sjis	Shift_JIS
Japanese	jis	ISO-2022-JP
Japanese	euc-jp	EUC-JP
Korean	euc-kr	EUC-KR
Latvian	latin4	ISO-8859-4
Lithuanian	latin4	ISO-8859-4
Norwegian	latin1	ISO-8859-1
Portuguese	latin1	ISO-8859-1
Polish	latin2	ISO-8859-2
Romanian	latin2	ISO-8859-2
Russian	cyrillic	ISO-8859-5
Spanish	latin1	ISO-8859-1
Swedish	latin1	ISO-8859-1
Turkish	latin3	ISO-8859-3
Turkish	latin5	ISO-8859-9
Unicode (All Languages)	utf8	UTF-8

Sorting

Google search provides three sorting options for search results:

- "Sort By Relevance (Default)" on page 37
- "Sort By Date" on page 37
- "Sort by Metadata" on page 38

Sort By Relevance (Default)

By default, Google combines hypertext-matching analysis and PageRank technologies to provide users with highly relevant results. Hypertext-matching analysis uses the design of the page, examining over 100 factors to determine the best result for your query term. PageRank considers the link structure of the entire index to understand how each page links to the other pages in the index.

Sort By Date

Google search engine can order search results by date in ascending or descending order. The date of a web document is defined by parameters configured by the search administrator. When a search request uses the sort-by-date feature, the date associated with each result document is used to determine the order of the results. Take note that the search appliance ignores the time of day for sorting, even if it's given by the "last-modified" date or other attributes.

When using the sort-by-date feature, the built-in filter of duplicate directories and duplicate snippets will group the highest result (newest or oldest depending on the sort order) with similar results regardless of their date. This can be disabled by adding the `filter=0` parameter to the search request when performing search by date.

Note: When sorting by date, the order of the results can also be effected by any relevant result biasing policies that are being used. See "Using Result Biasing to Influence Result Ranking" in *Creating the Search Experience.*

Example

The following request returns the first 10 top results that match the query "chicken teriyaki" in the "test" collection:

```
GET /search?q=chicken+teriyaki&output=xml&client=test&site=test&sort=date:D:S:d1
```

Results are sorted by date and relevancy.

Details

To sort the results by date, include the sort parameter in the search request, formatted as follows:

```
date:<direction>:<mode>:<format>
```

The following tables shows the possible values for `<direction>`, `<mode>`, and `<format>`.

<direction> Value	Description
A	Sort results in ascending order.
D	Sort results in descending order.

<mode> Value	Description
S	Return the 1000 most relevant results, sorted by date.
R	Get all results, sort by date, and return the first 1000 results. You can use this option when freshness is more important than relevancy. Do not use this filter if your collection contains more than 50,000 documents.
L	Return the date information for each result. No sorting is done.

<format> Value	Description
d1	The format of the value returned for each search result is set to YYYY-MM-DD.

Sort by Metadata

The Google Search Appliance can order search results by values that are included in individual documents. This makes it possible to sort documents by prices, dates, authors, or any other value that is relevant for your documents. The sorting occurs only on the 1000 most relevant results for the specific query.

When sorting by metadata, the total length of the metadata attr:value pair cannot exceed 121 characters. Exceeding the maximum character limit causes results to be unsorted. For more information, see “Meta Tags” on page 42.

When using the sort-by-metadata feature, the automatic quality filter sometimes re-orders results when performing result grouping. This can be disabled by adding the filter=0 parameter to the search request when performing search by date.

Using this feature will cause search performance to decrease. The performance decrease depends on, but is not limited to, the following factors

- How many results are returned
- How much metadata exists for each document
- The sorting options specified

The value used to sort each document is available in the XML output in the FS tag.

How Sorting Works

When a search request is submitted with the sorting parameter specified as described in the following sections, the Google Search Appliance retrieves the value corresponding to the given meta tag name for each search result. In some instances, as described below, some processing of the value will occur. These values will then be sorted according to the specific parameter specified. If two documents have the same value, they will be ordered according to their original relevance.

- *Multivalued metadata*—If you have specified certain meta tags as multivalued in **Index > Index Settings**, then only the first value in those tags is used for sorting.
- *Multiple meta tags with the given name*—If a document has multiple meta tags with the given name, all of which have the exact same value, then that value will be used for sorting. If the multiple meta tags have different values, then none will be used and the document will behave as if it has no meta tag with the given name.
- *No meta tag with the given name for a document*—If a document does not contain a meta tag with the given name, then it will be placed after documents that do have meta tags with the given name. All of these documents without such meta tag will be ordered by their original relevancy ranks.
- *Date*—If a meta tag has been specified as a date in Index > Index Settings, then the value will be normalized into a YYYY-MM-DD format before being sorted.

Details

To sort results by metadata, include the sort parameter in the search request, formatted as follows:

```
meta:<name>:<direction>:<mode>:<language>:<case>:<numeric>
```

All values after the name are optional and can be left blank. For instance, if you want to specify only the name and the language, you could use the following format:

```
meta:<name>:::<language>
meta:<name>:::<language>::
```

The following tables show the possible values for the options.

<name> Value	Description
any string	The name of the meta tag that should be used to sort by. This string must be double-URL-encoded.

<direction> Value	Description
A	Sort results in ascending order. Default.
D	Sort results in descending order.

<mode> Value	Description
E	Return the 1000 most relevant results, then sort by metadata. Default.
ED	Same as mode E, but also sort dates chronologically. Supported in GSA version 7.2.0.G.230 and later.
S	Return the 1000 most relevant results, then sort by metadata, then apply Advanced Score Reporting, Unification biasing, and filtering.
SD	Same as mode S, but also sort dates chronologically. Supported in GSA version 7.2.0.G.230 and later.

<language> Value	Description
any ISO 639-1 code	A 2-character language code indicating the language rules to use to sort. en is the default.

<case> Value	Description
D	Do not consider case when sorting. Default.
U	Sort uppercase version of a letter before the lowercase version of that letter. Note that this does not sort all uppercase characters before all lowercase characters.
L	Sort lowercase version of a letter before the uppercase version of that letter.

<numeric> Value	Description
D	Numeric sorting is disabled, so 123 comes before 2 because 1 is less than 2. Default. The order is ascending (the default). Examples: 123 2 34 ABC123XYZ ABC2XYZ ABC34XYZ
Y	Numeric sorting is enabled, so 2 comes before 123 because 2 is less than 123. This only sorts positive integers, but does classify numbers inside longer alphanumeric strings. So ABC2XYZ will come before ABC123XYZ. Examples: 2 34 123 ABC2XYZ ABC34XYZ ABC123XYZ
F	This is similar to Y, but also identifies and sorts negative and floating-point numbers. It will identify proper punctuation based on the language specified, so a decimal point is a . for English, but a , for German.
N	Can be used to sort pure English numbers (only containing digits and +-. punctuation) faster than using Y or F. But values like ABC2XYZ will not be sorted.

For more information about the language options, case options, and numeric options D and Y, see the ICU Collation documentation.

Examples

The following examples show sort parameters and how values would be sorted.

Sorting authors or other alphabetic values

sort=meta:<name>	Agatha Christie C. S. Lewis Henry David Thoreau
sort=meta:<name>:D	Henry David Thoreau C. S. Lewis Agatha Christie

Change case sorting

sort=meta:<name>:A:E:en:D	aa aA Aa Ab
sort=meta:<name>:A:E:en:U	Aa aA aa Ab
sort=meta:<name>:D:E:en:L	aa aA Aa Ab
sort=meta:<name>:D:E:en:U	Ab aa aA Aa

Sort numbers that have the same format

If all the numbers have the same format (for instance, if every number has a dollar sign, then two digits, then a comma, then three digits, then a period, then two digits: $xx,xxx.xx).

sort=meta:<name>:::::Y	$34,827.45 $84,671.11 $93,243.55

Sort currency

sort=meta:<name>:::::F	$0.01 $1.00 $34 $1,234.56

Sort English-looking numbers

This is a very fast option, if all the numbers are in the following formats.

sort=meta:<name>::::N	-12345
	-1234.56
	-3
	0
	34.9
	+35.172
	+321
	16003.58

Sort dates

sort=meta:<name>::ED	January 30, 2012
	February 18, 2013
	October 2, 2013

Sort date-like words

Dates in word format are sorted alphabetically.

sort=meta:<name>::E	February 18, 2013
	January 30, 2012
	October 2, 2013

Meta Tags

The Google Search Appliance provides search parameters and special query terms that enable you to leverage the meta tags that are available in your content. These make it possible to find matches specifically in meta data content, rather than content occurring anywhere in the document.

There are no restrictions on the number of meta tag matches a results page can display.

The maximum number of characters for a metadata attr:value pair for the search request parameters `requiredfields` and `partialfields` and special query term `inmeta` is 121. If the combination of the metadata attr:value is bigger than 121 characters then those particular meta tag contents will be visible at serving time although not searchable via the search request. Also, it will not be visible in dynamic navigation.

Take note that if the metatada attr:value contains term values that have less than 121 characters, those terms may still be searchable via the `partialfields` request parameter and the `inmeta` special query term using the ~ operator (see the example below).

At search time, if the encoded value of the search attribute (`requiredfields`, `partialfields`, `inmeta`) plus the attr:value is greater than 121, then the search won't produce any results. For example, the following meta tag value is not searchable because the encoded value of the search attribute plus the combination of attr:value is greater than 121 characters:

```
<meta name="gamc"
content="12345678901234567890123456789012345678901234567890123456789012345678901
2345678901234567890123456789012345678901234567890"/>
```

In the following example, the term 1234 would be searchable using a partialfields or inmeta request, as the attr:value meets the 121 character limit:

```
<meta name="gamc2" content="1234
1234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890"/>
```

This section describes the following methods of using meta data:

- Requesting meta tag values using the `getfields` parameter (see “Requesting Meta Tag Values” on page 43)
- Filtering by meta tags using the `requiredfields` or `partialfields` parameters (see “Filtering by Meta Tags” on page 44)
- “Using inmeta to Filter by Meta Tags” on page 48

Requesting Meta Tag Values

Use the `getfields` parameter in a search request to specify meta tag values to return with the search results. The search engine returns only meta tag information for results that actually contain the meta tags. The search for meta tags is case-insensitive. Use only whole words in the `getfields` parameter, not partial words or word “stems.” There are limits to the number of characters returned for each meta tag when using `getfields`. The character limits include the meta tag name and content. These are the limits:

- For Latin characters: name + value = 1500 characters; chars_AND_name <= 1500/2
- For characters in multibyte languages (Japanese, Chinese, and Korean): 500 characters

Usage

```
GET /search?q=[search term]&output=xml&client=test&site=test&getfields=[meta tag
name]
```

Example

The following search request returns the first 10 results that match the query “books” in the “test” collection:

```
GET /
search?q=books&output=xml&client=[test]&site=[test]&getfields=author.title.keywo
rds
```

If any of the results contain the `author`, `title` or `keywords` meta tags, then the values of those meta tags are returned with the respective results. For example, the following tags could be returned with this search request:

```
<meta name="author" content="Jakob Nielsen">
<meta name="title" content="Usability Engineering">
<meta name="keywords" content="Usability, User Interface, User Feedback">
```

Details

To specify multiple meta tag values to be returned, list all meta tag names separated by a period (`.`) as in the first example. To request all available meta tags for each search result, specify an asterisk (`*`) as the value for the `getfields` parameter.

When meta tag values are requested, they are not displayed in results requested in the default HTML format. You can use the custom HTML or XML output options, or set the XSLT variable `show_meta_tags` to display meta tags in results. For more information, see “Advanced Customization Topics” in *Creating the Search Experience*.

All specified meta tag names and values must be double URL-encoded (see “Appendix B: URL Encoding” on page 108).

Filtering by Meta Tags

The search appliance can filter results by the values of the results’ meta tags. This section describes how to use the `requiredfields` and `partialfields` input parameters to filter results using meta tag values. You can use these parameters to include only search results that contain specified meta tag values.

The term `partialfields` refers to part of the meta tag content, rather than part of a word. For information on other filtering techniques, see “Filtering” on page 31.

You can use the operators in the following table when filtering by meta tags.

Operator	Description
AND (`.`)	Include results when both filters are true.
OR (`|`)	Include results when at least one filter is true.
NOT (Exclusion) (`-`)	Exclude from the result set any results that contain the specified meta tag condition.

A search can be performed to find all documents containing a set of words and/or metadata, such as A AND B AND C. These terms can also be negated, such as A AND B AND NOT C. The search appliance can also use OR conditions for querying.

Usage

```
GET /search?q=[search term]&output=xml
                           &client=test
                           &site=test
                           &requiredfields=[meta tag name]:[meta tag content]
```

The `q=` parameter is optional when using `requiredfields` or `partialfields` parameters, however, the whole query needs to have at least one positive term, be it part of the query or in the metadata restricts.

Examples

Example 1:

The following search request returns the first 10 results that match the query “checks” in the “test” collection and also contain either of the following meta tags (the %2520 operator in the GET statement shows double encoding where %20 (space) is double encoded so that the % character (hexadecimal 25) is appended to the hexadecimal 20):

```
<META NAME="department" CONTENT="Human Resources">
<META NAME="department" CONTENT="Finance">

GET /search?q=checks&output=xml&client=test
                              &site=test
                              &requiredfields=department:Human%2520Resources|dep
artment:Finance
```

Example 2:

The following search returns the first 10 results that match the query “checks” in the “test” collection that do NOT contain the following meta tag:

```
<META NAME="department" CONTENT="Engineering">

GET //search?q=checks&output=xml&client=test
                              &site=test
                              &requiredfields=-department:Engineering
```

Example 3:

The following search request returns the first 10 results that match the query “books” in the “test” collection, and also contain the word “Scott” somewhere in the “author” meta tag. Some example meta tags that satisfy this search request are:

```
<META NAME="author" CONTENT="Sir Walter Scott">
<META NAME="author" CONTENT="F. Scott Fitzgerald">

GET /search?q=books&output=xml
                  &client=test
                  &site=test
                  &partialfields=author:Scott
```

Details

Multiple meta tag constraints can be specified using the `requiredfields` and `partialfields` parameters. To filter for the presence of a meta tag, indicate the name of the meta tag to be found. To filter on a specific meta tag value, indicate the name of the meta tag followed by the colon “`:`” character and then the specific value. The `partialfields` parameter matches complete words, not parts of words.

To combine multiple name-value pairs, use the following Boolean operators.

- **Boolean OR [|]**

  Returns results that satisfy either meta tag constraint.

  Example: `department:Sales|department:Finance`

- **Boolean AND [.]**

  Returns results that satisfy both meta tag constraints.

  Example: `author:William.author:Jones`

- **Combined OR and AND with [( )]**

  Evaluates conditions in parentheses first: `(department=Sales OR department=Finance) AND (author=Williams OR author=Jones).`

  Example: `(department:Sales|department:Finance).(author:William|author:Jones)`

Boolean operators are left associative with equal precedence. You can use parentheses to change the order of precedence. For example, A . (B | C | D) evaluates the OR (|) operators in the parentheses before the AND (.) operator. It is advisable to use brackets, braces, or parenthesis to clarify the precedence in complex queries.

Nested Boolean Filtering Using Meta Tags

Using the Google Search Appliance, the user can search over the meta tags in documents by writing complex queries using AND, OR, NOT operators nested within each other. Using nested metadata queries gives the user more power with the expressive capabilities of search requests.

Arbitrarily nested boolean queries can be written using `requirefields` and `partialfields` in conjunction with AND (.), OR (|), and NOT (-) operators. However, there is no way to specify range search with `requirefields` and `partialfields`, as noted. Nested boolean queries cannot be used with `inmeta`. Because precedence cannot be specified in the search box, when you use `inmeta`, the normal precedence operators take over and the query is executed. However, a single query can include both `inmeta` for range search and a nested boolean statement using `requirefields` and `partialfields`.

Before executing a search, the search appliance simplifies the search query by pushing NOTs down the query tree. This process is an application of De Morgan’s Laws and Double Negation Elimination. As a result of this process, if there are any NOT nodes in the query, they are just above the leaf nodes.

For example, consider the following query:

NOT (a OR b)

The following simplified query is the result of pushing NOTs down the query tree. The NOT nodes are above the leaf nodes:

(NOT a) AND (NOT b)

Not all combinations of operators are valid for searches. The following queries in a simplified query tree are invalid:

- A query in which there is any OR node with even a single child as a NOT node.
- A query in which there is any AND node with all children as NOT nodes.

The following table contains examples of invalid queries.

Query	Simplified Query	Reason
NOT(a OR b OR c)	(NOT a) AND (NOT b) AND (NOT c)	Invalid because AND has 3 children (NOT a, NOT b, NOT c) and all are NOT’ed
a OR b OR NOT (NOT (NOT c))	a OR b OR (NOT c)	Invalid because OR has one child, which is a NOT

Searches with unsupported expressions are not performed and do not return results.

Non-Alphanumeric Characters

By default, non-alphanumeric characters in a `partialfields` query separate the query terms in the same way as space characters. Generally use spaces as separators even when the original content used different content as a separator. For example if you were trying to do a `partialfields` query for the following meta tag:

```
<meta name="part" content="aaa-bbb+ccc*ddd-fff">
```

You should use queries like:

```
partialfields=part:aaa%20bbb
partialfields=part:bbb%20ccc
```

The following non-alphanumeric characters are exceptions:

Character	Description

Decimal point (.)	A double URL-encoded (see "Appendix B: URL Encoding" on page 108) decimal point can act as a decimal point in a number (for example, 250.01). For example to query for a meta tag like:
	`<meta name="number" content="1.1222">`
	Use a `partialfields` query like:
	`partialfields=number:1%252E1222`
	When a meta tag contains a decimal point with no numbers use the space as a separator as previously mentioned. For example for a meta tag like this:
	`<meta name="pet" content="dancing.parrot">`
	Use a `partialfields` query like (%2520 is a double URL-encoded space character):
	`partialfields=pet:dancing%2520parrot`
	If a meta tag contains a number that has letters immediately before or after it, a space should be used as a separator. For example, in the meta tag:
	`<meta name="serialnumber" content="A1.2"`
	Use a `partialfields` query like:
	`partialfields=serialnumber:A1%202`

Ampersand (&)	Not treated as a separator. For example for the meta tag:
	`<meta name="letters" content="a&b">`
	Use a `partialfields` query like this (%2526 is a double URL-encoded ampersand character):
	`partialfields=letters:a%2526b`

Underscore (_)	Not treated as a separator. For example for the meta tag:
	`<meta name="letters" content="a_b">`
	Use a `partialfields` query like this:
	`partialfields=letters:a_b\`

Using inmeta to Filter by Meta Tags

The special query term `inmeta` provides meta tag filtering directly from the search box. In combination with simple operators, `inmeta` filters by meta tags in the same way as the `requiredfields` or `partialfields` search parameters. You can further refine `inmeta` filtering using the double-period (`..`) separator and the `daterange` query term to search by number and date range. (For more information, see "Query Terms" on page 22.)

The special query term `inmeta` and relevant search parameters map to each other in this way:

inmeta Syntax	Search Parameter Syntax	Description
`inmeta:[meta tag name]`	`&requiredfields=[meta tag name]`	Returns results that contain the specified meta tag.
`inmeta:[meta tag name]=[meta tag content]`	`&requiredfields=[meta tag name]:[meta tag content]`	Returns only results that match the exact meta tag content value specified.

`[meta tag name]~[meta tag content]`	`&partialfields=[meta tag name]:[meta tag content]`
	
	`&requiredfields=[meta tag name]?[meta tag content]`
		Returns results that have the specified meta tag with a value that matches some or all of the specified meta tag content (that is, the partial value).

`inmeta:[meta tag name]~[partial value]*`	`&requiredfields=[meta tag name]:[partial value]*`	Returns results that have the specified meta tag name with a value that matches the wildcard search for all of the specified meta tag content (that is, content starting with a match on the partial value).
`inmeta:[meta tag name]~*[partial value]*`	`&requiredfields=[meta tag name]?*[partial value]*`	Returns results that have the specified meta tag name with a value that matches the wildcard search for some or all of the specified meta tag content (that is, content that includes a match on the partial value).

Usage Notes:

1. By default documents that contain ALL query terms are returned. This behavior is similar to a boolean AND. Note though that there is no AND query term. It is the default way of processing query terms. The default behavior can be changed by using the boolean or query term OR or the boolean not query term '-'. Also note that it is not possible to use the NOT operator in an OR statement, for example test OR -test1. Also, there is no way to do nesting of boolean logic using parenthesis.

2. The `OR` keyword separating query terms in which a date or numeric range appears returns inconsistent results.

   Examples:

   The following example returns one result when each portion of the query already returns one different result:

   ```
   inmeta:TainoParrot6:1..244227 OR inmeta:TainoParrot6=244228
   ```

   The following example returns two results and both are correct:

   ```
   inmeta:TainoParrot6=244227 OR inmeta:TainoParrot6=244228
   ```

   The following example returns 112 results when `empty` alone returns 112 results and the number range query returns 3 results:

   ```
   empty OR inmeta:TainoParrot6:244227..244229
   ```

   The following example returns 113 results and is correct:

   ```
   empty OR inmeta:TainoParrot6=244228
   ```

   The following example returns three results when `yvette` alone returns the same results and no results from the date range query appear:

   ```
   yvette OR inmeta:TainoParrot6:1..244228
   ```

3. An `OR` of two `inmeta` range terms does not return results.

   If a set of documents each contain a meta tag with numerical content declared, whether in fixed point notation (with a period) or integer notation, two `inmeta` range searches that return results in isolation do not return results when combined with an `OR`. For example, if one document contains `<META NAME="price" VALUE="20.00">` and another contains `<META NAME="price" VALUE="40.00">`, the search `inmeta:price:15..25` returns the first document, while the search `inmeta:price:35..45` returns the second document. However, the search `inmeta:price:15..25 OR inmeta:price:35..45` does not return results.

   If you have two `inmeta` range searches that in isolation return result sets that overlap, combining them with `AND` returns the intersection of those sets correctly, regardless of the notation used for the meta tag content or the range search itself.

4. An `inmeta` search for a number range returns results only when a number contains six or fewer digits.

   For example, if a document contains a meta tag of `<meta name="NumDateRange" content="20081230">`, then a search query of `inmeta:NumDateRange=20081230` works correctly, or a search where the six significant digits are respected, such as querying for `inmeta:NumDateRange=1..101230`. You can use a six digit number for dates with two digits for the year, two digits for the month, and two digits for the day. If a search is made where the range includes more than six digits, then no results occur, such as with `inmeta:NumDateRange=20081201..20101231`.

5. An `inmeta` search for a number range is unable to handle negative numbers and ignores them.

6. An `inmeta` search is unable to search by multiple keywords or perform phrase searches.

   For example, consider the following meta tags:

   ```
   <meta name="department" content="Human Resources">
   <meta name="department" content="Finance">
   ```

   The following query does not work correctly:

   ```
   checks inmeta:department=Human+Resources+OR+checks inmeta:department=Finance
   ```

   Instead, use multiple inmeta query terms, for example:

   ```
   inmeta:department~Human OR inmeta:department~Finance
   ```

7. Special characters in metadata names must be escaped for use in `inmeta`.

   For example, to match metadata tag `<meta name="s.pos" content="orange" />`, the following query will not work:

   ```
   inmeta:s.pos~orange
   ```

   You must use the following query, in which the special character is escaped:

   ```
   inmeta: s%2Epos~orange
   ```

8. An `inmeta` search of meta text with special characters, such as "." and using the operator "~" doesn't work, but using operator "=" with the full meta text does work.

9. When using `daterange` or `inmeta` queries, spelling suggestions are not returned.

   To view spelling suggestions, use the `requiredfields` parameter instead of `inmeta`.

10. When the search appliance indexes a Microsoft Office 2007 Word document, the following metadata in meta tags becomes available for `inmeta` search queries:

    ```
    <meta name="Author" content="Polly Hedra"></meta>
    <meta name="Keywords" content="Resume"></meta>
    <meta name="last saved by" content="Ray Polanco"></meta>
    <meta name="revision number" content="1"></meta>
    <meta name="last print date" content="5/27/2009 14:03:00"></meta>
    <meta name="creation date" content="4/27/2009 13:15:00"></meta>
    <meta name="Last Saved Date" content="4/27/2009 13:44:00"></meta>
    <meta name="template" content="Taino Parrot Resume Template.dotx"></meta>
    <meta name="edit minutes" content="23"></meta>
    <meta name="page count" content="3"></meta>
    <meta name="word count" content="220"></meta>
    <meta name="character count" content="1512"></meta>
    <meta name="source" content="Microsoft Office Word"></meta>
    <meta name="security" content="0"></meta>
    <meta name="Count Lines" content="12"></meta>
    <meta name="Count Paragraphs" content="3"></meta>
    <meta name="Scale Crop" content="no"></meta>
    <meta name="company" content="Coqui Parrot Inc."></meta>
    <meta name="links up to date" content="no"></meta>
    <meta name="Count Characters with Space" content="1729"></meta>
    <meta name="shared doc" content="no"></meta>
    <meta name="Links Dirty" content="no"></meta>
    <meta name="Application Version" content="12.0000"></meta>
    ```

11. Metadata can have multiple attributes with the same name. For example:

    ```
    <metadata>
      <meta name="Name" content="Jenny Wong"/>
      <meta name="Phone" content="x12345"/>
      <meta name="Phone" content="x789"/>
      <meta name="Floor" content="3"/>
    ```

    If multiple values are available and if any of the attribute values match the search query, a link to the document appears in the search results.

12. While inmeta supports wildcard search, it does not support boolean logic. Use `requiredfields` instead to combine wildcard search and boolean logic.

Examples

Example 1. These first query examples show how search requests are related to meta tags in the following example of a web page.

```
<html>
   <head>
      <title>My Title</title>
      <meta name="myFloat" content="1.23456">
      <meta name="myInteger" content="8" />
      <meta name="myCurrency" content="123.45" />
      <meta name="myDate" content="2011-03-05" />
   </head>
   <body>
      Hello world.
   </body>
</html>
```

The following search request is for a match to the lower bound value within the currency range:

```
inmeta:mycurrency:60.00999..
```

The following search request is for an exact match to the float value:

```
inmeta:myfloat:1.23456..1.23456
```

The following search request is for an exact match to a date or date range:

```
inmeta:mydate:2011-03-05..2011-03-05
- or -
inmeta:mydate:daterange:2011-03-05..2011-03-05
```

Example 2. The following search request returns results that contain the word “Scott” somewhere in the “author” meta tag. Some example meta tags that satisfy this search request are:

```
<meta name="author" content="Sir Walter Scott">
<meta name="author" content="F. Scott Fitzgerald">

books inmeta:author~Scott
```

Example 3. The following search request returns results that contain “size” meta tag values between 30 and 50 inches:

```
flat+panel+TV inmeta:size:30..50
```

Example 4. The following is an open-ended date range search request that returns results containing “date” meta tag values later than 2007-01-01:

```
Monica inmeta:date:daterange:2007-01-01..
```

Date meta tags must contain only the date information. If you want to filter by date meta tags, make sure the meta tag content fields do not contain any information other than a date.

Limitations

For information about search request limitations, see Specifications and Usage Limits.

Chapter 3

Results Format

This section covers the following topics:

- "Custom HTML" on page 53
- "XML Output" on page 54

Custom HTML

This section describes the custom HTML results.

- "Custom HTML Output Overview" on page 53
- "Internationalization" on page 54

Custom HTML Output Overview

Google Search Appliance has a built-in XSLT (eXtensible Stylesheet Language Transformation) server, and can generate custom HTML using your XSL stylesheet. Search requests that include the `output` parameter set to `xml_no_dtd` and a valid `proxystylesheet` parameter value are automatically processed by the XSLT server as requests for custom HTML output.

Using the XSL stylesheet specified by the `proxystylesheet` parameter, the XSLT server applies the transformation rules found in the XSL stylesheet to the standard Google XML results. Although this document assumes that the output generated by applying the XSL stylesheet is HTML, almost any output format can be generated by using appropriate XSL stylesheet rules. For any front end, the default XSL stylesheet can be customized or replaced by the search administrator.

To customize the XSL stylesheet used to generate custom HTML output, see "XML Output" on page 54 to determine the XML tags that may be transformed using a customized XSL stylesheet.

Additionally, you can leverage the `proxycustom` parameter to pass custom XML tags to the XSLT server. Because including custom XML does not generate search results, this feature is useful for implementing additional static search pages, such as an advanced search page.

Customizations to XSLT stylesheets may result in vulnerability to cross-site scripting (XSS) attacks. Google recommends that you run XSS test after customizing an XSLT stylesheet.

Notes:

- XSL stylesheets used by the XSLT server are cached for 15 minutes. To force the XSLT server to use the latest version of an XSL stylesheet, set the `proxyreload` input parameter to a value of `1` in your search request.
- XSL stylesheets that include other files may not be used with the Google search engine. An XSL stylesheet that contains the following tags generates an error result:
  - `<xsl:import>`
  - `<xsl:include>`
  - `xmlns:`
  - `document()`
- When you request cached results in custom HTML output, the `BLOB` XML tag and associated value are automatically converted to the original text before the XSL stylesheet rules are applied. When using an XSL stylesheet that customizes cache results, simply use the values of the `CACHE_LEGEND_TEXT`, `CACHE_LEGEND_NOTFOUND` and `CACHE_LEGEND_HTML` XML tags directly instead of applying a rule on the `BLOB` subtag.
- If you use input or output encodings other than `latin1`, see “Internationalization” on page 54 for more details.
- More information about XSL and XSLT can be found on the W3C (http://www.w3.org/Style/XSL/) web site.

Internationalization

The Google Search Appliance handles over 20 character encoding schemes. This section discusses special considerations for the custom HTML output format with encoding schemes other than `latin1`.

To support all the encoding schemes supported by Google, the XSLT server follows a process to ensure that the results are returned in the correct encoding scheme. When requesting search results through the XSLT server, the server translates the results to the UTF8 encoding scheme before applying the selected XSL stylesheet. After the XSL stylesheet rules are applied to generate the results, the results are converted to the encoding scheme that is specified by the output encoding parameter, `oe`. The one exception to this rule is cached result pages, which get converted to the encoding scheme of the cached document after XSLT processing.

Each front end for your search appliance is associated with an underlying stylesheet. All XSL stylesheets must be in `latin1` or `UTF8` formats.

XML Output

The description of the XML results format contains the following sections:

- “XML Output Overview” on page 55
- “Character Encoding Conventions” on page 55
- “Google XML Results DTD” on page 55
- “Google XML Tag Definitions” on page 56

XML Output Overview

For maximum flexibility, Google provides search results in XML format. Using the Google XML results, you can use your own XML parser to customize the display for your search users. If you are using an XSL stylesheet to transform the XML results instead of developing your own XML parser, proceed to "Custom HTML" on page 53.

Notes:

- Element values are valid HTML and are suitable for display, unless otherwise noted in the XML tag definitions. Some values are URLs and must be HTML-encoded to be displayed.
- To remain forward-compatible, your XML parser that parses Google search results should ignore attributes or tags that are not documented. By ignoring unknown tags, your custom XML parser can continue working without modification when Google adds more features to the XML output in the future.
- For custom parameters that contain spaces, each space is replaced with "_". You can still retrieve the unmodified value from the `original_value` attribute. For example:

```
<param name="temp" value="token_ring" original_value="token+ring" />
```

Character Encoding Conventions

The first line of the XML results indicates which character encoding is used. See XML Standard for information about character encoding (http://www.w3.org/TR/1998/REC-xml-19980210#charencoding).

Certain characters must be escaped when they are included as values in XML tags. These characters are documented in XML Standard (http://www.w3.org/TR/1998/REC-xml-19980210#dt-escape), and are shown in the table that follows. All other characters in the XML results are presented without modification.

Character	Escaped Form
`<`	either `&lt;` or `&#60;`
`&`	either `&amp;` or `&#38;`
`>`	either `&gt;` or `&#62;`
`'`	either `&apos;` or `&#39;`
`"`	either `&quot;` or `&#34;`

Google XML Results DTD

Google XML results can be returned with or without a reference to the most recent DTD (Document Type Definition) describing Google's XML format. The DTD is a guide to help search administrators and XML parsers understand the XML results output. Because Google's XML grammar may change from time to time, do not configure your parser to use the DTD to validate the XML results.

XML parsers should not be configured to fetch the DTD every time a search request is performed. Because the DTD is updated infrequently, these fetches create unnecessary delay and bandwidth requirements.

To get results in XML output format, use one of the following parameters in the search request:

- `output=xml_no_dtd` (recommended), or
- `output=xml`

When you use the `xml` output format, the XML results include the line:

```
<!DOCTYPE GSP SYSTEM "google.dtd">
```

The DTD is available on the Google Search Appliance at `http://<appliance_hostname>/google.dtd`.

Google XML Tag Definitions

This section contains an index of Google's XML tags.

Subtags legend:

`?`	zero or one instance of the subtag
`*`	zero or more instances of the subtag
`+`	one or more instances of the subtag
`|`	Boolean OR

BLOB

Format/Parent

Text (See Definition)

CACHE_HTML, CACHE_LEGEND_NOTFOUND, CACHE_LEGEND_TEXT

Subtags

None

Definition

This tag contains HTML data in the encoding format that is specified in the attribute. The data is Base64 encoded to preserve the data integrity of cached results that are encoded in a different encoding scheme than the requested results.

Attributes

Name	Format	Description
`encoding`	Text (Encoding Scheme)	The encoding scheme of the HTML data (See “Internationalization” on page 35 for a list of common encoding values)

C

Format/Parent

HAS

Subtags

None

Definition

Indicates that the “cache:” special query term is supported for this search result URL.

Cached results are suppressed and this element is not returned if the <head> tag of the document contains the following <meta> tag: `<meta name="ROBOTS" value="noarchive">`.

Attributes

Name	Format	Description
`SZ`	Text (Integer + “k”)	Provides the size of the cached version of the search result in kilobytes (“k”). This field is not populated if no cached version of a document is available, which can be the case if robots “noarchive” meta tags are used.
`CID`	Text	Identifier of a document in the Google Search Appliance cache. To fetch the document from the cache, send a search term of the form: "cache:" + CID text + ":" + encoded URL. The encoded URL is available in the `UE` tag. Send this search term normally, as you would type it into the search form.
`ENC`	Text	The encoding of the document in the cache. See “Internationalization” on page 35 for a list of common values.

CACHE

Format/Parent

GSP

Subtags

`CACHE_URL, CACHE_REDIR_URL, CACHE_LAST_MODIFIED, CACHE_LEGEND_FOUND?, CACHE_LEGEND_NOTFOUND?, CACHE_CONTENT_TYPE, CACHE_LANGUAGE, CACHE_ENCODING, CACHE_HTML`

Definition

Encapsulates the cached version of a search result.

Attributes

None

CACHE_CONTENT_TYPE

Format/Parent

Text (MIME type)

CACHE

Subtags

None

Definition

MIME type of the cached result, as specified in the HTTP header that is returned when the document is crawled.

Attributes

None

CACHE_HTML

Format/Parent

Text (HTML) (Custom HTML output only)

CACHE

Subtags

`BLOB?` (XML output only)

Definition

The cached version of the search result. All search results are stored in HTML format.

Attributes

None

CACHE_ENCODING

Format/Parent

Text

CACHE

Subtags

None

Definition

The encoding scheme of the cached result, as specified in the HTTP header that is returned when the document is crawled. (See “Internationalization” on page 35 for a list of common values.)

Attributes

None

CACHE_LANGUAGE

Format/Parent

Text (Google language tag)

CACHE

Subtags

None

Definition

The language of the cached result as determined by Google's automatic language classification algorithm. The value of this tag is the same as the values used for the automatic language collections without the "lang_" prefix (see "Automatic Language Filters" on page 32).

Attributes

None

CACHE_LAST_MODIFIED

Format/Parent

Text

CACHE

Subtags

None

Definition

Date that the document was crawled, as specified in the Date HTTP header when the document was crawled for this index. The crawler fetches documents from its cache if the web server responds with a 304 (not modified) status code to an if-modified-since request. In this case, the CACHE_LAST_MODIFIED is the date when the document was originally crawled and not the date of the if-modified-since request.

Attributes

None

CACHE_LEGEND_FOUND

Format/Parent

CACHE

Subtags

`CACHE_LEGEND_TEXT*`

Definition

Encapsulates query terms that are found in the visible text of the cached result returned.

Attributes

None

CACHE_LEGEND_NOTFOUND

Format/Parent

Text (Custom HTML output only)

CACHE

Subtags

`BLOB?` (XML output only)

Definition

Details of any query terms that are not visible in the cached result returned.

Attributes

None

CACHE_LEGEND_TEXT

Format/Parent

Text (Custom HTML output only)

CACHE_LEGEND_FOUND

Subtags

`BLOB` (XML output only)

Definition

Details of a query term that is visible in the cached result. Query terms found in the cached result are automatically highlighted using the colors described in the attributes of this tag.

Attributes

Name	Format	Description
`fgcolor`	Color attribute	The foreground color of the query term in the cached result. This value can be used directly in a color attribute for HTML tags.
`bgcolor`	Color attribute	The background color of the query term in the cached result. This value can be used directly in a color attribute for HTML tags.

CACHE_REDIR_URL

Format/Parent

Text (Absolute URL)

CACHE

Subtags

None

Definition

Final URL of cached result after all redirects are resolved.

Attributes

None

CACHE_URL

Format/Parent

Text (Absolute URL)

CACHE

Subtags

None

Definition

Initial URL of cached result.

Attributes

None

CRAWLDATE

Format/Parent

Text

R

Subtags

None

Definition

An optional element that shows the date when the page was crawled. It is shown only for pages that have been crawled within the past two days.

Attributes

None

CT

Format/Parent

HTML

GSP

Subtags

None

Definition

Search comments.

Example comment: Sorry, no content found for this URL

Attributes

None

CUSTOM

Format/Parent

GSP

Subtags

(Custom XML specified in the search request)

Definition

Encapsulates custom XML tags that are specified in the `proxycustom` input parameter.

Attributes

None

ENT_SOURCE

Format/Parent

R

Subtags

None

Definition

Identifies the application ID (serial number) of the search appliance that contributes to a result.

Example:

```
<ENT_SOURCE>S5-KUB000F0ADETLA</ENT_SOURCE>
```

Attributes

None

ENTOBRESULTS

Format/Parent

GSP

Subtags

OBRES

Definition

Encapsulates the results returned by OneBox modules.

Attributes

None

FI

Format/Parent

RES

Subtags

None

Definition

Indicates that document filtering was performed during this search.

See “Automatic Filtering” on page 31 for more details

Attributes

None

FS

Format/Parent

R

Subtags

None

Definition

Additional details about the search result.

Attributes

Name	Format	Description
`NAME`	Text	Name of the result descriptor
`VALUE`	Text	Value of the result descriptor

GD

Format/Parent

Text (HTML)

GM

Subtags

None

Definition

Contains the description of a KeyMatch result.

Attributes

None

GL

Format/Parent

Text (URL)

GM

Subtags

None

Definition

Contains the URL of a KeyMatch result.

Attributes

None

GM

Format/Parent

GSP

Subtags

`GL, GD?`

Definition

Encapsulates a single KeyMatch result.

Attributes

None

GSP

Format/Parent

This is the root element.

Subtags

`(CT?, CUSTOM?, ENTOBRESULTS, GM*, PARAM+, Q, RES?, Spelling?, Synonyms?, TM) | CACHE`

Definition

GSP = “Google Search Protocol”

Encapsulates all data that is returned in the Google XML search results.

Attributes

Name	Format	Description
`VER`	Text	Indicates version of the search results output. The current output version is “3.2”.

HAS

Format/Parent

R

Subtags

`L?, C?`

Definition

Encapsulates special features that are included for this search result.

Attributes

None

HN

Format/Parent

Text (URL-encoded web directory, see “Appendix B: URL Encoding” on page 108)

R

Subtags

None

Definition

Indicates that filtering has occurred and that additional results are available from the directory where this search result was found. The value of this tag is ready to be used with the `site:` query term (see “Directory Restricted Search” on page 26).

Attributes

Name	Format	Description
`U`	Text	Server and path components of the directory’s URL.

L

Format/Parent

HAS

Subtags

None

Definition

Indicates that the “link:” special query term is supported for this search result URL.

Attributes

None

LANG

Format/Parent

Text

R

Subtags

None

Definition

Indicates the language of the search result. The LANG element contains a two-letter language code. See “Automatic Language Filters” on page 32 for language codes.

Attributes

None

M

Format/Parent

Text (Integer)

RES

Subtags

None

Definition

The estimated total number of results for the search.

The estimate of the total number of results for a search can be too high or too low. See “Appendix A: Estimated vs. Actual Number of Results” on page 106.

Attributes

None

MT

Format/Parent

R

Subtags

None

Definition

Meta tag name and value pairs obtained from the search result.

Only meta tags (see “Meta Tags” on page 42) that are requested in the search request are returned.

Attributes

Name	Format	Description
`N`	Text	Name of the meta tag
`V`	Text	Value of the meta tag

NB

Format/Parent

RES

Subtags

`PU?, NU?`

Definition

Encapsulates the navigation information for the result set.

The NB tag is present only if either the previous or additional results are available.

Attributes

None

NU

Format/Parent

Text (Relative URL)

NB

Subtags

None

Definition

Contains a relative URL pointing to the next results page.

The NU tag is present only when more results are available.

Attributes

None

OBRES

Format/Parent

ENTOBRESULTS

Subtags

The contents of the OBRES element are provided by the OneBox module, and must conform to the OneBox Results Schema. See the specific OneBox module's documentation for details. See also the *Google OneBox for Enterprise Developer's Guide*.

Definition

Encapsulates a result returned by a OneBox module.

Attributes

None

OneSynonym

Format/Parent

HTML

Synonyms

Subtags

None

Definition

A related query for the submitted query, in HTML format.

Attributes

Name	Format	Description
`q`	Text	The URL-encoded version of the related query (see "Appendix B: URL Encoding" on page 108)

PARAM

Format/Parent

GSP

Subtags

None

Definition

The search request parameters that were submitted to the Google Search Appliance to generate these results.

Attributes

Name	Format	Description
`name`	Text	Name of the input parameter
`value`	HTML	HTML-formatted version of the input parameter value
`original_value`	Text	Original URL-encoded version of the input parameter value (see "Appendix B: URL Encoding" on page 108)

PARM

Format/Parent

RES

Subtags

PC, PMT*

Definition

Encapsulates all dynamic navigation results.

Attributes

None

PC

Format/Parent

Text (Integer 0 or 1)

PARM

Subtags

None

Definition

Indicates whether the counts are exact or partial. 0-exact, 1-partial.

None

PMT

Format/Parent

PARM

Subtags

PV+

Definition

Encapsulates results for one attribute. A maximum of 5k values (PV) are returned after sorting all by count or value as configured and discarding the rest.

Attributes

Name	Format	Description
`NM`	Text	Metatag name
`DN`	Text	Display name
`IR`	Text (Integer)	Attribute is range type (1) or not (0)
`T`	Text (Integer)	Attribute type: 0-String, 1-Integer, 2-Float, 3-Currency, 4-Date

PU

Format/Parent

Text (Relative URL)

NB

Subtags

None

Definition

Contains relative URL to the previous results page.

The PU tag is present only if previous results are available.

Attributes

None

PV

Format/Parent

PMT

Subtags

None

Definition

Encapsulates one value count information.

Attributes

Name	Format	Description
`V`	Text	Value (empty for range attributes)
`L`	Text	Contains low range value (empty for non-range attribute)
`H`	Text	Contains high range value (empty for non-range attribute)
`C`	Text (Integer)	Doc count matching this value or under this range

Q

Format/Parent

HTML

GSP

Subtags

None

Definition

The search query terms submitted to the Google search appliance to generate these results.

Attributes

None

R

Format/Parent

RES

Subtags

`CRAWLDATE, FS?, HAS, HN?, LANG, MT*, RK, S?, T?, U, UD, UE`

Definition

Encapsulates the details of an individual search result.

Attributes

Name	Format	Description
`N`	Text (Integer)	The index number (1-based) of this search result.
`L`	Text (Integer)	The recommended indentation level of the results. This value is 1 unless Duplicate Directory Filtering occurs (see “Automatic Filtering” on page 31). In this case, the second directory result has a value of 2.
`MIME`	Text	The MIME type of the search result.

RES

Format/Parent

GSP

Subtags

`FI?, M, NB?, PARM?, R*, XT?`

Definition

Encapsulates the set of all search results.

Attributes

Name	Format	Description
`SN`	Text (Integer)	The index (1-based) of the first search result returned in this result set.
`EN`	Text (Integer)	Indicates the index (1-based) of the last search result returned in this result set.

RK

Format/Parent

Text (Integer in the range 0-10)

Subtags

None

Definition

The `RK` parameter assigns a ranking score to each page on a scale from 0 (least important) to 10 (most important) based on how well the result matches the query. When search results are sorted by relevancy, the `RK` value is in decreasing order (highest to lowest).

To see the `RK` values, you must view search results in raw XML, as described in the following steps:

1. On the search page, enter a query and get results.
2. If not already selected, click on **Sort by relevance**.
3. On the **Advanced Search** page, edit the query parameters:
    a. Change the `output` parameter to `&output=xml`
    b. Remove `&proxystylesheet=default_frontend`
    c. Add `&getfield=*`
4. Renter the query.

The XML results show the `RK` parameter for each result, for example: **<RK>10</RK>**.

Attributes

None

S

Format/Parent

Text (HTML)

R

Subtags

None

Definition

The snippet for the search result.

Query terms appear in bold in the results. Line breaks are included for proper text wrapping.

In documents larger than 300KB, snippets may not contain query terms that occur beyond the first 300KB of the document. For non-HTML documents, the 300KB limit applies to the converted version, not the original document.

Attributes

None

SCOREBIAS

Format/Parent

Text (XML)

R

Subtags

None

Definition

The `SCOREBIAS` tag can appear zero or more times as a child of the R tag (see “R” on page 77) for each result. The `SCOREBIAS` tag appears for each result biaser that is applied.

The `NAME` attribute is the name of the result biaser.

The `VALUE` attribute indicates the effect of the biaser. For biasers where the strength is expressed symbolically, such as source or collection biasing and metadata biasing.

The search appliance does not include any information about the exact change in score or rank, or the weight of the result biaser.

The following example indicates a medium increase in the `PatternScorer` result biaser:

```
<SCOREBIAS NAME="PatternScorer" VALUE="2">
```

Attributes

Attribute	Value	Format	Description
`NAME`	`PatternScorer`	Text	Used for both source biasing and collection biasing.
	`DateBias`	Text	Used for date biasing.
	`Metadata`	Text	Used for metadata biasing.
`VALUE`	`3`	Text	For a strong increase.
	`2`	Text (integer)	For a medium increase.
	`1`	Text (integer)	For a weak increase.
	`0`	Text (integer)	For no change.
	`-3`	Text (integer)	For a strong decrease.
	`-2`	Text (integer)	For a medium decrease.
	`-1`	Text (integer)	For a weak decrease.

For biasers that do not use a symbolic change, such as date biasing, `VALUE` has these numerical values:

- 1 for an increase in score
- -1 for a decrease in score
- 0 for no change (probably won’t ever see this)

Spelling

Format/Parent

GSP

Subtags

`Suggestion+`

Definition

Encapsulates alternate spelling suggestions for the submitted query. Only one spelling suggestion is returned at this time.

Attributes

None

Suggestion

Format/Parent

HTML

Spelling

Subtags

None

Definition

An alternate spelling suggestion for the submitted query, in HTML format.

Attributes

Name	Format	Description
`q`	Text	The spelling suggestion.
`qe`	Text	Internal-only attribute of the spelling suggestion. This attribute works when the search results are transformed on the search appliance, but not on external parsers.

Synonyms

Format/Parent

GSP

Subtags

`OneSynonym+`

Definition

Encapsulates the related queries for the submitted query. Up to 20 related queries may be returned, depending on the related queries list that is associated with the front end.

Attributes

None

T

Format/Parent

Text (HTML)

R

Subtags

None

Definition

The title of the search result.

Attributes

None

TM

Format/Parent

Text (Floating-point number)

GSP

Subtags

None

Definition

Total server time to return search results, measured in seconds.

Attributes

None

U

Format/Parent

Text (Absolute URL)

R

Subtags

None

Definition

The URL of the search result.

Attributes

None

UD

Format/Parent

Text (URL to display for non-ASCII URLs)

R

Subtags

None

Definition

The URL string to display when the URL that is in the U parameter is non-ASCII. Displays UTF-8 characters and IDNA domain names properly.

Attributes

None

UE

Format/Parent

Text (URL-encoded version of the URL)

R

Subtags

None

Definition

The URL-encoded version of the URL that is in the U parameter.

Attributes

None

XT

Format/Parent

RES

Subtags

None

Definition

Indicates that the estimated total number of results specified in this search result is exact.

See “Automatic Filtering” on page 31 for more details.

Attributes

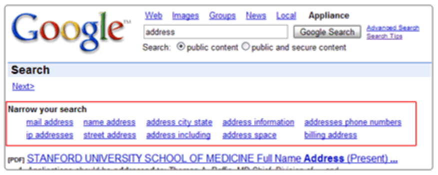
None

Chapter 4

Dynamic Result Clustering Service / cluster Protocol

Dynamic result clustering narrows searches by providing dynamically formed subcategories that appear at the top or right side of the search results.

The following illustration shows the dynamic result clustering at the top of the search results (enclosed in the red box):


Web Images Groups News Local Appliance
address
Google Search
Advanced Search
Search Tips
Search: public content public and secure content
Search
Next>
Narrow your search
mail address name address address city state address information addresses phone numbers
ip addresses street address address including address space billing address
[PDF] STANFORD UNIVERSITY SCHOOL OF MEDICINE Full Name Address (Present) ...

The search appliance generates alternative search queries by analyzing indexed documents based on a user’s current search query. The results appear as query suggestions to help the user modify the query.

You can enable dynamic result clustering for a front end in the Admin Console at **Search > Search Features > Front Ends** > **Output Format** > **Search Results** > **Dynamic result clusters**.

After enabling dynamic result clustering for a front end, the search appliance enables the XSLT spreadsheet variables to enable the feature and specify the position on the search results page for the dynamic result clustering:

```
<!-- *** dynamic result cluster options *** -->
<xsl:variable name="show_res_clusters">1</xsl:variable>
<xsl:variable name="res_cluster_position">position</xsl:variable>
```

Where *position* can be `right` or `top`.

When a user enters a query, the search appliance:

1. Uses the `http://`*`Search_Appliance`*`/cluster.js` JavaScript to provide the dynamic result clustering.
2. Fetches the `/cluster` content.

3. Triggers an AJAX call to the cluster service to populate the cluster position holders. The cluster position holders have the following DOM Ids depending on their position:

```
<xsl:when test="$res_cluster_position = ’top’">
  <table>
    <tr>
    <td id=’cluster_label0’></td>
    <td id=’cluster_label2’></td>
    <td id=’cluster_label4’></td>
    <td id=’cluster_label6’></td>
    <td id=’cluster_label8’></td>
    </tr>
    <tr>
    <td id=’cluster_label1’></td>
    <td id=’cluster_label3’></td>
    <td id=’cluster_label5’></td>
    <td id=’cluster_label7’></td>
    <td id=’cluster_label9’></td>
    </tr>
  </table>
</xsl:when>
<xsl:when test="$res_cluster_position = ’right’">
  <ul>
    <li id=’cluster_label0’></li>
    <li id=’cluster_label1’></li>
    <li id=’cluster_label2’></li>
    <li id=’cluster_label3’></li>
    <li id=’cluster_label4’></li>
    <li id=’cluster_label5’></li>
    <li id=’cluster_label6’></li>
    <li id=’cluster_label7’></li>
    <li id=’cluster_label8’></li>
    <li id=’cluster_label9’></li>
  </ul>
</xsl:when>
```

The default style sheet activates dynamic result clustering using **onload** attribute of the `<body>` tag on the search result page. The following is an example of the body opening tag:

```
<body onload="cs_loadClusters(’{search query}’, cs_drawClusters);">
```

Where `{search_query}` is the current search request, as shown in the following example (broken for readability):

```
q=culebra&btnG=Google+Search&access=p&client=default_frontend&output=xml_no_dtd&
proxystylesheet=default_frontend&sort=date%3AD%3AL%3Ad1&entqr=3&entsp=a&oe=UTF-
8&
ie=UTF-8&ud=1&site=default_collection
```

The default XSLT stylesheet provides the `clustering` CSS `id` value for the page heading, cluster position, and loading message.

```
<div id=’clustering’>
  <h3>Narrow your search</h3>
...
```

For more information, see “Using Dynamic Result Clusters to Narrow Searches” in *Creating the Search Experience*.

Note: The `cluster.js` file depends on additional JavaScript files listed in the application.

Dynamic Result Clustering Request

Administrators can test the `/cluster` feature by submitting a custom HTTP POST form.

The search appliance processes cluster requests:

1. The cluster request inherits all request parameters and the search appliance transports the parameters into an internal search query. If any of the `/search` parameters (see “Search Parameters” on page 10) are present in the parameter list for the request to `/cluster`, they are passed to the internal search request.

2. If custom parameters exist, the search appliance submits the parameters without filtering.

   The POST request must have all the parameters encoded in the URI.

   The clustering service recognizes the following parameter (in addition to the `/search` parameters, see “Search Parameters” on page 10).

Parameter	Description	Default Value
`coutput`	Cluster output type: `json` or `xml`. Indicates the output you requested. Specify `json` for JSON output on `/cluster` POST requests. Specify `xml` for XML output as either a GET or POST. The `xml` value is generally used with `/cluster` as a RESTful service and the GET method. All request parameters must appear in the URI of a POST request.	`json`

3. The search appliance stylesheet adds all parameters to the request related to the current search query, as well as the custom parameters. Although the search appliance passes all parameters, not all are used.

Dynamic Result Clustering JSON Request and Response

The following example HTML provides a POST form that you can use to get JSON output (statements are wrapped for readability). The query is for the island of Culebra.

```
<html>
<head> <title> HTTP POST to view JSON for dynamic result clustering </title> </
head>
<body>
<!-- Post parameters contiguous in a URL -->
<form method='post' action='http://Search_Appliance/cluster?q=culebra&
btnG=Google+Search&access=p&entqr=0&ud=1&sort=date%3AD%3AL%3Ad1&
output=xml_no_dtd&oe=UTF-8&ie=UTF-8&client=default_frontend&
proxystylesheet=default_frontend&site=default_collection'>
<input type=submit value='Post'></form>
</body>
</html>
```

Click the `Post` button to view the JSON response.

The search appliance returns the following JSON response:

```
{ "clusters": [
    { "algorithm": "Concepts",
      "clusters": [
        { "label": "canada chile culebra",
          "docs": [ 18,19,20,21,23,26,27,29,30,32]
        },
        { "label": "dewey culebra",
          "docs": [ 1,9,36]
        }
      ]
    }
  ],
  "documents": [
    { "url": "http://server.example.com/file42.pdf",
      "title": "TLA Annual Report 2009--Acronyms in the Public Sector
                 <b>...</b>",
      "snippet": "<b>...</b> Soy Flz (<b>Culebra</b>) <b>Culebra</b>
                 34,102 34,102 2.28 <b>...</b> Soy Flz (<b>Culebra</b>)
                 was re-elected<br> Executive Director of <b>Culebra</b>,
                 effective May 1, 2009. <b>...</b>"
    },
    ...,
    { "url": "http://server.example.com/turtle_island.html",
      "title": "Puerto Rico Travel",
      "snippet": "<b>...</b> rentals and useful information about <b>Culebra</b>
                 <b>...</b>"
    }
  ]
}
```

The top-level entries are described in the following table.

Entry	Description
`clusters`	The output from different clustering algorithms. There is only one supported cluster algorithm, so the value of `algorithm` must be `Concepts`. The `clusters` category consists of: • A series of `algorithm` and subordinate `clusters` pairs. The `algorithm` is the name and `Concepts` is the only supported algorithm. • The subordinate `clusters` is a series of `labels` and the array of `docs` that have that label. • The label is a query suggestion. The `docs` are indexes into the `documents` section that follow. Each `label` provides an alternative query, and each `docs` array tells the document location indices.
`documents`	A sequence of the URL, title, and snippet for each of up to 100 top search results from a search query. The search appliance creates the `docs` arrays from the `documents` list.

The dynamic result clustering service's default JavaScript client ignores the `documents` element and does not use the `docs` array.

Dynamic Result Clustering XML Request and Response

The POST form returns XML output by adding the `coutput=xml` parameter to the `action=` URL:

```
<form method='post' action='http://Search_Appliance/
    cluster?q=culebra&coutput=xml&btnG=Google+Search&access=p&entqr=0&ud=1&
    sort=date%3AD%3AL%3Ad1&output=xml_no_dtd&oe=UTF-8&
    ie=UTF-8&client=default_frontend&
    proxystylesheet=default_frontend&site=default_collection'>
  <input type=submit value='Post'>
</form>
```

The search appliance returns the following XML response:

```
<?xml version="1.0"?>
<toplevel>
  <Response>
    <algorithm data="Concepts"/>
    <t_cluster int="75"/>
    <cluster>
      <gcluster>
        <label data="canada chile culebra"/>
        <doc int="18"/>
        <doc int="19"/>
        <doc int="20"/>
        <doc int="21"/>
        <doc int="23"/>
        <doc int="26"/>
        <doc int="27"/>
        <doc int="29"/>
        <doc int="30"/>
        <doc int="32"/>
      </gcluster>
      <gcluster>
        <label data="dewey culebra"/>
        <doc int="1"/>
        <doc int="9"/>
        <doc int="36"/>
      </gcluster>
    </cluster>
  </Response>
  <t_fetch int="134"/>
  <document>
    <url data="http://server.example.com/file42.pdf"/>
    <title data="TLA Annual Report 2009--Acronyms in the Public Sector <b>...</
b>"/>
    <snippet data="<b>...</b> Soy Flz (<b>Culebra</b>) <b>Culebra</b>
    34,102 34,102 2.28 <b>...</b> Soy Flz (<b>Culebra</b>)
    was re-elected<br> Executive Director of <b>Culebra</b>,
    effective May 1, 2009. <b>...</b>"/>
  </document>
  <!-- ... -->
  <document>
    <url data="http://server.example.com/turtle_island.html"/>
    <title data="Puerto Rico Travel"/>
    <snippet data="<b>...</b> rentals and useful information about <b>Culebra</b>
    <b>...</b>"/>
  </document>
</toplevel>
```

The top-level entries are described in the following table.

Entry	Description
`<cluster>`	The output from different clustering algorithms. There is only one supported cluster algorithm, so the value of `<algorithm>` must be `Concepts`. The `<cluster>` category consists of: • A series of `<algorithm>` and subordinate `<gcluster>` pairs. • The subordinate `<gcluster>` is a series of `<label>` statements and the array of `<doc>` elements that have that label. • The label is a query suggestion. The `<doc>` statements are indexes into the `<document>` section that follows. Each `<label>` provides an alternative query, and each `<doc>` array provides the document location indices.
`<document>`	A sequence of the URL, title, and snippet for each of up to 100 top search results from a search query. The search appliance creates the `<doc>` arrays from the `<document>` list.

The dynamic result clustering service's default JavaScript client ignores the `<document>` element and does not use the `<doc>` array. The XML response is very basic, and does not use any validations such as a DTD or XML.

The following DTD defines the XML rules, however the XML output is not validated against these rules:

```
<?xml version="1.0"?>
<!ELEMENT toplevel (Response, t_fetch, document+)>
<!ELEMENT Response (algorithm, t_cluster, cluster)>
<!ELEMENT cluster (gcluster+)>
<!-- each gcluster element is an alternate query and its location indexes from the
top results -->
<!ELEMENT gcluster (label, doc+)>
<!-- each document element is search result, complete with url, title, and snippet
-->
<!ELEMENT document (url, title, snippet)>
<!ELEMENT algorithm EMPTY>
<!ELEMENT t_fetch EMPTY>
<!ELEMENT label EMPTY>
<!ELEMENT doc EMPTY>
<!ELEMENT url EMPTY>
<!ELEMENT title EMPTY>
<!ELEMENT snippet EMPTY>
<!ATTLIST algorithm
  data (Concepts)>
<!ATTLIST t_cluster
  int CDATA #REQUIRED>
<!ATTLIST label
  data CDATA #REQUIRED>
<!ATTLIST doc
  int CDATA #REQUIRED>
<!ATTLIST url
  data CDATA #REQUIRED>
<!ATTLIST title
  data CDATA #REQUIRED>
<!ATTLIST snippet
  data CDATA #REQUIRED>
```

Chapter 5

Query Suggestion Service /suggest Protocol

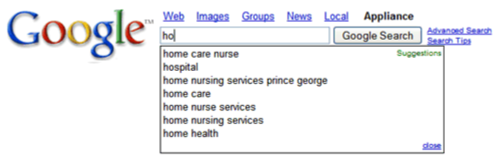
The query suggestion service provides suggestions that complete a user's search query. As a user enters a query in the search box, a drop-down menu appears with suggestions to complete the query. The search appliance uses the most popular search queries of its users to determine the top suggestions that list for a query.

Only queries that returned results are used to build the database of query suggestions. Queries with special terms, such as `inmeta:`, `info:`, `link:`, `daterange:`, etc are excluded when building the database of query suggestions.

In addition, if activated, the search appliance adds user-added results to the list of suggestions.

For information on retrieving and updating the suggestion blacklist using Java and the Google Data API, see "Query Suggestion Blacklist" in the *Administrative API Developer's Guide: Java*. For information on user-added results, see "Providing User Results" in *Creating the Search Experience*.

The following example shows query suggestions:



You can use the query suggestion feature to:

- Capture JSON response output from query suggestions and filter the information, before displaying suggestions to the user.
- Upload (and retrieve) a blacklist of bad words using the Google Data API, so that these words do not appear in the list of suggestions.
- Add information to a custom interface for search implementations that consume search results in XML form.

Enable the query suggestion client for a front end from **Search > Search Features** > **Front Ends** > **Output Format** > **Search Box** > **Query suggestions**.

The query suggestion service adds latest query data to its list of suggestions once every 24 hours. However you can force the service to pick up changes immediately by disabling and enabling the **Query Suggestions** checkbox. When you enable the checkbox, the suggest service is sent a `HUP` signal so that it updates its data structure from the latest query logs.

Queries with Special Query Terms, such as inmeta, are excluded from the Query Suggestion database. So if this type of query is used extensively, then a possible solution is to use the equivalent Search Parameters, such as partialfields and requiredfields.

After enabling query suggestions:

1. The search appliance sets the XSLT stylesheet element `show_suggest` element:

   ```
   <xsl:variable name="show_suggest">1</xsl:variable>
   ```

2. The search appliance provides access to the `http://`*`Search_Appliance`*`/ss.js` JavaScript file. The `ss.js` file is for version 6.2 and later only, the version 6.0 uses the `suggest_js.js` file.

3. When a user starts a query, the JavaScript in the client makes calls to the query suggestion URI and fetches the results, responding with JSON output. The AJAX response handler in the JavaScript client populates the list of suggestions.

For more information, see “Providing Query Suggestions” in *Creating the Search Experience*.

Query Suggestion JavaScript Variables

After enabling query suggestion, the `show_suggest` XSLT variable is set to 1 and the JavaScript variables become usable in the XSLT stylesheet.

```
<xsl:variable name="show_suggest">1</xsl:variable>
```

The query suggestion features supports the following JavaScript variables in the XSLT stylesheet.

ss_allow_non_query

A `true` or `false` flag used by the `os` (OpenSearch) and `rich` output formats to suppress user-added results. This variable eliminates user-added results from the suggestions, while retains those based on popularity among other users’ queries.

Default value: `true`

ss_form_element

The value of the attribute `id` of the `<form>` used for the search box. This variable must be overwritten with the correct value when using suggest in a custom form outside of the search appliance, but still using the existing JavaScript client. If this parameter is incorrect, AJAX calls aren’t sent.

The XSLT spreadsheet sets this variable as:

```
var ss_form_element = 'suggestion_form';
```

The form request occurs under the following XSLT condition (wrapped for readability):

```
<xsl:when test="$show_suggest = '1' and (($type = 'home')
  or ($type = 'std_top'))">
  <xsl:text disable-output-escaping="yes">&lt;form id="suggestion_form"
    name="gs" method="GET" action="search"
    onsubmit="return (this.q.value == '') ? false : true;"</xsl:text>
</xsl:when>
```

Default value: `suggestion_form`

ss_g_max_to_display

The maximum number of query suggestions to show from the suggest server. If set to 0, allows an unlimited number of suggestion types.

Default value: `10`

ss_g_more_names_to_display

A literal string that displays for multiple suggestions. This value appears to the right of the query suggestion box. The default value listed in the next column is for the English language version. The value is internationalized and the actual value depends on the current language setting.

Default value: `Suggestions`

ss_g_one_name_to_display

A literal string that displays for a single suggestion. This value appears to the right of the query suggestion box. The default value listed in the next column is for the English language version. The value is internationalized and the actual value depends on the current language setting.

Default value: `Suggestion`

ss_max_to_display

The total number of rows allowed in the suggestion box. The smaller of `ss_max_to_display` and `ss_g_max_to_display` takes effect in limiting the number of suggestions.

Default value: `12`

ss_non_query_empty_title

If a user-added results entry's title is missing or is an empty string, this value is used. This is a token that you can internationalize in the XSLT stylesheet. Set this value to `Unknown`, `Noname`, `---` , or anything that can be visually shown. Otherwise, the empty string becomes an element that introduces confusion, as the user may not know whether it is a suggestion or just a separation line. The default value listed in the next column is for the English language version. The value is internationalized and the actual value depends on the current language setting.

Default value: `No Title`

ss_popup_element

The value of the attribute `id` of the `<table>` as the placeholder for search suggestions. The `<table>` initially is empty, and is positioned beneath the search box, aligned to the left. If the `<table>` HTML code is incorrect, suggestion results do not display. The default value is:

```
var ss_popup_element = 'search_suggest';
```

For an example of the search_suggest in a table, see "Query Suggestion Table Class" on page 97.

Default value: `search_suggest`

ss_protocol

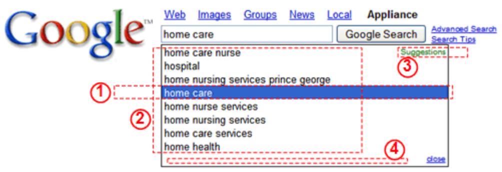
The three values are:

Value	Description
`legacy`	Provides backward compatibility with the version 6.0 query suggestion feature for the `token` and `max_matches` variables. This setting excludes user-added results from the response. If an unknown format is set, `legacy` is assumed.
`os`	Supports the OpenSearch format.
`rich`	Rich text format (default for version 6.2 and later).

Default value: `rich`

Query Suggestion CSS Classes in the XSLT Stylesheet

Programs can change the following areas of the suggestion display using CSS:



Google Suggest provides the following CSS classes that correspond to the numbers in the illustration:

Area	CSS Class	Style Description
1	`.ss_gac_a`	Suggestion box table row `<tr>` when an entry is not selected.
1	`.ss_gac_b`	Suggestion box table row `<tr>` when an entry is selected.
2	`.ss_gac_c`	Suggestion box table cell `<td>`. This class affects all suggestion entries.
3	`.ss_gac_d`	Suggestion box table cell `<td>`. This class is defined by the `ss_g_more_names_to_display` and `ss_g_one_name_to_display` XSLT variables.
4	`.ss_gac_e`	Suggestion box last table row `<tr>`.

Query Suggestion Table Class

The `.ss_gac_m` CSS class provides the style for the `<table>` element whose `id` value is referenced by the JavaScript `ss_popup_element` variable:

```
<table cellpadding="0" cellspacing="0" class="ss-gac-m"
 style="width: 365px; visibility: hidden;" id="search_suggest"></table>
```

Query Suggestion Requests and Responses

The output format controls the query suggestion request and response:

- Legacy Format—Backward compatibility with version 6.0 (see “Legacy Format” on page 98)
- OpenSearch Format—Supports the OpenSearch protocol (see “OpenSearch Format” on page 99)
- Rich Output Format—Version 6.2 and later default format (see “Rich Output Format” on page 100)

Legacy Format

The `legacy` format is similar to the suggest feature in the version 6.0 search appliance. The `token` and `max_matches` were in version 6.0, and the `client`, `format`, and `site` parameters were introduced in version 6.2.

Parameter	Description	Default Value

`callback`	Provides a JSONP compatible response from suggest. If you set callback=test, it will return:
	
	`/* GSA Suggest Service JSONP Response. */`
	`test(<WHAT WAS SUPPOSED TO BE RETURNED>);`
	
	The prefix `/* GSA Suggest Service JSONP Response. */` is hard-coded in the response to safeguard against cross-site scripting attacks on the JSONP callback name. You must remove the prefix before the response can be parsed by JavaScript.	

`client`	Front end name.	`default_frontend`
`format`	The output format in which the client wants the results.	`legacy`
`max_matches`	The maximum number of results that the suggest server should return. The minimum is `0`, which indicates that the server should return an empty set, however this result would not be meaningful. The maximum is not defined. If this parameter is not set, then the default value is 10 possible matches. If fewer suggestions are configured, then they are returned.	`10`
`site`	Collection name.	`default_collection`
`token`	The partial query string that a user enters in the search box. The minimum size is one character. If set to `0`, that is, if the search box is empty, then the suggest client side JavaScript doesn’t send a request to query suggestion. Even if an administrator implements a custom interface, sending an empty token returns an empty set as the result. The maximum size of the `token` parameter is not defined.	None

Request:

```
/suggest?token=<query>&max_matches=<num>&use_similar=0
```

Response:

```
[ "<term 1>", "<term 2>", ..., "<term n>" ]
```

Or, if no result:

```
[]
```

OpenSearch Format

The `os` format uses the OpenSearch protocol.

Parameter	Description	Default Value

`callback`	Provides a JSONP compatible response from suggest. If you set callback=test, it will return:
	`/* GSA Suggest Service JSONP Response. */`
	`test(<WHAT WAS SUPPOSED TO BE RETURNED>);`
	The prefix `/* GSA Suggest Service JSONP Response. */` is hard-coded in the response to safeguard against cross-site scripting attacks on the JSONP callback name. You must remove the prefix before the response can be parsed by JavaScript.	

`client`	Front end name.	`default_frontend`
`format`	The output format in which the client wants the results.	`os`
`max`	The maximum number of results that the suggest server should return. The minimum is `0`, which indicates that the server should return an empty set, however this result would not be meaningful. The maximum is not defined. If this parameter is not set, then the default value is 10 possible matches. If fewer suggestions are configured, then they are returned.	`10` (matches)
`q`	The partial query string that a user enters in the search box. The minimum size is one character. If set to `0`, that is, if the search box is empty, then the suggest client side JavaScript doesn’t send a request to query suggestion. Even if an administrator implements a custom interface, sending an empty token returns an empty set as the result. The maximum size of the `token` parameter is not defined.	None
`site`	Collection name.	`default_collection`

Request:

```
/suggest?q=<query>&max=<num>&site=<collection>&client=<frontend>&
   format=os
```

Response:

```
[
  "<query>",
  [ "<term 1>", "<term 2>", ... "<term n>" ],
  [ "<content 1>", "<content 2>", ..., "<content n>" ],
  [ "<url 1>", "<url 2>", ..., "<url n>" ]
]
```

The client distinguishes between suggest and user-added results as follows:

- For suggest results, the term is non-empty while the content and the URL are empty.
- For user-added results, the term is empty, content is optional, and the URL is non-empty.

A client can choose to display both suggest and user-added results or just one of them. The default front end provided in the Admin Console's XSLT stylesheet, displays both results. User-added results display after suggest results and appear in italics.

If no result occurs, the OpenSearch format provides the following response:

```
[ <query>, [] ]
```

Rich Output Format

The `rich` format uses the rich protocol for search-as-you-type suggestions.

Parameter	Description	Default Value

`callback`	Provides a JSONP compatible response from suggest. If you set callback=test, it will return:
	`/* GSA Suggest Service JSONP Response. */`
	`test(<WHAT WAS SUPPOSED TO BE RETURNED>);`
	The prefix `/* GSA Suggest Service JSONP Response. */` is hard-coded in the response to safeguard against cross-site scripting attacks on the JSONP callback name. You must remove the prefix before the response can be parsed by JavaScript.	

`client`	Front end name.	`default_frontend`
`format`	The output format in which the client wants the results.	`rich`
`max`	The maximum number of results that the suggest server should return. The minimum is `0`, which indicates that the server should return an empty set, however this result would not be meaningful. The maximum is not defined. If this parameter is not set, then the default value is 10 possible matches. If fewer suggestions are configured, then they are returned.	`10` (matches)
`q`	The partial query string that a user enters in the search box. The minimum size is one character. If set to `0`, that is, if the search box is empty, then the suggest client side JavaScript doesn't send a request to query suggestion. Even if an administrator implements a custom interface, sending an empty token returns an empty set as the result. The maximum size of the `token` parameter is not defined.	None
`site`	Collection name.	`default_collection`

Request:

```
/suggest?q=<query>&max=<num>&site=<collection>&client=<frontend>&format=rich
```

Response (wrapped for readability):

```
{
  "query": "<query>",
  "results": [
  { "name": "<term 1>", "type": "suggest"},
  { "name": "<term 2>", "type": "suggest"},
  { "name": "<term 3>", "type": "uar", "content": "Title of UAR",
    "moreDetailsUrl": "URL of UAR"}
  ...,
  ]
}
```

Clients can distinguish between suggest and user-added results by looking at the `type`. The type can be `suggest` or `uar` (user-added result) to identify the type of suggestion.

The `style` value is reserved and not used in version 6.2. For suggest entries, a term must be present. For user-added results entries, `moreDetailsUrl` must be present.

If no result occurs, the rich format provides the following response:

```
{ "query": <query>, "results": [] }
```

Chapter 6

Advanced Search Reporting Service / click Protocol

Advanced search reporting enables administrators to see what types of links a user chooses on a search results page, and more generally to track all actions that a user performs such as clicking navigational links. This information enables administrators to improve access and latency of search results, and to understand user click behavior. This document contains reference information about request parameters and the information logged by the search appliance about user click behavior.

An administrator enables advanced search reporting in the Admin Console's **Search > Search Features > Front Ends** > **Output Format** page. The search appliance then modifies search result pages by inserting JavaScript on the page for tracking all links that a user clicks. When a user clicks a link in the search results, the JavaScript executes in the browser, requesting a URL from the search appliance. The URL starts with `/click` and contains information about the link. The arguments given in the URL are logged on the search appliance, and the search appliance returns a response to the browser. The browser then retrieves the URL on which the user clicked.

The `/click` URL is not visible to users. This URL has little effect on user-perceived latency because the processing is performed on the client side, and both the request and response are as minimal as possible.

This document enables advanced administrators to understand the `/click` URL information that is sent to the search appliance. Administrators who monitor network data may see this information. Programmers can also use this information in custom applications.

For information about configuring advanced search reporting in the Admin Console, see "Gathering Information about the Search Experience" in *Creating the Search Experience*. For information on search requests, "Request Format" on page 6.

The following describes how an advanced search reporting request is handled by the search appliance.

1. The user submits a search request to the Google Search Appliance.
2. The search appliance sends back search results.
3. The user clicks a URL in the search results.
4. The browser sends the associated URL to the search appliance.
5. The browser also sends a hidden `/click` URL to the search appliance containing advanced search reporting parameters about the link that the user clicked. For more information, see “Request Parameters” on page 103.
6. The search appliance responds to the browser by sending an HTTP status code of 204 (no content) to acknowledge receipt of the `/click` URL. For more information, see “Responses” on page 105.
7. The search appliance takes the `/click` URL information and uses the URL to write advanced search reporting information to a log file. Administrators view the log on the Admin Console. For more information on advanced search reporting and the Admin Console, see “Gathering Information about the Search Experience” in *Creating the Search Experience*.

Request Parameters

Advanced search reporting requires the `site` parameter. Ensure that you include `&site=`*`collection`* in request URLs, where *collection* is the collection being queried.

The end user does not see a `/click` URL. If an administrator uses a program such as **tcpdump** to view data on the Internet connection, the `/click` information is visible. Administrators or programmers can use this information for debugging or in an application. The parameters in the `/click` URL are as follows.

URL Parameter	Description	Example
`cd`	Click data. This is additional, user-provided information to give more information or context about the click. This is not interpreted by the search appliance and is just logged. This would typically only be used by people who perform advanced customization and need to log additional information.	`cd=malta`
`ct`	Required parameter. Click type. A value that identifies the type of link that a user clicks. For the value, use underscores or a dot without spaces and use alpha-numeric characters. For a complete list of click types, see “Click Types in Advanced Search Reports” in *Creating the Search Experience*. This is an extensible field. You can add your own values to this parameter. We recommend that you do not use any values that Google has already defined. We recommend that you use a sequence containing alpha-numeric characters, underscores, hyphens, and periods.	`ct=c`
`q`	Query text	`q=island+countries`
`r`	Rank of the result on which the user clicks.	`r=7`
`s`	Specifies the index number of the first entry in the result set.	`s=24`
`url`	URL that the user clicked.	`url=http%3A//www.foo.com/`
`site`	Required parameter. If this parameter does not have a valid value, other parameters in the query string do not work as expected. Limits search results to the contents of the specified collection. You can search multiple collections by separating collection names with the OR character, which is notated as the pipe symbol, or the AND character, which is notated as a period. For more information about the `site` parameter, see “site” on page 19.	`&site=`*`collection`*

The following is an example request for advanced search reporting parameters:

```
/click?ct=desk.news&r=7&url=http%3A//www.foo.com/
bar.html&q=olympics&site=default_collection
```

The request URL contains the following parameters:

Value	Description
`/click`	Start of the advanced search reporting URL.
`ct=desk.news`	User clicked the **News** link at the top of the search page.
`r=7`	The rank priority of the click is 7.
`url=http%3A//www.foo.com/bar.html`	The URL on which the user clicked.
`q=olympics`	The search query the user entered that created the search page.
`site=default_collection`	The collection that is searched.

When a user clicks a cluster label the following example appears:

```
ct=cluster&cd=spanish+cuisine&q=spain&site=default_collection
```

This URL contains the following information:

Value	Description
`ct=cluster`	User clicked a cluster label on the search page.
`cd=spain+cuisine`	The click data identifies the click for "spain cuisine."
`url=http%3A//www.foo.com/bar.html`	The URL on which the user clicked.
`q=spain`	The search query the user entered that created the search page.
`site=default_collection`	The collection that is searched.

Responses

In response to the `/click` URL, the search appliance always sends the same HTTP response back to the browser to acknowledge receipt. The response has a status code of 204 (no content) and a MIME type of `image/gif`.

The Google implementation running on the client generates a URL with JavaScript as a dummy image object. The result is ignored, because the response does not affect the client behavior. Its purpose is to log the user interactions on the server side.

Appendices

This section contains:

Appendix A: Estimated vs. Actual Number of Results

The Google Search Appliance does not guarantee the ability to return a particular number of results for any given search query. The total count of results is an estimate of the actual number of results for the search request. This section covers issues relating to this topic.

In search appliance software version 6.2 and later, the estimated number of results is different depending on whether filtering is enabled.

- When filtering is not enabled, you see the estimated total number of results.
- When filtering is enabled, for all but the last page of results you see the estimated total number of results. If you have requested the last page of results, then you see the total number of filtered results, which is likely to be much smaller than the estimated total number of results.

You can use the `rc` search parameter to request an accurate result count for up to 1M documents, but it might introduce high latency.

Counting Results in Secure Search

The total count of search results is not provided when a secure search is performed, regardless of which type of output format, XML or HTML, is used. A secure search request includes the parameters access=a or access=s.

How the Google Search Appliance Determines the Number of Results to Return

When search results are returned, the number of results is determined by one of the following conditions:

- If the Google Search Appliance has results to satisfy the search request, then the requested number of results are returned.
- If the Google Search Appliance has fewer results than the number requested in the search request, the last page of results is returned. The last page is determined by dividing the total number of results into pages based on the number of results requested.
- If no results are found, then an empty result set is returned.

To determine if a results page is the last page of available results, check for any of the following conditions:

- The first result number returned does not match the first result number requested.
- The number of results returned is less than the number of results requested.
- The results returned do not contain a link to the next result set.

Navigation

When the total number of results returned is an estimate, the navigation structure for search results is based on this estimate. Google recommends two approaches for generating a navigation scheme for your search results:

1. Only provide the search user with the ability to navigate to the previous results page and the next results page. The output format can be configured to provide links to the previous and next result set when appropriate.
2. Provide the search user with the ability to jump to any search page within the estimated number of results. If the user requests a results page beyond which results are actually available, the last results page is returned. The navigation structure is updated when the last page is displayed. This is the behavior you see in the default output of the Google Search Appliance.

Automatic Filtering

When the automatic filtering feature is active, the number of results returned is significantly reduced. Automatic filtering reduces undesirable results such as duplicate entries. You can disable this feature using the instructions in “Automatic Filtering” on page 31.

Filtered search results are identified in the returned results. For example, the `<FI/>` XML tag is present in XML search results where automatic document filtering occurs.

Google recommends that the search results page displays a message on the last page similar to the following, when automatic filtering occurs:

In order to show you the most relevant results, we have omitted some entries very similar to the search results already displayed. If you like, you can *repeat the search with the omitted results included*.

This is the behavior you see in the default output format of the Google Search Appliance.

The underlined text in the message should be a hypertext link to submit the same search again with the parameter `filter=0`. Google finds that this method of informing users about automatic document filtering is effective. This method is used on the Google Internet search site.

If you are using OneBox modules to provide additional query results to your users, note that the results served through a OneBox module are reported separately. The number of OneBox results are not added to the number of standard results.

Appendix B: URL Encoding

Some characters are not safe to use in a URL without first being encoded. Because a Google Search Appliance request is made by using an HTTP URL, the search request must follow URL conventions, including character encoding, where necessary.

The HTTP URL syntax specifies that only alphanumeric characters, the special characters `$-_.+!*'()`, and the reserved characters `;/?:@=&` can be used as values within an HTTP URL request. Since reserved characters are used by the search engine to decode the URL, and some special characters are used to request search features, all non-alphanumeric characters used as a value to an input parameter must be URL-encoded.

To URL-encode a string, replace each non-alphanumeric character with its hexadecimal ASCII value, in the format of a percent sign (%) character followed by two hexadecimal digits. Such an ASCII value may be referred to as an escape code. Spaces can be replaced by the plus sign (`+`) character for query parameters except when requesting search results by meta name or values.

If you are using the search box on the search appliance, you single-encode the special characters `$-.+!*'()`. Underscores (_) do not need to be URL-encoded in the search box.

If you are using special characters in a search query, you double-encode the special characters `$-.+!*'()`.

Underscores (_) do not need to be URL-encoded in the search box or in a search query.

Some input parameters require that the values passed to Google search are double-URL-encoded. This requirement means that you must apply the URL encoding to the string twice in succession to generate the final value. See the input parameter descriptions ("Search Parameters" on page 10) for more information.

Special characters in a query are the ones described as query term separators (see "Special Characters: Query Term Separators" on page 22) and meta tags names and values. Special characters within the document content do not get indexed so they are not searchable. For example, an indexed document containing a paragraph ending with "the *end" is not searchable using query "%2Aend" in the GSA search box. Only 'end' is indexed.

For more information about URL encoding, see W3C (http://www.w3.org/TR/html401/interact/forms.html#form-content-type) and IETF (http://www.ietf.org/rfc/rfc1738.txt) web sites.

Examples

Original String	URL-Encoded String
`chicken -teriyaki`	`chicken+%2Dteriyaki`
`admission form site:www.stanford.edu`	`admission+form+site%3Awww.stanford.edu`

Original String	Doubly URL-Encoded String
`William Shakespeare`	`William%2BShakespeare`
`admission form site:www.stanford.edu`	`admission%2Bform%2Bsite%253Awww.stanford.edu`

Appendix C: Date Formatting

The search appliance recognizes dates in most reasonable formats. However, dates that only mention the year (YY or YYYY), such as 2008, are not used. For dates in the format month year, the date is assumed to be the first of the month. The search appliance currently recognizes most Latin1 month names, but not Chinese, Japanese, or Korean month names.

Format	Description	Example
YYYY	All digits in a year	`2008`
YY	Last two digits of a year	`08`
YR	All four digits or only the last two digits of the year	`YY`, `YYYY`
M	Month represented by one or two digits	`9` or `09`
D	Day of the month represented by one or two digits	`7` or `07`
MM	Month represented by two digits	`04`
DD	Day of the month represented by two digits	`07`
WK	Day of the week	`Monday` or `Mon`
MON	Month	`March` or `Mar`
O	The relationship of local time to Universal Time (UT). O is used in a standard date format that follows ISO/IEC 8824. O is denoted by a plus sign (+), a minus sign (-), or the letter Z. A minus sign indicates that the local time is ahead of UT; a plus sign, behind UT; and the letter Z, equal to UT.	Pacific Standard Time would be a minus sign because it is ahead of UT.

Acceptable Date Formats

The following table lists date formats that you can use with the Google Search Appliance.

Format	Separator	Example
YYYY-M-D	Hyphen	2008-2-27
YYYY-D-M	Hyphen	2008-27-2
YYYY.M.D	Period	2008.2.27
YYYY.D.M	Period	2008.27.2
YYYY/M/D	Slash	2008/2/27
YYYY/D/M	Slash	2008/27/2
D-M-YYYY	Hyphen	20-2-2008
M-D-YYYY	Hyphen	2-23-2008
D.M.YYYY	Period	20.2.2008
M.D.YYYY	Period	2.23.2008
D/M/YYYY	Slash	20/2/2008
M/D/YYYY	Slash	2/23/2008
YY-MM-DD	Hyphen	09-04-27
DD-MM-YY	Hyphen	27-04-09
MM-DD-YY	Hyphen	04-27-09
YY.MM.DD	Period	09.04.27
DD.MM.YY	Period	27.04.09
MM.DD.YY	Period	04.27.09
YY/MM/DD	Slash	09/04/27
DD/MM/YY	Slash	27/04/09
MM/DD/YY	Slash	04/27/09
WK, D MON, YR	Comma	Tue, 3 March, 2009
WK, MON D, YR	Comma	Tue, March 3, 2009
D MON, YR	Space and comma	2 Jan, 09
MON YYYY	Space	March 2009
MON D, YR	Space and comma	Mar 03, 09
MON YY	Space	Mar 09
YYYYMMDDHHmm	(none)	200903211642 (see Note 1 below)
YYYYMMDDHH	(none)	2009082116
YYYYMMDD	(none)	20090323
YYYYMM	(none)	200903
YYYY	(none)	2009
DDMMYYYY	(none)	23032009

Format	Separator	Example
`MMDDYYYY`	(none)	`03232009`
`YYMMDD`	(none)	`090225`
`DDMMYY`	(none)	`150209`
`MMDDYY`	(none)	`021509`
`YYYY`	(none)	`2009`

Date Formatting Notes

1. The `YYYYMMDDHH` and `YYYYMMDDHHmm` patterns for specifying dates are supported, however, the search appliance has no notion of sorting search results based on the difference of time in document dates. For example, if a document has a meta tag with a value of `200910212150` and a second document with a value of `200910210900` then the search appliance discards both dates and sets document dates to their modification time (because the `YYYYMMDDHHmm` format does not get parsed).

2. Use meta tags with dates in the ISO-8601 format (YYYY-MM-DD) to avoid the confusion caused by multiple dates and multiple formats in the title or text of the documents.

3. The date of each file is returned in the date field of the results. This cannot be turned off, but you can choose not to display it on the front end to your users. To learn more about sorting by date, see “Sorting” on page 37.

4. If no date is found for a file, it is indexed without date data. Results that do not contain date data are displayed at the end of the results with dates, sorted by relevance.

5. If you have documents that contain exceptions to the default dates rule, enter the specific URL or pattern for the file and place these rules at the top of your list. The rules are handled in the order in which they are specified in the rule list. The first rule containing a valid date for the document determines the date of the document.

To specify rules for dates of documents:

1. Click Crawl and Index > Document Dates.

2. In the Host or URL Pattern column, enter the host or pattern to which the rule will apply.

3. Use the drop-down list in the Locate Date In column to select the location of the date for the documents in the specified URL pattern.

4. If you select Meta Tag, specify the name of the meta tag in the Meta Tag Name column.

5. To add more rules, click the **Add More Lines** button.

6. After all the rules are specified, click the **Save Changes** button.

Examples of Rules

Rule #	Host or URL Pattern	Date Located In	Meta Tag Name
1	`www.foo.com/example/`	Title	
2	`www.foo2.com/archives/`	URL	
3	`www.foo.com/`	Meta Tag	`publication_date`
4	`www.foo2.com/`	Body	
5	`/`	Last Modified	

Because the document `http://www.foo.com/example/foo.html` matches the URL pattern in rule 1, the search appliance first checks for the date in the title of the document. The URL doesn't match rule 2, so the search appliance checks against rule 3. If the search appliance is unable to find a valid date in the title or the URL, the search appliance looks for the date in the meta tag named publication_date according to rule 3. If the search appliance is unable to find a valid date in the meta tag, the search appliance defaults to the last modified date of the HTTP server, according to rule 5.

The search appliance extracts the date from the `http://www.foo2.com/archives/20040605/abc.html` URL.

Because the document `http://www.foo.com/foo.html` does not match the URL pattern in rule 1, the search appliance looks for the date in the meta tag, according to rule 3 and defaults to rule 5 if the search appliance cannot find a valid date in rule 3.

For the document `http://www.foo2.com/foo.html`, the search appliance looks for the date in the body and defaults to the last-modified date.

For the document `http://www.foo3.com/foo.html`, the search appliance looks for the date only on the last-modified header as it only matches the URL pattern of rule 5.

Appendix D: Compressed Results

The Google Search Appliance supports serving compressed results.

The search appliance serves compressed results to browsers that support compression. The browser must send the following HTTP header to the search appliance:

```
Accept-Encoding: gzip
```

The search appliance will then serve compressed results. The browser uncompresses the results.

This applies to both XML and XSLT-transformed results. If the `Accept-Encoding: gzip` header is not present, the results are not compressed.

Index