Google Search Appliance Connectors

Deploying the Connector for Databases

Google Search Appliance Connector for Databases software version 4.0.4
Google Search Appliance software version 7.2

January 2015

Table of Contents

About this guide

This guide is intended for anyone who needs to deploy the Google Search Appliance Connector 4.0.4 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 4.0.4 for general information about the connectors, including:

- What's new in Connectors 4.0?
- General information about the connectors, including the configuration properties file, supported ACL features, and other topics
- Connector security
- Connector logs
- Connector Dashboard
- Connector troubleshooting

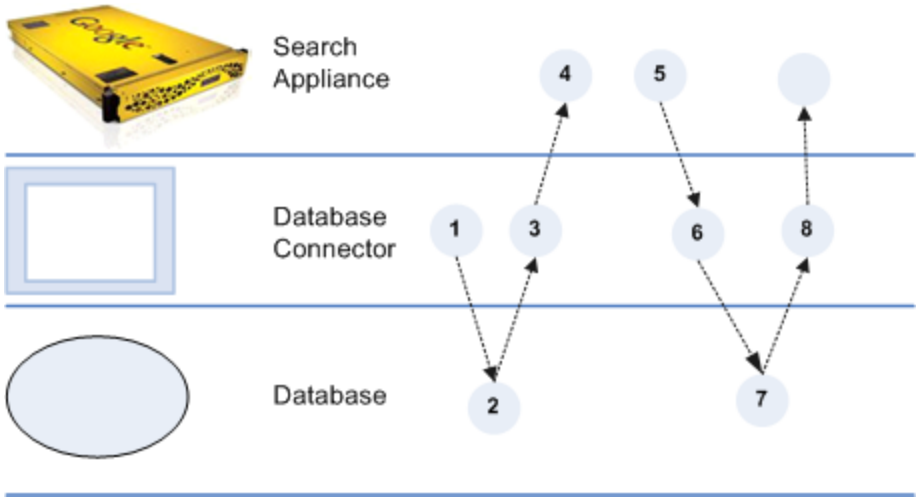
For information about using the Admin Console, see the Google Search Appliance Help Center.

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

Overview of the GSA Connector for Databases

The Connector for Databases enables the Google Search Appliance to crawl and index content from Oracle or Microsoft SQL Server 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 it 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.
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 modes of operation for the connector.
9. The search appliance continues to crawl the repository.

Automatic updates every 15 minutes

The connector starts monitoring for changes immediately by sending a SQL `SELECT` query to the database at intervals 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.

After it receives the SQL `SELECT` query, the database sends updates of DocIds to the connector, which constructs URLs from them and pushes them to the search appliance with a status of crawl immediately.

Supported operating systems for the connector

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

- Windows Server 2012
- Windows Server 2008 R2
- Windows Server 2003
- Linux

Supported databases

The Connector for Databases supports the following databases:

- Oracle 11g
- Microsoft SQL Server 2008

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`

Configuration variables

To use the Connector for Databases, you must set configuration variables. The Windows installation wizard provides the **GSA Hostname and other required configuration values** page where you can specify each value. For Linux or Windows command-line installations, you must set the configuration values in the `adaptor-config.properties` file. You can also change any configuration values by editing the `adaptor-config.properties` file.

The configuration variables set values for:

- Java JDBC driver
- Database URL
- Database user and password
- Database unique key
- Lister query
- Retriever query
- Database modes of operation
- Database metadata columns
- Update SQL statement and timezone used for timestamps
- Database ACL SQL statement
- Database ACL Delimiter

Java JDBC driver

A Java Database Connectivity (JDBC) driver enables the connector to interact with the database. An example of a full classname of a JDBC driver is `oracle.jdbc.OracleDriver`. The JDBC driver is required.

For Windows installation, indicate the JDBC driver by entering values in the **Specify JDBC jar file (Database driver)** field and the **Full classname of Java JDBC Driver** field on the **GSA Hostname and other required configuration values** page.

For Linux or command-line installation, set the Java JDBC driver in the `adaptor-config.properties` file by using the `db.driverClass` variable. For example:

```
db.driverClass=oracle.jdbc.OracleDriver
```

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
```

For Windows installation, enter the URL in the **Database URL (to talk to the database)** field on the **GSA Hostname and other required configuration values** page.

For Linux or command-line installation, set the Database URL in the `adaptor-config.properties` file by using the `db.url` variable. For example:

```
db.url=jdbc:oracle:thin:@45.62.11.99:1521:MY_ORACLE
```

Database username and password

The connector uses the **Database Username** and **Database Password** to query the database. The Database Username and password are required.

An example database username is `sys_as_adaptor`, with the password `pr@ducti@n`.

For Windows installation, enter the username in the **Database Username** field and the password in the Database Password field on the **GSA Hostname and other required configuration values** page.

For Linux or command-line installation, set the username in the `adaptor-config.properties` file by using the db.user variable. For example:

```
db.user=sys_as_adaptor
```

To set the password of `db.user`, use the `db.password` variable. For example:

```
db.password=pr@ducti@n
```

Take note that you can encode the value for **Database Password** and copy it to the **GSA Hostname and other required configuration values** page or the `adaptor-config.properties` file. Do this by using the **Storing Sensitive Values** section on the Connector Dashboard. For more information, see “Encode Sensitive Values” in the Administration Guide.

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 unique key can be a combination of column names which produce a unique permutation from the corresponding values. A database unique key might include

columns such as Last_Name, First_Name, SSN, Birth_Date. You must map a unique key to its type, for example:

```
customer_id:int
```

The valid types of unique key are `int`, `string`, `timestamp`, `date`, `time`, and `long`. This value must be a java type, such as java.lang.String, instead of a specific database type, such as VARCHAR in MS SQL Server.

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

For Windows installation, enter the unique key and its type in the **Database unique key** field on the **GSA Hostname and other required configuration values** page.

For Linux or command-line installation, set the unique key and map it to its type in the `adaptor-config.properties` file by using the `db.uniqueKey` variable. For example:

```
db.uniqueKey=customer_id:int
```

Lister query

A lister query is a SQL statement that 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. The lister query is required.

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

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

For Windows installation, enter the lister query in the **SQL statement that provides all IDs of all documents to be indexed** field on the **GSA Hostname and other required configuration values** page.

For Linux or command-line installation, set the lister query in the `adaptor-config.properties` file by using the `db.everyDocSql` variable. For example:

```
db.everyDocSql=select customer_id from oe.customers
```

Retriever query

A retriever query is a SQL statement that provides one document's content. 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. The retriever query 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. The unique key fields must provide the column names for the field to substitute with the ?.

For Windows installation, enter the retriever query in the **SQL statement that provides one document's content** field on the **GSA Hostname and other required configuration values** page.

For Linux or command-line installation, set the retriever query in the `adaptor-config.properties` file by using the `db.singleDocContentSql` variable.

For example:

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

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 the lister query.

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

Given 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 retriever query.

```
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 lister query for this example is the same as shown in this section. 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 must be `employee_id:int,dept:string`. Suppose `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 lister query and the substituted unique key values.

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
- BLOB 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 GSA 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 on the **GSA Hostname and other required configuration values** page.

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 on the **GSA Hostname and other required configuration values** page.

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 on the **GSA Hostname and other required configuration values** page.

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
```

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 on the **GSA Hostname and other required configuration values** page.

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
```

BLOB mode

In BLOB (binary large object) mode, the GSA extracts a BLOB from a particular column in the row. The GSA also indexes other columns in the row, which might include columns that contain metadata. 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 **blobColumn** from the pull-down menu in the **Database Mode of operation** field on the **GSA Hostname and other required configuration values** page.

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

```
db.modeOfOperation=blobColumn
```

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.

Database metadata columns

Database metadata columns specify columns to treat as metadata and provides a string to use as a key. For example:

```
name:name, dbcol:gsa_metaname
```

The default is an empty string, which means no column will be used as metadata. Database metadata columns are optional.

For Windows installation, enter the column names in the **Database metadata columns** field on the **GSA Hostname and other required configuration values** page.

For Linux or command-line installation, set the metadata in the `adaptor-config.properties` file by using the `db.metadataColumns`, for example:

```
db.metadataColumns=name:name, dbcol:gsa_metaname
```

and

```
db.metadataColumns=name:name, age:Age
```

Update SQL statement and timezone used for timestamps

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 where
order_placed_time > ?
```

For Windows installation, enter the statement in the **SQL statement to retrieve recently change documents by their timestamp** field on the **GSA Hostname and other required configuration values** page.

For Linux or command-line installation, enter the statement in the `adaptor-config.properties` file by using the `db.updateSql` variable.

For example:

```
db.updateSql=select customer_id, order_placed_time as GSA_TIMESTAMP
where order_placed_time > ?
```

If there is a difference between the database server 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. This config is optional. 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.

Database ACL SQL statement

The Database ACL SQL statement 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 = ?
```

For Windows installation, enter the statement in the **SQL statement to retrieve users/groups that are permitted or denied access to a document** field on the **GSA Hostname and other required configuration values** page.

For Linux or command-line installation, enter the statement in the `adaptor-config.properties` file by using the `db.aclSql` variable. For example:

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

The `db.aclSqlParameters` variable specifies the sequence of key parts to use in a `db.aclSql` query, for example:

```
customer_id,customer_id,modified_date
```

The value defaults to the sequence of column names in the unique key.

Database ACL delimiter

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

For Windows installation, enter the delimiter in the **Delimiter on retrieved results of above access control SQL statement** field on the **GSA Hostname and other required configuration values** page.

For Linux or command-line installation, enter the delimiter in the `adaptor-config.properties` file by using the `db.aclPrincipalDelimiter` variable. For example: `db.aclPrincipalDelimiter=;`

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 7.2.0.G.90 or higher
  To download GSA software, visit the Google for Work Support Portal (password required).
- Java JRE 1.7 update 6 or higher installed on computer that runs the connector
- JDBC driver jar for your database. Suggested is Oracle JDBC version 6 for Oracle Database 11g, and Microsoft SQL Server JDBC version 4 for Microsoft SQL Server.
- Connector for Databases 4.0.4 JAR executable
  For information about finding the JAR executable, see Step 2 Install the Connector for Databases
- SQL account (db.user+db.password) must have read access to necessary database tables
- 512 MB of RAM on the connector host
- Sufficient hard disk for log files on the connector host

Deploy the Connector for Databases

Because the Connector for Databases is installed on a separate host, 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. Add the URL provided by the connector to the search appliance's crawl configuration follow patterns.
2. 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.
3. Set up connector security.

Add the URL

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:5678/doc/
   where connector.example.com is the hostname of the machine that hosts the connector.
   By default the connector runs on port 5678.
3. Click **Save**.

Add the IP address

To add the IP address of the computer that hosts the connector to the list of trusted IP addresses:

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.

You can install the Connector for Databases on any host running one of the supported operating systems.

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

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 4.0.4 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-4.0.4.exe`.
4. Start installing the file by double clicking `database-install-4.0.4`.
5. On the **Introduction** page, click **Next**.
6. On the **GSA Hostname and other required configuration values** page, enter values for the following options:
   - **GSA Hostname or IP address** of the GSA that will use the connector.
     For example, enter `gsa.hostname=`**`yourgsa.example.com`**
   - **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 5678.

    - **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 5679.
    - **Specify JDBC jar file (Database driver)**
    - **Full classname of Java JDBC Driver**
    - **Database URL**
    - **Database username**
    - **Database password**
    - **Database Unique Key**
    - **SQL statement that provides all IDs of all documents to be indexed**
    - **SQL statement that provides one document's content**
    - **Database Mode of operation**
    - **Whether or not to run the connector after the installer finishes**
7. Click **Next**.
8. On the **Choose Install Folder** page, accept the default folder or navigate to the location where you want to install the connector files.
9. Click **Next**.
10. On the **Choose Shortcut Folder**, accept the default folder or select the locations where you want to create product icons.
11. 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**.
12. On the **Pre-Installation Summary page**, review the information and click **Install**. The connector Installation process runs.
13. 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 4.0.4 software downloads page at http://googlegsa.github.io/adaptor/index.html.
4. Download the Connector for Databases JAR executable (`adaptor-database-4.0.4-withlib.jar`) at https://github.com/googlegsa/database/releases/download/v4.0.4.
5. Create a directory on the host where the connector will reside. For example, create a directory called databases_connector_40.
6. Copy the Connector for Databases 4.0.4 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.everyDocSql=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:

```
.level=INFO
handlers=java.util.logging.FileHandler,java.util.logging.ConsoleHandler
java.util.logging.FileHandler.formatter=com.google.enterprise.adaptor.CustomFor
matter
java.util.logging.FileHandler.pattern=logs/adaptor.%g.log
java.util.logging.FileHandler.limit=10485760
java.util.logging.FileHandler.count=20
java.util.logging.ConsoleHandler.formatter=com.google.enterprise.adaptor.Custom
Formatter
```

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 host machine by using a command like the following example:

```
java -Djava.util.logging.config.file=logging.properties -cp
$JDBC_JAR$:adaptor-database-4.0.4-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.

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 Required configuration variables. See also "Common configuration options" in the the Administration Guide.

Take note that Oracle column names need to be all capital letters to match the names that Oracle provides.

Variable	Description	Default value
`server.dashboardPort`	Port on which to view web page showing information and diagnostics.	5679
`db.driverClass`	Sets the JDBC driver for the connector. Required.	
`db.url`	Sets the Database URL. Required.	
`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. Required.	
`db.everyDocSql`	Sets the lister query. Required.	
`db.singleDocContentSql`	Sets the retriever query. Required.	
`db.metadataColumns`	Identifies columns to treat as metadata and provides string to use as key. For example, name:name	Empty string
`db.singleDocContentSqlParameters`	Specifies sequence of unique key 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`	Sequence 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 where order_placed_time > ?`	Empty string, which means no updates for queries 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.	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,`	Empty string, which means ACLs are not queried and

	`denied_users as GSA_DENY_USERS from my_acl_table where customer_id = ?`	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.blobColumn.columnName`	For BLOB mode, specify the name of the column that contains the content of the document.	
`db.modeOfOperation.blobColumn.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.blobColumn.contentTypeCol` or `db.modeOfOperation.blobColumn.contentTypeOverride`, but not both.	
`db.modeOfOperation.blobColumn.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.	
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/

Upgrade from the GSA built-in database crawler

The configuration of the Connector for Databases 4.0.4 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.

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	not supported
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	not needed
BLOB Content Field	`db.modeOfOperation` `db.modeOfOperation.blobColumn.columnName`

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

Uninstall the Google Search Appliance Connector for Databases

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**.