GSA | Notes from the Field

Google Search Appliance

Introduction to Content Integration

August 2014

Content Integration

This paper discusses features that you can use to integrate content from various repositories into the Google Search Appliance (GSA).

About this document

The recommendations and information in this document were gathered through our work with a variety of clients and environments in the field. We thank our customers and partners for sharing their experiences and insights.

What's covered	This paper covers using feeds, connectors, and cloud connect.
Primary audience	Project managers, GSA administrators, and connector developers.
IT environment	GSA and various external data repositories, such as enterprise content management systems, LDAP, and Google Apps.
Deployment phases	Initial configuration of the GSA.
Other resources	● Learngsa.com provides educational resources for the GSA. ● GSA product documentation provides complete information about the GSA. ● Google for Work Support Portal provides access to Google support.

Contents

Introduction

In order to be able to search, we need to index all the content first. There is a range of possibilities:

- **Crawling:** Is the process to follow the links from a starting point, downloading all the content and adding it to the index.
- **Crawling through a Proxy:** Same as the previous process, but using a proxy in between e.g., to add Metadata (i.e. Microdata, security) or change content.
- **Database Crawling:** Allows indexing databases by using SQL queries. It is done by using a JDBC-SQL query that will map from the data model to the Feed format.
- **Feeding:** Is an XML-document that represents the document/content and permits to control the metadata and security of each document.
- **Connector:** Is the software that traverses the document from the repository, usually a DMS or CMS system, and sends it to the GSA in order to index it.

In the following chapters we will review the different indexing options in depth and will present advantages and disadvantages of each solution.

Chapter 1 Custom Feeds

Overview

A *feed* is an XML document that tells the Google Search Appliance about the contents that you want to index. It is used to push data to the search appliance. There are many reasons that you might consider using feeds, including:

- Content access
- JavaScript links
- External metadata
- Per-URL Access Control Lists (ACLs)
- Incorrect status code
- Urgent indexing requirements

Content access

You might consider using feeds because there is no web interface or other way to access the content through standard crawl. Sending a content feed is the only option in such cases.

JavaScript links

In some cases, web pages contain JavaScript-generated content or links that the GSA cannot crawl. The GSA's support of JavaScript-enabled links is limited to certain scenarios of embedded JavaScript. In some cases, especially with largely dynamically generated content, the only reliable way to index all the content of a website is through feeds.

External metadata

You might need to associate external metadata with documents. Properties embedded in documents and meta tags in HTML headers are automatically associated and indexed alongside the documents. But if there is additional metadata to be included—for example, from a content management system—it can be sent to the GSA by using feeds.

Per-URL Access Control Lists (ACLs)

If you need to use advanced features of per-URL ACLs for authorization, such as namespaces and case sensitivity, you must use a feed.

Incorrect use of HTTP 200 status code

In some cases, deleted pages do not return a 404 status code. If a URL returns an error page but with a 200 (OK) return code, the GSA treats it as an existing page and indexes it. The best approach to prevent this from happening is to send in "delete" feeds.

Urgent indexing requirements

You might need new or updated documents to be indexed immediately. For press releases and other types of time-sensitive content with a need for immediate visibility in the index, the best option is to send in feeds because even if the URL is already in the index or the content has changed, the GSA will move them to the front of the crawl queue.

Feed types

The Google Search Appliance supports the following two types of feeds:

- Web/Metadata-and-URL feeds
- Content feeds

Web/Metadata-and-URL feeds

A web/metadata-and-URL feed provides the search appliance with a list of URLs and metadata. It's still the GSA that's responsible for crawling the URLs. Sending a metadata-and-URL feed not only attaches additional metadata to a document, but also triggers a recrawl of the content. URLs embedded in the documents will also be crawled if they match the patterns defined in **Follow Patterns**.

Content feeds

A content feed provides the search appliance with both URLs and their content. A content feed can be either of the following types:

- Full feed
- Incremental feed

With full feeds, old content with the same URLs are removed first before new content is indexed. With incremental feeds, if content is empty with only metadata sent, the metadata will be updated and vice versa. URLs embedded in these documents will also be crawled if they match the patterns defined in **Follow and Crawl** patterns.

Updating feeds

Full feeds can't be modified (if they keep the same datasource name). If you send in full feeds first, and then send in incremental feeds, documents from the original feed are removed from index. This means that if fed content is ever likely to change, always use an incremental feed type with content feeds.

If you send an incremental feed with changed metadata, the new metadata replaces existing metadata.

Note: You cannot update individual metadata items—you must update all fields at once.

Feeds integrity

A well-designed feed client not only sends in feeds, but also makes sure the fed-in documents are indexed properly. The feeds are first pre-processed to verify the format, matched against both **Follow and Crawl** URL patterns and **Do Not Follow Patterns**. If the feeds are of the type metadata-and-URL, the GSA tries to crawl them. Otherwise the content is indexed directly. There might be errors during this stage. A feed client can use the GSA Administrative API to verify the status in both stages:

- Retrieving Data Source Feed Information
- Retrieving Document Status

Best practices

A well-developed solution using feeds should have the following traits:

- A small number of data source names.

  GSA keeps track of feeds status in files. If there are too many data sources, there will be too many files and it will negatively affect the Admin Console performance. Most often, one data source corresponds to one content source.

- A large feed file with many documents.

  The feed file has a maximum size of 1 GB. It's far more efficient for the GSA to process a large feed file with many documents rather than many small feed files, each containing one or only a few documents. Of course, it has to be balanced with the ease of error recovery: if a large feed fails, many documents will have to be resent.

- Compression of large feeds, used to improve performance.

- Base64 encoding, if your metadata includes non-ASCII characters

- "last-modified" property to enable the GSA to process feed files faster.

- If you're sending in content feeds that are purely text based, the throughput is significantly more than if you were sending in actual files that are base64 encoded that would then need to be decoded and potentially converted into HTML (as we need to do for Word docs, PDFs, etc)

- The feeds backlog can be read at http://<GSA>:19900/getbacklogcount. A good feed client will have a configuration to specify the maximum backlog. The feed client would then check this on the GSA before sending each feed and pause if the backlog setting is exceeded.

- To further improve feeds processing speed, round-robin your feeds across a few different datasource names (for example, 3), since these will be processed in parallel.

- When feeding ACLs, leverage inheritance to reduce the impact on both the content system and GSA index.

- ACL case sensitivity cannot be changed once they are fed in. The behavior of case sensitivity depends on the content system and should be considered in the initial design of the feed client.

Security

If metadata-and-URL feeds are used, the `authmethod` attribute in the feeds record helps the GSA to decide which authentication protocol to use when crawling the URLs. It also indicates whether or not to remove the URLs from public search (any value other than `authmethod="none"`). Since GSA v7.2 it is recommended that you always use the `authmethod` attribute—even for `none`.

However, using `authmethod` is not flexible. If changes are ever to be made, feeds have to be resent. A better approach is always to:

1. Set the attribute to a fixed value such as `httpbasic`.
2. Set up Crawler Access or Forms Authentication under **Content Sources > Web Crawl > Secure Crawl** in the Admin Console.

You can mark content as public or secure on the fly by checking or unchecking the **Make Public** checkbox on the **Content Sources > Web Crawl > Secure Crawl > Crawler Access** page.

Large scale deployments

The following considerations apply when designing feeds for advanced deployments:

- In a mirroring configuration, feeds can only be sent to the master node.
- In a distributed crawling and serving configuration, feeds can only be sent to the master node(s).
- In a unification configuration, feeds can be sent to any node.

Chapter 2 General Connector Considerations

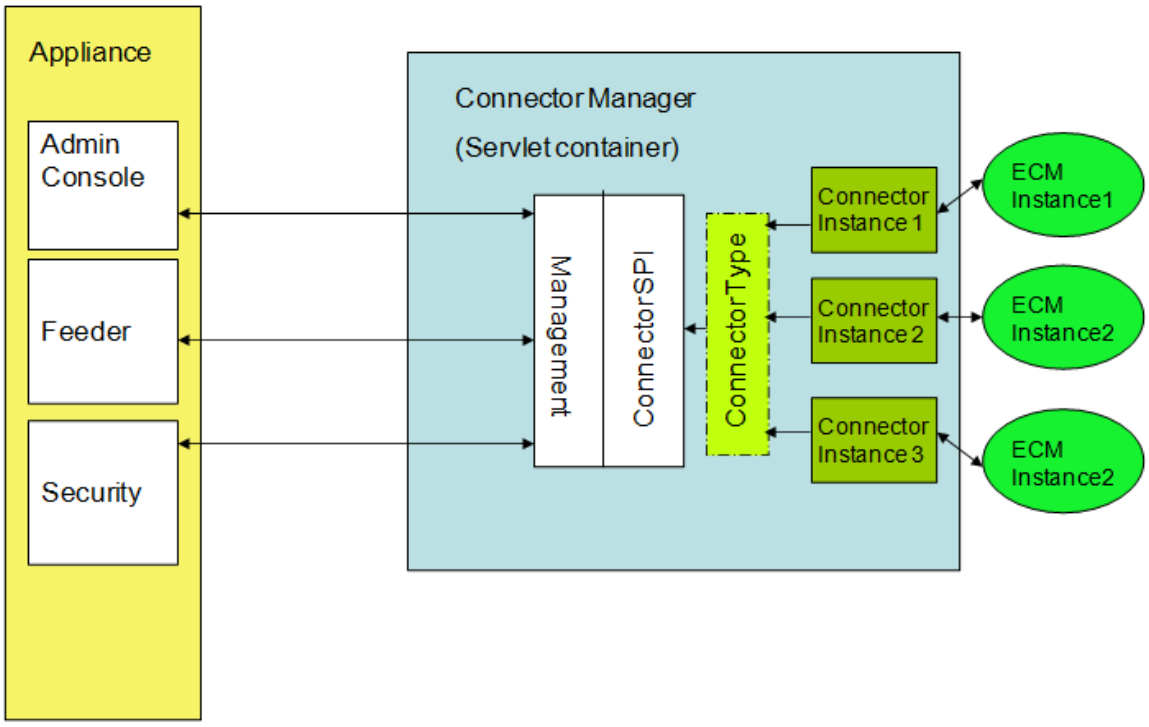
Overview

Connectors can extend the reach of the Google Search Appliance to non-web repositories, such as enterprise content management (ECM) systems, making more content available for search. The Google Search Appliance provides connectors for the following document repositories:

- Microsoft SharePoint
- Microsoft Active Directory
- File systems
- Databases
- Lotus Notes
- Open Text Livelink
- EMC Documentum
- IBM FileNet
- LDAP

GSA 7.2 introduced beta versions of the connector 4.0. This document currently focuses on the stable 3.2 version of the connectors.

To discover documents in a document repository through a connector, the Google Search Appliance does not crawl the content. Instead, it uses a process called “traversal,” in which the connector issues queries to the repository to retrieve document data to feed to the search appliance for indexing. The following diagram presents an overview of the components of a connector-based solution.



Connectors not only enable the Google Search Appliance to search and serve documents stored in non-web repositories, but they also enable the GSA to crawl web interfaces under limiting circumstances, such as:

- When the search appliance cannot reach all of the documents because of the complexity of the web sites, such as the extensive use of JavaScript
- When metadata associated with documents is not embedded in the documents (for example, when a separate database contains additional metadata about documents)

There are several factors to consider when planning connector deployment, including:

- Content repository size
- Security
- Physical location
- Connector customization

Content repository size

The first factor to consider is content repository size. Larger repositories take longer to traverse. With some connectors, traversal time can be reduced by using multiple connector instances, though you need to take care to avoid duplication of content.

The content size might also affect the deployment topology. Connectors are typically deployed on servers separate from the GSA. However, several connectors are built into the search appliance and can be run without additional hardware. These “on-board” connectors include SharePoint, File Share and LDAP.

Google recommends using on-board connectors if available only when the repository has fewer than 500,000 documents. Larger deployments require the on-board connectors to consume more resources on the GSA, impacting the overall performance of the search appliance. On the other hand, “off-board” connectors, which are external to the search appliance, allow for customizable hardware configurations and greater control of connector operations, troubleshooting and upgrades.

Security

Another major factor in planning a connector deployment is security. Connectors can handle both authentication (AuthN) of a user, group resolution and authorization (AuthZ) of search results. Typically, connectors are used to handle only authorization. They can use native APIs available from content repositories.

The authentication for secure search needs to be carefully planned and consolidated to one or two mechanisms supported by the search appliance such as Kerberos, forms or SAML. The goal is to establish the identity of the user so that connectors can perform authorization.

Physical location

When the Google Search Appliance is used to index content sources in different data centers, the physical location of connectors is an important consideration. If a connector handles authorization during serving, there must be at least three pieces in the deployment architecture:

- Google Search Appliance
- Content sources
- Connector

To reduce latency, at least two of the three should be collocated.

Connector customization

Sometimes it is necessary to modify the behaviors of an existing connector. For example, you might want to add authentication ability to the Google supported Database connector. Connector Manager is designed in such a way to make the customization easy. Connector Manager uses Spring Framework—it allows you to easily replace existing implementations of Connector Manager SPI with your own.

Another important and relatively new feature of the Connector Manager is Document Filter. Document filters can add, remove, or modify a document's properties, including the document content. Multiple document filters may be chained together, forming a document processing pipeline. For example, if certain documents in the content system are encrypted, you can decrypt them in a custom built Document Filter before allowing the documents to be fed to GSA.

Performance

Virtual machines can make traversal much slower than real servers if the shared hardware is over capacity. If you are using virtual servers, ensure the hosting server is appropriately sized and make sure the virtual servers have more capacity than the minimum required by connectors:

- 2G Memory (per CM)—you can select the size during installation.
- 20G hard disk
- 2.4GHz CPU

Also, include `filter=p` or `filter=0` in the query parameters during searches against the GSA. This minimizes the number of authorization requests required for each search query. For details about these query parameters, see Search Protocol Reference.

It's recommended that you monitor CPU, Memory and Disk resources to troubleshoot capacity issues with the connector or the underlying server.

Best practices for building a connector

Connectors using the Connector Framework Library implement a set of programming interfaces. The Connector Framework Library is open source. This means that in addition to the connectors supported by Google and its partners, you can extend the reach of your deployment with custom connectors for whatever content sources you need.

The Java-based Connector Manager provides the following benefits as a framework to build your custom connector:

- Well-defined interfaces to define traversal and security protocols.
- Built-in support of multiple threads for efficient traversal, scheduling, and host-load adjustment.
- Utility classes that handle common tasks.
- A fully supported framework that is updated regularly to take advantage of latest features on the Google Search Appliance.

The following list contains some best practices for building a connector:

- Start with the “Hello World” connector in Connector Developer’s Guide
- Always answer the following three questions:
    - How (fast) can I get the content into GSA for the first time?
    - How do I keep track of changed or added documents and how fast can they be discovered?
    - How do I remove deleted documents from the index?
- The purpose of the prefix to `googleconnector://` is for GSA to identify which connector instance is responsible for the authorization of certain URLs in the index. It’s fine to modify the Connector Manager to send in true URLs instead of this artificial one. The benefit could be improved relevancy.
- The connector does not have to perform authorization. For example, it’s perfectly fine for a SAML provider to do it. Sometimes not using the connector for authorization will simplify the design architecture and improve performance. For example, a SAML service or custom web service is available for authorization in an existing application server.
- If Per-URL-ACLs are defined on these URLs, the ACLs will be used for authorization.

Most likely the connector authentication is not needed because some kind of silent authentication for all or most of the content systems will be used and the connector only needs to be concerned with document authorization.

Chapter 3 Feeds or Crawler + Proxy or Adaptors Framework or Connector Framework?

Overview

You might need to decide whether a connector based on the Google Connector Framework, Google Adaptor framework (aka Plexi), crawling via proxy, or a feed client is the appropriate choice. Here are some arguments to help you to decide.

Using the Connector Framework

The Google Connector Framework is built on top of feeds with the following main features:

- Configuration, scheduling, UI integration, and many other utilities
- Specially formed URLs that allow the GSA to direct the authorization requests to the connectors (that is, googleconnector://)

The Connector Framework offers many useful features, but implementing a custom connector for the first time can involve a steep learning curve. Developing a custom connector would be appropriate if:

- Late-binding authorization is required, and the content system offers an API for verifying a user's access to documents
- The connector is to be reused for other GSA deployments

Even if you do not use the Connector Framework as a whole, there are still utilities that you can use as part of your feeds-based solution, such as the GSA Feed Manager, described in the next section. The Connector Framework can be found at http://code.google.com/p/google-enterprise-connector-manager/

Using Adaptors Framework aka Plexi

The new version of the Connector framework is called adaptor, and the internal name is Plexi. This new framework is a complete rewrite of the previous version and the major update comes with the architecture process, now based in Lister/Retriever pattern.

The **lister/retriever** architecture of adaptors provides many advantages over using content feeds when you can randomly access content: simplicity, scalability, self-healing, statelessness, transparent operation (easy monitoring and debugging), and it reuses existing infrastructure.

In this Developer's FAQ you can find answers to questions related to development with the adaptors framework.

In this link you can find the repository of the adaptors framework: https://code.google.com/p/plexi/

Using the crawler through a Proxy

The proxy can be used with the crawler in two scenarios:

- As a systems' requisite to control the GSA crawler, and use the proxy cache to avoid network overload or just being able to control the GSA Crawler.
- As a tool to rewrite the HTML on the fly and include in the metadata or in the content new information, extracted from the content or even from other sources, like databases.

It's in this second scenario, with the rewrite proxy, that the Content Integration process can benefit more, especially in scenarios without metadata. Metadata can be extracted from the content and added as normal metadata from the header of the document, e.g., microformats is a typical use case for this.

Using a feed client

It's worthwhile to consider using feeds as an alternative to building custom connectors. Since feeds are easier to build, it might be a better choice when:

- There is an easy way to query new/changed content
- There is an event trigger or monitoring system in place for new or changed content
- The content system doesn't support client API based in Java
- The development language of choice is not Java
- The implementation is a one-off integration (that is, ongoing indexing isn't required)
- Documents are considered public and not secured, or per-URL-ACL-based authorization is to be used

To push a feed to the search appliance, you require a feed client. One tool for using feeds to submit data to the GSA is the Feeds Manager. For more information about the Feeds Manager, see http://code.google.com/p/gsafeedmanager/. The GSA Feed Manager is not supported by Google for Work Support.

For more information about feeds, see the Feeds Protocol Developer's Guide.

Chapter 4 Google Search Appliance Connector for SharePoint

Overview

The Microsoft SharePoint connector enables the Google Search Appliance to index and search content files and metadata that are stored in Microsoft SharePoint. To use the Microsoft SharePoint connector, you typically need the following components:

- The connector running either on-board or on a separate host
- The set of custom web services called Google Services for SharePoint
- [Optional] The Google Search Box for SharePoint, which replaces the built-in search box in the SharePoint web front end

When Kerberos cannot be enabled on the Google Search Appliance, you will also need the SAML Bridge for Windows.

The normal deployment process consists of the following steps:

1. Install Google Services for SharePoint on SharePoint Web Front Ends.
2. Install and configure the SharePoint Connector and verify content is being indexed by the GSA.
3. Setup search-time security (authentication and authorization) and verify that silent authentication works, if required.
4. (Optional) If you'd like to allow users to use the Google Search Appliance without leaving the SharePoint UI, install Google Search Box for SharePoint and verify that secure search works from within SharePoint Web Front Ends.

Traversal

Traversal performance, which is how long it takes the connector to finish the traversal during initial deployment, is affected by many factors, including:

- The configured traversal rate
- The performance of the server that hosts the SharePoint connector
- The performance of the SharePoint Web Front End
- The number of documents, number of site collections and the depths of the sites

On average, the connector can traverse close to 1 million documents per day. You can increase the traversal speed, by:

- Increasing the traversal rate
- Setting up multiple connector instances

Increasing the traversal rate

Make sure the available memory is sufficient by monitoring the memory consumption of the hosting server. The default size of the allocated memory is 2GB. You can increase it if it's not enough. Also make sure that the SharePoint server has the capacity to handle the query load.

Setting up multiple connector instances

Set up multiple connector instances with each one handling part of the sites by using the exclusion patterns on the configuration page. For example, if there are hundreds or thousands of site collections, divide the top-level site connections to be allocated by multiple connector instances, with instance A handling site names starting from "a" to "c" and instance B handling site names starting from "d" to "f."

Early Binding of Groups

Starting with 3.0, a new Google Search Appliance Connector for Active Directory Groups was introduced. It's a solution for security instead of content acquisition. It replaces the previous releases' method of late binding for group resolution with early binding by traversing and storing both Active Directory groups in its own database.

SharePoint Connector 3.0 comes embedded with an AD Groups Connector. Besides Active Directory Groups, SharePoint local groups are also needed for ACL based authorization. SharePoint connector will retrieve and store both in a database. For a large Windows domain or SharePoint farm, this will add to the time it takes to complete the traversal.

Where is my file?

While going through all the site collections and sub sites, the SharePoint connector uses a depth-first traversal. If the site collection is really deep, it can take a long time for the content from the next site collection to show up.

Before you try to verify whether a particular document is indexed, check the value of `FullRecrawlFlag` for a web site in `sharepoint_state.xml` in the connector instance directory. There is an XSL file to help make the state file more readable. You can download the file (`SharePointStateReport.xslt`) from the following URL: http://code.google.com/p/google-enterprise-connector-sharepoint/downloads/list

If you have set up exclusion patterns in the SharePoint connector, it can take a lot of time to index a small number of documents because a lot of documents are discovered but discarded.

If metadata-and-url feeds are used, it's straightforward to locate the file in the **Index > Diagnostics > Index Diagnostics** section in GSA admin console. If content feeds are used, URLs from SharePoint are not used as the record URL in feeds. The Display URL will be the actual URL in the SharePoint server. By default, items in the SharePoint lists are indexed using the syntax `googleconnector://...?docid=http://...AllItems.aspx?<ItemID>`, and the GSA will crawl them as separate documents. To find the documents in **Index Diagnostics**, you need to find the record URL through the Display URL. Since GSA 6.8, you can use Display URL for site: and inurl. First, perform a search using the following syntax: `inurl:<fragment of the SharePoint URL>`. Make sure you have permission to access the file. Locate the file in the search results, and click on "cached" link. At the

top of the cached page, you can find the record URL starting with “googleconnector://”. You can then use this URL to locate the document through **Index Diagnostics**.

To find out whether a document has been discovered but excluded, you can check the `excluded_url.txt` file in the connector instance directory to make sure the connector is not excluding anything that you don’t intend to skip.

The SharePoint connector uses different regular expression pattern parsing libraries than the one used by GSA. Make sure to follow the documentation when entering the include/exclude patterns.

Presentation

SharePoint content can be served from the Google Search Appliance directly, or from within SharePoint Web Front End. The latter is achieved by installing the Google Search Box for SharePoint.

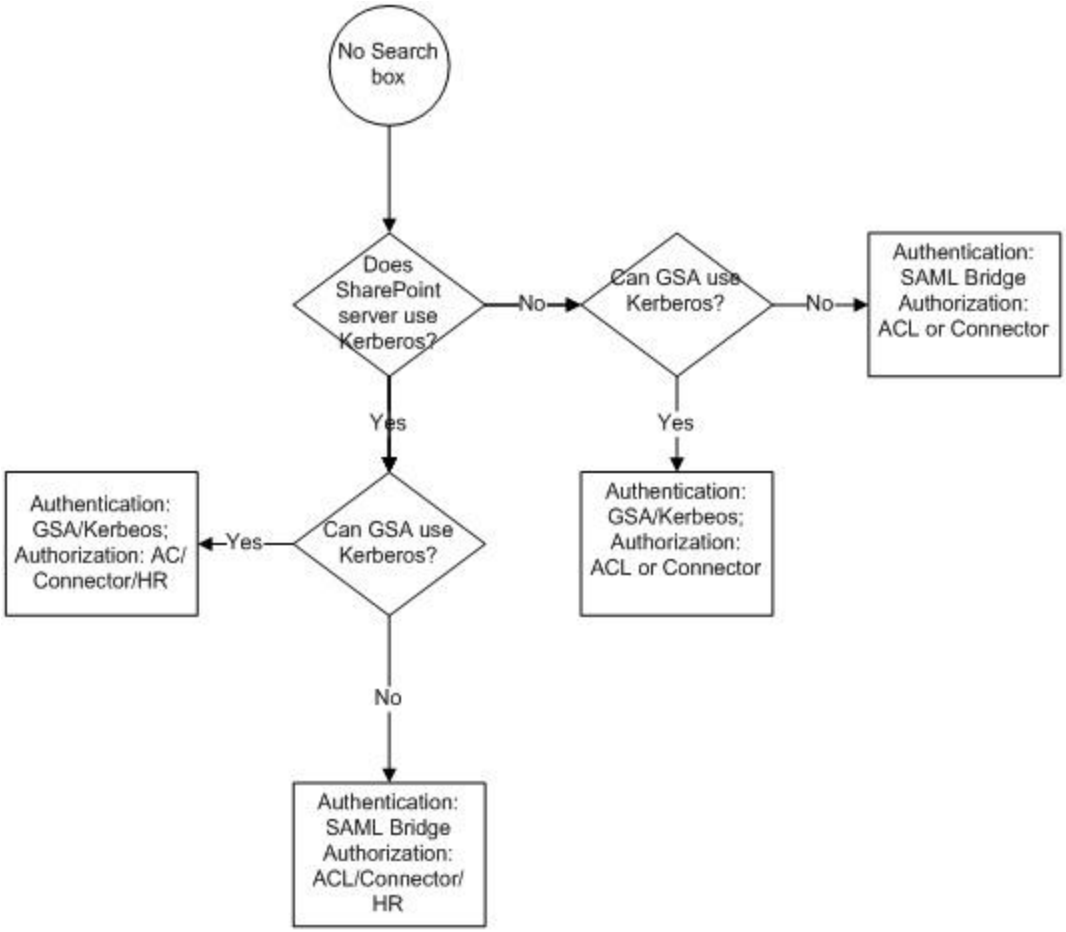
Google Search Box for SharePoint

Google Search Box for SharePoint provides an integrated experience and a similar look and feel to the SharePoint native search. Instead of using SharePoint’s search engine, the search box sends requests to the search appliance. It provides silent authentication experiences for an authenticated user if the security protocol is Integrated Windows Authentication, and also supports public search (that is, unsecured documents).

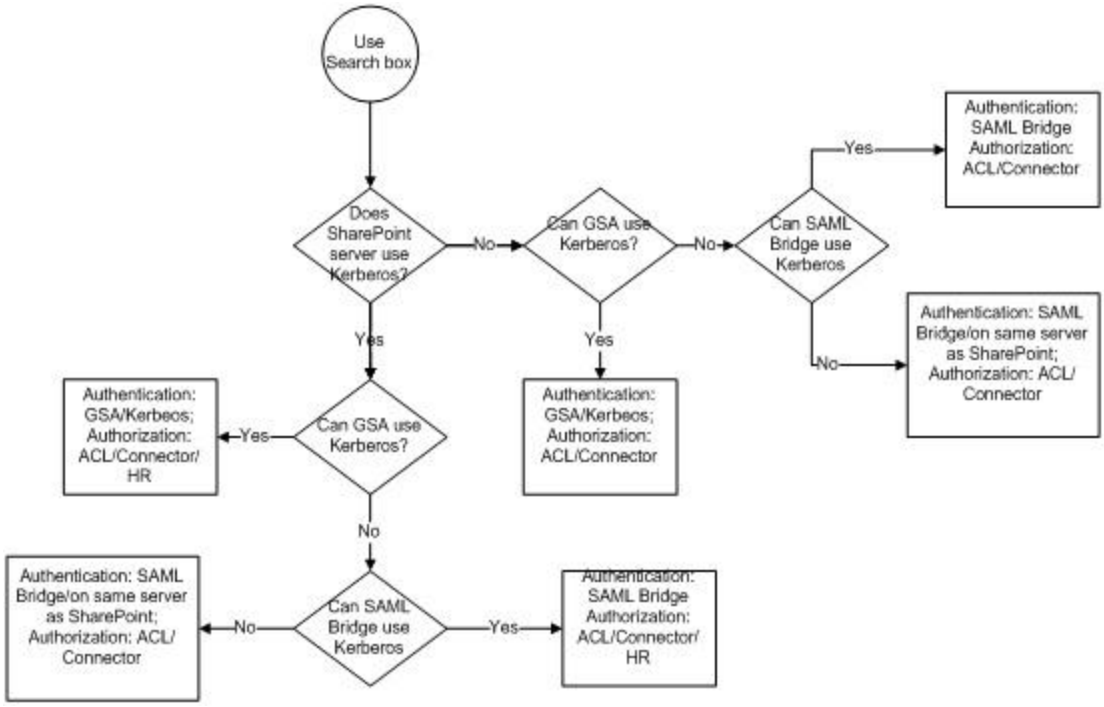
Secure search

Silent authentication is a very common requirement for secure search. It can be achieved using different approaches. The following two flowcharts present how to decide which components to use.

The following flowchart shows Silent Authentication *without* the Search Box for SharePoint. (HR means HTTP head request.)



The following flowchart shows Silent Authentication *with* the Search box for SharePoint. “No Kerberos” means using NTLM.



For details on the Google Search Box for SharePoint, visit the google-enterprise-connector-sharepoint wiki page. For details on SAML Bridge for Windows, see Enabling Windows Integrated Authentication.

Serving performance

To ensure optimal serving performance, make sure you follow these guidelines (in the order of preference):

- Use ACL for authorization. ACL support was added in release 2.8—this feature may provide the best performance. Since Connector release 3.0, the group resolution (both Active Directory Groups and SharePoint groups) became early binding through the use of Google Search Appliance Connector for Active Directory Groups. These newly added features greatly improved the authorization performance.
- Use the connector to perform authorization instead of head requests, if possible. (Both the GSA configured with Kerberos and with the SAML Bridge can perform head requests.)
- If the SAML Bridge is used for authorization, make sure to enable **Use batched SAML Authz Requests** on the **Search > Secure Search > Access Control** page.

On board or off-board?

For SharePoint Connector 3.0, we recommend always using off-board with an external database.

For official GSA partners, please review the GSA Watchpoints at Google for Work Connect (Deployment Section).

Chapter 5 Google Search Appliance Connector for File System

Overview

Google recommends using the File System connector for indexing file share. The built-in file system gateway was deprecated in 7.2.

Significant changes have been made to the File System connector in v3.2 and it works quite differently from the other connectors.

Traversal

The connector release 3.0 is based on the "lister/retriever" model—it discovers documents, sends the file properties and ACLs to GSA using metadata-and-url feeds. GSA will crawl the files through the File System Connector. The crawling frequency and other behavior are the same as web crawling.

Where is my file?

In release 3.0, the File System connector feeds the files to GSA using a URL in the following format: `http://<connector host:port>/connector-manager/getDocumentContent?ConnectorName=<connector name>&docid=smb://<path to file>`. Notice that the URL no longer starts with "googleconnector://".

Because the link ends with the path to the actual file, the query term `filetype` works. It also means that the **Do Not Follow Patterns** works for the file extension exclusion.

The best way to check whether a file is indexed is to use the **Index > Diagnostics > Index Diagnostics** page or to search by `inurl:<path to file>`.

The File System connector also filters by MIME types, by comparing a file's MIME type with the MIME type exclusion settings of the connector.

Presentation

The link to the file in the search results starts with `file://`. When clicked, it doesn't always open the file. If you want the clicking to cause the actual file to be opened, you need to customize the GSA's front end. Make sure to consider variances among browsers.

Secure search

The File System connector sends ACL entries for each file that it discovers, and per-URL ACL's are stored in the index of the GSA. This enables the File System connector to scale to handling millions of documents.

As of release 7.0, the Google Search Appliance supports deny ACLs and ACL inheritance.

AD Groups connector is needed to resolve groups and use the ACLs from the File System Connector.

On-board or off-board?

In v6.8 and higher, the GSA provides an on-board File System connector that can be used for light-weight deployments. However, off-boarding the File connector has several advantages:

- Scalable to handle very large file shares
- Works with GSA mirroring
- Easier to patch when there is a problem
- Advanced configurations are not available via the Admin Console
- For official GSA partners, please review the GSA File System Watchpoints at Google for Work Connect (Deployment Section).

Chapter 6 Google Search Appliance Connector for Database

Overview

The Google Search Appliance supports two ways to crawl databases:

- The built-in database crawler (deprecated in 7.2)
- The Database connector

Using the Database connector is preferable to using the built-in database crawler in the search appliance. It offers two important features that are not provided by the built-in database crawler:

- Secure search
- Scheduled crawling

Traversal

The Database connector can work in different scenarios, including those where:

- Each row is treated as a separate document
- There is a BLOB/CLOB field in the query results column
- A URL is associated with a database record

Each row is treated as a separate document

This scenario involves indexing database records with each row treated as a separate document. Take note that the connector query may be able to consolidate many database records into a single “record” from the connector’s perspective.

People choose this option instead of using a OneBox module for database serving mainly because of the following two reasons:

- When database records are indexed, they offer better relevancy
- Because serving of database connector traversed content does not involve querying of the database, it generally offers far better performance because the results are directly retrieved from the index

Here is how records are indexed in this scenario:

- Each row is converted to an XML file with column names as XML element names.
- Content feeds are generated, the primary URL is in the following format:
  `googleconnector://localhost/<host name>/DocID`
  The DocID is a Base64 encoding of the primary keys in the traversal query.
- Display URLs are in the following format:
  `dbconnector://<host name>/<database name>/<docId>`

Before the document is sent to the search appliance, an XSL stylesheet file is applied to transform the XML file to an HTML format. There is a default stylesheet provided with the connector, that can be customized for specific requirements such as:

- Indexing some columns as metadata by transforming associated XML elements as HTTP metadata headers.
- By following SEO best practices, creating HTML files that would make certain database columns more relevant (for example, HTML headings, bolded text, font sizes).

A BLOB/CLOB field in the query results column

When there is a BLOB/CLOB field in the query result column, content feeds are generated with the data of the BLOB/CLOB field treated as a document and the rest of the columns as metadata. The MIME type of the document is skipped if the MIME type is to be excluded, based on the Connector Manager's settings.

A URL is associated with a database record

When there is a URL associated with a database record, the connector sends in metadata-and-URL feeds.

Traversal rate

The Database connector keeps track of whether a record has changed by taking a snapshot of each file. When querying a large data set, the Database connector returns results in batches. If the traversal rate is too small, it'll take too long to finish. If the traversal rate is too large, the database query can hang. It's important to find the right traversal rate. Google recommends setting the traversal rate between 1000 and 1500 documents per minute.

Where is my record?

The document IDs of indexed documents are the checksum of primary key values. That's why there is no straightforward way to find a record in the crawl diagnostics. The best way to verify is to search for certain values directly.

Presentation

When each record is treated as a separate document and content feeds are sent, the display URL starts with "dbconnector://". In the default front end, they are transformed into links to cached results. No query is sent to GSA during serving.

Secure search

When the Database connector is configured to send in metadata-and-URL feeds, authorization of documents does not go through the connector.

The Database connector supports authorization by using a database query for content feeds, which requires a verified identity. A verified identity can be obtained in various ways. For details about verified identity, refer to Managing Search for Controlled-Access Content. The administrator needs to specify the authorization query with placeholders for user name and document ID's.

When an authorization SQL statement is configured, all feed records will have an attribute `authmethod="httpbasic"` indicating that it's secure content.

The Database connector does not provide an interface for connector authentication for secure search. The authentication must be performed by other GSA components such as Kerberos, cookie cracking, or the connector must be customized to provide an implementation for the AuthenticationManager SPI.

For detailed information on configuring secure search for the connector, see the google-enterprise-connector-database wiki page.

Chapter 7 Google Search Appliance Connector for Lotus Notes

Overview

The Lotus Notes connector version 3.0 is a major update from the previous versions of the Notes Connector. The new connector:

- Runs in the Connector Manager and Tomcat
- Uses the Notes client to access the Domino server
- Does not use the Agent Manager or HTTP Server on the Domino server
- Uses GSA Hierarchical ACLs for Notes application database ACLs and document reader names
- Uses a single configuration database in Domino
- Improves scalability to millions of documents
- No longer maintains metadata or reporting in the configuration database

Traversal

Traversal is managed by the connector manager; however, documents are temporarily queued in the configuration database and on the filesystem before they are sent to the GSA. The connector:

1. Polls the configured databases for changed/updated documents and creates a stub with the document id in the **Crawl Queue** of the connector database.

2. The crawler threads retrieve the documents and attachments from the source database and perform metadata transformation and place the documents in the **Submit Queue**.

3. The connector manager builds a feed from the documents in the **Submit Queue** and sends these to the GSA. Once the document has been sent to the GSA, only limited information is kept about the document in a local connector manager H2 database.

Where is my document?

The connector polls on a scheduled basis based on the configuration in the GSA Administration console. In addition, you can set how often an individual database is polled in the configuration database.

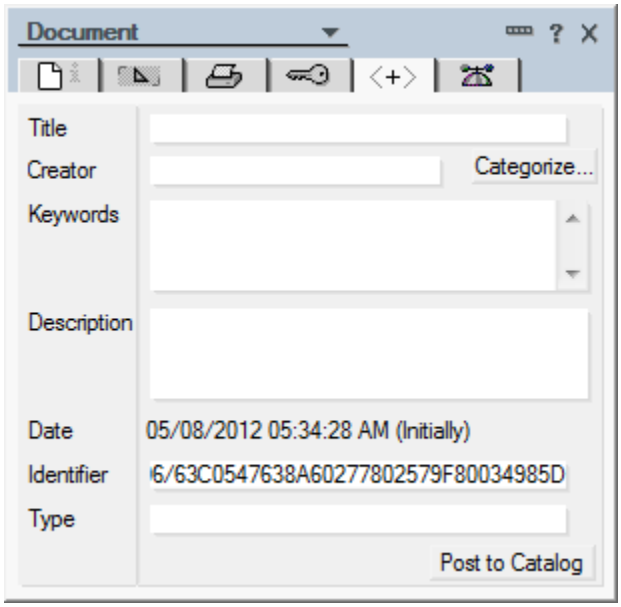
To verify that a document has been processed you should:

- Turn the logging level up to ALL to get all diagnostic information in the logs.

- Restart traversal for the database by following these steps:

    a. Open the connector configuration database
    b. Open the database configuration document
    c. Click **Edit**
    d. Click **Restart Traversal**
    e. Click **Save** and then **Exit**

- Find the UNID of the document by opening it in the Notes client and in the document properties dialog, go to the information <+> tab. In the identifier, the UNID is the string following the final / character.



- Search through the log looking for the document UNID to make sure the document is submitted correctly to the GSA.
- Check the **Content Sources > Feeds** page in the Admin Console to ensure that feeds were successful
- Review the **Index > Diagnostics > Index Diagnostics** page to check for documents that should have been crawled.

Finally, if a database has been processed but some documents have not been indexed you should:

1. Check the template configured for that database to ensure that the selection formula includes that document.
2. Check that the Notes user id you are using has access to the document (open the document in the Notes client using the Notes user id). The Notes connector does not index documents marked as "Replication / Save Conflicts"

Presentation

The default presentation of search results by the on-board XSLT is to link to the document through the HTTP protocol. Since the Notes protocol link is included in a metadata field in the search results, the XSLT can be customized to use this for search result links. With this method, assuming the Notes client is installed on the user's desktop (and registered with the browser), search results are opened using the Notes client. An example of this follows.

```
<xsl:when test="starts-with(U, $db_url_protocol)">
  <xsl:value-of disable-output-escaping='yes'
    select="concat('db/', $temp_url)"/>
</xsl:when>

<!-- *** BEGIN: Added for Notes URL replacement *** -->
<xsl:when test="starts-with(lower-case($stripped_url), 'newyork')">
  <xsl:value-of select="MT[@N='dom_noteslink']/@V"/>
</xsl:when>
<!-- *** END: Added for Notes URL replacement *** -->

<!-- *** URI for smb or NFS must be escaped because it appears in the URI query
*** -->
<xsl:when test="$protocol='nfs' or $protocol='smb'">
  <xsl:value-of disable-output-escaping='yes'
    select="concat($protocol,'/',$temp_url)"/>
</xsl:when>
```

Secure search

The default secure search mechanism uses connector authorization to authorize search results at serve time.

An alternative is to send ACLs to the GSA. This early-binding approach allows for much faster secure search performance, however, the following limitations will apply:

- For GSA V6.14 and prior, only the database ACL will be sent. Document level security using Reader and Author names fields will not be respected.
- "No Access" entries in the database ACL are not supported.

Tips & hints

Do not use “No Access” in the database access control list

Currently “No Access” entries in an ACL are not supported when using early binding (Per-URL ACLs). To work around this, set access as follows:

- Default is “No Access”
- Add groups to the ACL and assign the group the appropriate access

Do not index fields with limited visibility

To maintain consistency, do not include fields with hidden security in the Note user interface. This can be done by modifying the template used to crawl that database.

Turn off filtering

To show all documents from a database and improve serving performance set `filter=0` or `filter=p`.

Chapter 8 Google Search Appliance Connector for Livelink

Overview

The Open Text Livelink connector enables the GSA to index and search content files and metadata that are stored in an Open Text Livelink ECM. The connector formats content and metadata from the repository and feeds it to the GSA as a content feed.

The connector indexes the same items that Livelink indexes, such as documents, folders, projects, and discussions. The exact types and locations to be indexed are configurable. For documents, only metadata and the original file of the current version are indexed. Renditions and earlier versions are not indexed.

For more details on this connector, see the following resources:

- Configuring the Connector for Livelink
- Google Search Appliance Connector for Livelink

Traversal

The connector traverses Livelink content through the Livelink API by using the traversal username and password specified in the connector configuration. The scope of Livelink content retrieved can be controlled either by:

- Using the object IDs specified in the connector configurations
- Restricting the access the traversal user has to content

In using the latter approach, leave the object IDs for traversal blank, indicating that the connector should retrieve everything the traversal user has access to.

You can configure the Livelink connector to traverse the Livelink repository in a number of ways. By default, in v2.6.10, the Livelink connector uses a traversal method that utilizes a Livelink system table called the DtreeAncestors table. This approach provides the fastest traversal performance, but is dependent on a table (DtreeAncestors) that can often be out-of-date or hold incorrect information.

Due to these limitations, Google recommends that you configure the connector to use the “Genealogist” traversal approach, which is slower in performance but more reliable. Specifically, use the BatchGenealogist method because it yields the best performance out of the Genealogist traversal modes. For more details, see Advanced Configuration.

Where is my record?

When troubleshooting why content does not appear to be in the GSA, perform the following actions:

- Ensure that the feeds were successful in the GSA admin console Feeds page.
- In the GSA Admin Console, check the crawl diagnostics page for the page(s) you are looking for. You need to know the object IDs of the documents in Livelink, as these will be reflected in the crawl diagnostics URLs.
- Check the connector logs to ensure there are no related errors. If necessary, increase the logging level on the connector, as described in Configuring the Connector for Livelink.

Presentation

The search results can be displayed with the default XSLT, which you can configure to display the content's metadata, if the connector was configured to acquire it.

When configuring the Livelink connector, you specify Livelink URL for search results. This typically looks like: `http://host::port/Livelink/livelink.exe`

This is usually the standard URL through which users access Livelink. Take note that the Livelink URL must include a fully-qualified host name for the Cached and Text Version links in the search results to work.

Secure search

The connector supports authentication against the Livelink database and against an external directory service. The connector obtains authorization from the Livelink server when a user asks to view a particular item and the contents of the item. The GSA does not require any additional special configuration to support Livelink's user authentication and authorization mechanisms.

Currently, this connector does not support the sending of ACLs to the GSA and as such does not support the early-binding authorization mechanism.

Chapter 9 Google Search Appliance Connector EMC Documentum

Overview

The EMC Documentum connector formats content and metadata from an EMC Documentum repository and feeds it to the Google Search Appliance.

The connector crawls and indexes content files of the object type dm_document by default. Other subtypes of dm_sysobject, subtypes of dm_document, and custom subtypes can be indexed and searched, but you must manually add the types on the **Advanced Configuration** page when you configure a connector.

For more detailed information, see Configuring the Connector for Documentum.

Where is my record?

When troubleshooting why content does not appear to be in the GSA, consider the following questions:

- Was the content skipped by the query? Ensure that the object type is configured for crawling and that the WHERE clause does not exclude your content.
- Was the content skipped by the connector? Ensure that the content wasn't excluded by the connector, which could happen for several reasons:
    - MIME type not supported
    - File size too big
    - Retrieval error (for example, old DFC object types)
- Was the feed rejected by GSA? Check hosts that the GSA trusts feeds from, **Follow and Crawl** URL patterns, and that the feeds are successfully processed by the GSA.

Presentation

The search results can be displayed with the default XSLT, which you can configure to display the content's metadata, if the connector was configured to acquire it.

You can also control the way search results are opened by manipulating the "Webtop URL" item on the connector configuration page. This enables you to open the documents in different ways, such as:

- Opening the document with a choice to view or edit it
- Viewing the document
- Downloading the document directly
- Accessing the properties of the document

For detailed information, see Deciding on the Webtop URL Format.

As an alternative, you can build a WDK customization or a small DFC application that serves content without context. The only requirement is that the connector must append the object ID to the Webtop URL that you provide, which need not have anything to do with Webtop. You must also authenticate the user and check their permissions before returning documents, unless you want to provide completely open access to the documents.

Secure search

For secure content, the connector can use any Documentum user authentication mechanism.

At serve time, the connector requests the credentials of the user submitting a search request. Those user credentials are passed to the Content Server, which authenticates the user and determines which results the user is authorized to view. The GSA does not require any additional special configuration to support Documentum's user authentication and authorization mechanisms.

When configuring the connector, you can make content public by selecting the **Make Public** checkbox on the **Content Sources > Web Crawl > Secure Crawl > Crawler Access** page in the Admin Console. This will allow the content to be publicly searchable, regardless of the permissions in the repository.

Currently, this connector does not support the sending of ACLs to the GSA and as such does not support the early-binding authorization mechanism.

Chapter 10 Google Search Appliance Connector for IBM FileNet

Overview

The IBM FileNet connector enables the GSA to traverse, index, and search content in an IBM FileNet content repository. For more detailed information, see Configuring the Connector for FileNet.

Currently, the connector only indexes entities associated with the Document class (or sub-class) or a Workflow process.

Traversal

Traversal begins with the earliest document modification date and works forward. After the initial traversal, the connector works in an incremental mode to index documents that are added or modified.

You can review the checkpoint state of the connector to determine what documents have been crawled. This is maintained in the file `WEB-INF\...\<connector-instance-name>_state.txt`

The checkpoint consists of:

- uuid—last document added to index
- lastModified—date of last added document
- uuidToDelete—last document deleted from the index
- lastRemoveDate—date of last deleted document.

Where is my record?

Because the content is acquired by way of a content feed, you can check the connector status and the **Content Sources > Feeds** page in the GSA Admin Console to confirm that the feeds are successful. Similarly, each document can be checked on the **Index > Diagnostics > Index Diagnostics** page on the Admin Console. To find the document of interest, you need to know the FileNet document ID and search for this by using the **Index Diagnostics** page.

Presentation

When search results from the FileNet connector are displayed, they can be configured to point to the document itself or its metadata, via the FileNet Workplace client. This is discussed further under the **Workplace URL** setting in Connector Instance Parameters.

Secure search

The connector provides a configuration parameter for determining the viewing of public and secure results. See details on the **Make Public** setting in Connector Instance Parameters.

For secure search, authentication and authorization for the connector are handled through the FileNet Java application programming interfaces (APIs). The connector handles authentication and authorization in real time, using the late-binding approach. At this stage, early binding is not supported with the FileNet connector.

Chapter 11 Google Search Appliance Connector for LDAP

Overview

The LDAP connector was introduced in GSA software release 6.8. It uses a content feed to push LDAP content and metadata into the GSA.

For information about this connector, see Configuring the Connector for LDAP.

Where is my record?

To confirm that content is being sent to the GSA, use the **Content Sources > Feeds** page in the Admin Console to ensure that a feed has been received and successfully processed from the LDAP connector.

Presentation

The primary purpose of the LDAP connector is for use with Expert Search, which can be customized to display results from the LDAP connector in a unique fashion that suits displaying a person's contact information. For more information on Expert Search, see the help page.

Alternatively, content acquired by the LDAP connector can be displayed alongside traditional search results along with other documents that match the user's search criteria. In this scenario, it might be desirable to customize the XSLT to display results from the LDAP connector in a fashion similar to that used by the People Search sidebar element.

Secure search

The LDAP connector only supports public search; it does not support secure search.

Limitations

The LDAP connector uses an LDAP feature called pagination to page through LDAP results, rather than get all matching objects in one request. Some LDAP servers do not support this feature. For those that do not, the LDAP property called "MaxPageSize" (or equivalent) should be set to a high enough number so that all the objects can be returned in a single search result.

Chapter 12 Connector Deployment Architecture

Overview

This chapter examines connector high-availability configurations. These configurations are based on the following assumption:

- GSA mirroring is used to provide search appliance high availability
- Crawl-time high availability is not critical and hence not required.
- Secure search is a requirement hence serve-time high availability is important
- Authentication is not through a connector but through other mechanisms such as Kerberized GSA, cookie cracker or SAML provider.

With secure search, we will focus on the two use cases where a connector is used and hence must be setup to support high availability:

- GSA uses late binding by way of an authorization API going through the connector.
- Per-URL-ACL. Since ACLs are stored in the GSA index, the connector will not be used for authorization, but might be needed for group resolution. The exact configuration will depend on how it's implemented by each connector. It can be further divided in these scenarios:
    - GSA or other external components provides group resolution, such as LDAP or SAML. In this case, the connector is not used at all during serving.
    - Connectors are used for group resolution. At the time of writing, only the SharePoint connector and AD Groups connector provide group resolution support.

In this chapter, we will present an architecture to support both use cases. There is only one configuration described in detail. However, this is by no means the only option. The number of 9's in high availability is both a technical and business decision. For example, the configuration described below allows non-interrupt operation. If several hours' service interruption is tolerable when one of the components is down, a load balancing approach might be replaced with periodically backing up the connectors' configurations and state files or related database. A cost and benefit analysis should be performed before an approach is taken.

These configurations involve the following four components:

- The Google Search Appliances
- The Connector Manager's web application container (Tomcat)
- The Connector Managers
- The connector instances within the Connector Managers

Distinguishing between the Connector Manager and its hosted connector instances is very important (for naming reasons) and will help avoid confusion in the design. It's also important to know that the on-board connectors do not work with GSA mirroring. External connectors have to be used.

In this scenario, one GSA is considered a "Master" and the other a "Replica". This scenario applies to:

- GSA release 7.0 and above
- Connector release 3.0 and above

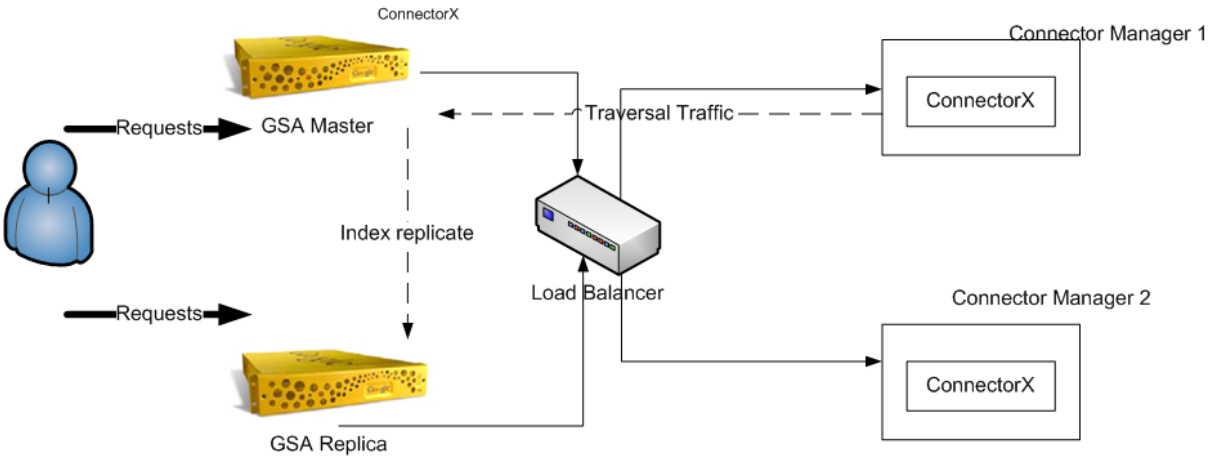
New releases of connector might change the deployment requirements.

Content is fed only to the "Master" GSA, with the entire search index then copied to the "Replica" GSA via the GSA replication mechanisms. The replica is created by using **GSA Mirroring Administration** on the **GSA^{n} > Configuration** page. Configuration of replication is not included in this document.

The replica gets its search indexes from the Master, so it **should not** be fed connector content directly. This configuration has ConnectorManager1 feeding the Master GSA, and ConnectorManager2 feeding no content at all.

During serve time, either connector would get requests from the GSA. In the case where the connectors are providing authorization directly, they will call out to the APIs provided by the content system. In the case where the connectors are providing group resolution, they will call out to a shared database to resolve current user's groups. Of course, the database server and the content system will have to be provisioned for high availability. This is out of the scope of our discussion.

The following diagram illustrates the connector high availability architecture in a mirrored configuration scenario.



Connector High Availability Deployment – With replication

Setting up a mirroring configuration involves the following major steps:

1. Configuring the GSA using the Admin Console
2. Manually configuring Tomcat
3. Adding the load balancer to the system

The following sections explain each of these steps in detail.

Step 1: Configuring the GSA using the Admin Console

On the master GSA:

1. Add the real IP address of ConnectorManager1 to the **List of Trusted IP Addresses** on the **Content Sources > Feeds** page.

   DO NOT enable "GSA Unification." This will break search authorization via the load balancer.

2. Add ConnectorManager1 by using **Register a New Connector Manager** on the **Content Sources > Connector Managers** page.

   The manager name should be unique to this Connector Manager (that is, "ConnectorManager1"). For **Location** or **Service URL**, specify the true IP address or host name of the machine (or vm) running ConnectorManager1, not the load balancer proxy address.

3. Add a new connector instance to ConnectorManager1 by using the **Content Sources > Connectors** page.

   For **Connector Name**, specify a non-unique name that will be shared by the connector instances in both Connector Managers. *The connectors must share this name so that load-balanced search Authentications and/or Authorizations will work.*

4. Ensure Traversals are enabled, and there is a connector traversal schedule.

5. Shut down ConnectorManager1.

6. Repeat Step 3, Registering ConnectorManager2.

7. Add a new connector instance to ConnectorManager2 using the **Content Sources > Connectors** page.

   For **Connector Name**, specify the same **non-unique** name that will be shared by the connector instances in both Connector Managers.

8. Ensure traversals are **disabled** by checking **Disable Traversal**.

Step 2: Manually configuring Tomcat

Add the IP address of the load-balancer to the list of allowed IP addresses. It's also recommended that you add the IP addresses of select system administration machines, because you will want it to troubleshoot this system.

The Connector Manager ships with a Tomcat Remote Address Valve enabled. This restricts traffic to the Connector Manager, allowing access only from the localhost and the GSA that was configured at installation time. In this step, you modify the Remote Address Valve to permit traffic from the load balancer. The complete instructions for modifying the Remote Address Valve are on the RemoteAddrValve wiki page.

For each Connector Manager:

1. Add the IP address of the load-balancer to the list of allowed IP addresses.
2. Recommended: add the IP addresses of select system administration machines, because you will want it to troubleshoot this system.

Step 3: Adding the load balancer to the system

In this step you are going to point the GSAs at the load balancer proxy rather than the connector managers.

1. Shut down the Connector Managers.

   This will allow you to delete the Connector Manager association with the GSAs, yet leave the connector configuration intact. Otherwise, the GSA will complain that it cannot delete a Connector Manager with running connectors. It will also complain when you try to add a new Connector Manager that has connector instances with existing names.

2. On the the master GSA, perform the following steps:

   a. On the **Content Sources > Connector Managers** page, unregister ConnectorManager1 and ConnectorManager2.

   b. Restart only ConnectorManager1.

   c. On the **Content Sources > Connector Managers** page, register the load balancer as a new Connector Manager.
   Its status should show a green ball. On the **Content Sources > Connectors** page, there should be an instance of the connector associated with the load balanced Connector Manager.

   d. Restart ConnectorManager2.

Note: At this point, you should no longer modify the connector's configuration or schedule via the Admin Console of either GSA. Doing so may result in an inconsistent, possibly corrupt state.

Replace a failed connector

In case of a connector server failure, the load balancer will automatically detect it and route the traffic to the active connector. The failed connector needs to be taken out of service, and a new connector needs to be registered. In the meantime, the service should not be interrupted. We will use the configuration and state migration approach instead of the shutting down/turning up approach during initial setup. This approach can also be used during initial HA setup. The procedure below describes general steps—it could vary based on connector type. For example, with the SharePoint connector, you will have to consider how to safely migrate state files when needed.

1. Install a connector of the same type on a new server—assuming that the failed server cannot be used. Make sure the connector service is stopped.

2. Copy the connector instance directory from the active connector server to the new connector server.

3. On GSA, edit the failed connector manager to point to the new server.

Chapter 13 Cloud Connect

Overview

The Cloud Connect functionality is a JavaScript based AJAX component that calls a service on the GSA, which in turn communicates with the Google Apps Search API. Responses from Google Apps are processed by the GSA for rendering. These results are rendered as sidebar elements with a limit of 10 results.

To configure personal content integration, use the **Content Sources > Google Apps** (Previous to 7.2 **Cloud Connect > Google Apps**) page in the Admin Console. In the configuration, the **Domain** to be entered is the Google Apps domain—without any prefix such as “www.” The **OAuth Consumer Key** represents a client (also a domain) registered with Google.

For example, search.yourcompany.com is an **OAuth Consumer Key** in the Google Apps Domain yourcompany.com. The two domains can also be totally unrelated. For example, yourcompany1.com is an **OAuth Consumer Key** in the Google Apps Domain yourcompany.com. The OAuth Consumer Key must be registered as a third-party authorized client to the Domain APIs.

For more information on how to register a client to a Google Apps domain, refer to: Using OAuth 2.0 for Web Server Applications.

Troubleshooting

The most common problem with Cloud Connect is no results are shown. To troubleshoot connectivity to Google Apps, use the following process:

1. Verify OAuth credentials
2. Verify primary verified ID
3. Verify firewall settings

The following sections explain each of these steps in detail.

Verify OAuth credentials

Because the call to the Google Apps Search service is made from a browser, it is difficult to troubleshoot what’s really happening. However, there is an OAuth Playground that can help test it: https://developers.google.com/oauthplayground/

To verify OAuth credentials in the OAuth Playground, take the following steps:

1. In section 1, **Choose your Scopes**, enter:
   https://www.googleapis.com/auth/appssearch

2. In section 2, **Modify the OAuth Parameters**, enter the following information:
   oauth_signature_method: HMAC_SHA1
   oauth_consumer_key: <user's consumer key from Google Apps Control Panel>
   consumer secret: <secret key from Google Apps Control Panel>

3. In section 3, click on the **Request Token** button to get a request token. It will populate the oauth_token.
4. In section 4, click on the **Authorize** button to authorize the request token. It will populate the oauth_token.
5. In section 6, click on the **Access token** button to upgrade to an access token. It will change the oauth_token to be an access token.
6. In section 6, paste the value after replacing text in "<>": https://www.googleapis.com/apps/search/v1r1?q=<some query term>&xoauth_requestor_id=<full Google Apps email address>
7. Click the **execute** button.

You should get a HTTP response of 200 and some results. If this is not working, then the OAuth credentials are not setup properly.

Verify primary verified ID

The primary verified identity (the user ID associated with the default credential group) is used as the ID for Cloud Connect. To verify the primary verified identity:

1. Open the **Search > Secure Search > Universal Login** page in the Admin Console.
2. Download the SegMgr log.
3. Check if the **Default** credential group is resolved properly.

Alias

If the user names obtained as the primary verified ID by the Google Search Appliance are different from the Gaia IDs of the Google Apps domain, these names must be added as aliases by the Google Apps domain administrator.

Verify firewall settings

Cloud Connect requires the GSA to be able to access https://www.googleapis.com/apps/search/v1r1/ over port 443. It will not work through a proxy. It is recommended that "google.com" also be enabled through the firewall for the GSAs to communicate openly with the Google APIs.