Google Search Appliance Connectors

Deploying the Connector for Databases

Google Search Appliance Connector for Databases software version 4.1.3
Google Search Appliance software versions 7.4 and 7.6

August 2017

Table of Contents

About this guide

This guide is intended for anyone who needs to deploy the Google Search Appliance Connector 4.1.3 for Databases. The guide assumes that you are familiar with Windows or LInux operating systems, databases, and configuring the Google Search Appliance by using the Admin Console.

See the Google Search Appliance Connectors Administration Guide for general information about the connectors, including:

- What's new in Connectors 4?
- General information about the connectors, including the configuration properties file, supported ACL features, and other topics
- Connector security
- Connector logs
- Connector Dashboard
- Connector troubleshooting

For information about using the Admin Console, see the Google Search Appliance Help Center.

For information about previous versions of connectors, see the Connector documentation page in the Google Search Appliance Help Center.

Before you deploy the Connector for Databases

Before you deploy the Connector for Databases, ensure that your environment has all of the following required components:

- GSA software version 7.4.0.G.120 or higher
  To download GSA software, visit the Google for Work Support Portal (password required).
- Java JRE 1.7u9 or higher installed on computer that runs the connector.
- If you want to use the DH (Diffie-Hellman) style of encryption and you are running the GSA with 2048-bit encryption, JRE 1.7u80 or 1.8.0_20 is required.
- JDBC driver jar for your database. Suggested is Oracle JDBC version 6 for Oracle Database 11g, Oracle Database 12c, and Microsoft SQL Server JDBC version 4 for Microsoft SQL Server.
- Connector for Databases 4.1.3 JAR executable or installer option for Windows
  For information about finding the JAR executable, see Step 2 Install the Connector for Databases.
- SQL account (db.user and db.password) must have read access to necessary database tables.
- Sufficient hard disk for log files on the connector host.

GSA host load, CPU, and memory recommendations

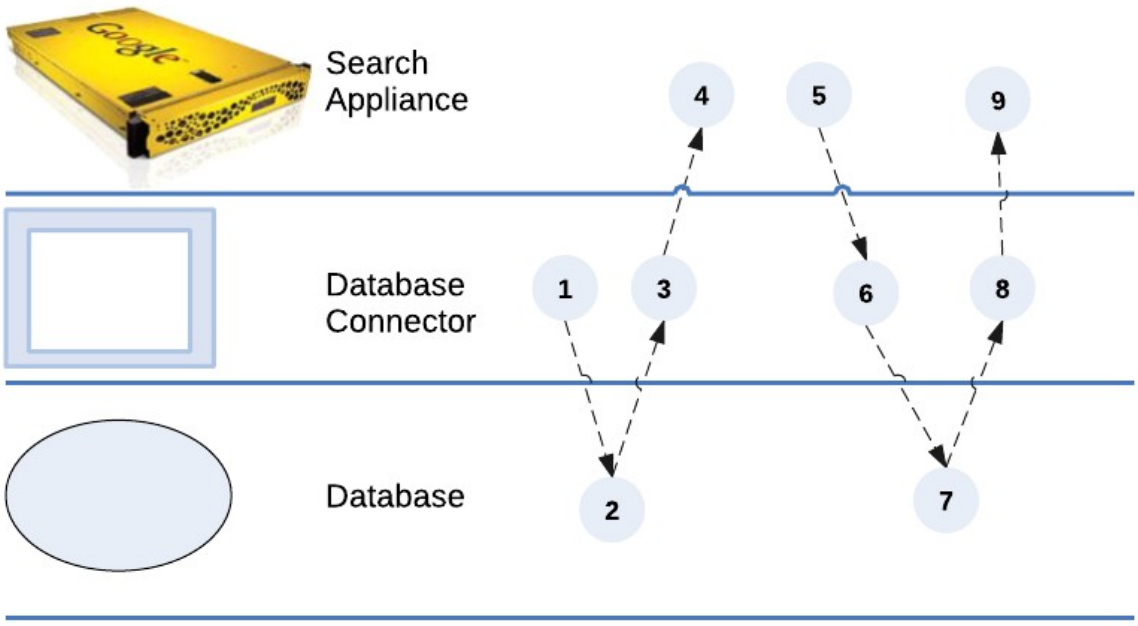
The following table contains GSA host load, CPU, and memory recommendations for the Connector for Databases.

Number of rows	Size of row	Size of ID	GSA host load	Recommended RAM	Recommended CPU cores
10,000,000	10MB	20 bytes	4	500 MB	4
20,000,000	10MB	20 bytes	4	900 MB	4
20,000,000	10MB	20 bytes	16	1.2 GB	8
20,000,000	10MB	60 bytes	16	2.5 GB	8
60,000,000	10MB	30 bytes	8	4 GB	6
60,000,000	25MB	30 bytes	8	4 GB	6

Overview of the GSA Connector for Databases

The Connector for Databases enables the Google Search Appliance to crawl and index content from several databases. A single connector instance can support a single database.

The following diagram provides an overview of how the search appliance gets content from the database through the Connector for Databases. For explanations of the numbers in the process, see the steps following the diagram.



1. The Connector for Databases sends a SQL query for all DocIds to the database.
2. The database streams DocIds to the connector.
3. The connector constructs URLs from the DocIds and pushes them to the search appliance in a metadata-and-URL feed. The feed file can contain a maximum of 5000 URLs. Take note that this feed does not include the document contents.
4. The search appliance gets the URLs to crawl from the feed.

   For urlAndMetadataLister mode, the search appliance gets a list of URLs and Metadata. The URLs point at some other repository (not the connector) and the GSA will crawl those sites according to its own crawl schedule.

The following steps only apply if the database mode of operation is Row to Text mode, Row to HTML mode, URL mode, File path mode, or Content mode.

5. The search appliance crawls the repository according to its own crawl schedule, as specified in the GSA Admin Console. At crawl time, it sends a GET request for a single URL to the connector.
6. The connector constructs a SQL query for the requested URL and sends it to the database.
7. The database extracts a row result set and sends it to the connector.
8. The connector constructs a document and sends it to the GSA. The actual format of the document depends on the database mode of operation for the connector.
9. The search appliance continues to crawl the repository.

Automatic updates every 15 minutes

The database connector monitors for changes when the configuration has an `db.updateSql` SQL select query, which provides updated and new document IDs. Also, if there is a difference between the database server's and the connector's time zones, also consider the setting of `db.updateTimestampTimezone` when setting `db.updateSql`.

The interval between updates is determined by the connector configuration option `adaptor.incrementalPollPeriodSecs`. The default interval value for automatic updates is 15 minutes, but you can configure it to suit your needs. For more information, see "Common configuration options" in the Administration Guide.

Supported operating systems for the connector

The Connector for Databases must be installed on one of the following supported operating systems:

- Windows Server 2016
- Windows Server 2012 R2
- Windows Server 2012
- Windows Server 2008 R2
- Linux

Certified databases

The Connector for Databases is certified for use with the following databases:

- Microsoft SQL Server 2016
- Microsoft SQL Server 2014
- Microsoft SQL Server 2012
- Oracle 12c Release 2
- Oracle 12c Release 1
- Oracle 11g Release 2

However, the Connector for Databases works with any database with a JDBC 4.0 compliant driver.

ACL support

The connector for databases supports controlling access to documents by using ACLs. By default, the connector does not query the database for ACLs and all documents are public. To implement ACL support, use the following optional configuration variables:

- `db.aclSql`
- `db.aclSqlParameters`
- `db.aclPrincipalDelimiter`

Providing a `db.aclSql` statement automatically enables secure serving. If you do not provide a `db.aclSql` statement, all content is public.

Deploy the Connector for Databases

Because the Connector for Databases is installed on a separate host from the GSA, you must establish a relationship between the connector and the search appliance.

To deploy the Connector for Databases, perform the following tasks:

1. Configure the search appliance
2. Install the Connector for Databases
3. Optionally, configure adaptor-config.properties variables
4. Run the Connector for Databases

Step 1 Configure the search appliance

For the search appliance to work with the Connector for Databases, the search appliance needs to be able to crawl database content and accept feeds from the connector. To set up these capabilities, perform the following tasks by using the search appliance Admin Console:

1. Set the GSA to crawl the connector.
2. Set the GSA to accept feeds from the connector.
3. Set up connector security.

Set the GSA to crawl the connector

To add the URLs provided by the connector to the search appliance's crawl configuration follow patterns:

1. In the search appliance Admin Console, click **Content Sources > Web Crawl > Start and Block URLs**.
2. Under **Follow Patterns**, add the URL that contains the hostname of the machine that hosts the connector and the port where the connector runs.
   For example, you might enter http://connector.example.com:6078/doc/
   where connector.example.com is the hostname of the machine that hosts the connector.
   When no port number is in configuration file, the connector runs on port 5678, however the Windows installer prompts you to specify a port number, with the default of 6078.
3. Click **Save**.

Set the GSA to accept feeds from the connector

To add the IP address of the computer that hosts the connector to the list of trusted IP addresses so that the search appliance will accept feeds from this address:

1. In the search appliance Admin Console, click **Content Sources > Feeds**.
2. Under **List of Trusted IP Addresses**, select **Only trust feeds from these IP addresses**.
3. Add the IP address for the connector to the list.
4. Click **Save**.

Set up security

For information about setting up security, see "Enable connector security" in the Administration Guide.

Step 2 Install the Connector for Databases

This section describes the installation process for the Google Search Appliance Connector for Databases on the connector host computer. This connector version does not support installing the connector on the Google Search Appliance.

Take note that you can encrypt the value for `db.password` before adding it to the file by using the Connector Dashboard or command-line tool, as described in "Encode sensitive values" in the Administration Guide.

Database modes of operation

The document that the connector constructs from a row result set and sends to the GSA depends on the database mode of operation for the connector:

- Row to Text mode
- Row to HTML mode
- URL mode
- File path mode
- Content mode
- urlAndMetadataLister mode

Set the database mode of operation by using the `db.modeOfOperation` configuration option. The database mode of operation is required.

Row to Text mode

In Row to Text mode, the connector converts a row into plain text. In this mode, the connector serves the row as a plain text document.

For Windows installation, set this mode by selecting **rowToText** from the pull-down menu in the **Database Mode of operation** field in the install wizard.

For Linux or command-line installation, set this mode in the `adaptor-config.properties` file by using the `db.modeOfOperation` configuration option:

```
db.modeOfOperation=rowToText
```

Row to HTML mode

In Row to HTML mode, the connector converts a row into HTML with XSLT. In this mode, the GSA serves the row as an HTML document.

For Windows installation, set this mode by selecting **rowToHtml** from the pull-down menu in the **Database Mode of operation** field in the install wizard.

For Linux or command-line installation, set this mode in the `adaptor-config.properties` file by using the `db.modeOfOperation` configuration option:

```
db.modeOfOperation=rowToHtml
```

URL mode

In URL mode, the connector gets the URL of the content from a particular column in the row. The GSA gets the content to index from the URL. The GSA also indexes other columns in the row, which might include columns that contain metadata.

For Windows installation, set this mode by selecting **urlColumn** from the pull-down menu in the **Database Mode of operation** field in the install wizard.

For Linux or command-line installation, set this mode in the `adaptor-config.properties` file by using the `db.modeOfOperation` configuration option:

```
db.modeOfOperation=urlColumn
```

You can cause the GSA to crawl secure URLs in the database by setting the configuration option `docId.isUrl=true`. However, with this setting alone, the connector does not provide metadata from other columns of the database. To retrieve metadata from all columns, use urlAndMetadataLister mode.

Setting the option `docId.isUrl=false` prevents crawling of secure URLs in the database. With this setting, the connector requests URLs and provides metadata, but prevents crawling secure URLs.

File path mode

In file path mode, the GSA extracts a file path from a particular column in the row. The GSA also indexes other columns in the row, which might include columns that contain metadata.

For Windows installation, set this mode by selecting **filepathColumn** from the pull-down menu in the **Database Mode of operation** field in the install wizard.

For Linux or command-line installation, set this mode in the `adaptor-config.properties` file by using the `db.modeOfOperation` configuration option:

```
db.modeOfOperation=filepathColumn
```

Content mode

In Content mode, the GSA extracts the content from a particular column in the row. The GSA also indexes other columns in the row, which might include columns that contain metadata. This mode supports SQL LOBs, binary, and character types, and Oracle BFILE. For a complete list of the types of data the GSA can index, see Indexable File Formats.

For Windows installation, set this mode by selecting **contentColumn** from the pull-down menu in the **Database Mode of operation** field in the install wizard.

For Linux or command-line installation, set this mode in the `adaptor-config.properties` file by using the `db.modeOfOperation` configuration option:

```
db.modeOfOperation=contentColumn
```

Note: `contentColumn` replaces blobColumn (which is still allowed).

urlAndMetadataLister mode

In urlAndMetadataLister mode, the GSA retrieves metadata for all URLs.

For Windows installation, set this mode by by selecting **urlAndMetadataLister** from the pull-down menu in the **Database Mode of operation** field in the install wizard. Selecting this mode automatically sets the configuration option `docId.isUrl=true`.

For Linux or command-line installation, set this mode in the `adaptor-config.properties` file by setting the `docId.isUrl=true` and using the `db.modeOfOperation` configuration option:

```
db.modeOfOperation=urlAndMetadataLister
```

To index metadata with this mode, specify the columns in the `db.metadataColumns` configuration option, and include those columns in the `SELECT` list of the `db.everyDocIdSql` query. For example:

```
db.everyDocIdSql=select id, name, dbcol from mytable
db.metadataColumns=name,dbcol:gsa_metaname
```

You must set the `db.uniqueKey` configuration option when using this mode of operation. For example:

```
db.uniqueKey=<URL Column>
```

Display URL column

In any mode, you can provide `displayUrlCol` in the `adaptor-config.properties` file. Set this variable when your database has a column whose values you'd like to use as search result links.

For example:

```
db.modeOfOperation.rowToHtml.displayUrlCol=URL
```

This setting causes the value in the column named "URL" to be used when displaying search results.

URLs for search results

For URL mode, the GSA displays the URL of the content in search results. For all other modes, the GSA displays a URL in the following format:

http://<adaptor:port>/doc/<unique-key-value>

For more information about URL patterns, see Constructing URL Patterns.

Windows installation

To install the Connector for Databases:

1. Log in to the computer that will host the connector by using an account with sufficient privileges to install the software.
2. Start a web browser.
3. Visit the connector 4.1.3 software downloads page at http://googlegsa.github.io/adaptor/index.html.
   Download the `exe` file by clicking **Database** in the Windows Installer table.

   You are prompted to save the single binary file, `database-install-4.1.3.exe`.

4. Start installing the file by double clicking `database-install-4.1.3`.
5. On the **Introduction** page, click **Next**.
6. On the **GSA Hostname and other required configuration values** page, enter a value for **GSA Hostname or IP address** of the GSA that will use the connector.

   For example, enter `gsa.hostname=`**`yourgsa.example.com`**

7. Enter the **Adaptor port number** for any crawlable documents this connector serves.

   Each instance of a Connector on the same machine requires a unique port. The default is 6078.

8. Enter the **Adaptor dashboard port number** for the Connector Dashboard.

   The value is the port on which to view a web page showing information and diagnostics about the connector. The default is 6079.

9. Specify the **JDBC jar file (Database driver)**.

   A Java Database Connectivity (JDBC) driver enables the connector to interact with the database. The JDBC driver is required.

10. Enter the **Full classname of Java JDBC Driver**.

    An example of a full classname of a JDBC driver is `oracle.jdbc.OracleDriver`.

11. Enter the **Database URL.**

    The connector uses the **Database URL** to communicate with the database. The Database URL is required. An example database URL is: `jdbc:oracle:thin:@45.62.11.99:1521:MY_ORACLE`
12. Enter the **Database username**.

    The connector uses the **Database Username** to query the database. The Database Username is required.

13. Enter the **Database password.** The Database password is required.

    When using MS SQL to authenticate without specifying username/password in the configuration file explicitly, you can modify your JDBC URL to use Windows Integrated Authentication. For more information, see https://msdn.microsoft.com/en-us/en%C2%ADus/library/ms378428.aspx.

14. Enter the **Database Unique Key**.

    The **Database unique key** is one or more column heading names (separated by commas) that provide a unique identifier for a database query result. The unique key allows each result row from a database query to be reliably identified by the retriever query. The Database unique key is required.

    A database unique key might include columns such as Last_Name, First_Name, SSN, Birth_Date. Optionally, you can map each unique key column to its type, for example: `customer_id:int`

    The valid types of unique key column are `int`, `long`, `numeric`, `string`, `timestamp`, `date`, and `time`.

    This is a fixed list of types. Generic SQL types are not supported.

    For a mapping between MS SQL Server's type to Java language type, refer to http://msdn.microsoft.com/en-us/library/ms378878(v=sql.110).aspx

For a mapping between Oracle's type to Java language type, refer to http://docs.oracle.com/cd/B19306_01/java.102/b14188/datamap.htm#CHDFJDIC and http://docs.oracle.com/cd/B19306_01/java.102/b14188/datamap.htm#CHDDABAA

15. Enter the **SQL statement that provides all IDs of all documents to be indexed**.

    This statement is a lister query, which provides all unique key values of all documents to be indexed. Each row result corresponds to a separate document. The information retrieved from the lister query provides the columns for indexing. This SQL statement is required.

    The following code shows the format of a SQL lister query.

    ```
    SELECT <table.keyColumn> [, <table.keyColumn>, ...,
    FROM <table>
    [WHERE some_condition]
    ```

16. Enter the **SQL statement that provides one document's content**.

    This statement is a retriever query. A SQL retriever query is used when a user clicks on a search result link, to retrieve and display the desired document data from the database. This SQL statements is required.

    The following example shows a retriever query:

    ```
    select * from oe.customers where customer_id = ?
    ```

    A retriever query displays result data using the "?" in the `WHERE` clause to allow for particular row selection and display.

17. Enter the **Database Mode of operation**.
18. Enter column names in the **Database metadata columns** field.

    Database metadata columns specify columns to treat as metadata and optionally provides a string to use as a key. Different metadata names can be supplied for the GSA index. For example, the following indexes the `name` column as `name`, and the `dbcol` column as `gsa_metaname`:

    ```
    name,dbcol:gsa_metaname
    ```

The default is an empty string. Take note that you can optionally provide a `gsa_metaname` for each column name.

19. Enter a **SQL statement to retrieve recently change documents by their timestamp**.

    The update SQL statement is used to retrieve recently changed documents by their timestamp. The update SQL statement is optional.

    For example:
    ```
    select customer_id, order_placed_time as GSA_TIMESTAMP from
    oe.customers where order_placed_time >= ?
    ```

    If there is a difference between the database server's and the connector's time zones, then specify the DB server's time zone in the **Timezone used for timestamps** field.

    The only exceptional case is that when you are using Microsoft SQL Server's `datetimeoffset`, use UTC or GMT regardless of the database server's time zone.

    For example:
    ```
    db.updateTimestampTimezone=America/Los_Angeles
    ```

    By default, the value is an empty string, which means it uses the connector machine's timezone.

    For valid values, refer to http://docs.oracle.com/javase/7/docs/api/java/util/TimeZone.html.

20. Enter the **SQL statement to retrieve users/groups that are permitted or denied access to a document**.

    This is a database ACL SQL statement, which specifies a SQL query to retrieve users and groups that are permitted or denied access to the row. The Database ACL SQL statement is optional.

    An example for `db.aclSql` is:

    ```
    select permitted_users as GSA_PERMIT_USERS, denied_users as
    GSA_DENY_USERS from my_acl_table where customer_id = ?
    ```

21. Enter the **Delimiter on retrieved results of above access control SQL statement**.

    This delimiter is used to separate principals in returned ACL column values. By default, this is a comma [,]. The Database ACL delimiter is optional.

22. Enter the **Maximum Java Heap size (in megabytes).**

    Refer to the entry in the table in GSA host load, CPU and memory recommendations that shows a suggested value for "Recommended RAM." Specify that value as the **Maximum Java Heap size**.

23. **Specify Whether or not to run the connector after the installer finishes.**
24. Click **Next**.
25. On the **Choose Install Folder** page, accept the default folder or navigate to the location where you want to install the connector files.
26. Click **Next**.
27. On the **Choose Shortcut Folder**, accept the default folder or select the locations where you want to create product icons.
28. To create icons for all users of the Windows machine where you are installing the connector, check **Create Icons for All Users** and click **Next**.
29. On the **Pre-Installation Summary page**, review the information and click **Install**. The connector Installation process runs.
30. On the **Install Complete** page, click **Done**. If you selected the option to run the connector after the installer finishes, the connector starts up in a separate window.

Linux installation

To install the connector:

1. Log in to the computer that will host the connector by using an account with sufficient privileges to install the software.
2. Start a web browser.
3. Visit the connector 4.1.3 software downloads page at http://googlegsa.github.io/adaptor/index.html.
4. Download the Connector for Databases JAR executable (`adaptor-database-4.1.3-withlib.jar`) at https://github.com/googlegsa/database/releases/download/v4.1.3.
5. Create a directory on the host where the connector will reside. For example, create a directory called databases_connector_413.
6. Copy the Connector for Databases 4.1.3 JAR executable to the directory.
7. Create an ASCII or UTF-8 file named `adaptor-config.properties` in the directory that contains the connector binary.

The following example shows the configuration variables you need to add to the `adaptor-config.properties` file (replacing boldface items with your actual configuration values):

```
gsa.hostname=yourgsa.example.com or IP address
db.driverClass=oracle.jdbc.OracleDriver
db.url=jdbc:oracle:thin:@45.62.11.99:1521:MY_ORACLE
db.user=sys_as_adaptor
db.password=pr@ducti@n
db.uniqueKey=customer_id:int
db.everyDocIdSql=select customer_id from oe.customers
db.singleDocContentSql=select * from oe.customers where customer_id = ?
db.modeOfOperation=rowToHtml
```

8. Create an ASCII or UTF-8 file named **`logging.properties`** in the same directory that contains the connector binary and add the following content:

```
# This file is UTF-8 encoded.  ☺
.level = WARNING
com.google.enterprise.adaptor.level = INFO
com.google.enterprise.adaptor.database.level = INFO

handlers = java.util.logging.FileHandler
# To enable console logging comment out above line and uncomment the line below
#handlers = java.util.logging.FileHandler,java.util.logging.ConsoleHandler

java.util.logging.ConsoleHandler.level = INFO
java.util.logging.ConsoleHandler.formatter =
com.google.enterprise.adaptor.CustomFormatter
# If your terminal can't handle colors and auto-detection fails, then use
# com.google.enterprise.adaptor.CustomFormatter$NoColor

java.util.logging.FileHandler.formatter=com.google.enterprise.adaptor.CustomFor
matter$NoColor
java.util.logging.FileHandler.pattern=logs/database-adaptor.%g.log
java.util.logging.FileHandler.limit=104857600
java.util.logging.FileHandler.count=5
```

9. Create a folder named `logs` in the same directory that contains `logging.properties`.

Step 3 Configure adaptor-config.properties variables

Optionally, you can edit or add additional configuration variables to the `adaptor-config.properties` file. For information, see Summary of configuration variables.

Step 4 Run the Connector for Databases

After you install the Connector for Databases, you can run it on a Linux host machine by using a command like the following example:

```
java -Djava.util.logging.config.file=logging.properties -cp
$JDBC_JAR:adaptor-database-4.1.3-withlib.jar
com.google.enterprise.adaptor.database.DatabaseAdaptor
```

On a Windows host machine, you can run it by using a command like the following example:

```
java -Djava.util.logging.config.file=logging.properties -cp
$JDBC_JAR$;adaptor-database-4.1.3-withlib.jar
com.google.enterprise.adaptor.database.DatabaseAdaptor
```

Where `$JDBC_JAR$` is the path of the jar for your database driver, for example, `sqljdbc4.jar`.

Verify that the connector has started and is running by navigating to the Connector Dashboard at:
`http://<CONNECTOR_HOST>:<nnnn>/dashboard` or
`https://<CONNECTOR_HOST>:<nnnn>/dashboard`

To run the connector as a service, use the Windows service management tool or run the `prunsrv` command, as described in "Run a connector as a service on Windows" in the Administration Guide.

Enable authentication by an Active Directory server

Optionally, you can configure the Connector for Databases to let an Active Directory server perform authentication rather than run as the local account on the SQL Server host (the logged-in user). Before setting up this configuration, ensure that the AD account has read access to the database on SQL Server.

To configure the Connector for Databases to let an Active Directory server perform authentication:

1. Download the SQL Server JDBC Driver 4.0 from Microsoft: http://msdn.microsoft.com/en-us/sqlserver/aa937724.aspx.
2. Go to Microsoft JDBC Driver 4.0 for `SQL Server\sqljdbc_4.0\enu` and copy `sqljdbc4.jar` and `sqljdbc_auth.dll` to `<Connector Installation Root> directory`
3. Update the following `adaptor-config` properties:

    - `db.driverClass=com.microsoft.sqlserver.jdbc.SQLServerDriver`
    - `db.url=jdbc:sqlserver://localhost;user=<ADUser>;password=<pwd>;` or `db.url=jdbc:sqlserver://localhost;integratedSecurity=true;`

      The keyword `integratedSecurity=true` indicates current Windows account credentials are used for authentication.

For more information on how to build the connection URL, see: https://msdn.microsoft.com/en-us/library/ms378428(v=sql.110).aspx

When running the connector you must include the `.jar` file in the adaptor `classpath`, and the `dll` in the adaptor `path`, for example:

```
 java -Djava.util.logging.config.file=logging.properties -classpath
"sqljdbc4.jar;adaptor-database-4.1.3-withlib.jar"
com.google.enterprise.adaptor.database.DatabaseAdaptor -path
"sqljdbc_auth.dll"
```

Summary of configuration variables

The following table lists the most important optional variables that pertain to the Connector for Databases, as well as their default values. You can change any configuration values by editing the `adaptor-config.properties` file. Take note that some variable names in the table are formatted for readability.

For additional information, see "Common configuration options" in the the Administration Guide.

Variable	Description	Default value
`server.dashboardPort`	Port on which to view web page showing information and diagnostics.	5679. Note: Windows installer sets 6079.
`server.port`	Unique value for the retriever port.	5678. Note: Windows installer sets 6078.
`db.driverClass`	Sets the JDBC driver for the connector. Required. For example: `db.driverClass=oracle.jdbc.OracleDriver`	
`db.url`	Sets the Database URL. Required. For example: `db.url=jdbc:oracle:thin:@45.62.11.99:1521:MY_ORACLE`	
`db.user`	The database user that the connector uses to query the database. Required.	
`db.password`	The password for the database user that the connector uses to query the database. Required.	

`db.uniqueKey`	One or more column heading names (separated by commas) that provide a unique identifier for a database query result. In some instances, you may need to specify the type for the column name, for example `employee_id:int` Valid types are `int, long, numeric, string, timestamp, date, time` Required.	
`db.everyDocIdSql`	Sets the lister query. Required.	
`db.singleDocContentSql`	Sets the retriever query. Required, except in urlAndMetadataLister mode.	
`db.metadataColumns`	The connector can automatically configure identity mappings, where the database column name is the same as the metadata key. This means that you can configure table rows without specifying the datatypes. If an entry in the list is a single token (rather than two separated by a colon), then that entry is an identity mapping, for example: `db.metadataColumns=id, name, acct:account`	Empty string (see note in db.includeAllColumnsAsMetadata section)
`db.includeAllColumnsAsMetadata`	If set to “true”, and if “db.metadataColumns” is left empty, then all columns retrieved by the lister or retriever queries, depending on the mode of operation, will be considered metadata.	false
`db.singleDocContentSqlParameters`	Specifies sequence of unique key	Sequence

	parts to use in `singleDocContentSql` query; allows for reusing same unique key part multiple times in `singleDocContentSql` query. For example, `customer_id,customer_id, modified_date`	of column names in uniqueKey
`db.updateSql`	Query to retrieve recently changed documents by their timestamp. For example, `select customer_id, order_placed_time as GSA_TIMESTAMP from customer where order_placed_time >= ?`	Empty string, which means no queries for updates are made
`db.updateTimestampTimezone`	Specifies the DB server's time zone when there is a difference between the database server and the connector's time zones. For example: `db.updateTimestampTimezone= America/Los_Angeles`	Empty string, which means it uses the connector machine's time zone
`db.aclSql`	Query to retrieve users and groups that are permitted or denied access to row. For example, `select permitted_users as GSA_PERMIT_USERS, denied_users as GSA_DENY_USERS from my_acl_table where customer_id = ?`	Empty string, which means ACLs are not queried and docs are public
`db.aclSqlParameters`	Specifies sequence of key parts to use in a `db.aclSql` query; allows for reusing same key part multiple times in a `db.aclSql` query. For example, `customer_id,customer_id,modified_date`	Sequence of column names in uniqueKey

db.aclPrincipalDelimiter	Delimiter used to separate principals in returned ACL column value.	,
db.modeOfOperation	Sets the database mode of operation for the connector. Required.	
db.modeOfOperation.[mode].displayUrlCol=URL	If docId.isUrl=false, you can provide displayUrlCol. The value is name of column which contains the URL. The URL is what is followed when user clicks a search result link. For example: db.modeOfOperation.rowToHtml.displayUrlCol=URL	
db.modeOfOperation.contentColumn.columnName	For Content mode, specify the name of the column that contains the content of the document.	
db.modeOfOperation.contentColumn.contentTypeCol	Specifies the name of the column that contains the type of the document. If empty, the GSA tries to infer the content type. You can specify either db.modeOfOperation.contentColumn.contentTypeCol or db.modeOfOperation.contentColumn.contentTypeOverride, but not both.	
db.modeOfOperation.contentColumn.contentTypeOverride	Overrides the content type of all the documents with the specified value. For example, text/plain, application/pdf	
db.modeOfOperation.filepathColumn.columnName	For File Path mode, specify the name of the column that contains the file path of the document.	
db.modeOfOperation.filepathColumn.contentTypeCol	Specifies the name of the column that contains the type of the document. If empty, the GSA tries to infer the content type. You can specify either	

	db.modeOfOperation.filepathColumn.contentTypeCol or db.modeOfOperation.filepathColumn.contentTypeOverride, but not both.	
db.modeOfOperation.filepathColumn.contentTypeOverride	Overrides the content type of all the documents with the specified value.	
db.modeOfOperation.rowToHtml.stylesheet	Specifies the path to the custom XSLT file used to render the document.	
db.modeOfOperation.urlColumn.columnName	For URL mode, specify the name of the column that contains the URL of the document. Must contain a valid value.	
db.modeOfOperation.urlColumn.contentTypeCol	Specifies the name of the column that contains the type of the document. If empty, the GSA tries to infer the content type. You can specify either db.modeOfOperation.urlColumn.contentTypeCol or db.modeOfOperation.urlColumn.contentTypeOverride, but not both.	
db.modeOfOperation.urlColumn.contentTypeOverride	Overrides the content type of all the documents with the specified value.	
server.docIdPath	Arbitrary part of document URLs. For example, /customer_db/orders/	/doc/
gsa.samlEntityId	URL of point of contact on the GSA where you can send SAML messages.	

Access-Controlled serving in secure mode

You can configure the database connector to serve access-controlled content to your users by setting up secure mode and using the GSA as a SAML IdP. So rather than use SQL to access database records, users can click the URLs that link to rows and view results in a browser. The connector only serves results that users are allowed to view.

This configuration requires a GSA running software release 7.4 or higher, which enables the GSA to act as a SAML Identity Provider (IdP). You must configure the V4 connector for secure serve by setting `server.secure=true`, and set up your connector to send SAML messages to the GSA.

Serving scenarios

The following scenarios illustrate how serving access-controlled content works.

User is already authenticated

In the first scenario, the user is already authenticated by the GSA.

1. In a browser, the user clicks a URL link in search results to get a database record from the connector.
2. The connector sends a message to the GSA to find out if authentication was previously completed for this user.
3. Because the user is already authenticated, the GSA sends a verified ID to the connector.
4. The connector serves the secure content to the verified user.

User is not already authenticated

In the second scenario, the user is not yet authenticated by the GSA.

1. In an email, the user clicks a link that was sent to her to get a database record from the connector.
2. The connector sends a message to the GSA to find out if authentication was previously completed for this user.
3. Because this user has not yet been authenticated, the GSA verifies the user through any configured authentication mechanism, then sends a verified ID to the connector.
4. The connector serves the secure content to the verified user.

Configure secure serve

To configure secure serve of access-controlled content, perform the following steps:

1. Configure an authentication mechanism on your GSA
2. Configure secure serve for your connector

Configure an authentication mechanism on your GSA

Use any of the following methods:

- Cookie cracking--Implement cookie cracking, as described in "Using Cookie Cracking" in Managing Search for Controlled-Access Content.
- Cookie authentication
- HTTP authentication
- Client certificate authentication
- Kerberos authentication
- SAML authentication
- LDAP authentication

To configure an authentication mechanism for your GSA, use the appropriate tab on the **Secure Search > Universal Login Auth Mechanisms** page in the Admin Console.

Configure secure serve for your connector

A prerequisite for secure serve is setting the database ACL SQL statement. If you don't set the database ACL SQL statement, then all the documents are public and are served without security.

Configure secure serve for your connector by performing the following steps:

1. Configure certificates and turn on security by setting the option `server.secure=true`. For detailed information about this step, see "Enable Connector Security" in the Administration Guide.
2. Add the following configuration option to `adaptor-config.properties` file: `gsa.samlEntityId`=[URL of point of contact on the GSA where you can send SAML messages.
   For example, `gsa.samlEntityId=http://google.com/enterprise/gsa/T2-QP2XQL6PGLWJT`
   For information about the SAML issuer entity ID, see the Admin Console help page for **Search > Secure Search > Access Control**.

Upgrade from the GSA built-in database crawler

Take note that the last release of the GSA that supported the built-in crawler was 7.2. The built-in crawler was deprecated in GSA 7.4 and removed from the GSA in release 7.6.

The configuration of the Connector for Databases 4.1.3 is similar to the GSA's built-in crawler's configuration. The connector requires the database connectivity information similar to the built-in crawler. However, the "follow pattern" will need to be adjusted. See Step 1 Configure the search appliance for more information.

The following table shows the mapping from the GSA database crawl settings to the Connector for Databases configuration variables. Take note that the Connector for Databases does not currently support a mode that sends secured URLs on a different host where the Database provides the metadata.

GSA Database Crawler	**Connector for Databases**
Database Type	`db.driverClass`
Hostname	`db.url`
Port	
Database Name	
Username	`db.user`
Password	`db.password`
Lock documents	not supported
Crawl Query	`db.everyDocIdSql`
Serve Query	`db.singleDocContentSql`
Unique Key Fields	`db.uniqueKey`
Default Stylesheet	`db.modeOfOperation` `db.modeOfOperation.rowToHtml.stylesheet`
Custom Stylesheet	`db.modeOfOperation` `db.modeOfOperation.rowToHtml.stylesheet`
Serve URL Field	`db.modeOfOperation.[mode].` `displayUrlCol=URL`

Document URL Field	`db.modeOfOperation` `db.modeOfOperation.urlColumn.columnName`
Document ID Field	not supported
Base URL	not supported
Incremental Crawl Query	`db.updateSql` `db.updateTimestampTimezone`
Action Field	GSA_ACTION column alias
BLOB Content Field	`db.modeOfOperation` `db.modeOfOperation.contentColumn.columnName`

BLOB MIME Type Field	`db.modeOfOperation`
	`db.modeOfOperation.contentColumn.columnName`
	`db.modeOfOperation.contentColumn.contentTypeCol`
	`db.modeOfOperation.contentColumn.contentTypeOverride`

Uninstall the Google Search Appliance Connector for Databases

If you have created a Windows Service for the Connector for Databases, before you uninstall the connector, you must remove the Windows Service.

To stop and remove the Windows Service, execute the following command:

```
prunsrv //DS//adaptor-database
```

To uninstall the Connector for Databases on Windows:

1. Navigate to the databases connector installation folder, **_GSA Database Adaptor Installation**.
2. Click `Uninstall GSA Database Adaptor.exe`.
   The **Uninstall GSA Database Adaptor** page appears.
3. Click **Uninstall**.
   Files are uninstalled.
4. Click **Done**.

Additional lister and retriever query examples

This section shows example lister and retriever queries for an employee database with these fields:

```
employee_id, first_name, last_name, email, dept
```

The following example shows a lister query where employee_id is enough to identify a unique row:

```
SELECT employee_id FROM employee
```

If instead three columns, `employee_id, first_name,` and `last_name`, that are combined to be the `uniqueKey`, the following example shows the lister query:

```
SELECT employee_id, first_name, last_name
FROM employee
```

The following example shows the lister query for selecting rows to add or delete:

```
SELECT employee_id, action as GSA_ACTION from employee
```

The value for the `GSA_ACTION` column alias must be `add` or `delete`.

The following example shows the retriever query with employee_id being enough to identify a unique row:

```
SELECT employee_id, first_name, last_name, email, dept
FROM employee
WHERE employee_id = ?
```

The `uniqueKey` field for this case must be `employee_id`. The ? signifies that this value is provided at serve time, from the search result that the user clicks.

For a table with multiple column unique keys, if the combination of `employee_id, dept` is unique, you can use multiple bind variables. The following example shows the retriever query:

```
SELECT employee_id, first_name, last_name, email, dept
FROM employee
WHERE employee_id = ? AND dept = ?
```

The `uniqueKey` field for this case may be `employee_id:int,dept:string.`

Here `employee_id` is of certain database column types that mapped to `JDBC INT` type, and `dept` is of certain database column types that mapped to `JDBC STRING` type.

Notes:

- You can validate SQL queries by using TOAD for SqlServer or SQLPlus/iSQLplus for Oracle.
- SQL keywords are in uppercase by convention. Uppercase is not required.
- The '?' is substituted with a real column value to identify a particular record to be displayed when a user clicks on a database search result.
- The URL accessed by the user is partly generated from the unique keys; the database query is made based on the retriever query and the substituted unique key values.