



# **Splunk® DB Connect**

## **Deploy and Use Splunk DB Connect 3.11.0**

Generated: 11/07/2022 8:51 pm

# Table of Contents

<b>Introduction.....</b>	<b>1</b>
About Splunk DB Connect.....	1
What DB Connect can do.....	1
Who DB Connect is for.....	1
Share data in Splunk DB Connect.....	1
How Splunk DB Connect collects data.....	2
Data Splunk DB Connect collects.....	2
Data DB Connect does not collect.....	9
How Splunk DB Connect works.....	9
How to get help and learn more about Splunk software.....	13
<b>Before you deploy.....</b>	<b>14</b>
System requirements.....	14
Architecture and performance considerations.....	16
Architecture and performance considerations.....	17
Performance considerations in distributed environments.....	17
General performance tuning considerations.....	18
<b>Install Splunk DB Connect.....</b>	<b>20</b>
Installation and setup overview.....	20
Upgrade an existing DB Connect deployment.....	20
Migrate DB Connect deployment to DB Connect 3.....	21
Install database drivers.....	21
Install and configure Splunk DB Connect on a single instance Splunk platform deployment.....	33
Install and configure Splunk DB Connect on a Splunk Enterprise On-Premise distributed platform deployment.....	34
Install and configure Splunk DB Connect on Splunk Cloud Platform.....	36
Check DB Connect installation health.....	37
<b>Configure and manage Splunk DB Connect.....</b>	<b>40</b>
Configure Splunk DB Connect settings.....	40
Configure Splunk DB Connect security and access controls.....	42
Create and manage identities.....	44
Create and manage database connections.....	46
Create and manage database inputs.....	49
Create and manage bulk operations of the database inputs.....	55
Create and manage database outputs.....	57
Create and manage database lookups.....	61
Use SQL explorer to make live reports.....	65
Execute SQL statements and stored procedures with the dbxquery command.....	69
Monitor Splunk DB Connect health.....	73
<b>Reference.....</b>	<b>76</b>
Configuration file reference.....	76
app-migration.conf.spec.....	77
db_connection_types.conf.spec.....	77
db_connections.conf.spec.....	79
database_typespec.....	82

# Table of Contents

## Reference

db_inputs.conf.spec.....	82
db_input_templates.conf.spec.....	85
db_outputs.conf.spec.....	86
db_lookups.conf.spec.....	88
identities.conf.spec.....	90
inputs.conf.spec.....	90
dbx_settings.conf.spec.....	91
Saved searches for Splunk DB Connect.....	92
Data models for Splunk DB Connect.....	92

## Troubleshooting.....93

Troubleshooting Tool for DB Connect.....	93
Set Up Troubleshooting Tool for DB Connect.....	93
Common issues for Splunk DB Connect.....	94
SQL tips and tricks.....	104

## Terminology.....108

Splunk DB Connect terminology.....	108
------------------------------------	-----

# Introduction

## About Splunk DB Connect

Splunk DB Connect 3 lets you combine your structured data from databases with your unstructured machine data, and then use Splunk Enterprise to provide insights into all of that combined data.

When you use Splunk DB Connect, you are creating additional data inputs for Splunk Enterprise. That is, you're giving Splunk Enterprise more sources of data to consume. Splunk DB Connect is what connects your relational database data to Splunk Enterprise and makes that data consumable by Splunk Enterprise. In addition, Splunk DB Connect can do the reverse—write Splunk Enterprise data back to your relational database.

For more information about how DB Connect works, see [How Splunk DB Connect works](#).

## What DB Connect can do

Splunk DB Connect lets you **import tables, rows, and columns from a database directly into Splunk Enterprise**, which indexes the data. You can then analyze and visualize that relational data from within Splunk Enterprise along with your existing Splunk Enterprise data.

DB Connect also lets you **output data** from Splunk Enterprise back to your relational database. You map the Splunk Enterprise fields to the database tables you want to write to.

DB Connect also performs **database lookups**, which let you reference fields in an external database that match fields in your event data. Using these matches, you can add more meaningful information and searchable fields to enrich your event data.

## Who DB Connect is for

Splunk DB Connect is great for users who:

- Want to quickly get data from a database into Splunk Enterprise.
- Want to run on-the-fly lookups from data warehouses or state tables within Splunk Enterprise.
- Want to index structured data stored in databases in streams or batches using Splunk Enterprise.
- Want to write Splunk Enterprise data into databases in streams or batches.
- Want to preview data and validate settings such as locale and time zone, rising column and metadata choice, and so on before indexing begins, to prevent accidental duplication or other problems in the future.
- Want to scale, distribute, and monitor database read/write jobs to prevent overload and receive notice of failures.
- Want to know what databases are accessible to which Splunk Enterprise users, to prevent unauthorized access.

For installation instructions, see Install Splunk DB Connect: [Single server deployment](#) or [Distributed deployment](#).  
Download Splunk DB Connect on Splunkbase.

## Share data in Splunk DB Connect

Splunk DB Connect includes the opt-in ability to send anonymized usage data to Splunk to help improve the add-on in future releases.

## How Splunk DB Connect collects data

If you opt in, the add-on enables an internal library to track basic usage and crash information. The library uses browser cookies to track add-on user visitor uniqueness and sessions and sends events to Splunk using XMLHttpRequest (XHR) in JSON format.

## Data Splunk DB Connect collects

If you opt in, Splunk DB Connect sends the following information to Splunk:

- The number of [identities](#) you defined on the Splunk platform instance.
- The number of [connections](#) you defined on the instance.
- The number of [connections](#) associated with each identity on the instance.
- Each connection's database type, whether SSL is enabled, and the connection's associated number of database inputs, outputs, and lookups
- The number of databases [inputs](#) defined on the instance.
- Each input's [type](#) (batch mode or rising column).
- The number of database [outputs](#) you defined on the instance.
- The number of database [lookups](#) you defined on the instance.
- Whether you enabled [search head clustering](#).
- Each search head's search head cluster mode.
- The daily license usage for each source type that DB Connect defines. DB Connect uses to send daily license usage. You can disable the saved search without harming DB Connect operations.
- The Splunk Enterprise license size.
- The name and version number of every [JDBC driver](#) you installed.
- The server roles you defined (provided by the server-roles attribute of the /services/server/info endpoint).

Event	Source Type	Description	Data sent includes common fields, plus		
			Field	Type	Description
Session start	mint:ping	Each ping event indicates that a new session has started.	fsEncrypted	N/A	Not used, always "NA"
			rooted	N/A	Not used, always false
Session end	mint:gnip	Each gnip event indicates that a session has ended.	ses_duration	int	How long the session lasted
Page views	mint:view	Triggered once per page view in the app.	current	string	The URL of the current web page, without the hostname.
			currentView	string	Not used. Hardcoded to 'examples'.
			domProcessingTime	int	Time spent to process the domain.
			domLookupTime	int	

Event	Source Type	Description	Data sent includes common fields, plus		
					Time spent to look up the domain name.
			elapsedTime	int	Time spent to render the page.
			host	string	The hostname in the URL.
			loadTime	int	Time spent to load the page.
			previous	string	The referrer URL.
App performance and configuration	mint:log	Usage and performance logs for the DB Connect that track dashboard memory usage, dashboard loading times, the number of accounts, inputs, and regions configured in the app, and non-sensitive input configuration parameters.	serverTime	int	Time spent to get a response from the server.
			level	int	Log level. For example, 60 means 'error'.
			log_name	any	Log content. See examples below.
API calls	mint:network	XMLHttpRequest calls, usually HTTP API calls from client side (browser) to the Splunk server.	failed	boolean	Indicates if the request failed or not.
			latency	int	Time spent before response received.
			protocol	string	Network protocol: either http or https.
			requestLength	string	N/A. Not used.
			responseLength	int	The size of the response.
			statusCode	string	HTTP response code.
			url	string	The request URL, without the hostname.

### Common fields

The data that the DB Connect sends to Splunk, if enabled, includes the following common fields. This set of fields includes several fields that are disabled or deliberately not used for the DB Connect for purposes of anonymization.

Field	Type	Description	Example value
apiKey	string	MINT API key for the DB Connect	4t2fk73n
appRunningState	Field is unused by the SDK. Shows a value of "NA" in all events.		
appVersionCode	Field is unused by the SDK. Shows a value of "NA" in all events.		

Field	Type	Description	Example value
appVersionName	string	The version name of the app sending data.	4.1.0
browser	string	The browser name.	Chrome
browserVersion	string	The browser version.	47.0.2526.111
carrier	Field is unused by the SDK. Shows a value of "NA" in all events.		
connection	Field is unused by the SDK. Shows a value of "NA" in all events.		
device	string	The device making the request.	MacIntel
extraData	JSON object	This field stores custom information for the app. This app uses <code>extraData.splunk_version</code> to store the version number of the Splunk platform instance.	6.3.1511
locale	string	The user locale set in the browser.	en-US
osVersion	string	The version code of the underlying operating system.	OS X 10.11.2
packageName	string	The package name of the DB Connect.	splunk_app_db_connect
platform	Not used for the DB Connect. Shows a value of "web" in all events.		
remoteIP	Not used for the DB Connect. Shows a value of "3.0.0.0" in all events.		
sdkVersion	string	The version of the SDK.	4.3
screenOrientation	Field is unused by the SDK. Shows a value of "NA" in all events.		
session_id	string	A unique string to identify a session.	a5026251
state	string	Indicator of whether the browser is online or not. Can be either CONNECTED or DISCONNECTED.	CONNECTED
uuid	UUID	A random identifier to track the user's uniqueness	837227ea-4569-4675-9a17-ccb39ca69505

### Example app performance and configuration events

DB Connect sends performance and configuration information using the `log_name` field in the `mint:log` source type. This `log_name` field contains two sub-fields, `name`, which indicates which type of logs are being transmitted, and `data`, the content of the tracking log.

There are three possible options for `name`:

- `track_performance`. When a user accesses a dashboard in the app, DB Connect sends performance logs for dashboard memory usage and loading times.
- `track_configuration`. When a Splunk admin visits the Configure page, DB Connect sends a log of the number of accounts, inputs, and regions configured in the app, and non-sensitive input configuration parameters.
- `track_usage`. When a Splunk admin visits the Configure page, DB Connect sends a log of the data volume that each input is responsible for.

The following examples demonstrate what data the DB Connect sends for each type of event.

<code>log_name.name</code>	Example JSON object
<code>track_performance</code>	<pre>{   "memory": {</pre>

log_name.name	Example JSON object
	<pre> "totalJSHeapSize":72200000, "usedJSHeapSize":39600000, "jsHeapSizeLimit":1620000000 }, "timing":{   "navigationStart":1453273923766,   "unloadEventStart":1453273923929,   "unloadEventEnd":1453273923930,   "redirectStart":0,   "redirectEnd":0,   "fetchStart":1453273923766,   "domainLookupStart":1453273923766,   "domainLookupEnd":1453273923766,   "connectStart":1453273923766,   "connectEnd":1453273923766,   "secureConnectionStart":0,   "requestStart":1453273923773,   "responseStart":1453273923927,   "responseEnd":1453273923929,   "domLoading":1453273923939,   "domInteractive":1453273923975,   "domContentLoadedEventStart":1453273923975,   "domContentLoadedEventEnd":1453273923975,   "domComplete":1453273926985,   "loadEventStart":1453273926985,   "loadEventEnd":1453273926987 } } </pre>
track_configuration	<pre> {   "identities": {     "count": 7,     "data": [       {         "connections_count": 1       },       {         "connections_count": 1       },       {         "connections_count": 0       },       {         "connections_count": 1       },       {         "connections_count": 2       },       {         "connections_count": 6       },       {         "connections_count": 1       }     ]   },   "connections": {     "count": 12,     "data": [ </pre>



log_name.name	Example JSON object
	<pre> {     "connection_type": "db2",     "jdbcUseSSL": false,     "inputs_count": 0,     "outputs_count": 0,     "lookups_count": 0 }, {     "connection_type": "informix",     "jdbcUseSSL": false,     "inputs_count": 0,     "outputs_count": 0,     "lookups_count": 0 }, {     "connection_type": "memsql",     "jdbcUseSSL": false,     "inputs_count": 0,     "outputs_count": 0,     "lookups_count": 0 }, {     "connection_type": "generic_mssql",     "jdbcUseSSL": false,     "inputs_count": 0,     "outputs_count": 0,     "lookups_count": 0 }, {     "connection_type": "mssql_jtds_win_auth",     "jdbcUseSSL": false,     "inputs_count": 0,     "outputs_count": 0,     "lookups_count": 0 }, {     "connection_type": "mysql",     "jdbcUseSSL": false,     "inputs_count": 4,     "outputs_count": 2,     "lookups_count": 3 }, {     "connection_type": "mysql",     "jdbcUseSSL": false,     "inputs_count": 0,     "outputs_count": 0,     "lookups_count": 0 }, {     "connection_type": "mysql",     "jdbcUseSSL": false,     "inputs_count": 0,     "outputs_count": 0,     "lookups_count": 0 }, {     "connection_type": "mysql",     "jdbcUseSSL": false,     "inputs_count": 0,     "outputs_count": 0,     "lookups_count": 0 } </pre>

log_name.name	Example JSON object
	<pre>       "inputs_count": 0,       "outputs_count": 0,       "lookups_count": 0     },     {       "connection_type": "oracle",       "jdbcUseSSL": false,       "inputs_count": 0,       "outputs_count": 1,       "lookups_count": 1     },     {       "connection_type": "oracle",       "jdbcUseSSL": true,       "inputs_count": 0,       "outputs_count": 0,       "lookups_count": 0     },     {       "connection_type": "postgres",       "jdbcUseSSL": false,       "inputs_count": 0,       "outputs_count": 0,       "lookups_count": 0     }   ] }, "drivers": {   "count": 0,   "data": [] }, "inputs": {   "count": 4,   "data": [     {       "mode": "batch"     },     {       "mode": "rising"     },     {       "mode": "batch"     },     {       "mode": "rising"     }   ] }, "outputs": {   "count": 3,   "data": [     {       "transactional": true,       "is_saved_search": false     },     {       "transactional": true,       "is_saved_search": false     }   ] } </pre>

log_name.name	Example JSON object
	<pre> {     "transactional": true,     "is_saved_search": false } ], "lookups": {     "count": 4,     "data": [         {             "input_fields_count": 0,             "output_fields_count": 0         },         {             "input_fields_count": 0,             "output_fields_count": 0         },         {             "input_fields_count": 0,             "output_fields_count": 0         },         {             "input_fields_count": 0,             "output_fields_count": 0         }     ] }, "serverstatus": {     "count": 1,     "data": [         {             "isSHCluster": false,             "javaVersion": "1.8.0_92"         }     ] } } </pre>
track_usage	<pre> {     "usage": [         {             "time": "2017-06-21",             "volumes": {                 "sakila_city_rising_input": "0"             }         },         {             "time": "2017-06-22",             "volumes": {                 "sakila_city_rising_input": "0"             }         },         {             "time": "2017-06-23",             "volumes": {                 "sakila_city_rising_input": "0.001021385"             }         }     ] } </pre>

log_name.name	Example JSON object
	<pre> {   "time": "2017-06-24",   "volumes": {     "sakila_city_rising_input": "0"   } }, {   "time": "2017-06-25",   "volumes": {     "sakila_city_rising_input": "0"   } }, {   "time": "2017-06-26",   "volumes": {     "sakila_city_rising_input": "0"   } }, {   "time": "2017-06-27",   "volumes": {     "sakila_city_rising_input": "0"   } } ] </pre>

## Data DB Connect does not collect

DB Connect does not collect or send the following kinds of data:

- Sensitive data such as usernames or passwords.
- Identifying information such as addresses, phone numbers, IP addresses, or hostnames.
- Indexed data that you ingest into your Splunk platform instance.

DB Connect does not collect any data that is not explicitly described in the [Data Splunk DB Connect collects](#) section above.

## How to opt in or out

Splunk DB Connect presents an opt-in request the first time that you launch the app. You can also opt in at any time. In **Settings**, check the box under the **Usage Collection** tab.

To opt out, leave the box unchecked when the app presents the **Help us improve Splunk products and services** box. If you previously opted in but want to change your response, uncheck the box at the bottom of the **Configure** page, and then click **Stop sending data**. If you opt out after having previously opted in, the app immediately stops sending data to Splunk.

For more information about how Splunk collects and uses data, please refer to the Splunk Privacy Policy.

## How Splunk DB Connect works

Splunk DB Connect is an add-on that bridges Splunk Enterprise with relational databases via Java Database Connectivity (JDBC). It enables Splunk Enterprise to connect to and exchange data with databases such as MySQL, Microsoft SQL

Server, Informix, DB2, and many others, enriching your Splunk Enterprise data by combining it with data that was previously only available to you directly from those databases.

Use the add-on to configure database queries and lookups in minutes via the Splunk Enterprise interface. By installing Splunk DB Connect, you can broaden the range of data inputs available to Splunk Enterprise, because they can now include your relational databases. Splunk DB Connect can also do the reverse and send Splunk Enterprise data back for storage in your relational database tables. Splunk DB Connect enriches and combines unstructured data with structured data, which allows users to cross-reference, augment, and correlate between events in machine logs and external databases.

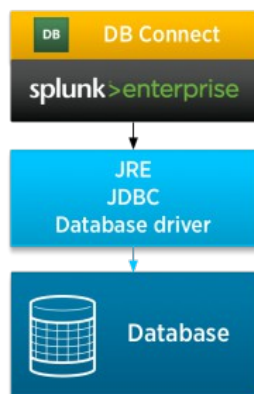
This topic provides an overview of how Splunk DB Connect works.

## Setup

Before you can get started, you need to set up Splunk DB Connect. Download the Splunk DB Connect add-on, and then follow the instructions in either the [single-server](#) or [distributed](#) deployment installation topics. You can use Splunk DB Connect on a heavy forwarder to support continual data gathering or output. For more interactive use, including lookups, you should install the add-on on a search head.

All DB Connect instances require Java Runtime Environment (JRE) version 8 or higher in order to enable JDBC. DB Connect uses a remote procedure call (RPC) server to manage communications with the Java subsystem. You must also install a Java Database Connectivity (JDBC) driver so that Splunk Enterprise can communicate with your databases. Review [Install database drivers](#) for more information and a listing of tested drivers.

A checklist of steps required for setting up Splunk DB Connect is available at [Installation overview](#).



## Identities

After installing prerequisites and Splunk DB Connect, you must create an **identity**. An identity is what defines the database user through which Splunk Enterprise will connect to your database(s). It consists of the username and password that you use to access your database(s). A single identity can be used by many connections, so that service accounts can be easily shared across multiple systems. This makes regular password changes easier to support.

**Be aware that these are database credentials, and are not the same as your Splunk Enterprise credentials.** When you configure an identity, you can specify the Splunk Enterprise roles that have read, read-write, or no access to the identity.

- Read access means that Splunk Enterprise roles will be able to use the identity.
- Read-write access means that Splunk Enterprise roles will be able to use and modify the identity.

By default, the Splunk Enterprise **admin** and **db\_connect\_admin** roles have read-write access to a new identity (**sc\_admin** role for the cloud customer), the **db\_connect\_user** role has read access, and all other roles have no access.

For more information about setting up and using identities, see [Create and manage identities](#).

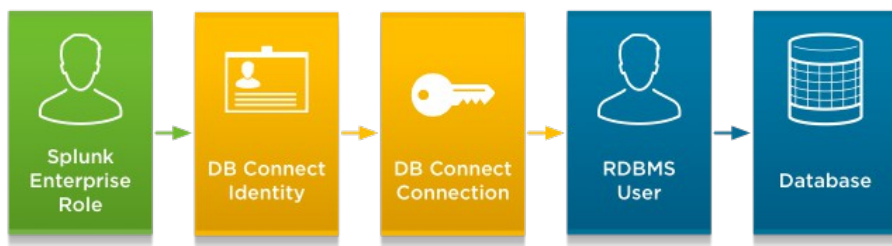
## Connections

Once you've created the necessary identities for your database environments, you'll need to create a **connection**. A connection is the information necessary to connect to a specific database. It consists of the address of your database (the host name), the database's type, and the name of the database.

When you configure a connection, you can specify which Splunk Enterprise roles have read, read-write, or no access to the connection. Read access means that Splunk Enterprise roles will be able to use the connection. Read-write access means that Splunk Enterprise roles will be able to use and modify the connection. By default, the Splunk Enterprise "admin" and "db\_connect\_admin" roles have read-write access to a new connection, the "db\_connect\_user" role has read access, and all other roles have no access.

It's important to remember that, while an identity can be used by several connections, each connection can only be assigned a single identity. When you create a new connection, you specify which identity you want to use with the connection. As you use Splunk DB Connect, you'll only need to specify the connection to use; it will use whatever identity you assigned it. This enables Splunk Enterprise users to work with database contents without knowledge of the database credentials stored in the identity.

For more information about setting up and using connections, see [Create and manage database connections](#).



## Database inputs

A **database input** enables you to retrieve and index data from a database using Splunk Enterprise. It's where you can start to narrow down the data you want to index by building a database query. You can either specify the catalog, schema, and table you want to access (in Automatic Query Mode), or enter a custom SQL query against the database (in Editor Query Mode). DB Connect also enables you to preview the results of your query, so that you know that your query is working the way you expect.

Several parameters also help Splunk Enterprise retrieve your data efficiently and in exactly the way you want. For instance, you can specify whether the input should be a batch input (everything dumped in at once), or whether the input has a rising column (a column that is continuously rising, such as an identifier number or timestamp). You can also specify whether to retrieve all rows or a certain number of rows, identify a timestamp format, and set how often to execute the query.

Once you create your database input, Splunk Enterprise uses DB Connect to query your database, and then indexes your data given the parameters you specified. Indexed data is available to searches, reports, and alerts.

For more information about setting up and using database inputs, see [Create and manage database inputs](#).

## Search

Once you've set up identities, connections, and database inputs, and Splunk Enterprise has indexed your data, you are ready to **search**. Indexed data obtained via Splunk DB Connect from relational databases is searchable just like the rest of your Splunk Enterprise data. To get started, see Searching and Reporting.

Some data is not suitable for indexing, but can be searched directly from Splunk Enterprise. DB Connect provides the *dbxquery* command for querying remote databases and generating events in Splunk Enterprise from the database query result set. The **dbxquery** command supports SQL queries and stored procedures that have been defined in your database. See [dbxquery](#) for command documentation.

For more information about searching in Splunk Enterprise, see the *Search Manual*.

## Database outputs

Splunk DB Connect also enables you to write Splunk Enterprise data back to your relational database using **database outputs**. You can do this interactively from a search head or by setting up an automatic output from a heavy forwarder. Both cases assume that you are connecting to the database using an identity with sufficient write permissions. DB Connect V3 provides a *dbxoutput* search command for running database outputs that you've defined in DB Connect. There is also a predefined **custom alert action** for using the **dbxoutput** command.

- For directions on how to create outputs in DB Connect, see [Create and manage database outputs](#).
- To learn more about Alert Actions in Splunk Enterprise, see Custom alert actions overview.

## Database lookups

Splunk DB Connect includes functionality for you to enrich and extend the usefulness of your Splunk Enterprise data through interactions with your external database. **Database lookups** give you real-time contextual information from a database during ad hoc explorations of data in Splunk.

An example of this functionality would be a lookup that takes a customer ID value in an event, matches that value with the corresponding customer name in your external database, and then adds the customer name to the event as the value of a

new `customer_name` field. Therefore, if you have an event where `customer_id="24601"`, the lookup would add `customer_name="ValJean, Jean"` to that event.

DB Connect V3 provides the *dbxlookup* command for performing lookups by using remote database tables as lookup tables. Use *dbxlookup* to enrich your indexed events with the information stored in external databases.

- For detailed description on how to use the *dbxlookup* command, see [dbxlookup](#)
- For instructions on creating lookups in DB Connect, see [Create and manage database lookups](#).

## Health monitoring

Splunk DB Connect includes a **health dashboard** that allows you to monitor numerous aspects of your database connections and transactions with Splunk Enterprise.

For more information about using the health dashboard, see [Monitor database connection health](#).

## How to get help and learn more about Splunk software

Splunk DB Connect is officially supported by Splunk.

### How to get help

To get help with the Splunk DB Connect add-on, first consult the [Troubleshooting](#) section within this documentation.

You can also log a support case via the Splunk Support Portal.

If your deployment is large or complex, you can engage a member of the Splunk Professional Services team. They will assist you in deploying the Splunk DB Connect add-on.

### Learn more about Splunk software

There are a variety of resources available to help you learn more about Splunk Enterprise and the Splunk DB Connect add-on, including:

- Splunk Enterprise documentation
- Splunk Answers
- The #splunk IRC channel on EFNET



# Before you deploy

## System requirements

Before you install this version of Splunk DB Connect, your environment must meet the requirements listed in this topic.

### Splunk platform requirements

- This version of Splunk DB Connect is compatible with Splunk Platform versions 7.2.0 and later.
- Splunk DB Connect is compatible with Splunk Cloud.
- Splunk DB Connect is not FIPS compliant. However, DBX is compatible with a FIPS compliant Splunk node, so if your Splunk Indexer or Search Head is running in a FIPS environment, DB Connect will work with that Splunk instance.
- You can change your python version to ensure compatibility with Splunk versions, if necessary.

### Operating systems and browsers

Splunk DB Connect runs Windows and \*nix-based operating systems. For version details, see supported operating systems of Splunk Enterprise.

You can use the following browsers to use Splunk DB Connect on your Splunk platform:

- Apple Safari (latest)
- Google Chrome (latest)
- Microsoft Internet Explorer 11
- Mozilla Firefox (latest)

### Splunk licenses and DB Connect

If you configure Splunk DB Connect to import data from a connected database into a Splunk Enterprise index, the amount of data Splunk Enterprise indexes counts towards your Splunk Enterprise license. A Splunk Enterprise license is needed in order to see your data while configuring your inputs, perform lookups, and configure outputs. Using the *dbxquery*, *dbxlookup*, and *dbxoutput* commands against a connected database in DB Connect does not count towards the license.

For more information about Splunk licenses, see [How Splunk licensing works](#).

### Splunk DB Connect Release Notes

For the latest known issues and fixed problems in Splunk DB Connect, see [Release Notes](#).

### User permissions

Before using DB Connect, a logged-in user must have the ability to write to the `$SPLUNK_HOME/var` directory (`%SPLUNK_HOME%\var` on Windows hosts) and to `$SPLUNK_HOME/etc/apps/splunk_app_db_connect` (`%SPLUNK_HOME%\etc\apps\splunk_app_db_connect` on Windows hosts) and its sub-directories. For more information, see [Use access control to secure Splunk data](#).

## Java Runtime Environment (JRE) requirements

Before deploying Splunk DB Connect:

- Download and install one of the following compatible versions of the Java Runtime Environment:
  - ♦ Java Platform, Java Runtime Environment (JRE) 8 from Java Platform, Standard Edition (deprecated).
  - ♦ Java Platform, Open Java Development Kit (OpenJDK) 8 from the OpenJava Project (deprecated).
  - ♦ Java Platform, Java Runtime Environment (JRE) 11 from Java Platform, Standard Edition.
  - ♦ Java Platform, Open Java Development Kit (OpenJDK) 11 from the OpenJava Project.
  - ♦ Java Platform, Java Runtime Environment (JRE) 17 from Java Platform, Standard Edition
  - ♦ Java Platform, Open Java Development Kit (OpenJDK) 17 from the OpenJava Project.
  - ♦ Java Platform, Java Runtime Environment (JRE) 18 from Java Platform, Standard Edition
  - ♦ Java Platform, Open Java Development Kit (OpenJDK) 18 from the OpenJava Project.
  - ♦ Java platform, AdoptOpenJDK 17 the AdoptOpenJDK Project
  - ♦ Java platform, AdoptOpenJDK 18 the AdoptOpenJDK Project
  - ♦ Java platform, Azul JDK 17 the Azul JDK product
  - ♦ Java platform, Azul JDK 18 the Azul JDK product
- Download and install a [supported database driver](#), and run locally or elsewhere on your network.

### **Configure Java Runtime Environment (JRE) for Splunk DB Connect**

1. Download and install supported versions of Java Runtime Environment (JRE). . Only use a supported JVM in server mode, not in client mode.
2. Once you have installed the JRE, write down the path to the JRE directory or set `$JAVA_HOME`. You need the file path when you are [configuring DB Connect](#). When DB Connect prompts you to input the **JRE Installation path** in **Configuration > Settings**, be sure to input the complete JRE file path you wrote down.

Splunk DB Connect define patch to java command in following sequence: Java patch taken from Splunk DB Connect configuration (UI or from file customized.java.path ) If path in point 1 is undefined than DB Connect try takes java from JAVA\_HOME system variable When verification in point 1 and 2 fail it try to take java from system PATH Algorithm which is used to define java for Splunk DB Connect is defined in command.sh in Technical Add-on code.

Validate proper java installation Login as user use for starting Splunk Define java patch (PATH\_TO\_JAVA) which will be use by Splunk DB Connect (algorithm describe above) Validate java installation: Run command from command line "PATH\_TO\_JAVA/java -version" it should return information about installed java. Output may look like "openjdk version "18.0.1.1" 2022-04-22 OpenJDK Runtime Environment Homebrew (build 18.0.1.1+0) OpenJDK 64-Bit Server VM Homebrew (build 18.0.1.1+0, mixed mode, sharing)" Validate binary run (It works for Java 11 and above) Create file named HelloWorld.java with content: public class HelloWorld {

```
public static void main(String[] args) {  
    System.out.println("Hello World!");  
}
```

} In the directory where the file was created, run the command: "PATH\_TO\_JAVA HelloWorld.java" output should be: "Hello World!" NOTE: If any of the steps fail it means that Java is installed or configured incorrectly.

If java is installed and configured properly but the Task Server still can not start please check Splunk logs.

Investigate problem with starting Task Server for Splunk DB Connect. Go to `SPLUNK_HOME/var/log/splunk` Open `splunkd.log` and search for "ERROR" and "splunk\_app\_db\_connect" Error message may look like: "ERROR ModularInputs [7699 MainThread] - <stderr> Introspecting scheme=server:

/opt/splunk/etc/apps/splunk\_app\_db\_connect/bin/server.sh: 33: exec: java: not found" Try to resolve problem described by error message

### ***Java Runtime Environment (JRE) for Splunk DB Connect running on Splunk Cloud Victoria Experience***

DB Connect running on Search Heads inside a Splunk Cloud Victoria Experience will use the JRE that is already installed on the Search Head:

1. If your Cloud Stack is version 8.2.2107 or below, DB Connect will be using Java 8.
2. If your Cloud Stack is version 8.2.2109 or above, DB Connect will be using Java 11.
3. If there are any issues with the installed Java version, file a support ticket.

If you are unsure whether you have the correct version of Java installed, access [DB Connect setup](#). On the first screen of the DB Connect setup or at **Configuration > Settings > General** you can specify the path to your JRE. If there is a problem with the Java install or the system `$JAVA_HOME` variable is incorrect, DB Connect displays an error message. It is important that you resolve any JRE issues before proceeding as DB Connect uses Java Database Connectivity (JDBC) to communicate with your databases.

### **Database and JDBC database driver requirements**

Splunk DB Connect supports connections to many databases. You must install a Java Database Connection (JDBC) driver or a JDBC driver add-on for your database before you can connect to databases with DB Connect.

For more information about the databases that Splunk DB Connect supports, see [Supported databases](#).

For more information about the JDBC driver add-ons, see [new drivers using JDBC Driver addons JDBC driver add-ons](#).

For information about connecting to unsupported databases with Splunk DB Connect, see [Other databases](#).

### **Architecture and performance considerations**

When adding Splunk DB Connect to your deployment, there are several architecture and performance considerations to take into account. You can install and run Splunk DB Connect on Splunk Enterprise deployments ranging from a single host (indexer and Splunk Web both running on the same system) to a large distributed deployment (multiple search heads, search head clusters, indexers, load-balanced forwarders, and so on). This topic provides guidance for setting DB Connect up and running in these environments. It also describes the kind of performance you can expect based on your deployment and capacity requirements.

#### **Database performance considerations**

If Splunk DB Connect retrieves a large amount of data from your database, it may affect your database performance, especially for the initial run. Subsequent runs of the same query may have less impact, as the database may cache results and only retrieve new data since the previous run of the query.

## Architecture and performance considerations

When adding Splunk DB Connect to your deployment, there are several architecture and performance considerations to take into account. You can install and run Splunk DB Connect on Splunk Enterprise deployments ranging from a single host (indexer and Splunk Web both running on the same system) to a large distributed deployment (multiple search heads, search head clusters, indexers, load-balanced forwarders, and so on). This topic provides guidance for setting DB Connect up and running in these environments. It also describes the kind of performance you can expect based on your deployment and capacity requirements.

### Database performance considerations

If Splunk DB Connect retrieves a large amount of data from your database, it might affect your database performance, especially for the initial run. Subsequent runs of the same query might have less impact, as the database might cache results and only retrieve new data since the previous run of the query.

### Performance considerations in distributed environments

To use Splunk DB Connect in a distributed search environment, including search head clusters, you must determine the planned use cases. For ad hoc, interactive usage of database connections by live users, install the app on search heads. For scheduled indexing from databases and output of data to databases, install the app on heavy forwarders.

When planning a large DB Connect deployment, the ideal configuration for your needs can depend on a number of factors, including:

- Total number of Forwarders in the deployment, and the hardware specifications of each.
- Total expected data volume to transfer.
- Number of [database inputs](#) per Forwarder.
- Dataset size, per input, per interval.
- **Execution Frequency**, the interval length between a database input's separate executions.
- **Fetch size** (Not all JDBC drivers use this parameter for returning result sets).

Overloading the system can lead to data loss, so performance measurement and tuning can be critical. Use [performance expectations](#) as the reference to plan your deployment, and monitor expected data returns for loss conditions.

### Performance expectations

This section provides measured throughput data achieved under certain operating conditions. Use the information here as a basis for estimating and optimizing the DB Connect throughput performance in your own production environment. As performance might vary based on user characteristics, application usage, server configurations, and other factors, Splunk can't guarantee specific performance results.

Splunk produced the performance data in the following table with the following test bed and DB Connect configuration (Increasing cores or RAM might improve scaling characteristics):

- Server: 8-core 2.60GHz CPU, 16GB RAM, 1Gb Ethernet NIC, 64bit Linux
- JVM config: MaxHeapSize = 4GB. (For more information about the JVM memory setting, see "[Performance tuning advice](#)".)
- Data Source: Oracle 11g

## Inputs

- Number of inputs: 1600
- Data payload (per input execution) : 250KB
- Duration = 45 minutes
- Interval: 1 minute

total data volume = data payload \* duration / interval \* number of inputs = 17.5 GB

Data payload per input execution is the same for different input modes (rising column and batch)

## Queries

Rows in data set					
DB Connect 3	1.2 seconds	1.3 seconds	1.6 seconds	4.1 seconds	22.9 seconds
DB Connect 2	1.4 seconds	1.5 seconds	2.4 seconds	11.4 seconds	103.5 seconds

## Lookups

Rows in data set			
DB Connect 3	1.2 seconds	2.8 seconds	36.0 seconds
DB Connect 2	0.2 seconds	4.3 seconds	70.0 seconds

## Outputs

Rows in data set					
DB Connect 3	2.1 seconds	1.9 seconds	3.0 seconds	9.1 seconds	67.2 seconds
DB Connect 2	1.0 seconds	1.5 seconds	10.0 seconds	83.9 seconds	644.0 seconds

## General performance tuning considerations

While it's impossible to provide prescriptive advice for maximizing performance in every situation, the following observations and tips can help you tune and improve performance in your unique distributed deployment:

1. **Only select columns if necessary.** A table can contain many types of columns. When ingesting data from a database into DB Connect, you likely don't need all of them. Therefore, instead of using a `SELECT * FROM ...` clause to retrieve all the columns, select only what you need by using a `SELECT columnNeeded1, columnNeeded2, ... FROM ...` clause. More columns means more memory claimed by the task server; omit those unnecessary columns to make smarter use of your available memory. See [SQL tips and tricks](#) for more details.
2. **Avoid reaching the 5MB/10MB limit.** Large column sizes can cause DB Connect to potentially run out of memory and behave erratically, so DB Connect has a column size limit of 10MB for data columns that hold two-byte data types and 5MB for one-byte data types. Splunk truncates data for columns with data that exceeds these limits. If possible, trim the amount of data stored per column so that you avoid the DB Connect hard caps.

3. **Adjust the fetch size based on your scenario.** The **Fetch Size** input parameter specifies the number of rows returned at a time from a database, which defaults to 300 rows. A higher fetch size means Splunk receives more records per database request, so you can use fewer database requests to retrieve the same total number of records. This increases resource utilization on the database and in DB Connect, but can lead to performance improvements. Lowering the fetch size parameter can help prevent the Task Server from hitting its memory limit. If you receive out of memory errors when you increase the fetch size, you might need to increase the memory heap size from its default of 1/4 of system RAM.
4. **Reduce the total number of database inputs.** It can increase the amount of data that each input is taking in. This helps ensure that CPU cores have to handle fewer processes within a given window of time. Small datasets can be slower than large because of environment initialization.
5. **Reduce the concurrency of scheduled database tasks.** Shifting start times for scheduled tasks reduces choke points during which inputs and outputs have to share resources. For more information, see ["Set parameters" in Create and manage database inputs](#).
6. **Adjust batch\_upload\_size field.** The `batch_upload_size` field defines the number of events sent to splunkd through HEC per request, which defaults to 1,000 records. A higher batch upload size means Splunk sends more records per HTTP post, so you can use fewer server transactions to index the same total number of records. This increases resource utilization on the Forwarder, but can lead to performance improvements. You can increase the `batch_upload_size field` under `$SPLUNK_HOME/etc/apps/splunk_app_db_connect/local/db_inputs.conf` to have better performance.
7. **Specify sufficient hardware.** In general, Splunk best practice is to use the same hardware specifications for DB Connect as required for Splunk Enterprise. Increased hardware might be necessary for increased indexing loads.
8. **Configure Java for performance.** Current Java engines automatically reserves 25% of the machine's RAM when opening. If your **JVM Options** setting is as `-Xmx1024m` (which is the default value from DB Connect version 2.0 to 2.2). You can remove it and use the default JVM setting. For more information about changing JVM options, see ["JVM Options" in Configure DB Connect Settings](#).
9. **Configure Splunk for performance.** Increase Splunkd's index queue size and number of Parallel Ingestion Pipelines to avoid concurrency limits.
10. **Configure DB Connect for performance** Set [SchedulerThreadPoolSize](#) to match the number of processor cores.

## Performance nonfactors

During testing, varying the following factors had a negligible effect on performance:

- There was no discernable performance difference between running in batch mode (all events processed) and running in rising column mode (only the new events processed) with the same dataset.
- The number of defined [database connections](#) does not limit performance. The number of connections is different from the number of database inputs.

## More performance help

If you are still experiencing performance issues, or want to receive feedback tailored to your setup, you have the following options:

- Post a request to the community on Splunk Answers.
- Contact Splunk Support.

# Install Splunk DB Connect

## Installation and setup overview

This topic provides an overview of how to install and set up Splunk DB Connect.

To deploy Splunk DB Connect on either a single instance of Splunk Enterprise or on a search head in a distributed deployment, you must meet the [system requirements](#).

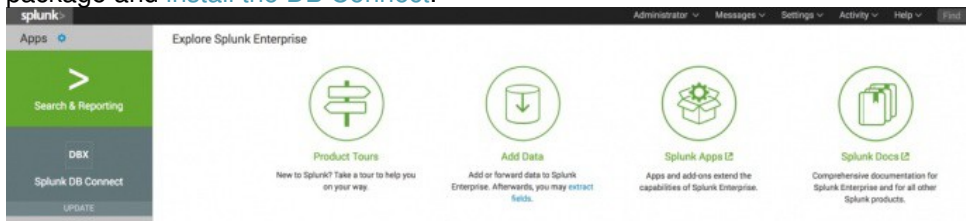
Once the system requirements are met, you can start the DB Connect installation process:

1. [Download and install](#) the DB Connect app.
2. Install a JDBC driver for your database. See [Install database drivers](#).
3. After installing DB Connect and restarting Splunk Enterprise, launch DB Connect.
4. Create a database identity and set up a database connection.
5. Create a new [database input](#) and [use it as a data input](#) in a Splunk Enterprise search.

For distributed deployments, there are further instructions for [deploying the distributed deployment](#).

## Upgrade an existing DB Connect deployment

Click the **Update** button and follow the wizard to upgrade your DB Connect using Splunk web, or you can download the package and [install the DB Connect](#).



There are some known limitations on upgrading DB Connect on certain circumstances, please review the following list before upgrading.

1. If you are running DB Connect on Windows platform and want to upgrade to the higher version. You need to:
  1. On the Splunk Web home page, click **Apps > Manage Apps**.
  2. On the **Apps** page, navigate to Splunk DB Connect, and click **Disable**.
  3. Upgrade DB Connect and restart Splunk platform.
  4. On the Splunk Web home page, click **Apps > Manage Apps**.
  5. On the **Apps** page, navigate to Splunk DB Connect, and click **Enable**.
2. If you are using DB Connect (version prior to 3.0.0) and want to upgrade to DB Connect 3, see [Migrate DB Connect deployment to DB Connect 3](#).
3. If you want to migrate DB Connect in Splunk Cloud, contact Splunk Support. Do not upgrade from previous versions yourself because you cannot migrate configuration files on your Splunk Cloud instance.

## Migrate DB Connect deployment to DB Connect 3

Due to changes introduced in DB Connect versions 3.10.0 and higher, it is no longer possible to have a direct migration from DB Connect 2 to versions later than 3.9.0. If you want to migrate from DB Connect 2, migrate to a version of DB Connect 3 first (3.0.0 - 3.9.0) and then upgrade to a later version. Migration instructions to version 3.9.0 can be found in the previous version of this documentation.

## Install database drivers

After you've [downloaded and installed](#) Splunk DB Connect, the first step in the DB Connect setup process is installing a Java Database Connectivity (JDBC) database driver.

The recommended way to install a JDBC driver on a Splunk instance is to install a JDBC driver add-on. After you add the database driver, continue with either the [single server](#) or [distributed deployment](#) instructions. You will be able to verify whether the database driver was installed successfully during DB Connect setup.

### Install new drivers using JDBC Driver add-ons

DB Connect officially supports connecting to databases using JDBC drivers made available through the following Splunk add-ons for DB Connect using specific databases:

- MS SQL Server
- MySQL
- Oracle
- Postgres
- Redshift
- Snowflake

To use a JDBC driver from one of the add-ons simply install the add-on following installation instructions and DB Connect will automatically use the JDBC driver provided by the add-on.

For all supported databases by drivers add-ons, to install the JDBC driver on a Splunk instance, follow these instructions:

1. Install the JDBC driver add-on for your database, if available.
2. **Reload** the driver under **Settings>Drivers**.

In case of missing add-on for your database please open idea in [<http://ideas.splunk.com/IdeasPortal>]

Please avoid installation of the same JDBC driver via the `$SPLUNK_HOME/etc/apps/splunk_app_db_connect/drivers` directory and a JDBC driver add-on. In such a case the JDBC driver provided by the add-on will take priority over the one in `$SPLUNK_HOME/etc/apps/splunk_app_db_connect/drivers` directory

The Java Runtime Environment (JRE) used by your deployment's `JAVA_HOME` must match the JRE version of the driver that you are installing.

List of supported database should be provided in each JDBC Driver addons



## Upgrading add-on from manual install method

The recommended way to install a JDBC driver on a Splunk instance is to install a JDBC driver add-on. Drivers installed manually can be upgraded to add-on based drivers by using the following procedure:

1. Remove appropriate manually installed JDBC driver from directory  
`$SPLUNK_HOME/etc/apps/splunk_app_db_connect/drivers`
2. **Reload** the driver under **Settings>Drivers**. Check if driver was disabled in drivers list
3. Install the appropriate JDBC driver from this manual: Install new drivers using DBX Driver add-ons

## Install unsupported drivers (customer managed only)

In addition to the supported databases that Splunk has tested and certified for use with DB Connect, you may also be able to use unsupported JDBC-compatible databases with Splunk DB Connect. You will need to provide the necessary JDBC drivers to add your own database types. You can add custom support to Splunk DB Connect for any database that has a JDBC driver, even if it doesn't appear in the [supported database matrix](#).

### Notes:

- ◆ Connecting to a database that is not listed in the supported database matrix is not supported by Splunk Support.
- ◆ At a minimum, Splunk DB Connect supports querying custom database connections. For some custom database connections, certain query-related features may not work.

Installing a custom database is a multi-step process:

1. [Download and install the JDBC driver file](#).
2. [Add the custom database to db\\_connection\\_types.conf](#).
3. [Troubleshoot or disable connection validation, if necessary](#).

### **Download and install the JDBC driver file**

Download the JDBC driver for the database you want to add, and copy the .JAR file to the

`$SPLUNK_HOME/etc/apps/splunk_app_db_connect/drivers` directory  
(`%SPLUNK_HOME%\etc\apps\splunk_app_db_connect\drivers` on Windows hosts).

### **Add the custom database to db\_connection\_types.conf**

When you add a custom database connection that Splunk DB Connect does not support by default, you must create a stanza to define the database connection in a copy of [db\\_connection\\_types.conf](#) under

`$SPLUNK_HOME/etc/apps/splunk_app_db_connect/local` (`%SPLUNK_HOME%\etc\apps\splunk_app_db_connect\local` on Windows hosts), for example

```
displayName = $your database name$
serviceClass = com.splunk.dbx2.DefaultDBX2JDBC
jdbcDriverClass = $JDBC driver class$
jdbcUrlFormat = $JDBC URI Format$
```

```
ui_default_catalog = $database$
```

For more information, see [Configuration file reference](#).

### **Database connection validation**

Each time DB Connect uses a database connection, it tries to validate that the database connection is actually working. If validation fails, you might see an error message, such as "ValidateObject failed".

DB Connect uses these two methods to validate a connection:

1. If a **testQuery** is specified in **db\_connection\_types.conf**, DB Connect executes that query, and receives a response that validates that the connection is working.
2. If **testQuery** is not specified, DB Connect uses the Java method **connection.isValid()**, and relies on the JDBC driver to answer. Some JDBC drivers do not implement this API call. (For instance, Apache Derby was built against Java 1.5, where JDBC doesn't have the method **isValid**.) The workaround is to specify a manual **testQuery**, such as `SELECT 1`.

After you add the custom database driver, continue with either the [single server](#) or [distributed deployment](#) instructions.

### **Enable SSL for your database connection**

DB Connect has built-in support for connecting via SSL to several supported databases. Though other supported databases may support SSL connections, DB Connect support for SSL means that Splunk has tested SSL for that database type and supports connecting over SSL using DB Connect.

If you want to deploy DB Connect in Splunk Cloud. You must enable SSL connection for your database. See [supported database matrix](#) for the database supported for cloud deployment.

DB Connect will detect whether it supports SSL for your database type,

- If DB Connect supports SSL connections for your database type, enabling an SSL connection is easy. When you [create a new database connection](#), select the **Enable SSL** checkbox.
- If DB Connect does not natively support SSL for your database type, you cannot select the **Enable SSL** checkbox in connection settings, it is grey out. However, you can add in SSL support by selecting **Edit JDBC URL** and editing the URL. Be aware that Splunk cannot certify that databases for which DB Connect does not have native support for SSL will connect via SSL correctly. Splunk Support will not help you troubleshoot such connections.

The possibility to configure one-way SSL authentication from the UI was added for MySQL, MSSQL and Oracle databases. While creating a connection users will be able to pass a certificate which will be automatically added to the Java TrustStore and used for the server authentication. If the **Enable SSL** checkbox will be selected, but the certificate field would be empty - DB Connect will ignore it and work as before.

To improve security following some properties have been removed from the default connection properties. The full list of changes can be found below.

1. MySQL - {"verifyServerCertificate":"false"}
2. MS-SQL Server Using MS Generic Driver - {"trustServerCertificate":"true"}
3. MS-SQL Server Using MS Generic Driver With Windows Authentication - {"trustServerCertificate":"true"}
4. MS-SQL Server Using MS Generic Driver With Kerberos Authentication - {"trustServerCertificate":"true"}
5. Oracle -  
{ "oracle.net.authentication\_services": "(TCPS)", "oracle.net.ssl\_cipher\_suites": "(SSL\_DH\_anon\_WITH\_3DES\_EDE\_CBC\_SHA, TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA)" }

#### 6. Oracle Service -

```
{ "oracle.net.authentication_services": "(TCPS)", "oracle.net.ssl_cipher_suites": "(SSL_DH_anon_WITH_3DES_EDE_CBC_SHA, TLS_RSA_WITH_AES_128_CBC_SHA)" }
```

#### 7. PostgreSQL - { "sslfactory": "org.postgresql.ssl.NonValidatingFactory" }

#### 8. AWS RedShift - { "verifyServerCertificate": "false" }

#### 9. AWS RedShift version 2.0 - { "verifyServerCertificate": "false" }

#### 10. AWS RDS Aurora - { "verifyServerCertificate": "false" }

#### 11. Hive - { "verifyServerCertificate": "false" }

#### 12. Spark SQL - { "verifyServerCertificate": "false" }

If you still want to use them, they can be added to `connection_properties` in the connection definition, or defined in the `jdbcUrl`.

### ***Use a different default driver for your database***

You can change the driver that DB Connect uses for your database type. This can be useful if you want to use a custom driver to connect to your database. To change the driver, you edit the `db_connection_types.conf` file.

First, get the JDBC driver vendor's Java driver class name. For example, the class name of the Microsoft JDBC Driver for SQL Server is **com.microsoft.sqlserver.jdbc.SQLServerDriver**. Once you've found the correct Java class name, do the following:

1. Using a text editor, open the **db\_connection\_types.conf** from within  
`$SPLUNK_HOME/etc/apps/splunk_app_db_connect/default/`  
(`%SPLUNK_HOME%\etc\apps\splunk_app_db_connect\default` on Windows hosts).
2. Find the stanza for the database type for which you want to configure a custom driver, select it, and then type Ctrl-C or Command-C to copy it.
3. Create a new **db\_connection\_types.conf** file (if one doesn't already exist) in  
`$SPLUNK_HOME/etc/apps/splunk_app_db_connect/local/`  
(`%SPLUNK_HOME%\etc\apps\splunk_app_db_connect\local` on Windows hosts). If the file is a new file, open it and type `[local]` and then two new lines.
4. Paste in the database stanza you copied from the default **db\_connection\_types.conf** file.
5. Change the entry next to `jdbcDriverClass` to match the Java class name for your custom driver.
6. If you want to retain the ability to choose the original database driver, change the name of the stanza and update the `displayName` attribute to differentiate it from the original driver.
7. Restart Splunk Enterprise.

## **Supported databases**

Splunk DB Connect supports the databases listed in the following matrix.

- The driver listed in the *JDBC driver name/link* column is the driver that Splunk has tested with Splunk DB Connect.
- DB Connect supports and has been tested with Java cryptography standard algorithm, if you need a stronger algorithm, you must install the "JCE Unlimited Strength Jurisdiction Policy Files." Legal restrictions may apply in your country. See Java Security Providers and JCE Download pages for details. Consult your database vendor's documentation for configuration instructions.

## Supported databases by JDBC Add-Ons

JDBC driver Add-Ons provides in documentation what version of java and database supports.

Database Name	JDBC driver Add-on	JDBC driver Addon Documentation	SSL support*
AWS RDS Aurora	Splunk Add-on for MySQL JDBC	JDBC Add-on for MySQL Documentation	Yes
AWS RedShift	Splunk Add-on for Redshift JDBC	JDBC Add-on for RedSift Documentation	Yes
MemSql	Splunk Add-on for MySQL JDBC	JDBC Add-on for MySQL Documentation	Yes
Oracle Database	Splunk Add-on for Oracle JDBC	JDBC Add-on for Oracle Documentation	Yes
MySQL	Splunk Add-on for MySQL JDBC	JDBC Add-on for MySQL Documentation	Yes
Postgres	Splunk Add-on for Postgres JDBC	JDBC Add-on for Postgres Documentation	Yes
Snowflake	Splunk Add-on for Snowflake JDBC	JDBC Add-on for Snowflake Documentation	Yes
Microsoft Sql Server	Splunk Add-on for Microsoft Sql Server JDBC	JDBC Add-on for Microsoft Sql Server Documentation	Yes
IBM DB2	Splunk Add-on for DB2 JDBC	JDBC Add-on for DB2 Documentation	Yes

## Other Supported Databases

Database	Database version	JDBC driver name	JDBC driver version tested	SSL support*	Cloud support**
AWS RedShift	1.0.1044	RedshiftJDBC41-1.2.1.1001.jar Not compatible with RedshiftJDBC42-1.2.1.1001.jar.			

1.1YesNoInformix12.10.FC5TLInformix JDBC Driver3.0YesNoSAP SQL Anywhere (aka Sybase SA)16.0.0.1948Sybase jConnect7.0NoNoSybase ASE16.0.02.00.1014Sybase jConnect7.0NoNoSybase IQ15.4.1.3019Sybase jConnect7.0NoNo

\* **SSL support:** This column specifies whether DB Connect tests and supports Secure Sockets Layer (SSL) connections to this database. Though other supported databases may support SSL connections, DB Connect support for SSL means that Splunk has tested SSL for that database type and supports connecting over SSL using DB Connect.

\*\* **Cloud support:** This column specifies whether the database is supported for Splunk Cloud. You must enable SSL connection if you want to use DB Connect in a Splunk Cloud deployment.

## Useful information about drivers configuration

### Notes:

- ◆ Your database connection must be set up to return results encoded in UTF-8. Consult your database vendor's documentation for instructions.
- ◆ DB Connect supports sending data that is in a multi-byte character set, such as Traditional Chinese, using a [database output](#). Depending on your database, you may need to change certain settings on the database side to the database to properly receive and store the data. See [Enable output to multi-byte character sets](#).

## IBM DB2

IBM DB2 is supported when the database is running on Linux. Splunk doesn't test or support DB2 on AS/400 or Windows.

**Note:** IBM DB2 is only supported when the database is running on Linux. Splunk doesn't test or support DB Connect with DB2 on AS/400 or on Windows.

If you want to use GSS API security mechanisms in IBM DB2, download *Java Cryptography Extension (JCE) Policy* to your `$JRE_Installation_Path/jre/lib/security` and set the `encryptionAlgorithm` parameter of JDBC URL as

```
jdbcUrlFormat =  
jdbc:db2://<host>:<port>/<database>;securityMechanism=9;encryptionAlgorithm=2;
```

Follow these instructions to install the DB2 JDBC Driver:

1. Go to DB2 JDBC Driver Versions and Downloads on the IBM website, and click the link for the latest **DB2 Version 10.5** driver.
2. Click the **Download** link for the **IBM Data Server Driver for JDBC and SQLJ** (JCC Driver).
3. Select the newest fix pack, and then click **Continue**. You will need to log in with your IBM ID, or create one if you don't already have one.
4. Click the **Download using your browser (HTTPS)** radio button, and then **Continue**.
5. Right-click the file name, and then save it to your hard disk.
6. Expand the downloaded file, and then expand the **db2\_db2driver\_for\_jdbc\_sqlj.zip** file.
7. Copy or move the **db2jcc4.jar** file to the `$SPLUNK_HOME/etc/apps/splunk_app_db_connect/drivers` directory.
8. **Reload** the driver under **Settings>Drivers**.

For additional information, see the Installing and Connecting to Clients documentation on the IBM website.

## Microsoft SQL Server

You have several options for connecting to a Microsoft SQL Server. Start by determining:

- Which operating system (Windows or Linux) is running Splunk DB Connect.
- Which driver you want to use.
- How your database server authenticates your connection.

See **Legends** below the table for a more detailed explanation.

Follow this table from left to right, choosing the options that correspond to your Microsoft SQL Server environment. For more information about the meaning of each column, see the legend that follows the table.

Operating System*	Driver**	Authentication***	How to install driver
Windows	MS	SQL Authentication	<a href="#">Install the SQL Server database driver using SQL server authentication</a>
	Generic	Windows Authentication	<a href="#">Install the SQL Server using MS Generic driver with Windows authentication</a>

Operating System*	Driver**	Authentication***	How to install driver	
	Linux	MS Generic	SQL authentication	Install the SQL Server database driver using SQL server authentication
			Kerberos authentication	Install the SQL Server database driver using the MS Generic driver with Kerberos authentication

## Legend

\* **Operating System:** The operating system on which DB Connect is running. DB Connect is supported on both Linux and Windows Server.

\*\* **Driver:** The JDBC driver that DB Connect will use to connect to your database:

♦ **MS Generic:** Microsoft JDBC Driver for SQL Server (the "generic Microsoft driver")

\*\*\* **Authentication:** The type of service account that DB Connect will use to connect to your database:

- **SQL Authentication:** Log into SQL Server as a SQL Server user. This option assumes you will [create an identity](#) that uses a username and password that is defined on the database.
- **Windows Authentication:** Log into SQL Server as a Microsoft Integrated Windows Authentication user ("domain user"). This assumes that you are [creating an identity](#) that uses a domain, user name, and password to connect to the database, and that the user you assign to the identity is part of an Active Directory user group that has been set up to log into your SQL Server.
- **Kerberos Authentication:** DB Connect uses a Kerberos client (Linux) to negotiate an authentication ticket with the Active Directory environment directly. This assumes that the user you assign to the identity is part of an Active Directory user group that has been set up to log into your SQL Server.

## Install the SQL Server database driver using a SQL server authentication

If you will log onto your SQL Server database using a SQL Server username and password (non-domain attached), follow these instructions:

### Install the SQL server database driver

To install the Microsoft JDBC Driver for SQL Server, please follow these instructions:

1. Install the MSSQL JDBC driver add-on.
2. **Reload** the driver under **Settings>Drivers**.

To install the SQL Server database driver in an on prem Splunk instance or to install a different version of the SQL Server Database driver than available via the add-on in a Splunk instance, please follow these instructions:

1. Download the appropriate JDBC driver for SQL Server:
  - ♦ For the Microsoft JDBC Driver for SQL Server download the driver from the Microsoft JDBC Driver page.
2. Move the driver file to the correct location:
  - ♦ For the MS Generic Driver, from inside the zip file, copy or move the proper jar file to the `$(SPLUNK_HOME)/etc/apps/splunk_app_db_connect/drivers` directory (`%SPLUNK_HOME%\etc\apps\splunk_app_db_connect\drivers` on Windows hosts).

- ◆ If you will need to use a database service account on Windows with the Generic driver, you will also need to install the JDBC Auth library:

1. From the Microsoft JDBC Driver for SQL Server download, locate the **.dll** file.
2. Copy the **.dll** file to **C:\Windows\System32** on your Splunk Enterprise server.
3. From the Windows Control Panel, go to **Services**, and then get properties on **Splunk Service**.
4. Click the **Log On** tab, and then change the **Log on as** setting from the Local System account to that of the logged on domain user.

**Note:** The domain user should have sufficient privileges to access the SQL Server instance.

5. Save your changes, and then restart the Splunk Enterprise server for the changes to take effect.

When you [create a connection](#) to this database in DB Connect, you must choose **MS-SQL Server Using MS Generic Driver** from the **Connection Types** popup menu.

### ***Install the SQL Server database driver using the MS Generic driver with Kerberos authentication***

If you're running DB Connect on Linux, you can connect to a Microsoft SQL Server using a Windows domain and user name by enabling Kerberos authentication.

Before starting the following procedure:

1. See the instructions on how to [Install the appropriate JDBC driver for SQL Server](#)
2. Enable Kerberos on the Microsoft SQL Server.

Now complete the following steps to enable DB Connect to use Kerberos authentication to connect to your SQL Server database:

1. First, get the domain name:
  1. Log onto the Windows server that is running your SQL Server database.
  2. At the command line, run `systeminfo`, and then copy down the domain name (next to the **Domain:** label).
2. Next, get the Kerberos Distribution Center (KDC) information:
  1. From the command line, run the following, replacing `<domain>` with the domain name you copied in the previous step: `nltest /DSGETDC:<domain>`
  2. Copy down the name of the domain controller, which is next to the **DC:** label. This is the KDC host name.
3. Back on the machine running DB Connect, use the `kinit` command to generate a **krb5.conf** file, which Kerberos will use to authenticate with the SQL Server for DB Connect.
  1. First, install `kinit` by entering the following at the command line: `apt-get install krb5-user` If you are prompted to enter a realm, enter the domain name you obtained in Step 1, but in all upper-case letters. For example, if your domain is `abc.dbx`, you'd enter `ABC.DBX`.
  2. Next, use `kinit` to initialize the Kerberos ticket cache. Enter the following at the command line, replacing `<username>` with the appropriate user name (such as `Administrator`) and `<DOMAIN>` with the domain name, in upper-case letters: `kinit <username>@<DOMAIN>`

The **krb5.conf** has been generated in the **/etc** directory.

**Note:** If the **krb5.conf** file is not in the **/etc** directory, set the following option in **JVM Options** under the **Settings** tab:

`-Djava.security.krb5.conf=/path/to/krb5.conf`

4. Create an identity in DB Connect.

1. Select **New Identity** under **Explorer > Identities**.
2. Enter **Identity Name**, **Username**, **Password**, and then click **save**.
  - **Username**: Enter the username of your Kerberos account. If you don't select **Use Windows Authentication Domain?**, you have to append @<DOMAIN> at the end of the username, e.g. Administrator@<DOMAIN>
  - **Password** and **Confirm Password**: Enter the password of your Kerberos account.
  - **Use Windows Authentication Domain?**: If you select it, you have to enter the <DOMAIN> in **Windows Authentication Domain** field, then you don't have to append @<DOMAIN> at the end of the **Username**.

**Note:** You can add multiple identities if you need to use the different users to do the Kerberos authentication.

5. Create a new connection in DB Connect as you normally would, keeping in mind the following:
  - ◇ **Database Types**: Be sure to choose **MS-SQL Server Using MS Generic Driver with Kerberos Authentication** from the popup menu.
  - ◇ **JDBC URL Format**: Once you've entered all the necessary connection information, add the following to the JDBC URL string to enable DB Connect to use the Kerberos authentication you just set up:

```
· integratedSecurity=true
· authenticationScheme=javaKerberos
```

After you're finished, the field should contain the following:

```
jdbc:sqlserver://
/<host>:<port>;databaseName=<database>;selectMethod=cursor;integratedSecurity=true;authenticationScheme=javaKerberos
```

**Note:** To debug Kerberos authentication, set the following option in JVM Options under the **Settings** tab:

```
-Dsun.security.krb5.debug=true
```

## Multiple SQL Server instances

If you have multiple instances of Microsoft SQL Server installed on your server, you will need to edit the JDBC connection string to add a parameter that explicitly references the instance you want to contact.

1. Follow the instructions in "[Override db\\_connection\\_types.conf](#)" to make a copy of the **db\_connection\_types.conf** file in the **local** directory and copy the stanza for the Microsoft SQL Server driver you're using into the file.
2. Edit the **jdbcUrlFormat** or **jdbcUrlSSLFormat** (if you're connecting using SSL) setting by appending it with the following: `;instanceName=`
3. Set the **instanceName** parameter to the name of the instance you want to connect to. For example:  

```
jdbc:sqlserver://dbx-sqlserver.mydomain.com:1433;databaseName=master;instanceName=test
```
4. Save and close the file, and then **Reload** the driver under **Settings>Drivers**

After you add the database driver, continue with either the [single server](#) or [distributed deployment](#) instructions. You will be able to verify whether the database driver was installed successfully during DB Connect setup. If you've already set up DB Connect and are adding a new database, click **Settings** in the top navigation bar and then the **Driver** tab to view the driver status. If you have trouble, see "[Troubleshoot driver connections](#)."

## Sybase ASE, SAP/Sybase SA, and SAP/Sybase IQ

Follow these instructions to download and install the jConnect for JDBC driver for Sybase Adaptive Server Enterprise (ASE), SAP/Sybase SQL Anywhere, and SAP/Sybase IQ:



1. Go to the SAP software downloads page and log in.
2. Search for and download the jConnect package.
3. Expand the file you just downloaded and locate the **jconn4.jar** file inside the **classes** directory.
4. Copy or move the **jconn4.jar** file to the `$SPLUNK_HOME/etc/apps/splunk_app_db_connect/drivers` directory (`%SPLUNK_HOME%\etc\apps\splunk_app_db_connect\drivers` on Windows hosts).
5. **Reload** the driver under **Settings>Drivers**.

## Spark SQL

Your Spark instance must be running the Thrift JDBC/ODBC service before Splunk will be able to connect to it. Follow these instructions to install the Simba Spark JDBC driver:

**Note:** SparkSQL support requires Thrift server setup for JDBC  
<http://spark.apache.org/docs/latest/sql-programming-guide.html#running-the-thrift-jdbcodbc-server>

1. Go to the Spark JDBC Driver page on the Simba website.
2. Click and download the ZIP file for the latest version of the driver. You can download a free 30 day trial or purchase it.
3. Expand the **Simba\_Spark\_JDBC\_Desktop.zip** file you just downloaded. There are two zip files in the **Simba\_Spark\_JDBC\_Desktop.zip** folder, expand **SimbaRTU\_SparkJDBC41\_Client\_1.0.2.1004.zip**.
4. Copy all the **SparkJDBC41.jar** file to `$SPLUNK_HOME/etc/apps/splunk_app_db_connect/drivers` and the other .jar files under `$SPLUNK_HOME/etc/apps/splunk_app_db_connect/drivers/SparkJDBC41-libs`.
5. Copy your Spark JDBC driver license file to the `$SPLUNK_HOME/etc/apps/splunk_app_db_connect/drivers` directory.
6. **Save and restart RPC server** under **Settings > General Settings**.

## Teradata

Follow these instructions to install the Teradata JDBC Driver:

1. Go to the Teradata JDBC Driver page on the Teradata website.
2. Click the link to download the ZIP or TAR file for the latest version of the driver. You need to log in with your Teradata user account.
3. Expand the file you just downloaded.
4. From inside the **TeraJDBC\_...** directory, copy or move the two .JAR files (**terajdbc4.jar** to the `$SPLUNK_HOME/etc/apps/splunk_app_db_connect/drivers` and **tdgssconfig.jar** to the `$SPLUNK_HOME/etc/apps/splunk_app_db_connect/drivers/terajdbc4-libs`) directory.
5. **Reload** the driver under **Settings>Drivers**.

For more information, including how to use the **tdgssconfig.jar** file, see the **readme.txt** file inside the **TeraJDBC\_...** directory, or the Teradata JDBC Driver Reference.

## Oracle

Follow these instructions to install the Oracle JDBC Driver in a Splunk instance:

1. Install the Oracle JDBC driver add-on.
2. **Reload** the driver under **Settings>Drivers**.

To install the Oracle JDBC driver in a different version of the Oracle JDBC driver than available via the add-on in a Splunk instance, please follow these instructions:

1. Go to the Oracle JDBC Driver Downloads page.
2. After you have downloaded the correct driver for your database, copy the .JAR driver file to the `$SPLUNK_HOME/etc/apps/splunk_app_db_connect/drivers` directory (`%SPLUNK_HOME%\etc\apps\splunk_app_db_connect\drivers` on Windows hosts). If you want to use `ojdbc7` to parse XML data type, add `xmlparserv2.jar` as a dependency
3. **Reload** the driver under **Settings>Drivers**.

For users of Oracle 11g, the `SSL_DH_anon_WITH_3DES_EDE_CBC_SHA`, `SSL_DH_anon_WITH_RC4_128_MD5`, and `SSL_DH_anon_WITH_DES_CBC_SHA` cipher suites are disabled by default in Java 8. To allow these cipher suites, see the Test or Revert changes to Oracle's JDK and JRE Cryptographic Algorithms section of the Java documentation.

## Connect to Oracle using SSL

You can connect to an Oracle database using Secure Sockets Layer (SSL) 3.0 (TLS) in three ways:

- Without using SSL authentication: Neither client (DB Connect) nor server verifies each other's certificate authority (CA).
- Using one-way SSL authentication: The client verifies the server's CA.
- Using two-way SSL authentication: Both client and server verify each other's CA.

If you are using one-way or two-way authentication, you will need to have set up an Oracle wallet on the server. Oracle wallets store credentials for connecting to Oracle databases. An Oracle wallet is not required on the client if you're not using SSL authentication.

The basic steps for setting up one of these SSL connections are:

1. Configure Oracle wallets:
  1. Create and configure a server wallet.
  2. Create and configure a client wallet.
  3. Enable the auto-login feature.
2. Enable SSL and configure settings on the Oracle database server:
  1. Set the server's auto-login Oracle wallet location in the **sqlnet.ora** and **listener.ora** files.
3. Configure DB Connect:
  1. Copy JDBC drivers to the appropriate DB Connect directory.
  2. Create a database connection using the correct JDBC URL.
  3. Specify the wallet location in your connection's stanza within the `db_connections.conf` file or add a connection property from the user interface. The property name should be `oracle.net.wallet_location` and the value should point to the wallet directory.

## Configure Oracle wallets

First, create and configure standard wallets for both the server and client using Oracle Wallet Manager, and enable the auto-login feature. Do one of the following:

- Follow the instructions in 9.3 How to Create a Complete Wallet: Process Overview, within Chapter 9, Using Oracle Wallet Manager, in the *Oracle Database Advanced Security Administrator's Guide*.
- To use the Wallet Manager's `orapki` command-line utility instead of the Wallet Manager UI, see Appendix F, "orapki Utility."

Make a note of the location of your Oracle wallets, particularly if you changed the default location.

After following the appropriate set of Oracle instructions, you will have created your Oracle wallets, imported the appropriate certificates into your wallets, and set the auto-login feature for your wallets. Be sure that, if you are setting up either one-way or two-way authentication, you have a wallet on both the DB Connect machine and the Oracle server. The DB Connect machine's wallet should contain the server wallet's CA.

### ***Enable SSL and configure settings on the Oracle database server***

Next, enable and configure SSL on the Oracle database server by adding the path to your server wallet to the **sqlnet.ora** and **listener.ora** files:

1. On your Oracle database server, navigate to the **.../network/admin** directory. For example:  
**/u01/app/oracle/product/11.2.0/dbhome\_1/network/admin**
2. Edit the **sqlnet.ora** file and add the following, changing the **DIRECTORY** path to the path to the server wallet: **WALLET\_LOCATION= (SOURCE= (METHOD=FILE) (METHOD\_DATA= (DIRECTORY=/server/wallet/path/)))**
3. In the same file, set **SSL\_CLIENT\_AUTHENTICATION** to **FALSE** if you're either connecting with no SSL authentication or with one-way SSL authentication. Set **SSL\_CLIENT\_AUTHENTICATION** to **TRUE** if you're connecting with two-way SSL authentication. For more information, see *Configuring Secure Sockets Layer Authentication* in the *Oracle Database Advanced Security Administrator's Guide*.
4. In the same directory, edit the **listener.ora** file. Add the same wallet location and set **SSL\_CLIENT\_AUTHENTICATION** in the same way.
5. Save and close both files.

Set up the SSL connection using a cipher suite:

- For no SSL authentication, use an anonymous Diffie-Hellman key exchange.
- For one-way or two-way SSL authentication, use a cipher suite like RSA or standard DH.

### ***Configure DB Connect and enable an SSL connection on the client***

Now, configure DB Connect. Start by copying the JDBC drivers to the appropriate DB Connect directory:

1. On the Oracle server, navigate to **\$ORACLE\_HOME/jlib**.
2. Using your network or removable media, copy **ojdbc6.jar** file to the **\$SPLUNK\_HOME/etc/apps/splunk\_app\_db\_connect/drivers** directory  
(**%SPLUNK\_HOME%\etc\apps\splunk\_app\_db\_connect\drivers** on Windows hosts) on the machine running DB Connect, and then copy the following **.jar** files under **\$SPLUNK\_HOME/etc/apps/splunk\_app\_db\_connect/drivers/ojdbc6-libs** directory  
(**%SPLUNK\_HOME%\etc\apps\splunk\_app\_db\_connect\drivers\ojdbc6-libs** on Windows hosts)
  - ◆ **oraclepki.jar**
  - ◆ **ojpse.jar**
  - ◆ **osdt\_cert.jar**
  - ◆ **osdt\_core.jar**

### ***(Optional) Connect to Oracle RDS over SSL***

Use the following steps to connect to Oracle RDS over SSL.

1. Open a command line window, and enter the following information;

```
openssl x509 -outform der -in rds-ca-2019-root.pem -out rds-ca-2019-root.der
keytool -import -alias rds-root -keystore keystore/default.jks -file rds-ca-2019-root.der
keytool -list -v -keystore
<pre>
</li>
```

```
<li>Navigate to $SPLUNK_HOME/etc/apps/splunk_app_db_connect/default, and make a copy of
db_connection_types.conf.$SPLUNK_HOME/etc/apps/splunk_app_db_connect/local, and paste the copy
of db_connection_types.conf.$SPLUNK_HOME/etc/apps/splunk_app_db_connect/local, open
db_connection_types.conf using a text editor.

```

<li>For every connection type that you use, add or replace the following parameter:

```
<pre>
```

```
connection_properties =
```

```
{ "javax.net.ssl.trustStore": "<path>/splunk_app_db_connect/keystore/default.jks",
  "javax.net.ssl.trustStoreType": "JKS", "javax.net.ssl.trustStorePassword": "changeme" }
```

2. Restart your Splunk platform instance.

Next, set up [a new identity](#), and then a new connection:

1. Follow the instructions in [Create and manage database connections](#) to set up a new connection.
2. Use your server information to set up the connection, and be sure to select the **Enable SSL** checkbox.
3. When you get to the **JDBC URL Format** field, click the "Click here" link beneath the field to edit the JDBC URL.
4. In the **JDBC URL Format** field, paste the following:

```
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps) (HOST=servername)
(PORT=2484)) (CONNECT_DATA=(SERVICE_NAME=service_name)))
```

5. Save the connection.
6. Go to `$SPLUNK_HOME/etc/apps/splunk_app_db_connect/local`, and edit the `db_connections.conf` file.
7. Find the connection stanza with the name you gave it, and edit the **connection\_properties** argument.

Here is an example:

```
connection_properties =
```

```
{ "oracle.net.authentication_services": "(TCPS)", "oracle.net.ssl_cipher_suites": "SSL_DH_anon_WITH_3DES_EDE_CBC"
```

**connection\_properties** is in JSON format, it represents JDBC connection properties. The properties you defined here need to be consistent with what you configured in Database. See [JDBC with Oracle 12c Database](#) for more information.

8. Save and close the `db_connections.conf` file.

## Install and configure Splunk DB Connect on a single instance Splunk platform deployment

This topic shows you how to install and configure Splunk® DB Connect on a single instance (the indexer and Splunk Web both running on the same system). Ensure you meet all [prerequisites](#) before installing.

### Install the Splunk DB Connect

To install Splunk DB Connect, use Splunk Web:

1. Log in to Splunk Web and go to **Apps > Find More Apps**.
2. Use the search box to find `db connect`.
3. Click the green **Install** button next to **Splunk DB Connect**.
4. Click **Restart Splunk**.

You can also download the app package from Splunkbase and then install it offline:

1. Download Splunk DB Connect and save it to a temporary location that you can access from your Splunk Enterprise instance.

2. Log in to Splunk Web, go to **Apps > Manage Apps**, then click **Install app from file**.
3. Navigate to the package that you downloaded; **splunk\_app\_db\_connect-<version>.tgz** and click **Upload**.
4. Click **Restart Splunk**.

You can also install DB Connect by copying its directory into your Splunk Enterprise apps directory:

1. Download Splunk DB Connect and save it to a temporary location that you can access from your Splunk Enterprise instance.
2. Un-tar the download.
3. Move the **splunk\_app\_db\_connect** directory into **\$SPLUNK\_HOME/etc/apps**.
4. Restart Splunk Enterprise.

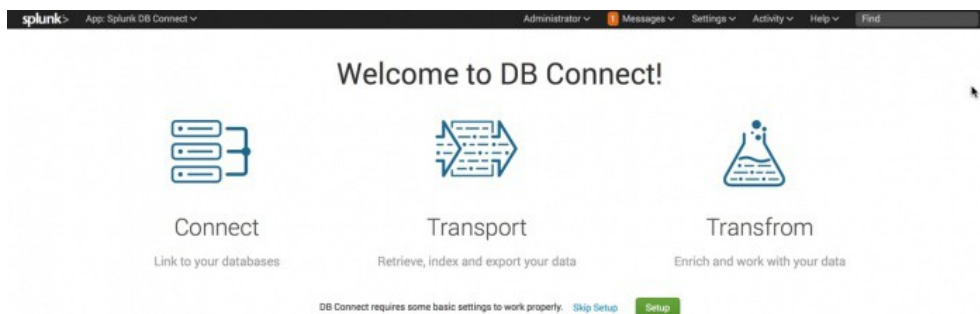
## Install database drivers

Before setting up Splunk DB Connect, install a JDBC driver for your database. See [Install database drivers](#). You can proceed without having first installed database drivers, but you need to do so before you can use DB Connect to connect to your database.

## Set up Splunk DB Connect

Before you can start using Splunk DB Connect, you need to set it up.

When you open Splunk DB Connect for the first time, you see the following screen:



Click **Setup** to access the general settings page. See [General Settings](#) for more detailed instruction on how to configure the settings of this page.

## Install and configure Splunk DB Connect on a Splunk Enterprise On-Premise distributed platform deployment

To use Splunk DB Connect in a distributed search environment, including **search head clusters**, you must install the app on search heads and heavy forwarders.

- In a distributed environment, the heavy forwarder needs to search your deployment's indexers in order to output to the DB. See Deploy a distributed search environment in the *Distributed Search* manual to learn how to set up distributed search on your deployment's heavy forwarders.
- DB Connect 3.x.x does not support resource pooling. See [migrate DB Connect 2.x.x to DB Connect 3.x.x on distributed deployment](#).

- DB Connect is incompatible with deployment server. Do not attempt to distribute DB Connect using a deployment server.
- DB Connect is incompatible with search head pooling, which Splunk no longer supports as of Splunk Enterprise 6.2.0.

## Deployment topologies

Design your deployment based on [architecture and performance considerations](#). This list specifies the typical deployment topologies in which you can install Splunk DB Connect. In all cases, Splunk best practice is to install DB Connect on a dedicated search head.

- Single search head, multiple indexers, load-balanced forwarders
- Multiple search heads, multiple indexers, load-balanced forwarders
- Indexer cluster, single search head
- Search head cluster, multiple independent indexers, load-balanced forwarders

For general information about configuring the topology components described in this section, see Distributed Splunk Enterprise overview, or any of the following topics:

- Single search head: Deploy a distributed search environment, Add search peers to the search head
- Search head clusters: About search head clustering, Search head clustering architecture
- Multiple indexers: Indexers in a distributed deployment
- Indexer clusters: About indexer clusters and index replication, The basics of indexer cluster architecture
- Load-balanced forwarders: About forwarding and receiving, Set up load balancing

## Deploy DB Connect on search head clusters

You can deploy Splunk DB Connect in a search head clustering environment. To install, use the deployer to distribute DB Connect to all of the search head cluster members. Be aware that you must use the cluster deployer, not Deployment Server, to distribute DB Connect to search head cluster members.

For more information about configuring search head clusters, see [Configure the search head cluster](#).

1. If you have not already done so, deploy and configure a search head cluster.
2. Install the database drivers for the databases you want to connect to with DB Connect. Access the instructions on the [Install database drivers](#) for details.
3. Install DB Connect on the deployer. Access the instructions on the [Single server deployment](#) for details.
4. Set up [identities](#) and [connections](#) for your databases.
5. Copy the **splunk\_app\_db\_connect** directory from **\$SPLUNK\_HOME/etc/apps/** to the **\$SPLUNK\_HOME/etc/shcluster/apps/** directory on the deployer. This includes all custom configuration files as well as JDBC drivers. You can't replicate the `kerberos_client.conf` and `identity.dat` files to other SHC nodes after making changes. You need to copy the files manually to other SHC nodes.
6. Deploy the configuration bundle by running the `splunk apply shcluster-bundle` command on the deployer: `splunk apply shcluster-bundle -target <URI>:<management_port> -auth <username>:<password>`
  - ◊ The `-target` parameter specifies the URI and management port for any member of the cluster, for example, `https://10.0.1.14:8089`. You select only one cluster member but the deployer pushes to all members. You must enter this parameter.
  - ◊ The `-auth` parameter specifies credentials for the deployer instance.
7. The deployer displays a message that asks you to acknowledge that the cluster members might restart. Select **Y** to acknowledge.

For more information about deploying a configuration bundle, see [Deploy a configuration bundle](#).

For full instructions about how to use the deployer to distribute apps, add-ons, and their configuration bundles, see [Use the deployer to distribute apps and configuration updates](#).

When you use DB Connect in a search head clustering (SHC) environment, use the deployer to push configuration changes to SHC members. If you prefer to use the DB Connect UI or modify `.conf` files and then replicate configuration to SHC members, restart Splunk Enterprise on SHC members after you have updated them with the new configuration. There are three reasons why you must restart SHC members after updating their configuration:

- When you make a configuration change on a search head, such as a change to the RPC server port, Splunk Enterprise replicates changes to the SHC members automatically. However, the SHC members might still use the old configuration until you restart them.
- Splunk Enterprise automatically replicates SHC for changes you make to most of the DB Connect-specific settings and objects through the REST API. Splunk Enterprise does not automatically replicate changes you make by editing `.conf` files on a search head. To ensure that Splunk Enterprise replicates all your changes, and to replicate any changes you made by editing `.conf` files, you must restart the search head on which you made the change.
- Splunk Enterprise does not automatically replicate changes you make by editing `kerberos_client.conf` and `identity.dat` files. You need to manually replicate the files to other SHC nodes.

## A note about indexes

When you [create a database input](#), you must select the index you want to index the data your database receives. When you select an index, by default you must select one of the indexes on that instance of Splunk Enterprise. This means that you cannot select an index that you have configured on a search peer but not distributed to the rest of the deployment.

To select an index that you have not configured on, for example, a forwarder or search head that is running DB Connect, you can create or edit an `indexes.conf` file, and then distribute it using Deployment Server. Although you cannot distribute DB Connect configuration using a Deployment Server, you can distribute `indexes.conf` files.

To configure peer indexes in a distributed deployment, follow the instructions in [Configure the peer indexes in an indexer cluster](#). First, you edit the `indexes.conf` file, and then you distribute it to peers. This practice ensures that you configure search heads and forwarders to send all logs to the indexer tier, which prevents this distribution of `indexes.conf` from causing Splunk Enterprise to create local indexes on search heads and forwarders.

Once you have distributed the configuration, applications like DB Connect know which indexes exist to validate configuration.

## Install and configure Splunk DB Connect on Splunk Cloud Platform

### Deploy DB Connect to Splunk Cloud Platform Victoria Experience

If you want to deploy DB Connect to Victoria Experience, see [Install an add-on for Splunk Cloud Platform](#) for details. You can only deploy DB Connect on Victoria Experience using Search Head instances. Search Head instances on Victoria Experience already have a supported version of JRE installed. If you install DB connect on a single instance then you can use full functionality on search heads. If you install DB Connect on search head cluster environments, consider limitation for setting up DB connect on distributed environment describer in [Performance considerations in distributed environments](#). Start the DB Connect installation process:

1. Splunk leaves the outbound ports closed by default on Victoria Experience. Splunk Cloud Platform provides the Admin Config Service (ACS) API which you can use to open ports to allow outbound network connections to specific IP subnets programmatically. For details on how to use the ACS API see [ACS API Documentation](#)
2. Follow the instructions to install DB Connect on a Search Head
3. Install a JDBC driver add-on for your database. See [new drivers using JDBC Driver addons](#)[JDBC driver add-ons](#)[JDBC driver add-ons](#). If an add-on is not available for your database, or you require a different version of the driver than provided by the add-on, contact Splunk Support for help.
4. After installing DB Connect and restarting Splunk, open DB Connect.
5. Create a database identity and set up a database connection.
6. Create a new [database input](#) and [use it as a data input](#) in a Splunk search.

## Migrate DB connect from Splunk Enterprise to Splunk Cloud Platform Victoria Experience

DB Connect has differences in managing it on Splunk Cloud Platform Victoria Experience. This list presents those differences:

- Installing JDBC drivers other than served by JDBC Add-ons is not possible on Splunk Cloud Platform Victoria Experience. In case of necessity of having other drivers idea need to be create on Idea Portal
- Migrate from Single Instance of Splunk Enterprise to Search Head Cluster environment in Victoria Experience consider limitation described in [Performance considerations in distributed environments](#).
- Splunk leaves the outbound ports on Splunk Cloud Platform Victoria Experience closed by default. It might prevent connecting to your database from Search Head with DB Connect installed. Follow the instructions to open the Outbound port from Search Head to your Database. Splunk Cloud Platform provides the Admin Config Service (ACS) API which you can use to open ports to allow outbound network connections to specific IP subnets programmatically. For details on how to use the ACS API see [ACS API Documentation](#).

## Deploy DB Connect to Splunk Cloud Platform Classic Experience

If you want to deploy DB Connect to Splunk Cloud Platform, contact Splunk Support for guidance and assistance. You cannot deploy DB Connect yourself because you cannot configure network access to databases on your Splunk Cloud Platform instance. See [Install an add-on in Splunk Cloud Platform](#) for details.

## Check DB Connect installation health

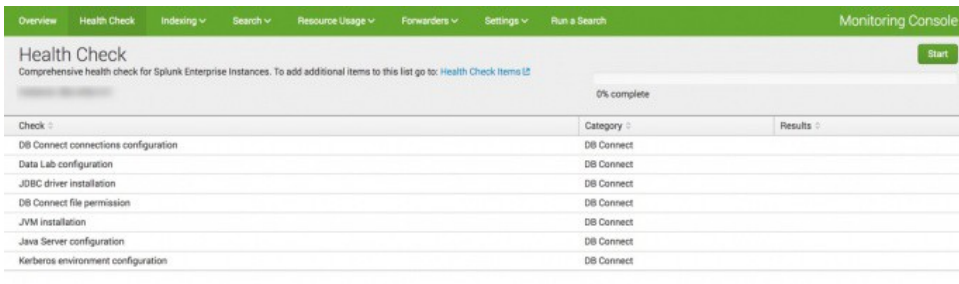
Beginning with version 3.1.0, DB Connect provides health checks for the Splunk Access and customize health check to help you diagnose common issues.

### Note:

- ◆ The Monitoring Console is only visible to users with the Splunk Enterprise admin role.
- ◆ The health check feature is available with Splunk version 6.5 and higher.

To use the health check, go to **Settings** of your Splunk instance and select **Monitoring console**. On the monitoring console page, select **Health Check** tab. There are seven configured health check items provided by DB Connect. You can check the health of these items by selecting **DB Connect** and clicking **Start**, or leave the filters blank to run all available health checks.





The Splunk DB Connect health check items are:

- **DB Connect connection configuration.**  
Checks that database connection configurations (identities and connections) are syntactically correct and that you can resolve all dependencies between stanzas.
- **Data lab configuration.**  
Checks that data access configurations (inputs, outputs, lookups) are syntactically correct and that you can resolve all dependencies between stanzas.

**Note:**

◇ Configuration entries with incorrect syntax don't display in the user interface, and you must correct them using configuration files.

- **JDBC driver installation.**  
Checks that connections configured in DB Connect have the corresponding JDBC driver and version installed correctly.
- **DB Connect file permission.**  
Checks the existence and permission settings of the folders that the DB Connect processes needs to read and write: DB Connect application folder, checkpoint folder and log folder.
- **JVM installation.**  
Checks the existence and version of the JVM that you configured DB Connect to use to run individual commands and the Task Server.
- **Java server configuration.**  
Checks the JVM bootstrap conditions of the DB Connect Task Server.
- **Kerberos environment configuration.**  
Checks the configuration file for Kerberos is correctly configured to support Microsoft Active Directory authentication. This item is only performed on Linux servers when at least one database connection uses Kerberos.

After the health check process is complete, the result summary (how many succeeds, how many errors or warning messages) displays on the page. You can expand each health check item to see the detailed result.

**Example 1:** After checking all the health check items, all the health check items are successful.

The screenshot shows the 'Health Check' section of the Splunk Monitoring Console. At the top, there's a navigation bar with links like Overview, Health Check, Indexing, Search, Resource Usage, Forwarders, Settings, and Run a Search. The 'Health Check' section has a 'Start Over' button and a completion status: 'Completed 6/16/2017, 10:48:43 AM'.

Below the status bar, there's a summary of health check results:

ALL	ERROR	WARNING	INFO	SUCCESS	N/A
7	0	0	0	7	0

A table lists the health check items:

Check	Category	Results
JDBC driver installation	DB Connect	✓ This health check item was successful. Everything is good here.
DB Connect file permission	DB Connect	✓ This health check item was successful. Everything is good here.
JVM installation	DB Connect	✓ This health check item was successful. Everything is good here.
Java Server configuration	DB Connect	✓ This health check item was successful. Everything is good here.
Kerberos environment configuration	DB Connect	✓ This health check item was successful. Everything is good here.
DB Connect connections configuration	DB Connect	✓ This health check item was successful. Everything is good here.
Data Lab configuration	DB Connect	✓ This health check item was successful. Everything is good here.

The 'JDBC driver installation' item is expanded, showing a description, message, and a results table.

**Description:** It checks whether the connections configured in DB Connect have the corresponding JDBC driver with correct version installed.

**Message:** This health check item was successful. Everything is good here.

**Results (1):**

severity	instance	details
✓		

**Example 2:** After checking all the DB Connect health check items, there are some issues in **Data lab configuration** and **JDBC driver installation**.

You can expand the health check item **JDBC driver installation**, it shows the error messages and suggested action provided by Splunk. You can also view the details of the issue in the **Result** table.

This screenshot shows a more detailed view of the 'JDBC driver installation' health check item. The summary bar now shows 2 errors and 6 successes.

ALL	ERROR	WARNING	INFO	SUCCESS	N/A
7	2	0	0	6	0

The table of health check items is updated:

Check	Category	Type	Results
Data Lab configuration	DB Connect	configuration	✗ One or more configuration values are invalid. Troubleshoot after DB Connect installation.
JDBC driver installation	DB Connect	installation	✗ One or more defined connections require the corresponding JDBC driver.
DB Connect connections configuration	DB Connect	configuration	✓ Troubleshoot item was successful. Everything is good here.
DB Connect file permission	DB Connect	configuration, installation	✓ Troubleshoot item was successful. Everything is good here.
JVM installation	DB Connect	configuration, installation	✓ Troubleshoot item was successful. Everything is good here.
Java Server configuration	DB Connect	configuration, installation	✓ Troubleshoot item was successful. Everything is good here.
Kerberos environment configuration	DB Connect	configuration, installation, operating system	✓ Troubleshoot item was successful. Everything is good here.

The 'JDBC driver installation' item is expanded, showing a description, message, and a results table with error details.

**Description:** It checks whether the connections configured in DB Connect have the corresponding JDBC driver with correct version installed.

**Message:** One or more defined connections require the corresponding JDBC driver.

**Suggested Action:** Install missing drivers by following the documentation instructions. Once installed, restart the Splunk service for the changes to take effect.

**Link:** [Learn more about DB Connect JDBC driver installation](#)

**Results (2):**

severity	instance	details
✗	db1	One or more invalid connections require the corresponding JDBC driver. Check whether the JDBC connection is valid, whether the user has the correct connection privileges, whether the user has the correct connection permissions.

Besides installation health check, DB Connect also provides pre-built panels for you to monitor DB Connect input health, input performance and the connection health with the database. See [Monitor database connection health](#) for details.

# Configure and manage Splunk DB Connect

## Configure Splunk DB Connect settings

Read this to set up DB Connect before you use it to access databases.

### General tab

1. Access **Configuration > Settings**.
2. The **General Settings** tab contains settings related to your Java Runtime Environment (JRE) and Task Server. Change any settings you want. When DB Connect 3.x prompts you to input the **JRE Installation path**, Make sure to input the complete JRE file path. See [Prerequisites](#) for further details.
3. Select **Save** to restart the Task Server's Java process. You do not need to restart Splunk Enterprise for changes on this page to take effect.

### JRE Installation Path (JAVA\_HOME)

DB Connect attempts to detect the JAVA\_HOME environment variables as the JRE installation path if possible. You can change it to the Java home path you want to use for DB Connect.

### JVM Options

This field lists Java Virtual Machine parameters. For more information about available JVM parameters, access Oracle's JVM documentation.

DB Connect saves the options in this field in `$SPLUNK_HOME/etc/apps/splunk_app_db_connect/jars/server.vmopts`.

### Task Server Port

This field contains the port number of the task server. DB Connect uses an RPC server to manage communications with the Java subsystem. The default port is 9998, but you can use any unassigned, unused port on your system.

### Drivers tab

This tab contains a list of supported database connection types, along with install status and version number information.

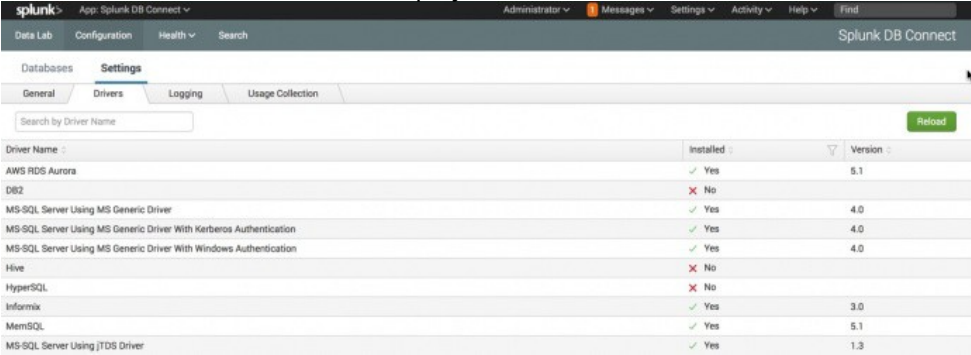
If there is no JDBC driver for a connection type, the **Installed** column shows an **X** icon and the word "No". By default, there are no drivers.

1. To install a JDBC driver, follow the instructions in "[Install database drivers](#)".
2. Once you have moved the appropriate JAR file to the `$SPLUNK_HOME/etc/apps/splunk_app_db_connect/drivers` directory, select the **Reload** button.

If you have installed a JDBC driver and it still does not register:

- Verify that you correctly installed the driver by repeating the steps in "[Install database drivers](#)".
- Access the "[Supported databases matrix](#)" to verify that DB Connect supports the driver and its version. If necessary, download a newer version of the driver.
- Follow the applicable steps in "[Troubleshoot driver connections](#)".

When DB Connect detects a driver, it displays a green checkmark icon and the word "Yes" next to the database, as shown in the screenshot. It also displays the version information of the driver.



The screenshot shows the 'Drivers' tab in the Splunk DB Connect settings. It features a search bar and a 'Reload' button. Below is a table listing various database drivers with their installation status and version.

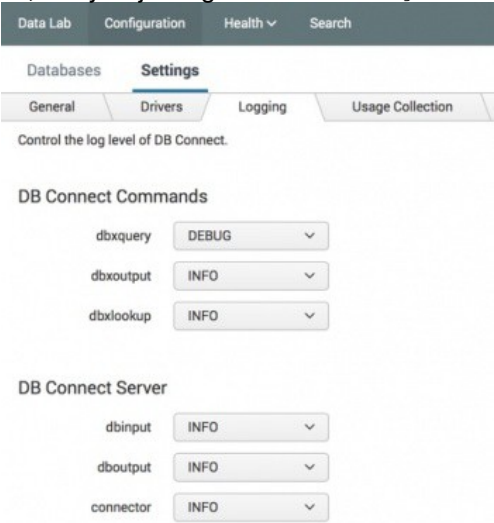
Driver Name	Installed	Version
AWS RDS Aurora	✓ Yes	5.1
DB2	✗ No	
MS-SQL Server Using MS Generic Driver	✓ Yes	4.0
MS-SQL Server Using MS Generic Driver With Kerberos Authentication	✓ Yes	4.0
MS-SQL Server Using MS Generic Driver With Windows Authentication	✓ Yes	4.0
Hive	✗ No	
HyperSQL	✗ No	
Informix	✓ Yes	3.0
MemSQL	✓ Yes	5.1
MS-SQL Server Using JTDs Driver	✓ Yes	1.3

## Logging levels

Versions 3.0.x and higher of Splunk DB Connect provides graphical configurations of the logging levels of DB Connect. DB Connect logs activity to files in `$SPLUNK_HOME/var/log/splunk` and automatically indexes to `_internal`. The relevant log files for DB Connect are:

- `splunk_app_db_connect_server.log`
- `splunk_app_db_connect_job_metrics.log`
- `splunk_app_db_connect_dbx.log`
- `splunk_app_db_connect_audit_server.log`

By default, DB Connect logs all SQL queries it executes at the INFO level. You can enable other logging levels using the UI, or by adjusting the `dbx_settings.conf` file at `splunk/etc/apps/splunk_app_db_connect/default/dbx_settings.conf`.



The screenshot shows the 'Logging' tab in the Splunk DB Connect settings. It contains two sections: 'DB Connect Commands' and 'DB Connect Server', each with three dropdown menus for log levels.

DB Connect Commands	
dbxquery	DEBUG
dbxoutput	INFO
dbxlookup	INFO

DB Connect Server	
dbinput	INFO
dboutput	INFO
connector	INFO

## Keystore tab

This tab contains a text input for setting a new keystore password. Only Splunk admin and DBX admin can run this action. The password must be at least 6 characters long.

## Usage Collection tab

This tab contains an option to grant permission for Splunk to collect statistics about how you use DB Connect. See [sending usage data to Splunk DB Connect](#) to learn more about the data that DB Connect sends to Splunk.

## Configure remote HTTP Event Collector (HEC)

Improve data ingestion performance by configuring a remote HTTP Event Collector (HEC). By default, Splunk DB Connect ingests data through a local HTTP Event Collection (HEC). Remote HEC can ingest data to remote forwarders, indexers, and indexer clusters, or through HEC directly. To configure remote HEC, run the following steps.

1. Open a command line interface window, and navigate to  
`$SPLUNK_HOME/etc/apps/splunk_app_db_connect/local/`.
2. Open `dbx_settings.conf` in a text editor. If `dbx_settings.conf` does not yet exist, navigate to  
`$SPLUNK_HOME/etc/apps/splunk_app_db_connect/default/`, copy the `dbx_settings.conf` file, and paste it in  
`$SPLUNK_HOME/etc/apps/splunk_app_db_connect/local/`.
3. In `dbx_settings.conf`, edit the following stanzas:  
hecUri: A list of HEC servers/ports, separated by comma.  
hecToken: A HEC token listing, when it is the same HEC token, configured on multiple indexers.

By default, these values are empty, and Splunk DB Connect ingests data thru your local HEC.

4. Save your changes.
5. Restart your Splunk platform instance.

If an error takes place, the scheduler performs a round robin of all HEC URIs, and Splunk DB Connect marks the current HEC URI as unavailable for 1 minute (not configurable), before trying the next available HEC URI.

## Configure Splunk DB Connect security and access controls

The role-based user access controls in Splunk Enterprise enable you to set up access permissions for identity and connection objects, as well as capabilities for inputs, outputs, and lookups in Splunk DB Connect. You can grant a user global access to all DB Connect features and available database connections, or limit a user's access to only specific database connections or capabilities.

Before using DB Connect, the logged-in user must have the ability to write to the `$SPLUNK_HOME/var` directory (`%SPLUNK_HOME%\var` on Windows hosts) and to `$SPLUNK_HOME/etc/apps/splunk_app_db_connect` (`$SPLUNK_HOME/etc/apps/splunk_app_db_connect` on Windows hosts) and its sub-directories. For more information, see [Use access control to secure Splunk data](#).

## Roles

When thinking about permissions in DB Connect, familiarize yourself with the Splunk Enterprise role-based user access system. You can create Splunk Enterprise users with passwords and assign them to roles. Roles determine the access

and permissions of any user you assign to that role, including whether a user can access DB Connect. Roles determine users' capabilities for many Splunk Enterprise tasks, including DB Connect tasks.

When you install DB Connect, it adds two new roles to Splunk Enterprise: **db\_connect\_admin** and **db\_connect\_user**. In addition, Splunk Enterprise gives DB Connect capabilities to its **admin** role.

- The **db\_connect\_user** role has 11 capabilities and inherits from the **user** role. Its capabilities involve reading DB Connect objects.
- The **db\_connect\_admin** role has 31 capabilities and inherits from the **db\_connect\_user** role. Its capabilities involve reading and writing DB Connect objects, plus actions involving the task server.
- The existing Splunk Enterprise **admin** role automatically gets all DB Connect-specific capabilities when you install DB Connect.

To set or view the capabilities of a specific role from within Splunk Enterprise, go to **Settings > Access controls > Roles**. From here you can also create new roles with specific capabilities.

## Permissions

You set permissions when you define the DB Connect **identities** and **connections**. An identity object contains encrypted database credentials. A connection object contains the necessary information for connecting to a remote database.

Use the **Permissions** table on the **New Identity** and **New Connection** setup pages to enter the Splunk Enterprise roles that have access to the identity or connection.

- **Read access** to an object means that Splunk Enterprise roles can use the object.
- **Write access** to an object means that Splunk Enterprise roles can use and modify the object.

By default, the Splunk Enterprise "admin" and "db\_connect\_admin" roles have read-write access to objects such as identities or connections, the "db\_connect\_user" role has read access. All other roles have no access.

The permissions you set when you create a new identity or a new connection associate with Splunk Enterprise roles, and are not related to database permissions. Because identities store encrypted database credentials, the level of database access that an identity has is directly related to the user account that you store in the identity. When you use an identity to connect to your database, Splunk limits your access to that database to the database access level of the user you have stored in that identity.

For example, if you added the credentials of *user1* to *identity1*, and *user1* only has access to table *abc* on the database and not to table *xyz*, a connection that uses *identity1* *only has access to table abc, and not xyz, on the database. That means that any database input, output, or lookup using that connection also has access to only table abc on that database.*

Therefore, you must consider which database credentials to store within a DB Connect identity. You might want to create DB Connect-specific users on the database who have access to only the database tables you want to access with Splunk Enterprise, and then assign those users to your identities.

## Capabilities

Every role in Splunk Enterprise has capabilities. In DB Connect, permissions define access for identities and connections, but capabilities define access for DB Connect-specific modular inputs, which include inputs, outputs, and lookups.

By defining the capabilities a role has, you define what that role can do with the inputs, outputs, and lookups you have created. For instance, the **db\_connect\_user** role can use inputs, outputs, and lookups, but the **db\_connect\_admin** role includes the additional capabilities that enable it to edit inputs, outputs, and lookups (specifically, "edit\_modinput\_mi\_input," "edit\_modinput\_mi\_output," and "edit\_modinput\_mi\_lookup").

## Read-only connections

Splunk DB Connect supports the **readonly** JDBC attribute which can restrict connections to read-only mode. This setting alone does not guarantee that users connecting using DB Connect can't write to the database.

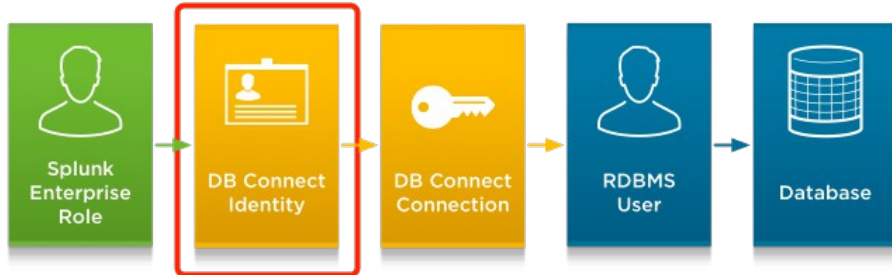
When you create a new [connection](#), select the **readonly** check box to restrict users to using `SELECT` statements with the database. The **readonly** check box cannot always guarantee read-only access; the database driver is what ultimately allows or prevents changes.

If you use the read-only option, ensure that, on the database itself, you limit the user to read-only access. To determine if your database supports read-only access, see [Supported databases](#) or check your database vendor's documentation.

## Create and manage identities

An **identity** object contains database credentials. It comprises the username and password that Splunk DB Connect uses to access your database.

Note that your database credentials are not the same as your Splunk Enterprise credentials. When you configure a DB Connect identity, you use the Splunk Enterprise role-based access control system to define access to the identity.



## Create a Basic identity

1. From within Splunk DB Connect, access the **Configuration > Databases > Identities** tab and click **New Identity > Basic Identity**.
2. Complete the following fields:
  - ♦ **Identity Name**
  - ♦ **Username**: Enter the name of the database user you want to connect as.

**Note:** Ensure that the database user has sufficient access to the data you want to search. For example, you might create a database user account whose access is limited to the data you want Splunk Enterprise to consume.

- ♦ **Password**: Enter the password for the user you entered in the **Username** field.

**Note:** Your password should only contain US-ASCII characters. DB Connect requires this

field to connect to your database. Your password is encrypted.

- ◆ **Use Windows Authentication Domain:** This setting is for identities that connect to Microsoft SQL Server databases. Enable this setting if you need to specify a Windows Authentication Domain.
- ◆ **Windows Authentication Domain:** If you selected the **Use Windows Authentication Domain** checkbox, enter the domain in this field. For more information about connecting to Microsoft SQL Server databases using Windows Authentication, see "[Microsoft SQL Server](#)."

3. In the **Permissions** tab, update the Splunk Enterprise permissions for this identity. For more information, see [Permissions](#).
4. Click **Save**.

## Create a CyberArk identity

You can create a new identity in DB Connect that uses your existing CyberArk account information.

From within Splunk DB Connect, access the **Configuration > Databases > Identities** tab and click **New Identity > CyberArk Identity**. Complete the following fields:

- **Identity Name**
- **Username:** Enter the name of the database user you want to connect as.
- **Connection Type:** Select HTTP or HTTPS.
- **URL:** Enter the CyberArk URL.

**Note:** Confirm the URL does not contain the connection type or the port number. It must specify only the domain portion, for example *my-cyberark-vault.com*.

- **Port:** Enter the CyberArk port.
- **Certificate:** This is the public certificate text.
- **AppID:** Enter the ID you created for DB Connect. This is used by Cyberark to identify who accessed your information.
- **Safe:** Where you store your CyberArk password.
- **Object:** The name of the password object.
- **Synchronization Frequency:** How often you want the password updated, in seconds. This keeps your identity information in sync.

## Create a HashiCorp identity

You can create a new identity in DB Connect that uses your existing HashiCorp Vault by utilizing the Key/Value and Databases secrets engine.

From Splunk DB Connect, access the **Configuration > Databases > Identities** tab and click **New Identity > HashiCorp Identity**. Complete the following fields:

- **Identity Name**
- **Username:** Enter the name of the database user you want to connect as.
- **Connection Type:** Select HTTP or HTTPS.
- **URL:** Enter the HashiCorp URL.

**Note:** Confirm the URL does not contain the connection type or the port number. It must specify only the domain part, for example *my-hashicorp-vault.com*.

- **Port:** Enter the HashiCorp port.
- **AppRole Authentication Method Path:** AppRole is the authentication method supported by DB Connect. In this field you must specify the path.
- **Role ID:** The role id for the role you have configured.



**Note:** Confirm the role you have configured has access to the secrets by adding ACL policies.

- **Secret ID:** The secret id obtained for the role id.

**Note:** Ensure that the secret id obtained has enough TTL to avoid failures during sync processes.

- **Secrets Engine Path:** The secret engine path for the specific engine you have configured.
- **Secrets Engine:** The secret engine you have configured.

**Note:** DB Connect supports Key/Value Version 1, Key/Value Version 2 and Database engines. Database engines only support Static Roles.

- **Secret Path:** The secret path in case you have selected Key/Value Version 1 or Key/Value Version 2 engine.
- **Key Name:** The key name in case you have selected Key/Value Version 1 or Key/Value Version 2 engine.
- **Role Name:** The role name in case you selected the Database engine.
- **Synchronization Frequency:** How often you want the password updated, in seconds. This keeps your identity information in sync.

## Edit Identities

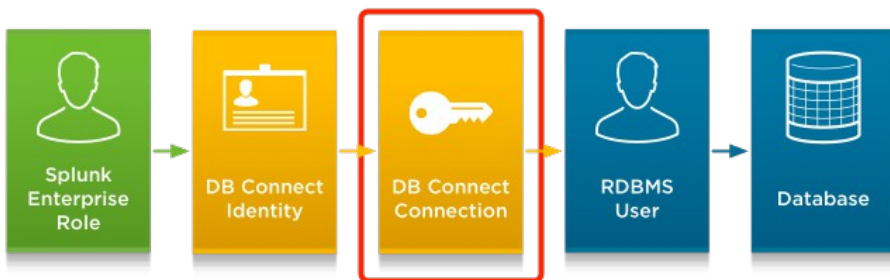
To see a list of the defined identities, go to **Configuration > Databases > Identities**. To see a list of identities, reference the table below.

To edit an identity, click its name. You can edit the following attributes of an identity, except where noted:

- **Status:** Disable an identity by clicking **Disable**. You cannot disable an identity if any connections are using it. In that case, DB Connect displays an error message that reads *Can not disable this identity!*.
- **Identity Name:** Not editable. To change the name of an identity, clone it, give the clone the name you want, and then delete the original identity.
- **Username, Password, Use Windows Authentication Domain?** checkbox and **Windows Authentication Domain** are the same as [Create an identity](#).
- **Permission:** The **Permissions** table is where you specify the Splunk Enterprise roles that have read, write, or no permissions to the identity. By default, the Splunk Enterprise **admin** and **db\_connect\_admin** roles have read-write access to a new identity, the **db\_connect\_user** role has read access, and all other roles have no access.
  - ♦ Read access means that Splunk Enterprise roles are able to use the identity.
  - ♦ Write access means that Splunk Enterprise roles are able to use and modify the identity.

## Create and manage database connections

A database **connection** object contains the necessary information for connecting to a remote database.



## Create a database connection

To create a new connection:

1. From within Splunk DB Connect, select the **Configuration > Databases > Connections** tab.
2. select **New Connection**.

**Note:** If you have not yet created an identity, the New Identity window appears. [Create a new identity](#), and then you can create a new connection.

3. On the New Connection page, complete the following fields:

- **Connection name**
- **Identity:** Select the identity you want to use with this connection.
- **Connection Type:** A list of supported databases. Select the type of database to which you're connecting.

**Note:** This list contains all supported databases, regardless of whether you installed their drivers. You must first [install the database driver](#) for the database type you want to use. You can't select a database from this list without first installing the corresponding driver. For information about the Microsoft SQL Server options listed here, see ["Microsoft SQL Server"](#).

- **Timezone:** If you want to convert data of date and time types read from the database into the Splunk server's local time zone, enter the source database time zone for the conversion. When you set the time zone, you have two time zone conversion options depending on the **localTimezoneConversionEnabled** setting in

`db_connections.conf`:

- ♦ When you don't define **localTimezoneConversionEnabled** or set it to false, the system only converts index time values read from table columns in the source database into the local time zone. This conversion option only applies to inputs.
- ♦ When you set **localTimezoneConversionEnabled** to true, the system converts all date and time data type (datetime, time, date, timestamp) values into the local time zone except source datetime values that already include time zone information (For example, `TIMESTAMP WITH TIMEZONE`). This conversion option applies to inputs, dbxqueries, and dblookups.

If you leave the **Timezone** field blank, the system assumes that the source database time zone and Splunk server's local time zone are the same and reads the date and time data as is.

### Example

	Timestamp in database	Database time zone	Splunk Server (JVM) time zone	Indexed timestamp value
Without timezone conversion	2006-02-15 04:34:33	UTC	Asia/Shanghai (GMT+8)	2006-02-15 04:34:33 in Asia/Shanghai time zone
With timezone conversion	2006-02-15 04:34:33	UTC	Asia/Shanghai (GMT+8)	2006-02-15 12:34:33 in Asia/Shanghai time zone

### Note:

- ♦ If the column in your database contains timezone information (For example, for the column with a `TIMESTAMP WITH TIMEZONE` type), Splunk ignores the timezone you set here.
- ♦ The timezone setting defaults to JVM time zone. If there are no JVM time zone settings, Splunk uses the time zone of your operating system.
- ♦ For MySQL database driver, you need to set `useLegacyDatetimeCode` setting to false in the JDBC URL if you want to use the timezone setting.

## JDBC URL Settings

- **Host:** Enter the address, or host, of the database.
- **Port:** (Optional.) Enter the port number of the database. You don't need to enter a port number here if your database is using its default port.
- **Default Database:** Enter the default database or catalog name for the database type you chose. The usage and meaning of this parameter varies between database vendors, so check your database vendor's documentation. For more information, see [supported databases matrix](#).
- **Enable SSL:** Select this check box to enable Secure Sockets Layer (SSL) encryption for the connection. SSL support is not available for all connection types. For further information, see [supported databases matrix](#) and [Enable SSL for your database connection](#). To find out how to connect to Oracle using SSL for encryption, see [Connect to Oracle using SSL \(for encryption only\)](#).
- **Certificate:** When you select the **Enable SSL** check box, the **Certificate** field appears for the MySQL, MSSQL and Oracle connection types. You can pass a certificate which Splunk automatically adds to the Java TrustStore and uses it for the server authentication. If you select the **Enable SSL** check box but leave the certificate field empty, then DB Connect ignores the certificate and functions as before.
- **JDBC URL Preview** A Java Database Connectivity (JDBC) Uniform Resource Locator (URL) is a URL that encodes all the necessary information for connecting to your database. The **JDBC URL Preview** field is not editable by default, but Splunk populates this field, using placeholder values, with the correct URL format according to the database type you have chosen.
- **Connection properties** Splunk uses the provided properties during the creation of the connection. Splunk provides properties either here or through the editing of the JDBC URL. You can find a list of all valid driver connection properties in the official database documentation. Splunk ignores incorrect properties.
- You can manually edit the URL format by selecting **Edit JDBC URL**. When you select this field, the **JDBC URL Preview** becomes editable. This is useful if you need to add customized JDBC URL parameters to the JDBC URL, but that is not Splunk best practice unless you already know what customizations you want to add. Connection parameters vary among JDBC drivers. Consult your database vendor's documentation for a list of supported parameters and values.

**Note:** DB Connect supports LDAP connection, you need to select the LDAP URL in **JDBC URL** field. Consult your database vendor's documentation on how to generate LDAP URL.

## Advanced Settings

- **Fetch Size:** (Optional.) Enter the number of rows to return at a time from the database. If you leave this field blank, it defaults to 300.
- **Readonly:** Select this check box to indicate your intention for users to only use `SELECT` statements with the database. Be aware that this cannot always guarantee read-only access. DB Connect tries to ensure that it's read-only, but it is the database driver that ultimately allows or prevents changes. If you intend to use the read-only option, ensure that, on the database itself, the user you're connecting as only has read-only access. See [Read-only connections](#) for more details.
- In the **Permissions** table, update the Splunk Enterprise permissions for this database connection. For more information, see [Permissions](#).
- Select **Save** to save the connection.

**Note:** If the connection you create is valid, you can save the connection successfully. Otherwise Splunk displays an error message for you to check the configuration of the connection and save again.

## Edit connections

Go to **Configuration > Databases > Connections** to see a list of the defined database connections.

**Note:** The list of connections that you can see depends on the permissions set on each connection. For more information,

see [Permissions](#).

### Connection actions

You can make changes to a connection using the following buttons on the connection page under the **Edit** tab:

- **Disable/Enable:** Disable/Enable a connection by selecting **Disable/Enable** here. You cannot disable a connection if any inputs, outputs, or lookups are using it. In that case, Splunk greys out this button.
- **Edit:** Edit the connection by clicking its name or **edit** button.
- **Clone:** Creates a copy of the connection. You must give the copy a new name.
- **Delete:** Deletes the connection. You cannot delete a connection if any inputs, outputs, or lookups are using it. In that case, Splunk greys out this button.

You can also edit the attributes of a connection listed in [Create a database connection](#), except its name. To change the name of a connection, clone it, give the clone the name you want, and then delete the original connection.

### Permissions

The **Permissions** table is where you select the Splunk Enterprise roles that have read, read-write, or no access to the connection.

- Read access means that Splunk Enterprise roles can use the connection.
- Write access means that Splunk Enterprise roles can use and modify the connection.

By default, the Splunk Enterprise "admin" and "db\_connect\_admin" roles have write access to a new connection, the "db\_connect\_user" role has read access, and all other roles have no access.

### Override db\_connection\_types.conf

For fine-grained control over your database connections, you can override the JDBC connection strings for your database driver. The `db_connection_types.conf` file lists the supported database types, driver parameters, and test queries. To override it, copy the `db_connection_types.conf` file under `$SPLUNK_HOME/etc/apps/splunk_app_db_connect/default` to `$SPLUNK_HOME/etc/apps/splunk_app_db_connect/local`.

You can now fine-tune your database connection by editing its JDBC parameters. If the stanza name in the **db\_connection\_types.conf** in **local** is identical to the name in the same file in **default**, Splunk overrides its connection settings. Consult your JDBC driver vendor's documentation for exact parameter syntax. The [db\\_connection\\_types.conf.spec](#) file helps you understand the individual settings. When you're done, restart Splunk Enterprise.

## Create and manage database inputs

A **database input** object lets you retrieve and **index** data from a database. Database inputs are what enable Splunk Enterprise to query your database, identify the data to consume, and then tag and index the data. Once you have set up a database input, you can use that in the same way that you use other data input you have defined in Splunk Enterprise.

Splunk DB Connect 3.5.x supports creating inputs using templates created in both Splunk Add-ons and in Splunk DB Connect. This saves the work of recreating basic content of inputs. Splunk lists the templates in the template field of DB Connect. These add-ons include:

- Splunk Add-on for Microsoft SQL Server
- Splunk Add-on for Oracle Database
- Splunk Add-on for McAfee ePO Syslog
- Splunk Add-on for Nagios Core

## Create a database input

You can create a new input from scratch. Additionally, you can create your new input using a template.

- If you have not yet created an identity or a connection, you must do so before you can create a database input.
- If you create the input using a template, to make a supported add-on appear in the **Templates** list in the DB Connect UI, you must copy the `db_input_templates.conf` file from your add-on's `default` directory to your `splunk_db_connect/local` directory. You must reconfigure templates when upgrading to the latest version of Splunk DB Connect.
- Inputs only support SQL queries such as "select...from...where", not DDL such as "create table..."

Complete the following instructions to create a database input in DB Connect:

1. In Splunk DB Connect, select **Data Lab > Inputs** and then **New Input**.
2. On the **Set SQL Query** page, complete the following steps and then select **Next** to go to the **Set Properties** page.
  1. **Choose table**. Select the database table you want to use with this input.
  2. **Specify SQL query**. Specify a query to run to retrieve data from your database.
  3. **Choose input mode**. Specify the input mode and related templates for this input.
  4. **Choose input type**. Specify the input type and related settings of this input.
3. On the **Set Properties** page, complete the following steps and then select **Finish**.
  1. **Basic information**. Specify the name, description and the application of this input.
  2. **Parameter settings**. Configure the fetch size, execution frequency and max row to retrieve of this input.
  3. **Metadata**. Specify the metadata of this input, Splunk uses the value to index your data events.

### Choose Table

1. Select a connection that you want to use for this input from the list under the **Connection** field. The list displays all the connections you have configured in DB Connect, you can also enter the connection name to search the connection you want to use.  
Once you have selected the connection, Splunk DB Connect validates the connection, and displays an error message if it is not able to do so. You cannot continue the new input setup process unless you specify a valid connection.
2. From the corresponding dialog menus, choose the **Catalog**, **Schema**, and **Table** that contain the data you want to pull into Splunk platform.

### Specify SQL query

After you select the table, Splunk displays the corresponding SQL query in **SQL Editor**, you can preview the result of the query. If you need to further edit or write your own SQL query, you can write it directly in **SQL Editor** and select **Execute SQL** to preview the result. You can make complex SQL statements easier to see by selecting **Format SQL**.

**Note:** If you want to use SQL query from template, select the template from the menu. Be aware that

using a template overwrites the previous SQL you specified.

**Note:** Inputs only support SQL queries such as "select...from...where", not DDL such as "create table..."

### ***Choose input mode***

Select an input mode for your query, either **Metric** or **Event**.

- **Metric.** An input mode that allows for the user to ingest performance metric data for DB Connect. Splunk does not currently support getting data into DB Connect as metrics.
- **Event.** An input mode that allows for the user to ingest event data.

After selecting the input mode, select the related templates for the input mode from the available list of templates in the **Template** menu.

If you create the input by using a template, Splunk sets all the settings from the template for the new input. You can change them based on your needs, but be aware that Splunk saves the changes to the input you create but not to the template.

### ***Choose input type***

Specify an input type for your query, either **Batch** or **Rising**. Then configure the related fields for the input type.

- **Batch.** A batch input invokes the same database query each time you run the input and returns all results.
- **Rising.** A rising input has a column that DB Connect uses to keep track of what rows are new from one input execution to the next.

#### **Batch input**

A batch input invokes the same database query each time you run the input and returns all results. It does not keep track of whether you add rows since the last time you ran the input. Batch input mode is ideal for unchanging historical data that you want to index into Splunk once. It can also be useful for reindexing data that updates through time, such as personnel records or state tables, though this has license impact; lookups might be a better choice.

To create a batch input type,

1. Select **Batch** under **Input Type** field.
2. Select the **Timestamp** column.

Specify which column contains the timestamp that you want to use to order this data in the Splunk index. Splunk uses this required value to populate the `_time` variable and index time-series data.

- **Current Index Time:** Assigns indexed data a timestamp value that is equal to index time. Select this option if your data has no timestamp.
- **Choose Column:** Select the column that contains the timestamp value.

If this column is in a format which Splunk can't parse as a timestamp by DB Connect, Splunk displays a field where a Java SimpleDateFormat you can enter a compatible parser so that Splunk can understand the timestamp value.

- **Column:** This option only appears if you selected **Choose Column** in the previous option. Select the column that contains the timestamps you want to use.
- **Query timeout:** Enter the number of seconds to wait for the query to complete. The default is 30 if you leave it blank.

## Rising input

A rising input has a column that DB Connect uses to keep track of what rows are new from one input execution to the next. When you create a rising input type, you must select the rising column. You can select a rising column as any column whose value increases or decreases over time, such as a timestamp or sequential ID. For example, you can use columns such as `row_id`, `transaction_id`, `employee_id`, `customer_id`, `last_updated`, and so on.

Timestamps are not ideal for rising columns, though they often are the best available choice. Using a timestamp for rising column can produce the following problem conditions:

- A high rate of event generation can cause data duplication or loss, because checkpointing in a stream of timestamp-distinguished records assumes there is never more than one row created in a given time. If you set the time to a one second level of resolution and get five records per second, you lose or duplicate four records from every run.
- Clock skew, NTP skew corrections, physical moves between timezones, and daylight savings events can cause data mis-ordering, duplication, or loss. If the skew is towards the future, then the resulting checkpoint value might temporarily or permanently stop data collection.
- Splunk does not evaluate Nonnumeric datetime values numerically, and lexical sorting can produce unpredictable results. If you order time series data lexically, then the resulting checkpoint value might temporarily or permanently stop data collection.

To create a rising input, select **Rising** and configure the following fields:

- **Rising column:** The rising column is the column from which DB Connect updates the checkpoint value each time you run the input.
- **Checkpoint value:** The checkpoint value is how DB Connect determines what rows are new from one input execution to the next. The first time you run the input, DB Connect only selects those rows that contain the value higher or lower than the value you specified in this column. Each time Splunk finishes running the input, DB Connect updates the input's checkpoint value with the value in the last row of the checkpoint column.
- Update the SQL query that includes a `?` symbol and order by clause for the checkpoint value. The question mark (`?`) as the checkpoint placeholder and an *order by* clause. Every time you run the input, DB Connect replaces the question mark with the latest checkpoint value .

### Note:

- ◆ From DB Connect 3.9.0 and higher, Splunk stores the rising column checkpoints of the input in the Splunk KV Store in the collection `dbx_db_input`. For each input DB Connect creates a separate entry in the collection.
- ◆ To see what column types (`varchar`, `number`, `timestamp`, and so on) Splunk supports as rising column value types in DB Connect-supported databases, see [supported rising column types by database](#).

With a rising column mode input, you can:

- Effectively create a "descending column", or a column wherein the checkpoint decreases every time the input runs.  

```
SELECT abc, xyz FROM table WHERE abc < ? ORDER BY abc DSC
```
- Customize the comparison operation to determine the checkpoint value by evaluating a SQL function result, while simultaneously avoiding using query wrapping.  

```
SELECT CONCAT(last_name, first_name) as NAME, ACTOR_ID, FIRST_NAME, LAST_NAME FROM actor WHERE  
CONCAT(last_name, first_name) > ? ORDER BY CONCAT(last_name, first_name)
```

- Customize the `WHERE` clause to add additional conditions other than the comparison expression that the rising column inputs specify.

```
SELECT *FROM
(SELECT * FROM ACTOR WHERE ACTOR_ID > ?) ACTOR
JOIN FILM_ACTOR
  ON actor.ACTOR_ID = FILM_ACTOR.ACTOR_ID
JOIN FILM
  ON FILM_ACTOR.FILM_ID = FILM.FILM_ID
ORDER BY  ACTOR.ACTOR_ID
```

- Use other advanced SQL features in the `WHERE` clause; for example, a `CASE` statement.

After you specify the input type, select **Next** to go to the **Set Properties** page.

### **Basic information**

Configure the following fields,

- **Name:** Specify the name of the input. Be aware the input name cannot contain space or special characters.
- **Description:** Optional. The description of the input.
- **Application:** The name of the Splunk Enterprise app where you want to save this input object. By default, Splunk sets the pop-up menu to Splunk DB Connect. This menu enables other apps to use DB Connect inputs, outputs, and lookups within their own context.

### **Parameter settings**

Complete the following fields,

- **Max Rows to Retrieve:** Optional. The maximum number of rows to retrieve with each query. If you set this to 0 or leave it blank, you can have unlimited rows.
- **Fetch size:** Optional. The number of rows to return at a time from the database. The default is 300 if you leave it blank.
- **Execution Frequency:** The number of seconds or a valid cron expression, such as `0 18 * * *` (every day at 6 PM).

### **Metadata**

Splunk uses the metadata fields to index your data events. As you search the indexed data using Splunk Enterprise, you can identify data from the query by the fields that you enter here.

- **Host:** Optional. Splunk uses the host defined on the connection if you leave it blank.
- **Source:** Optional. Splunk uses the input name if you leave it blank.
- **Source type:** Enter a sourcetype field value for Splunk Enterprise to assign to queried data as you index. select the field and enter a value, or select an existing value from the menu that appears.
- **Index:** Enter an index value for the index in which you want Splunk Enterprise to store indexed data. You can enter the index name or select it from the typeahead menu.

**Note:** If you want to use the customized index, you have to make sure the index exists in the Splunk platform. Otherwise you have to create the index in Splunk Enterprise first to prevent data loss.

## **Edit database inputs**

To see a list of the defined database inputs, first select the **Data Lab>Inputs** tab. Splunk displays a list of your database inputs.



To edit a database input, select its name. You can make changes to a database input using the following buttons on the input page:

- **Enable/Disable:** Disable an input by selecting **Enable/Disable** here.
- **Edit:** Edit the input by selecting the name or the **Edit** button.
- **Clone:** Creates a copy of the input. You must give the copy a new name.
- **Delete:** Deletes the input.

You can also edit any of the attributes of a database input listed in [Create a database input](#), except its name. To change the name of an input, clone it, give the clone the name you want, and then delete the original input.

## Supported rising column types by database

The following matrix summarizes what column types (`varchar`, `number`, `timestamp`, and so on) Splunk supports as rising column value types in DB Connect-supported databases.

Database	varchar	int, float, real, bigint, number	timestamp	datetime	date
<b>AWS RDS Aurora</b>	x	x	x	x	x
<b>AWS RedShift</b>	x	x	x		
<b>DB2/Linux</b>	x	x	x		
<b>Informix *</b>					
<b>MemSQL</b>	x	x	x	x	x
<b>Microsoft SQL Server</b>	x	x	x	x	x
<b>MySQL</b>	x	x	x	x	x
<b>Oracle</b>	x	x	x	x	x
<b>PostgreSQL</b>	x	x	x		
<b>SAP SQL Anywhere</b>	x	x	x		
<b>Sybase ASE</b>	x	x	x		
<b>Sybase IQ</b>	x	x	x		
<b>Teradata</b>	x	x	x		

The column marked **x** means Splunk supports this column.

\* Information for Informix databases is not currently available.

## Use database inputs

Once you've configured a database input and Splunk Enterprise has indexed your data, you can use that input as you do any other data input you've defined in Splunk Enterprise. Use the Search Processing Language (SPL) to write a series of commands and arguments.

For a quick view of what an input returns to Splunk Enterprise, go to the database inputs page by selecting **Data Lab>Inputs** in the top navigation bar, select the name of the input to view, and then select the **Find Events** button. The search app opens with a pre-populated search that searches on your input.

You can search for keywords and use Boolean operators such as `AND`, `OR`, and `NOT`, plus wildcard characters ("`*`"):

```
manufacturing (widgets AND gadgets OR gewgaw*)
```

Fields in Splunk Enterprise correspond to columns in your database. Search for fields using the syntax: `fieldname="field value"`

```
source="db2input"
```

Chain search commands and arguments together using the pipe ("`|`") character. For example, the following search retrieves indexed events from the database input `db2input` that contain the term "manufacturing" and, for those events, reports the most common `LOCATION` values:

```
manufacturing source="db2input" | top LOCATION
```

To refine your search further, show only the fields you want using the `fields` keyword. The following search only shows the five fields listed after `fields`, plus a timestamp, for each event retrieved:

```
source="db2input" | fields ADMRDEPT DEPTNAME DEPTNO LOCATION MGRNO
```

To learn more:

- Read the Search Tutorial. Since you've already gotten your data into Splunk Enterprise, start with Part 4, Using Splunk Search.
- See Search and reporting in the *Splunk Enterprise Overview* manual for a guide to documentation based on your level of familiarity with Splunk Enterprise and on what you want to do with your data.
- For information specific to SQL users, see [SQL tips and tricks](#) in this manual, and Splunk SPL for SQL users in the *Splunk Enterprise Search Reference* manual.

## Create and manage bulk operations of the database inputs

At some point, you might need to create, change or delete multiple inputs from different connections at the same time. From Splunk DB Connect 3.1.0, you can do this by performing a bulk operation. The bulk operation includes:

- [Bulk creation](#)
- [Bulk edit \(including enable and disable\)](#)
- [Bulk delete](#)

### Create multiple inputs

The bulk creation lets you create multiple inputs using different connections at the same time. To run a bulk creation, go to **Data lab > Input** and select **Create** under the list of **Bulk Actions**. Complete the following procedures to bulk create multiple inputs and then select **Finish**.

1. [Select Connections](#)
2. [Input Settings](#) and preview the result
3. [Confirmation](#)

#### Select Connections

Select the connections used by the new inputs you create. If you want to select all connections, check the check box in the table header.

Once you have selected the connections, Splunk DB Connect validates them and displays the status of the connections on the side of the page. If you have a connection which is not valid, you can edit it on the [create and manage database connection](#) page and validate it again by clicking **Revalidate all**.

**Note:** Be aware that you can continue to the next step even if there are invalid connections, but the invalid connections might impact the data you want to retrieve from the database and Splunk might not create the input successfully. Splunk best practice is to make sure all the connections you select are valid before the next step.

## ***Input settings***

Configure the following settings and then preview the result of one connection before clicking **Next**.

**Note:** The meaning and verification of bulk input settings is the same as single input. If you need more information, see [create a database input](#) for details.

- **Template:** Select the template you want to use for the inputs. If you do not want to use the template, leave this field blank. If you create inputs by using a template, Splunk sets all the settings from the template for the inputs. You can change them based on your needs, be aware that Splunk saves the changes to the input you create but not to the template.
- **Input Type:** Select the input type for the inputs, either **Batch** or **Rising**.
- **SQL Query:** Specify the SQL query that you want to use.
- **Timestamp:** Specify the timestamp column you want to use. You can use **Current Index Time** or select the timestamp column by selecting **Choose Column**.
- **Query Timeout:** Optional. The number of seconds to wait for the query to complete. The default is 30 if you leave it blank.
- **Name Prefix:** Specify the prefix for the inputs you create. All the inputs you create in one bulk creation operation has the same prefix.
- **Description:** Optional. The description of the inputs.
- **Application:** The name of the Splunk Enterprise app where this input object gets saved. By default, the pop-up menu uses Splunk DB Connect.
- **Max row to retrieve:** Optional. The maximum number of rows to retrieve with each query. If you set this to 0 or leave it blank, you can have unlimited rows.
- **Fetch size:** Optional. The number of rows to return at a time from the database. The default is 300 if you leave it blank.
- **Execution Frequency:** The number of seconds or a valid cron expression. For example, "0 18 \* \* \*" (every day at 6 PM).
- **Host Value:** Optional. Splunk uses the host defined on the connection if you leave it blank.
- **Source:** Optional. The source field value for Splunk Enterprise to assign to queried data when indexing.
- **Source Type:** The sourcetype field value for Splunk Enterprise to assign to queried data when indexing.
- **Index:** The index in which you want Splunk Enterprise to store indexed data. You can enter the index name or select it from the typeahead menu.

After configuring the input settings, you can select one connection to preview the result. Select one connection from the list and select **Execute SQL**. The result of this connection displays on the preview page. To proceed, select **Next**. Otherwise, go back and edit the input settings until you get the expected result.

On the confirmation page, review the input settings and then select **Finish**.

## **Edit multiple inputs**

This bulk operation let's you edit multiple inputs at the same time. The bulk edit operations available depend on the inputs selected and the nature of the fields you want to change.

To run a bulk editing, go to **Data lab** -> **Input** and select **Edit** under the list of **Bulk Actions**.

Then complete the following steps,

- **Select Inputs.** Select the inputs you'd like to run the bulk operation on, and select **Next**.
- **Edit Inputs.** Select the fields which you want to edit by checking the field's name and enter the value of the field. Once you have modified the value of the field, that field modifies all inputs you selected. For the detailed description about each field, see [input settings](#).

**Note:** To make the inputs consistent, there are some constraints and dependency on editing the input fields.

- ◆ If you want to edit the input type, configure the following dependent fields based on the input type you select.
  - ◇ Batch input type: SQL Query field, timestamp field.
  - ◇ Rising input type: SQL Query field, Rising Column field (checkpoint value) and timestamp field.
- ◆ If you want to edit the SQL Query field, you must update the following dependent fields.
  - ◇ Batch input type: timestamp field.
  - ◇ Rising input type: Rising Column field (checkpoint value) and timestamp field.

The dependent fields get greyed out in Splunk Web before you select the **Input Type** or **SQL query** field if they have different values.

- Select an input to preview the result and then select **Next**.
- On the confirmation page, review the settings you edit for the selected inputs and then select **Finish**.

## Delete multiple inputs

The bulk operation lets you delete multiple inputs at the same time. To run a bulk deletion, go to **Data lab** -> **Input** and select **Delete** under the list of **Bulk Actions**.

On the **Delete inputs** page, select the input names you want to delete and select **Next**. A confirmation dialog box pops up and lists all the inputs you want to delete. If you are OK to delete all the inputs, select **Yes, delete all** or otherwise **Cancel**.

## Create and manage database outputs

A **database output** object lets you define how to send data from Splunk Enterprise to a database on a recurring basis. Defining database outputs is useful if you want to store your indexed Splunk Enterprise data in a relational database.

### Create a database output

From within Splunk DB Connect, click the **Data Lab** > **Outputs** tab and click **New Output**.

Complete the following steps to create a database output.

1. [Set up search](#). Perform a SPL or a saved search to search the event data in Splunk platform.
2. [Choose table](#). Choose the table from the database which Splunk event data will be exported to.
3. [Fields mapping](#). Map the Splunk Enterprise fields you selected to columns in your database.
4. [Set properties](#). Configure the basic settings and parameters of the output.

## Set up search

In this step, you define the Splunk Enterprise fields that you want to output to the database table as columns:

1. Perform a search of your Splunk Enterprise data. You can either enter a search using the Search Processing Language (SPL), or you can run a saved search.
2. To select a saved search, click **Saved Search** and then choose the one you want to use from the drop-down list. Splunk Enterprise performs the search and displays the results in a table. Each column corresponds to a Splunk Enterprise field.
3. Specify the time range you want to run the search. Be aware that you cannot choose a real-time time range since it will keep running the search all the time. It is a known issue in DB Connect 3.1.0.

### Tips:

- ◇ You can fine-tune the format of your output directly from this search field by using standard search commands. For example, if you want Splunk Enterprise to display and use output to send data to your database as key-value pairs, use the `eval` search command here. If you want to change the quoting pattern, use `rex`.
- ◇ If you want Splunk Enterprise to send a specific number of rows each time the output runs, use the `head` search command. For example, to specify that the output should send no more than 1000 rows each time it runs, define your search as follows:

```
index=main sourcetype=foo status=ERROR | head 1000
```

- ◇ If you want to use the **Saved Search**, set the permission of the saved search to **This app only (splunk\_app\_db\_connect)** or **All apps**.

4. Run the search and if you are ok with the result, click **Next**.

## Choose table

In this step, you choose the table from the database which Splunk event data will be exported to.

1. **Connection:** Choose the [database connection](#) you want to use with this output. DB Connect validates the connection and displays an error message if it is not able to do so. You cannot continue the new output setup process unless you have specified at least one valid connection from the **Connection** dialog.
2. From the corresponding dialog menus, choose the **Catalog**, **Schema**, and **Table** that contain the columns you want to map the Splunk event data to.
3. Once you have chosen the table, you can preview the data on the **Table Schema Preview** page. It lists the column name, data type, column size and related fields of the table.
4. Click **Next**.

## Fields mapping

In this step, you map Splunk data field(s) to the database column(s).

- **Fields Mapping.** Click **Add Search Field** and choose the field name of Splunk data you want to output. Then select the column name in the **Table Column** field. The data of the selected field will be exported to the column in your database table. You can add multiple Splunk fields/table columns pairs if you need to.
- **UPSERT Configuration.** UPSERT allows either inserting a row, or on the basis of the data of the column already existing, UPDATE that existing row instead. If you choose to **Enable UPSERT**, then specify the **Key** column in your table.

**UPSERT Example:** UPSERT is enabled and the **id** column is set as the key column.

id	name
1001	Jack
1002	Michael

If there are two data events to be exported to this table, one is 'id=1001, name=Daniel', the other is 'id=1003, name=Leo', then the updated table is

id	name
1001	Daniel
1002	Michael
1003	Leo

## Set properties

### Basic information

- **Name:** The output name cannot contain any spaces. Do not use special characters in the output name.
- **Description:** Optional. The short description of this output.
- **Application:** The name of the Splunk Enterprise app in which DB Connect saves this output object. By default, the pop-up menu selects **Splunk DB Connect**. This menu enables other apps to use DB Connect inputs, outputs, and lookups within their own context.

### Parameter Settings

- **Query Timeout:** Enter the number of seconds for a single query to wait to complete. The default is 30 if you leave it blank.
- **Batch size:** Enter the size of a batch. The default is 1000 if you leave it blank.
- **Scheduling:** Configure **Execution Frequency** if you want to schedule this output, you can enter the number of seconds or a valid cron expression e.g. 0 18 \* \* \* (every day at 6PM).

## Edit database outputs

To see a list of the database outputs you defined, navigate to the **Configuration > Outputs** tab.

To edit a database output, click its name. You can make changes to a database output using the following buttons on the output page:

- **Enable/Disable:** Enable or disable an output.
- **Edit:** Edit an output by clicking its name or the **Edit** button.
- **Clone:** Create a copy of the output. You must give the copy a unique name.
- **Delete:** Delete the output.

You can also edit any of the attributes of a database output listed in [Create a database output](#), except its name. To change the name of an output, clone it, give the clone the name you want, and then delete the original output.

## Enable output to multi-byte character sets

DB Connect can send data that is in a multi-byte character set, such as Traditional Chinese, using a [database output](#). Depending on your database, you may need to change certain settings to the database to properly receive and store the data.

- **MySQL:** When creating a connection to a MySQL database, customize the JDBC URL by adding some additional query parameters. For more information, access MySQL documentation for Inserting unicode UTF-8 characters into MySQL.
- **PostgreSQL:** By default, this database supports multi-byte character sets. You do not need to take additional steps.
- **Microsoft SQL Server:** Ensure your database columns' data types are N-variant types, such as NVARCHAR versus VARCHAR).
- **Oracle:** Change your database character set to AL32UTF8. For more information, access Oracle's documentation for Supporting Multilingual Databases with Unicode.
- **Other databases:** Consult your database's documentation for more information about enabling multi-byte character sets.

## Use database outputs

Database outputs run automatically at the frequency you set during the "[scheduling](#)" step of the setup process. To verify that database outputs are working properly, query your database after a few executions of the output operation to ensure that DB Connect is sending your Splunk Enterprise data properly.

- ◆ DB Connect 3 does not support running scheduled tasks (input or output) on the search head in the Search head cluster deployment. You must run the scheduled task on a heavy forwarder.
- ◆ In a distributed environment, the heavy forwarder will need to be able to search the Indexers in order to output to the DB. See Deploy a distributed search environment in the *Distributed Search* manual to learn how to set up distributed search on your deployment's heavy forwarders.

## Use modular alert to run database output

DB Connect provides a modular alert which allows users to actively respond to events and send alerts. You can configure the **DBX output alert action** on the Alert Actions Manager page. See alert action manager.

To use the DB Connect modular alert:

1. Navigate to the **Search** page in DB Connect.
2. Create a search, then select **Save as > Alert**.
3. Enter the title and optional description.
4. In the **Trigger Actions** field, select **DBX output alert action**.
5. Enter the **Output Name**. The output name must exist in DB Connect.

For details about custom alert, see Create custom alert.

## Use dbxoutput command to run database output

`dbxoutput` is a search command you can use to run [database outputs](#) that you have defined in DB Connect.

## Syntax

```
dbxoutput output=<string>
```

## Required Arguments

output

**Syntax:** output=<string>

**Description:** Name of a configured database output object.

chunksizesize

**Syntax:** <integer>

**Description:** Specifies the number of events that will be processed to be output to the database in a single operation (not to be confused with the `batch_size` property of the DB Output).

**Default:** 1000.

## Example

The following example uses the output `dbx_output` to send the results of a search query to a database.

```
<search query> | dbxoutput output="dbx_output"
```

## Create and manage database lookups

A **database lookup** object enables you to enrich and extend the usefulness of your Splunk Enterprise data through interactions with your external database.

For example, a database lookup is a lookup that takes a customer ID value in an event, matches that value with the corresponding customer name in your external database, and then adds the customer name to the event as the value of a new `customer_name` field. Therefore, if you have an event where `customer_id="24601"`, the lookup adds `customer_name="Valjean, Jean"` to the event.

You can run lookups against your database by specifying the `dbxlookup` command in your search query.

The database data DB Connect transfers when it performs lookups does not count toward your daily Splunk Enterprise indexing quota.

## Select between database lookups and Splunk lookups

Splunk provides lookups by default which enable high-speed lookup functionality by holding the desired data in memory or the KV store. CSV lookups are cached in memory for high speed access. This is ideal behavior for data sets that are infrequently changed, smaller than available RAM, or both. KV Store lookups are not fully cached in memory, but they do consume resources. In either case, these lookups can be created and updated from an external database using the `dbxquery` command in a scheduled search. For instance, the following search could be used to maintain a CSV lookup of IDs and names:

```
| dbxquery query="SELECT actor_id,first_name,last_name FROM \"sakila\".\"dbo\".\"actor\""
connection="ms-sql"
| inputlookup append=true actor-lookup.csv
| dedup actor_id
| outputlookup actor-lookup.csv
```



A database lookup is different from a Splunk lookup because each execution will contact the database for the current state of the rows in question. This is ideal behavior for data sets that are rapidly changing, very large, or both, because the user receives the freshest data possible.

## Create a database lookup

From within Splunk DB Connect, navigate to the **Data Lab > Lookups** tab and click **New Lookup**. You have to [Create a new identity](#) and [create a new connection](#), before you can create a new database lookup.

Complete the following steps to create a database lookup.

- [Set Reference Search](#). Perform a Splunk Enterprise search to select the fields on which to base the lookup.
- [Set Lookup SQL](#). Specify a reference search to fetch data from your database.
- [Field Mapping](#). Map your selected database table column, plus any new column you want to add, to Splunk Enterprise fields.
- [Set Properties](#). Configure the basic settings and parameters of the lookup.

### ***Set Reference Search***

In this step, you perform a search or use the saved search of your Splunk indexed data.

1. Perform a search of your Splunk Enterprise data. You can either enter a search using the Search Processing Language (SPL), or you can run a report (saved search).
2. To select a saved search, click **Saved Search** and then choose the one you want to use from the drop-down list. Splunk Enterprise performs the search and displays the results in a table. Each column corresponds to a Splunk Enterprise field.  
If you want to use the **Saved Search**, set the permission of the saved search to **This app only (splunk\_app\_db\_connect)** or **All apps**.
3. Click **Next**.

### ***Set Lookup SQL***

In this step, you specify a search to fetch data from your database.

1. Choose a connection that you want to use for this lookup from the drop-down list under the **Connection** field. The drop-down list lists all the connections you have configured in DB Connect, you can also enter the connection name to search the connection you want to use.  
Once you have selected the connection, Splunk DB Connect will try to validate the connection, and will display an error message if it is not able to do so. You cannot continue the new lookup setup process unless a valid connection is specified.
2. From the corresponding dialog menus, choose the **Catalog**, **Schema**, and **Table** that contain the columns you want to base for this lookup.
3. After you choose the table, the corresponding SQL query will be displayed in **SQL Editor**, you can preview the result of the query. If you need to further edit or write your own SQL query, you can write it directly in SQL Editor and click **Execute SQL** to preview the result. You can make complex SQL statements easier to read by clicking **Format SQL**. This is a one-way operation.  
The reference search fields of Splunk data will be listed at the right side of the page. You can use it as a reference to specify your SQL query.
4. Click **Next**.

## Field Mapping

In this step, you map the selected Splunk fields with the database table column(s), then you can add the table columns as new Splunk data fields to enrich your Splunk data.

1. **Search fields mapping.** Click **Add Search Field** and choose the field name you want to match to the database table column. Then select the column name in the **Table Column** field. You can add multiple pairs of fields and columns if you need to.
2. **Add table column(s) as new Splunk fields.** Click **Add Column** and choose the column name on the drop-down list, the column you select will be added as new fields in Splunk data.
3. **Set alias name to the table columns.** (Optional) To make the name of the database column easy to understand, You can rename the database column by entering a new name in **Aliases** field. The name must be unique, and must not overlap with the field name. When you choose a database column in the first part of this step, do not then choose the same database column in the second part of this step.
4. The corresponding search will be displayed in **Preview Results** field. You can run the search to preview the data by clicking **Open in Search**.

## Set properties

Configure the following fields:

- **Name:** The output name cannot contain any spaces. Do not use special characters.
- **Description:** The description of the lookup
- **Application:** The name of the Splunk Enterprise app in which DB Connect saves this lookup object. By default, the DB Connect selects **Splunk DB Connect**. This menu enables other apps to use DB Connect inputs, outputs, and lookups within their own context.

You can use the generated `dbxlookup` command in the summary page to enrich your Splunk data. See more on [Use dbxlookup to perform lookups in DB Connect](#).

## Edit database lookups

To see a list of the defined database lookups, first click the **Data Lab>Lookup** tab. To edit a database lookup, click its name. You can make changes to a database lookup using the following buttons on the lookup page:

- **Enable/Disable:** Enable/Disable a lookup by clicking **Disable** here.
- **Edit:** Edit a lookup by clicking its name or the **Edit** button.
- **Clone:** Creates a copy of the lookup. You must give the copy a new name.
- **Delete:** Delete the lookup.

You can also edit any of the attributes of a database lookup listed in [Create a database lookup](#), except its name. To change the name of a lookup, clone it, give the clone the name you want, and then delete the original lookup.

## Use dbxlookup to perform lookups in DB Connect

### Description

`dbxlookup` is a search command for performing lookups by using remote database tables as lookup tables. Use `dbxlookup` to enrich your indexed events with the information you have stored in external databases. If you are not using Verbose search mode, you must explicitly reference input fields in the search.

## Syntax

There are two ways to use the `dbxlookup` command.

```
dbxlookup chunksize=<integer> lookup=<lookup_name>
```

The argument `<lookup_table_name>` refers to the lookup you defined in DB Connect UI.

From DB Connect 3.1.0, `dbxlookup` command allows users to declare Splunk fields/table column mapping directly in the options. The syntax is similar as `lookup`. Users do not have to create a lookup in the UI before using it in the `dbxlookup` command.

```
dbxlookup connection=<connection name> query=<SQL query> chunksize=<integer> <database_key_column> AS  
<splunk_key_field> OUTPUT <column_destname> AS <field_destname>
```

## Argument

If you want to use the lookup defined in DB Connect, use the following

lookup

**Syntax:** `lookup=<lookup_name>`

**Description::** Required. It refers to the lookup name you defined in the UI.

chunksize

**Syntax:** `<integer>`

**Description:** Optional. Specifies the number of events that will be used to look up the database of each query.

**Default:** 1000.

If you want to define the connection, SQL query and mappings directly in `dbxlookup` command, use the following

connection

**Syntax:** `connection=<connection name>`

**Description::** Required. Specifies the connection name you want to use for the lookup.

query

**Syntax:** `query=<SQL query>`

**Description::** Required. Specifies the SQL query to search the data in your database.

chunksize

**Syntax:** `<integer>`

**Description:** Optional. Specifies the number of events that will be used to look up the database of each query.

**Default:** 1000.

`<database_key_column>`

**Syntax:** `<string>`

**Description::** Refers to a table column in the database table to match the key value against the field in Splunk. You can specify multiple `<database_key_column>` values separated.

`<splunk_key_field>`

**Syntax:** `<string>`

**Description::** Refers to a Splunk field from which to acquire the value to match in the database table. You can specify multiple `<splunk_key_field>` values.

<column\_destname>

**Syntax:** <string>

**Description::** Refers to a table column in the database table to be copied into the Splunk field. You can specify multiple <column\_destname> values separated by commas.

<field\_destname>

**Syntax:** <string>

**Description::** The alias name of <column\_destname> in Splunk. You can specify multiple <field\_destname> values separated by commas.

**Default::** The value of the <field\_destname> argument.

When using the `dbxlookup` command, if an `OUTPUT` clause is not specified, all the table columns that are not the key field are used as output fields. If the `OUTPUT` clause is specified, the output lookup fields overwrite existing fields.

### Examples

**Example 1:** This example uses the lookup "GTS\_Product\_Lookup" to enrich the results of a Splunk Enterprise search query.

```
sourcetype=GTS_Sessions | dbxlookup lookup="GTS_Product_Lookup" | stats sum(product_price) as "spend" by uid
```

**Example 2:** If you are not using Verbose search mode, you must explicitly reference input fields in the search:

```
sourcetype=GTS_Sessions | fields pid, uid | dbxlookup lookup="GTS_Product_Lookup" | stats sum(product_price) as "spend" by uid
```

**Example 3:** This example defines the connection, SQL query, mappings and alias name directly in the `dbxlookup` command.

```
sourcetype=GTS_Sessions | dbxlookup connection="sh-oracle" query="SELECT * FROM \"TEST\".\"DBX\" ID AS customer_id OUTPUT data AS customer_data, name AS customer_name
```

**Example 4:** If there is no alias name, the table column name will be set as the corresponding Splunk field name.

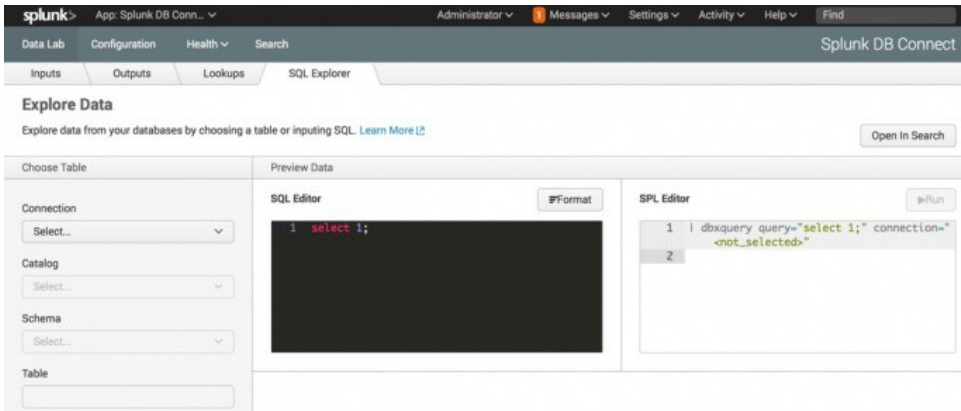
```
sourcetype=GTS_Sessions | dbxlookup connection="sh-oracle" query="SELECT * FROM \"TEST\".\"DBX\" ID OUTPUT data, name AS \"Last Name\"
```

## Use SQL explorer to make live reports

DB Connect 3.x.x provides a SQL Explorer interface to help you use [dbxquery](#) to query your database in real time. You can browse the database connections, catalogs, and schema, write and test SQL commands, and modify results with Splunk's SPL commands. You can use the resulting command in Splunk Enterprise searches to create reports, alerts, and dashboard panels.

### Use SQL Explorer to build queries

You can reproduce the examples on this page in your test environment using a free tool such as MySQL Sakila sample database.



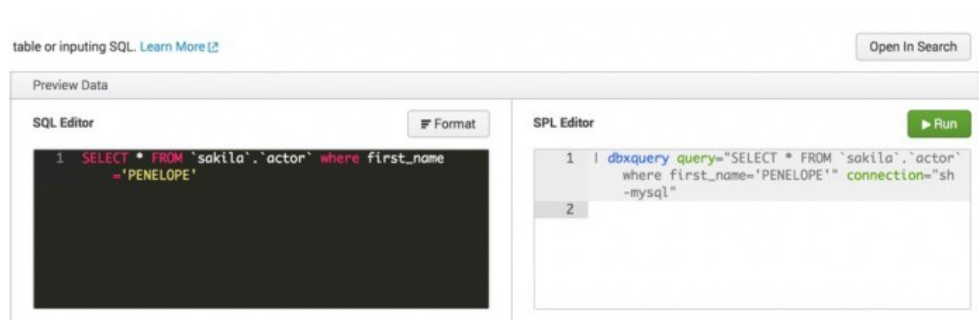
1. Navigate to **Data Lab>SQL Explorer** in DB Connect.
2. Select a **Connection** that you defined in [create and manage database connections](#).
3. Browse to the desired **Catalog**, **Schema**, and **Table** from the type-ahead lists on the left. Refer to the basic query of the table you selected in the SQL Editor window.
4. Replace or customize the query by writing SQL in **SQL Editor**. Click **Format SQL** to make complex SQL statements easier to read. This is a one-way operation.
5. DB Connect generates the `dbxquery` command and displays it in the **SPL Editor**. If you want to further customize the command, write SPL in this window, then click **Run** to see the results.
6. Click **Open in Search** to copy the final `dbxquery` command to Splunk Search in a new Tab. Add visualizations, look for patterns, or save the query as a dashboard panel, alert, saved search, or report. Each use of these new elements generates a live query to the database. Review [access controls](#) and [performance considerations](#) before you deploy in production.

## Use variables in dbxquery to build dashboard panels

DB Connect allows users to use variables in `dbxquery` to interactively query databases and create dashboards. You can use variables to enter an input, select multiple time pickers, or dynamically drill down. In the example below, create a dashboard form in which the user enters a first name into a DB Connect search from the sakila sample database.

1. Create a SQL statement by choosing **Connection**, **Catalog**, **Schema** or **Table** under **Data Lab > SQL Explorer**.

```
SELECT * FROM `sakila`.`actor` WHERE first_name='PENELOPE'
```



2. DB Connect generates the `dbxquery` command and presents it in the **SPL Editor**. Edit the SPL if you wish, then click **Open In Search**.

```
| dbxquery query="SELECT * FROM `sakila`.`actor` where first_name='PENELOPE'" connection="sh-mysql"
```

3. DB Connect generates the SPL and presents it in the Searching and Reporting app, where you can further edit the SPL or use visualizations.
4. Click **Save As > Dashboard Panel > New Dashboard** to create a dashboard. If you are not familiar with how to create and edit dashboards, see create dashboards for details.
5. Add a form element for the user to use. Click **Edit > Add Input > Text** to edit the Token. Set the input field name to **First Name** and the token to `first_name` in this form, then click **Apply**. See token usage in dashboards for further details.

**T Text** General

☒ Radio

▼ Dropdown

☒ Checkbox

▼ Multiselect

[Link List](#)

☒ Time

Label

Search on Change ☐

Token Options

Token?

Default?

Initial Value?

Token Prefix?

Token Suffix?

6. Click **Edit Search** to edit the SPL, and replace `where first_name='PENELOPE'` with `where first_name='$first_name$'`. This variable name is the token name from your input form, bracketed with dollar signs. Click **Apply** to continue.

**Edit Search**

Title

Search String

[Run Search](#)

Time Range Scope

Auto Refresh Delay?

Refresh Indicator

7. Test the form by entering 'RICHARD' in the **First Name** input field. The dashboard updates with the value you entered in this field.

## Use dynamic variables in dashboard panels

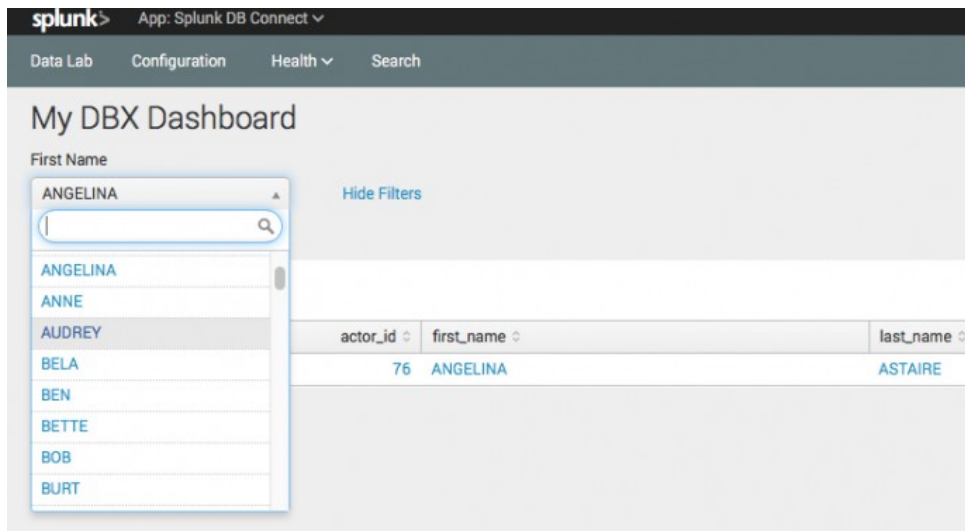
Now, replace the **First Name** form with a drop down that allows the user to select from existing names in the table.

1. Return to SQL Explorer and edit the SQL statement to select a distinct list of first names from the actors table.

```
SELECT DISTINCT first_name FROM `sakila`.`dbo`.`actor`
```

Test the result, then Open in Search to copy the resulting `dbxquery` command.

2. Edit the dashboard again, then edit the input form element. Change the type from **Text** to **Dropdown**. Under **Dynamic Options**, set the "Search String" field to the **dbxquery** command for selecting a distinct list of names: `dbxquery query="SELECT DISTINCT first_name FROM `sakila`.`dbo`.`actor`" connection="sh-mysql"`. Set the "Field For Label" field to **first\_name**, and set the "Field for Value" field to **first\_name**. Click **Apply**. See token usage in dashboards for further details.
3. Test the form by selecting 'ANGELINA' from the dropdown input field. The dashboard updates with the value you entered in this field.



## Limitations of time-based inputs

Splunk Enterprise dashboards and Splunk DB Connect support many types of variables in a dashboard, including string, numeric, and time values. The native Splunk Enterprise dashboard time selector is not always a good choice for dashboards that use DB Connect to perform time-based queries. The time selector allows users to select exact time representations (for instance, `earliest=%m/%d/%Y:%H:%M:%S`) and relative time representations (for instance, `[+|-]<time_integer><time_unit>`) in the same form. These two types of WHERE conditions use different SQL which can cause SQL errors. To avoid errors, customize the form to limit available time input formats. See [date and time format variables](#) for details.

## Execute SQL statements and stored procedures with the `dbxquery` command

Splunk DB Connect has the `dbxquery` command for executing SQL statements and stored procedures within Splunk Enterprise searches and dashboards. [Use SQL explorer to edit your query](#) or write the `dbxquery` based on the syntax below.

### Description

`dbxquery` is a custom search command for querying remote databases and generating events in Splunk Enterprise from the database query result set.

### Syntax

```
dbxquery connection=<string> [fetchsize=<int>] [maxrows=<int>] [timeout=<int>] [shortnames=<bool>]  
query=<string> OR procedure=<string> [params=<string1,string2>]
```

### Required Arguments

`connection`

**Syntax:** `connection=<string>`

**Description:** Name of a configured database connection object.

`query` or `procedure`

`query`

**Syntax:** `query=<string>`

**Description:** A SQL query. You can also use a URL-encoded SQL query, but you must percent-encode all spaces (`%20`).

`procedure`

DB Connect supports stored procedures in databases beginning with version 3.0.0. A stored procedure is a set of Structured Query Language (SQL) statements with an assigned name. It is stored in the database in compiled form so that several programs can share it. You can call the stored procedures of your database by using a `procedure` argument instead of a `query` argument. Many stored procedures expect variables in the form of an ordered list of arguments. You can pass these variables from Splunk to the stored procedure using the `params` argument. Note that the `dbxquery` command is a generating command and needs to be the first command in a search. See [Cross-database queries](#) on this page for examples of the syntax needed to pass parameters to `dbxquery`.

**Syntax:** `procedure=<string>`



**Description:** A stored procedure invocation.

This table lists `procedure` syntax formats for different types of databases.

Supported databases	Syntax
AWS RDS Aurora Informix MemSQL Microsoft SQL Server MySQL SAP SQL Anywhere Sybase ASE Sybase IQ IBM DB2 for Linux Teradata	<code>dbxquery procedure="{call &lt;procedure-name&gt;}"</code>
Oracle	<code>dbxquery procedure="{call &lt;procedure-name&gt;(?)}"</code>
Postgres AWS RedShift	<code>dbxquery procedure="{?=call &lt;procedure-name&gt;}"</code>

Important notes on stored procedure support in DB Connect:

- ◆ `dbxquery` only supports stored procedures which return a single result set.

#### IBM DB2 examples

- ◆ If you use an **IBM DB2 for Linux** or **Teradata** database, you must return the result as Cursor type and open it within the stored procedure. See the IBM DB2 for Linux stored procedure example below for details.

#### Stored procedure example (IBM DB2 for Linux)

```
CREATE OR REPLACE PROCEDURE TEST_IBMSP (IN actor_id varchar(30))
  DYNAMIC RESULT SETS 1
  LANGUAGE SQL
BEGIN
  DECLARE result_set CURSOR WITH RETURN TO CLIENT FOR
  SELECT * FROM ACT WHERE ACTNO <= actor_id;
  OPEN result_set;
END;
```

#### Oracle examples

- ◆ If you use an **Oracle** database, you must store the result in the first parameter and set it as OUT `SYS_REFCURSOR`. See the Oracle Stored procedure example below for details.

#### Stored procedure examples (Oracle)

Create stored procedure 1:

```
CREATE OR REPLACE PROCEDURE test_orasp_1(
  p_ref_cursor OUT SYS_REFCURSOR,
```

```

    p_var_in      IN  VARCHAR)
AS
BEGIN
    OPEN p_ref_cursor FOR
    SELECT 'you passed-in: '|| p_var_in out_var FROM dual;
END test_orasp_1;

```

#### Use stored procedure 1:

```
| dbxquery connection=splunk_test procedure="{call test_orasp_1(?,?) }" params="foo"
```

#### Create stored procedure 2:

```

CREATE OR REPLACE PROCEDURE TEST_ORASP_2(
    ref_cursor OUT SYS_REFCURSOR,
    id         IN  VARCHAR)
AS
BEGIN
    OPEN ref_cursor FOR
    SELECT * FROM soe.customers WHERE customer_id = id;
END TEST_ORASP_2;

```

#### Use stored procedure 2:

```

| dbxquery connection=splunk_test procedure="{call test_orasp_2(?,?) }" params="50865"
| makeresults count=1
| eval cust_id="50865"
| map search="| dbxquery connection=splunk_test procedure=\"{call test_orasp_2(?,?) }\"
params=\"\$cust_id\$\" "

```

#### Stored procedure example (MS SQL Server)

```

CREATE PROCEDURE test_sp_no_param
AS
BEGIN
    SET NOCOUNT ON;
    SELECT * FROM city as c1 LEFT JOIN country as c2 on c1.country_id=c2.country_id;
END
GO

```

## Optional Arguments

fetchsize

**Syntax:** fetchsize=<int>

**Description:** The number of rows to return at a time from the database. To avoid running out of memory, the query result set is divided into multiple pieces and returned to DB Connect one piece at a time. This argument specifies how many rows are in each of those pieces. Depending on the RPC server's maximum heap size and whether the target database table contains any unusually large columns, you may want to specify a smaller value for this argument. The maximum value for this option is 10,000.

**Default:** varies per database type

maxrows

**Syntax:** maxrows=<int>

**Description:** The maximum number of rows to return. If you do not specify a value for maxrows, dbxquery returns 100,000 rows at most. There is no maximum value for this argument, but retrieving a very large number of records may result in performance issues or out-of-memory messages. In this case, you should experiment with setting maxrows to a lower number that is manageable by your Splunk Enterprise server hardware.

**Default:** maxrows=100000

params

**Syntax:** params="string1,string2"

**Description:** The value(s) of the variable(s) you defined in query or procedure. The value of the params is in CSV format.

**Note:** The value of params="BOB" is different from params=" BOB". The space character is not skipped inside the quotation marks.

**Example:** The values of actor\_id and actor\_name are defined in params="3,BOB", which is 3 and BOB respectively.

```
dbxquery query="select * from actor where actor_id > ? and actor_name = ?" connection="mysql" params="3,BOB"
```

Another example for using stored procedure, 150 and BOB are the values for the variables in the procedure *sakila.test\_procedure*.

```
dbxquery procedure="{call sakila.test_procedure(?, ?)}" connection="mysql" params="150,BOB"
```

shortnames

**Syntax:** shortnames=<bool>

**Description:** By default, the returned fields will be in the TABLE> form.<COLUMN>.<DATATYPE>. Set the shortnames argument to true to return fields called <COLUMN>.

**Default:** shortnames=true

**Note:** To emulate the output format of the dbquery command from DB Connect 1.x.x, set the output to csv and shortnames to true.

timeout

**Syntax:** timeout=<int>

**Description:** Specifies the timeout of your query in seconds. Set to zero to allow unlimited execution. Typically the value should be less than query job lifetime, which defaults to 10 minutes (600 seconds). For more information, see [http://docs.oracle.com/javase/7/docs/api/java/sql/Statement.html#setQueryTimeout\(int\)](http://docs.oracle.com/javase/7/docs/api/java/sql/Statement.html#setQueryTimeout(int)).

**Default:** timeout=600

## Examples

```
| dbxquery query="SELECT actor_id, first_name as fname, last_name as lname, last_update FROM actor"
connection="mySQL"
| dbxquery query="SELECT actor_id, first_name as fname, last_name as lname, last_update FROM actor"
connection="mySQL" maxrows=100
```

## Cross-database queries

DB Connect 2.2.0 and later includes improved support for using the `dbxquery` command in complex Search Processing Language (SPL) statements. You can now use `dbxquery` with subsearches, so that you can effectively perform cross-database queries—enriching or filtering data on the fly.

Using a command such as `append` with `dbxquery`, you can run a separate database search and add the output to the first search. For example, the following search query appends results obtained from a SQL Server database connection object to results obtained from a MySQL database connection object:

```
| dbxquery connection=mysql query="SELECT * FROM sakila.city" | append [dbxquery connection="sql_server"
query="SELECT * FROM sakila.actor"]
```

You can use any of the following search commands to produce cross-database query results:

- `append`: Appends the results of a subsearch to the current results.
- `appendcols`: Appends the columns of the subsearch results with the input search results.
- `join`: Combines the results of a subsearch with the results of a main search. One or more of the columns must be common to each result set.
- `map`: Runs a search repeatedly for each input record or result. You can run the `map` command on a saved search, a current search, or a subsearch.

For more information about subsearches:

- Read About subsearches in the *Splunk Enterprise Search Manual*.
- Read How to use the search command in the *Splunk Enterprise Search Manual*.

## Incompatible Arguments

The following `dbxquery` arguments are not functional. They do not produce errors if you leave them in the command line.

- `wrap`
- `output`

You can't use `INSERT` or `UPDATE` statements with `dbxquery`. Use `dbxoutput` if you want to write data to your database.

## Monitor Splunk DB Connect health

Splunk DB Connect provides pre-configured health dashboards that allow you to monitor and troubleshoot several aspects of your database connections and inputs from inside Splunk Enterprise. Using numerous pre-built Splunk visualization panels, you can monitor DB Health and Input Metrics.

Data model acceleration for DB Connect dashboards is disabled by default. If your DB Connect dashboards are slow, enable data model acceleration.

## Permissions

A logged-on user must be assigned a role that has access to both the `_internal` index and "dbx\_health" source type in order to see the health dashboards. If none of the user's roles has permission to either search `_internal` or view data with `sourcetype="dbx_health"`, DB Connect displays a "Permission denied" error message.

## DB Connect input health

This dashboard provides pre-built panels to give you insights about your input operation health.

To use the health log dashboard, go to Splunk DB Connect, then click the **Health > DB Connect Input Health** tab.

The pop-up menus along the top of the window are controls you can use to refine the data in the dashboard. By default, they show all available data from the last 24 hours.

- **Total Errors:** The total number of errors for all the inputs.
- **%Error:** The percentage of the errors among all input jobs.
- **Number of Input Jobs over Time:** The number of input Jobs over the time you choose.
- **Input Jobs Errors over Time:** The total errors of input jobs. over the time you choose.

## DB Connect Input Performance

This dashboard gives you insights about the inputs performance.

To use the input performance dashboard, go to Splunk DB Connect, then click the **Health > DB Connect Input Performance** tab.

- **Input Jobs Median Duration:** The median duration of input jobs over the time you choose.
- **Input Jobs Median Time Distribution:** The median distribution time for each input.
- **HEC Median Duration:** The median duration of HEC.
- **HEC Median Throughput:** The median throughput of HEC
- **HEC Connection Status:** HEC connection status over the time you choose.
- **HEC Upload Time:** HEC upload time over the time you choose.
- **HEC Median Upload Time(ms) by Input Name:** HEC median upload time for each input.
- **HEC Median Throughput(MB) by Input Name:** HEC median throughput for each input.

## DB Connect Connection Health

This dashboard gives you insights about the connection health.

To use the input performance dashboard, go to Splunk DB Connect, then click the **Health > DB Connect Connection Health** tab.

- **Total Errors:** The total number of errors of the connection you choose.
- **%Error:** The percentage of the errors among all input jobs.
- **Number of Each Connection over Time:** The number of connections per each connection over the time period you select.
- **Connections Errors over Time:** The number of errors per each connection over the time period you select.
- **Number of Each Operation over Time:** The number of operations (dbxinput, dbxoutput and dbxlookup) over the time period you select.
- **Operations Errors over Time:** The errors of each operation over the time period you select.

## Connection Pool Health (scheduled inputs/outputs)

- **Active Connections:** The number of active connections of each connection pool over the time period you select.
- **Idle Connections:** The number of idle connections of each connection pool over the time period you select.
- **Pending Connections:** The number of pending connections of each connection pool over the time period you select.
- **Total Connections Pool:** The total number of connections (including active, pending and idle connections) of each connection pool over the time you select.
- **Connections Wait Time:** The waiting time for each connection pool.
- **Connections Usage Time:** The usage time for each connection pool.
- **Connections Wait Median Duration:** The median waiting duration for each connection pool.
- **Connections Usage Median Duration:** The median duration usage of each connection pool.

# Reference

## Configuration file reference

Splunk DB Connect includes the following custom configuration spec files:

- [app-migration.conf.spec](#)
- [db\\_connection\\_types.conf.spec](#)
- [db\\_inputs|db\\_inputs.conf.spec](#)
- [db\\_outputs|db\\_inputs.conf.spec](#)
- [db\\_lookups.conf.spec](#)
- [identities.conf.spec](#)
- [inputs.conf.spec](#)
- [dbx\\_settings.conf.spec](#)

The most current versions of these spec files exist in the `$SPLUNK_HOME/etc/apps/splunk_app_db_connect/README/` folder, `%SPLUNK_HOME%\etc\apps\splunk_app_db_connect\README\` directory on Windows.

## Configuration files for version upgrades

Configuration files for Splunk DB Connect have changed in DB Connect version 3. DB Connect does not transfer your configuration files to the new version automatically. See [Migrate DB Connect deployment to DB Connect 3.0.0](#)

The following sections describe the .conf files where settings existed in previous versions of DB Connect, and where they exist in DB Connect version 3.

### *DB Connect version 2 configuration files*

- **app-migration.conf**: Specifies options for migrating from DB Connect 1.x.x to DB Connect 2.x.x.
- **db\_connection\_types.conf**: Lists the database types, driver parameters, and test queries DB Connect supports.
- **db\_connections.conf**: Stores all configuration necessary for connecting to a specific database, unless overridden by parameters from [identities.conf](#).
- **healthlog.conf**: Manages the behavior of the self-monitoring health dashboard in DB Connect.
- **identities.conf**: Stores credentials that DB Connect uses to connect to databases in the standard Splunk Enterprise credential store.
- **inputs.conf**: Configures database indexing, lookup, and output behavior using modular inputs. Also includes per-input Java options.

### *DB Connect version 3 configuration files*

- **app-migration.conf**: Specifies options for migrating from DB Connect 2.x.x to DB Connect 3.x.x.
- **db\_connection\_types.conf**: Lists the database types, driver parameters, and test queries that DB Connect supports.
- **db\_connections.conf**: Stores all configuration necessary for connecting to a specific database, unless overridden by parameters from [identities.conf](#).
- **db\_inputs.conf**: Configures database indexing behavior using modular inputs.
- **db\_lookups.conf**: Configures database lookup behavior using modular inputs.
- **db\_outputs.conf**: Configures database output behavior using modular inputs.
- **inputs.conf**: Keeps the Task Server running using modular inputs.

- **identities.conf**: Stores credentials DB Connect uses to connect to databases in the standard Splunk Enterprise credential store. (The credentials use obfuscated hash text.)

For more information about configuration files, see [About configuration files in the \*Splunk Enterprise Admin Manual\*](#). For more information about how Splunk Enterprise determines which configuration files take precedence, see [Configuration file precedence in the \*Splunk Enterprise Admin Manual\*](#).

## app-migration.conf.spec

The **app-migration.conf** file specifies options for [migrating from DB Connect v1 to DB Connect v2 using the migration scripts](#).

```
[<migrationId>]
# splunk identifies stanzas by their migrationId
# splunk uses a migrationId to decide which migration business logic to load and apply

STATE = <string>
# JSON encoded state store for all migrations of type <migrationId>

DEST_CONF = <string>
# by default the DEST_CONF is <migrationId>. Use if you need to override,
# for example mi_input:// , mi_output://, and mi_lookup:// are all in inputs.conf but
# require different business logic for migration.
```

## db\_connection\_types.conf.spec

The **db\_connection\_types.conf** file lists the supported database types, driver parameters, and test queries.

Compared to version 2.x, in DB Connect 3, the fields

- **cwallet\_location** and **oracle\_cipher\_suites** get merged into **connection\_properties**.
- Splunk has removed the **sslConnectionType**.

```
# @copyright@
# The file contains the specification for database connection types

[<name>]

serviceClass = <string>
# required
# Splunk Java class that provides JDBC support for this database connection type.

supportedVersions = <string>
# optional
# list (comma separated) supported versions(majorVersion.minorVersion) of JDBC driver implemented by
serviceClass
# installed driver major version must == MajorVersion and minor version >= MinorVersion for Splunk to
consider a driver as supported.

jdbcUrlFormat = <string>
# required
```



```

# JDBC Connection URL used for this database connection type. Supported variables: host, port, database,
informixserver.

jdbcUrlSSLFormat = <string>
# optional
# JDBC Connection URL used for this database connection type if you enabled jdbcUseSSL. Supported variables:
host, port, database, informixserver.

jdbcUseSSL = [true | false]
# optional
# default is false, whether this type of connection uses SSL connection.

jdbcDriverClass = <string>
# optional
# Driver vendor Java class that provides JDBC support for this database connection type.

testQuery = <string>
# optional
# simple SQL to test validation for this database type.
# JDBC 4 compliant drivers do not need this parameter.

displayName = <string>
# optional
# Describe the database connection type for end users.

database = <string>
# required if used in jdbcUrlFormat or jdbcUrlSSLFormat.
# JDBC URL variable for the default database or schema used for this database connection type.

port = <integer>
# required if used in jdbcUrlFormat or jdbcUrlSSLFormat.
# JDBC URL variable for the network port used for this database connection type.

useConnectionPool = [true | false]
# optional, default is true.
# The connection pooling uses Hikari, see https://github.com/brettwooldridge/HikariCP

ui_default_catalog = <string>
# optional
# ignored since 3.0

ui_default_schema = <string>
# optional
# ignored since 3.0

maxConnLifetimeMillis = <integer>
# optional, default is 120000 = 120 seconds
# valid when useConnectionPool = true
# The maximum lifetime in milliseconds of a connection in the connection pool.
# Splunk never retires an in-use connection, but only removes a connection when it gets closed.
# A value of zero means the connection has an infinite lifetime.

```

```

maxWaitMillis = <integer>
# optional, default is 30000 = 30 seconds
# valid when useConnectionPool = true
# The maximum number of milliseconds that the pool waits (when there are no
# available connections) for Splunk to return a connection before throwing an exception.
# [250, 300000] milliseconds is a reasonable value to wait to establish a connection.
# The max wait time is 300000 milliseconds (300 seconds).

maxTotalConn = <integer>
# optional, default is 8 connections
# valid when useConnectionPool = true
# The maximum number of active connections that Splunk allocates from this pool at the same time, or
# negative for no limit.

connection_properties = <string>
# optional
# Set JDBC variables for this database connection type.
# For example: {"useLegacyDatetimeCode": false}

```

## db\_connections.conf.spec

The **db\_connections.conf** file stores all configuration necessary for connecting to a specific database, unless overridden by parameters from [identities.conf](#).

Compared to DB Connect 2, in DB Connect 3, the fields

- **username** and **password** have been removed.
- **cwallet\_location**, **sslConnectionType** and **oracle\_cipher\_suites** have been removed, configure these fields in **connection\_properties**.

```

# @copyright#
# The file contains the specification for database connections

[<name>]

serviceClass = <string>
# optional
# inherits from db_connection_types.conf if not configured
# java class that serves the jdbc service for this type.

customizedJdbcUrl = <string>
# optional
# the user customized jdbc url used to connect to the database, empty or missing means use template to
# generate jdbc url.
# see jdbcUrlFormat and jdbcUrlSSLFormat defined in db_connection_types.conf.spec

jdbcUseSSL = [true | false]
# optional
# inherits from db_connection_types.conf if not configured
# default is false, whether this type of connection will support SSL connection.

```

```

jdbcDriverClass = <string>
# optional
# inherits from db_connection_types.conf if not configured
# java jdbc vendor driver class.

testQuery = <string>
# optional
# inherits from db_connection_types.conf if not configured, if still not provided, JDBC isValid API will be
used.
# simple SQL to test validation for this database type.

database = <string>
# required only when other parameters refer to.
# inherits from db_connection_types.conf if not configured
# The default database that the connection will use

connection_type = <string>
# required
# The type of connection configured in db_connection_types.conf that the connection refer to

identity = <string>
# required
# The database identity that the connection will use when connecting to the database
# an identity provides username and password for database connection.

isolation_level = <string>
# optional
# The transaction isolation level that the connection should use
# valid values are: TRANSACTION_NONE, TRANSACTION_READ_COMMITTED, TRANSACTION_READ_UNCOMMITTED,
TRANSACTION_REPEATABLE_READ, TRANSACTION_SERIALIZABLE and DATABASE_DEFAULT_SETTING
# default: DATABASE_DEFAULT_SETTING.

readonly = true|false
# optional
# default to false
# Whether the database connection is read-only. If it is read only, any modifying SQL statement will be
blocked

host = <string>
# required only when other parameters refer to.
# The host name or IP of the database server for the connection
# Possible variable from jdbcUrlFormat.

port = <integer>
# required only when other parameters refer to.
# inherits from db_connection_types.conf if not configured

```

```

# The TCP port number that the host database server is listening to for connections
# Possible variable from jdbcUrlFormat.

informixserver = <string>
# optional
# Required option for informix server to compose proper jdbc connection url.
# This attribute is used for informix server connection setup only.

useConnectionPool = true|false
# optional
# boolean to make connection use a connection pool
# defaults to true

connection_properties = <string>
# optional, differs via different databases.

fetch_size = <integer>
# optional, default is 100, the number of rows retrieved with each trip to the database.

maxConnLifetimeMillis = <value>
# optional, default is 120000 = 120 seconds
# valid when useConnectionPool = true
# The maximum lifetime in milliseconds of a connection. After this time is exceeded the connection will fail
the next activation, passivation or validation test.
# A value of zero or less means the connection has an infinite lifetime.

maxWaitMillis = <value>
# optional, default is 1800000 = 1800 seconds
# valid when useConnectionPool = true
# The maximum number of milliseconds that the pool will wait (when there are no available connections) for a
connection to be returned before throwing an exception.
# [250, 300000] milliseconds is a reasonable value to wait to establish a connection. The max wait time is
300000 milliseconds (300 seconds).

idleTimeout = <integer>
# optional, default is 600000 = 10 minutes
# The maximum amount of time that a connection is allowed to sit idle in the
# pool in milliseconds.
# valid when useConnectionPool = true and when minIdle is defined to be less
# than maxTotalConn.
# Whether a connection is retired as idle or not is subject to a maximum
# variation of +30 seconds, and average variation of +15 seconds.
# A connection will never be retired as idle before this timeout.
# Once the pool reaches minIdle connections, connections will no longer be
# retired, even if idle.
# A value of 0 means that idle connections are never removed from the pool.
# The minimum allowed value is 10000ms (10 seconds).

maxTotalConn = <value>
# optional, default is 8 connections
# valid when useConnectionPool = true
# The maximum number of active connections that can be allocated from this pool at the same time, or
negative for no limit.

timezone = <time zone identifier>

```

```
# optional, default uses JVM time zone
# The identifier could be:
# an offset from UTC/Greenwich, that uses the same offset regardless given date-time e.g. +08:00
# an area where a specific set of rules for finding the offset from UTC/Greenwich apply e.g. Europe/Paris.

localTimezoneConversionEnabled = [true | false]
# optional, default is false
# valid when a time zone is set
# When turned on, time-related fields are read from the DB using the configured time zone and then
translated to the JVM time zone.
# For example, with a DB using UTC and the JVM using GMT+8, the datetime field defined in the DB
# 2017-07-21 08:00:00 will be displayed 2017-07-21 16:00:00
```

## Example:

```
[oracle_example]
connection_type = oracle
database = orcl
disabled = 0
host = oracle.host.com
identity = oracle_identity
jdbcUseSSL = true
port = 2484
readonly = false
customizedJdbcUrl =
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps) (HOST=oracle.host.com) (PORT=2484)) (CONNECT_DATA=(SERVICE
_NAME=orcl)))
connection_properties = {"oracle.net.authentication_services": "(TCPS)", "oracle.net.ssl_cipher_suites":
"(SSL_DH_anon_WITH_3DES_EDE_CBC_SHA, TLS_RSA_WITH_AES_128_CBC_SHA)", "javax.net.ssl.trustStore":
"/opt/splunk/cwallet/cwallet.sso", "javax.net.ssl.trustStoreType": "SSO", "javax.net.ssl.keyStore":
"/opt/splunk/cwallet/cwallet.sso", "javax.net.ssl.keyStoreType": "SSO"}

[informix_example]
connection_type = informix
database = test
disabled = 0
host = informix-demo.host.com
identity = informix_win
informixserver = demo_on
jdbcUseSSL = false
port = 9088
readonly = false
customizedJdbcUrl =
jdbc:informix-sqli://informix-demo.host.com:9876/test:informixserver=demo_on;DELIMIDENT=Y
```

## database\_typespec

## db\_inputs.conf.spec

```
[<name>]
description = <string>
# optional
# Description for this input

interval = <integer|string>
# required
# interval to fetch data from DB and index them in Splunk
```

```

# It could be a number of seconds or a cron expression

index = <string>
# optional
# index to store events imported in Splunk
# If not specified default index is used
# default to 'default'

source = <string>
# optional
# source associated to events indexed
# By default, the stanza name will be used

sourcetype = <string>
# required
# source type associated to events indexed

host = <string>
# optional
# host associated to events indexed

mode = [batch|rising|advanced]
# required
# Operational mode.
# advanced is deprecated since DBX v3.1, use rising instead.

connection = <string>
# required
# Indicates the database connection to work on.

query = <string>
# required
# SQL statement to retrieve data from remote database connection.

query_timeout = <integer>
# optional
# the max execution time of a SQL, the default is 30 seconds.

max_rows = <integer>
# optional
# the max rows of data retrieval. The default is 0, which means unlimited.

fetch_size = <integer>
# optional
# The number of rows to return at a time from the database. The default is 300.

batch_upload_size = <integer>
# optional
# Number of rows to be uploaded to HEC in one batch. Default is 1000.

tail_rising_column_name = <string>
# optional if batch mode
# at tail mode, the rising column is the column which is always rising as the checkpoint of the tail
loading.
# when tail_rising_column_name and tail_rising_column_number both exist, the column name has higher
priority,
# unless by using it, the code fails to locate the value of the rising value in which case, the code will
# try to use column number

tail_rising_column_fullname = <string>
# deprecated since DBX v3.1.0, using tail_rising_column_number instead.
# optional if batch mode

```

```

# fullname of input tail rising column, currently this value is used by front end only.

tail_rising_column_number = <integer>
# required for advanced mode
# 1-based position of rising column in the data loading.

tail_rising_column_init_ckpt_value = <json>
# internal used only
# Do not update this manually

input_timestamp_column_name = <string>
# deprecated since DBX v3.1.0, using input_timestamp_column_number instead
# optional
# the input timestamp column name, the data of this column will be the event time. If not set, dbinput will
use the current timestamp as the event time.

input_timestamp_column_fullname = <string>
# deprecated since DBX v3.1.0, using input_timestamp_column_number instead
# optional
# fullname of input timestamp column, currently this value is used by front end only.

input_timestamp_column_number = <integer>
# optional
# 1-based column number of input timestamp.

input_timestamp_format = <string>
# optional
# Specifies the format of input timestamp column, in JavaSimpleDateString format.
# with index_time_mode as dbColumn, if this property is provided then DBX will use ResultSet#getString to
get the value
# and try to parse the timestamp with given format, else if this property is not present DBX will try to use
ResultSet#getTimestamp
# to get the timestamp.

index_time_mode = [current|dbColumn]
# required
# Specifies how to set the index time.
# current: use current time as index time
# dbColumn: use a DB column as index time.

max_single_checkpoint_file_size = <integer>
# optional
# Max checkpoint file size before archiving checkpoint file in bytes. Default is 10MB, max is 100MB.

ui_query_mode = [simple|advanced]
# deprecated since DBX v3.1.0, it is ignored by the UI
# optional
# Specifies whether the ui should use simple mode or advanced mode for SQL queries

ui_query_catalog = <string>
# deprecated since DBX v3.1.0, it is ignored by the UI
# optional
# in simple mode, this value will be pre-populated into the catalog dropdown

ui_query_schema = <string>
# deprecated since DBX v3.1.0, it is ignored by the UI
# optional
# in simple mode, this value will be pre-populated into the schema dropdown

```

```

ui_query_table = <string>
# deprecated since DBX v3.1.0, it is ignored by the UI
# optional
# in simple mode, this value will be pre-populated into the query dropdown

template_name = <string>
# optional
# if provided and the value is not empty it means this input is created based on a template and the
# value is the name of the template.

input_type = [event|metric]
# optional
# Defines which type of metric it is. If not given default value is event.

```

## **db\_input\_templates.conf.spec**

```

[<name>]
description = <string>
# optional
# short description about the input template

interval = <integer|string>
# required
# interval to fetch data from DB and index them in Splunk
# It could be a number of seconds or a cron expression

sourcetype = <string>
# required
# source type associated to events indexed

mode = [batch|rising]
# required
# Operational mode.

query = <string>
# required
# SQL statement to retrieve data from the database

rising_column_index = <integer>
# required in rising mode
# The column number of the rising column you specified (1-based).

input_timestamp_column_index = <integer>
# required when the index_time_mode is dbColumn
# The column number of the timestamp column you specified (1-based).

input_timestamp_format = <value>
# optional
# specify the format of input timestamp column, in JavaSimpleDateString format.
# with index_time_mode as dbColumn, if this property is provided then DBX will use ResultSet#getString to
get the value
# and try to parse the timestamp with given format, else if this property is not present DBX will try to use
ResultSet#getTimestamp
# to get the timestamp.

index_time_mode = [current|dbColumn]
# required
# Specifies how to set the index time.
# current: use current time as index time
# dbColumn: use a DB column as index time.

```



```

input_type = [event|metric]
# optional
# Defines which type of metric template tracks. If not given default value is event.

connection_type = <string>
# optional
# Defines for which connection type the template will be shown. If empty - will be shown for all
connections.

```

## db\_outputs.conf.spec

```

[<name>]
description = <value>
# optional
# Description for this output

interval = <string>
# required
# Specifies the interval to fetch data from Splunk and export to a database
# Valid only if scheduled is enabled
# It could be a number of seconds or cron expression

connection = <string>
# required
# Specifies the database connection to persist splunk data.

table_name = <string>
# required
# Specifies the table name used to export data.

using_upsert = [true|false]
# optional
# Specifies if upsert is used when exporting data to the table

unique_key = <string>
# required if upsert is enabled
# Specifies the column used as key to verify if the row needs to be inserted or updated.

query_timeout = <integer>
# optional
# Specifies the timeout (in seconds) for the SQL statement while exporting data.
# If the database supports batch operations, the timeout may apply to the whole
# batch or to each single statement. This behavior depends on the JDBC driver.
# For more information please consult the documentation of the driver about the
# following API Statement#setQueryTimeout.
# Default to 30 seconds.
# 0 means unlimited.

time_out = <integer>
# deprecated since DBX v3.1, use search_time_out and query_timeout instead.

search_timeout = <integer>
# optional
# Specifies the max time (in seconds) for a search job to run, if the search
# takes longer than this limit the job will be finalized then only part of the
# data will be exported.
# default is 0, which means unlimited.

search = <string>
# required
# Specifies the splunk search to pull data for output.

```

```

# If is_saved_search is enabled then this value specifies the saved search name.

batch_size = <string>
# optional
# Specifies the size of a batch.
# If it is not provided, the default value is 1000.

query_earliest_time = <string>
# optional
# Specifies the earliest time applied to the search.
# The format is defined by Splunk time modifiers.
# default to null, which means no lower bound limit.

query_latest_time = <string>
# optional
# Specifies the latest time applied to the search.
# The format is defined by Splunk time modifiers.
# default to null, which means no upper bound limit.

is_saved_search = [true | false]
# optional
# specify the max time (in seconds) a search job can run, if time exceeded then search job will
# be finalized then partial events will be outputted.
# default is 0, which means unlimited.

customized_mappings = <string>
# required
# Specifies the output data name (fieldx) and database column number (1...n) mappings.
# The expected format is:
#   field1:column1:type1,field2:column2:type2,â,fieldN:columnN:typeN
# For compatibility reason the following format is also supported but is deprecated:
#   field1:column1,field2:column2,â,fieldN:columnN

scheduled = [true|false]
# required
# Specifies whether the output is scheduled.
# An unscheduled output is typically used in dbxoutput command context.

ui_mappings = <string>
# deprecated since DBX v3.1.0, it is ignored by the UI
# optional
# JSON mappings, purely for storage purposes

ui_selected_fields = <string>
# deprecated since DBX v3.1.0, it is ignored by the UI
# optional
# JSON array of selected fields, purely for storage purposes

ui_saved_search_str = <string>
# deprecated since DBX v3.1.0, it is ignored by the UI
# optional
# saved search string of the current saved search

ui_query_sql = <string>
# deprecated since DBX v3.1.0, it is ignored by the UI
# optional

ui_query_mode = [simple|advanced]
# deprecated since DBX v3.1.0, it is ignored by the UI
# optional

ui_query_catalog = <string>

```

```
# optional
# Specifies the table used to initialize the UI editor

ui_query_schema = <string>
# optional
# Specifies the schema used to initialize the UI editor

ui_query_table = <string>
# optional
# Specifies the table used to initialize the UI editor
```

## Example

```
[test_output]
description = this is a test output
interval = 40 * * * *
connection = test_connection
table_name = `test_table`
using_upsert = 0
search = index=main | stats count(*) by test_column
is_saved_search = 0
time_out = 6000
customized_mappings = test_column:varchar_col:12
ui_mappings =
[{"inputField":"test_column","outputField":"test_output_column","valType":"VARCHAR","sqlType":"12"}]
ui_selected_fields = [{"value":"skip","label":"Skip this
Column"}, {"value":"test_column","label":"test_column"}]
ui_query_catalog = test_catalog
ui_query_schema = NULL
ui_query_table = output_test
```

## db\_lookups.conf.spec

```
[<name>]
description = <string>
# optional
# Description for this lookup

lookupSQL = <string>
# required
# Specifies the SQL query for lookups.

connection = <string>
# required
# Specifies the database connection to use.

output_column_map = <string>
# required
# Key/value pairs of database columns to search result column in JSON format.

input_column_map = <string>
# required
# Key/value pairs of search result column to database column in JSON format.

ui_input_spl_search = <string>
# optional
# the splunk spl search which will be used for choosing lookup input_fields

ui_input_saved_search = <string>
# optional
```

```

# the splunk saved search which will be used for choosing lookup input_fields

ui_use_saved_search = [true|false]
# optional
# if true, then the UI will use ui_input_saved_search
# if false, then the UI will use ui_input_spl_search

input_fields = <string>
# deprecated since DBX v3.1.0, it is replaced by input_column_map
# Specifies the input fields for lookups.

output_fields = <string>
# deprecated since DBX v3.1.0, it is replaced by output_column_map
# Specifies the output fields after lookups.

ui_query_mode = [simple|advanced]
# deprecated since DBX v3.1.0, it is ignored by the UI
# optional
# specify whether the ui should use simple mode or advanced mode for SQL queries

ui_query_catalog = <string>
# deprecated since DBX v3.1.0, it is ignored by the UI
# optional
# in simple mode, this value will be pre-populated into the catalog dropdown

ui_query_schema = <string>
# deprecated since DBX v3.1.0, it is ignored by the UI
# optional
# in simple mode, this value will be pre-populated into the schema dropdown

ui_query_table = <string>
# deprecated since DBX v3.1.0, it is ignored by the UI
# optional
# in simple mode, this value will be pre-populated into the query dropdown

ui_column_output_map = <string>
# deprecated since DBX v3.1.0, it is ignored by the UI
# optional
# JSON mapping from db result column to field name

ui_field_column_map = <value>
# deprecated since DBX v3.1.0, it is ignored by the UI
# optional
# JSON mapping from search result field to db column

ui_query_result_columns = <value>
# deprecated since DBX v3.1.0, it is ignored by the UI
# optional
# JSON encoded array of query result columns
# stores the columns from the associated lookupSQL

```

### Example:

```

[test_lookup]
lookupSQL = SELECT * FROM `sakila`.`actor`
connection = test_connection
input_fields = test_input_field
output_fields = actor_id
ui_query_mode = simple
ui_query_catalog = sakila
ui_query_schema = NULL

```

```

ui_query_table = actor
ui_input_spl_search = index=main | stats count(*) by test_input_field
ui_use_saved_search = 0
ui_query_result_columns =
[{"name":"actor_id"}, {"name":"first_name"}, {"name":"test_input_field"}, {"name":"last_update"}]
ui_column_output_map =
[{"removable":false, "label":"actor_id", "value":"actor_id", "name":"actor_id", "alias":"output_actor_id"}]
ui_field_column_map =
[{"name":"test_input_field", "selected":true, "removable":true, "label":"test_input_field", "value":"test_input_field", "alias":"test_input_field"}]

```

## identities.conf.spec

The **identities.conf** file stores credentials used to connect to databases in the standard Splunk credential store (obfuscated hash text).

```

# @copyright@
# The file contains the specification for database identities (username/password)

[<name>]

username = <string>
# required
# the username for this database connection identity

password = <string>
# required
# The encrypted value of the password for this database connection identity.

domain_name = <string>
# optional
# Specifies the windows domain name which the username belongs to

use_win_auth = [true|false]
# optional
# Specifies whether the Windows Authentication Domain is used

```

### Example

```

[mysql_local]
username = admin
password = UdxsEmbJvU/lcINqMVGAQgBKT4DLIx/2c9Ka+3oUOVy=
use_win_auth = false

```

## inputs.conf.spec

Instead of configuring inputs, outputs and lookups in one **inputs.conf** file, in DB Connect 3, inputs, outputs and lookups are configured in discrete files **db\_inputs.conf**, **db\_outputs.conf** and **db\_lookups.conf**. This increases configurability and performance.

**Note:** Modifying **inputs.conf** file stanzas outside of the DB Connect app, such as in the search app or manager context, is not supported.

```

# @copyright@
# The file contains the specification for all db connect modular inputs
# # server - the modular input that runs java server

```

```
[server://<name>]
config_file = <string>
# required
# If the value is an absolute path, taskserver will treat it as the config_file path.
# If the value is a relative path, taskserver will prepend SPLUNK_HOME and generate the actual config_file path.
# On UNIX systems, a path is absolute if its prefix is "/".
# On Microsoft Windows systems, a path is absolute if its prefix is a drive specifier followed by "\\ ", or if its prefix is "\\\\".

keystore_password = <string>
# required
# Specifies the java keystore password
# The keystore contains the certificate to set up the HTTPS connection to the
# task server.

interval = <integer>
# required
# Specifies the interval used by Splunkd to monitor Java servers. If Java process
# is stopped, Splunkd will spawn a new Java process.
```

## dbx\_settings.conf.spec

The *dbx\_settings.conf* file contains DB Connect settings such as the **\$JAVA\_HOME** path, the JVM Options string, and Splunk usage collection. These settings can also be configured in DB Connect on the [Settings page](#).

```
# @copyright@

[java]
javaHome = <string>
# optional
# Specifies the path to Java home. The path defined will be used to resolve the
# java command location. Typically, the /bin/java or \bin\java.exe will be
# appended to this path to resolve the command location on *nix and windows
# respectively.
# If not specified the JAVA_HOME environment variable is used.
# If JAVA_HOME is also undefined, the JAVA command is directly used. In that case,
# java command must be defined in the PATH environment variable.

[loglevel]
dbxquery = [TRACE | DEBUG | INFO | WARN | ERROR]
# set the log level for dbxquery command

dbxoutput = [TRACE | DEBUG | INFO | WARN | ERROR]
# set the log level for dbxoutput command

dbxlookup = [TRACE | DEBUG | INFO | WARN | ERROR]
# set the log level for dbxlookup command

dbinput = [TRACE | DEBUG | INFO | WARN | ERROR]
# set the log level for db inputs

dboutput = [TRACE | DEBUG | INFO | WARN | ERROR]
# set the log level for db outputs

connector = [TRACE | DEBUG | INFO | WARN | ERROR]
# set the log level for all the interactions with DBs

[hec]
# when HEC failed to forward data to Splunk Indexer, it will return 503 error,
# then DBX will retry 3 times by default with exponential backoff
```

```

maxRetryWhenHecUnavailable = <integer>

# maxHecContentLength setup max event size, default is 10MB
maxHecContentLength = <integer>

# HEC URIs: https://<idx-host1>:8088,https://<idx-host2>:8088,...
hecUris = <comma-seperated-hec-uri>

# HEC token for HEC URIs above
hecToken = <hec-token>

# Lower case field name for backward compatibility with dbx 1.x
# By default it is false
hecFieldNameLowerCase = <true|false>

```

## Saved searches for Splunk DB Connect

Splunk DB Connect includes the following saved searches.

To enable or disable a saved search:

1. From the **Settings** menu, choose **Searches, reports, and alerts**.
2. Under the **Action** column of the saved search, choose **Edit > Enable/Disable** to enable or disable it.

Name	Purpose	Action required
splunk_app_db_connect.apimetricscollector: calculate indexed data volume	Calculates the amount of data indexed in your Splunk platform by sourcetype.	Automatically enabled. No action required. Scheduled to run once daily at twenty minutes past midnight.

## Data models for Splunk DB Connect

Splunk DB Connect includes four data models to support health and performance dashboards.

Name	Purpose	Accelerated	Action required
DBX_Job_Metrics	Supports the DB Connect Input Performance dashboard.	No	None
DBX_Health_Metrics	Supports Task server health providing JVM heap size, REST API metrics, etc	No	None
DB_Connection_Metrics	Supports the DB Connect Connection Health dashboard.	No	None
DB_Health_Audit	Supports the DB Connect Connection Health dashboard.	No	None

# Troubleshooting

## Troubleshooting Tool for DB Connect

Splunk provides you with this command line utility that goes with Splunk DB Connect that enables you to troubleshoot basic configuration problems from your network, database, and DB Connect, saving you time and effort.

### Functions

- Test connectivity between DBX and DB instances
- Test DB credentials
- Test JDBC driver compatibility
- Test JVM options

### Current Limitations

The self-service troubleshooting tool has these current limitations.

- Test scheduled inputs
- Test DBX search commands(dbxquery, dbxlookup, dbxoutput)
- Run SQL targeting on a DB instance
- Data loss/duplication
- Performance issues
- HEC errors

The tool can only troubleshoot issues related to connectivity for DB Connect.

### Applicability

Use this tool if the following applies to you:

- Other database clients show expected results but there are errors with DB Connect
- You see DB connect errors in the configuration UI or logs

## Set Up Troubleshooting Tool for DB Connect

The troubleshooting tool for DB Connect enables you to use a command line utility for troubleshooting your own configuration for DB Connect, saving you time and effort.

### Compatible Databases

- Oracle
- MS SQL
- Postgres
- MySQL
- MemSQL



## Prerequisites

- You have an existing connection in DB Connect (click 'save anyway' even if there is an error).
- You have added the JDBC driver for your connection to the 'drivers' folder in DB Connect.
- You have CLI access to the instance you want to test.
- You should have Java 8 or higher JDK installed and can run the 'javac' command from the CLI.

## Configuration Steps

These are the troubleshooting tool configuration steps for both Windows and Mac/Linux.

### *On Windows*

1. Navigate to the `$SPLUNK_HOME\etc\apps\splunk_app_db_connect\bin` directory in DB Connect.
2. Make sure the following environment variables are defined and have the following paths:
  1. JAVA\_HOME: C:\Program Files\Java\jdk1.8.0\_79\bin
  2. PATH: C:\Program Files\Java\jdk1.8.0\_79\jre\bin\server
  3. PATH: C:\Program Files\Java\jdk1.8.0\_79\bin
  4. Download and install Microsoft Visual C++ Build Tools (command-line tools subset of Visual Studio) for Python 3.5 and 3.6: <https://visualstudio.microsoft.com/downloads/>

### *On Mac/Linux*

1. Navigate to the `$SPLUNK_HOME/etc/apps/splunk_app_db_connect/bin` directory in DB Connect.

### *Access the troubleshooting tool*

1. Run 'python dbx\_connection\_diag.py' or 'python3 dbx\_connection\_diag.py' or 'py dbx\_connection\_diag.py' depending on how your python3 command is set up.
2. Choose the JDBC driver associated with the connection you wish to test. The utility should show all of the drivers in the DB Connect 'Drivers' directory.
3. Provide the credentials for the local Splunk instance. The user will need to be an administrator.
4. After verifying that DB Connect is installed on the local Splunk instance, you will be shown the connections available for testing. Choose the connection you want to test.
5. Provide the database password for the user shown (This will be the user associated with this database connection).
6. The utility will attempt to ping the database server. If successful, you will be shown the current settings for the database connection and be given a chance to modify them.
7. The results of the test along with diagnosis and recommendations will display.

## Common issues for Splunk DB Connect

Troubleshoot common Splunk DB Connect issues using the Troubleshooting Tool for DB Connect. If the Troubleshooting Tool for DB Connect does not solve your issue, try the following steps.

## Answers

Have questions? In addition to these troubleshooting tips, go to Questions related to Splunk DB Connect on Splunk Answers to see what questions and answers the Splunk community has about using Splunk DB Connect.

## Health dashboard

When you're trying to figure out the cause of degraded performance or figure out how failure rates correspond to transaction type or database user, the place to start is the health dashboard in Splunk DB Connect.

The health dashboard is a pre-configured dashboard that let's you monitor and troubleshoot several aspects of your database connections from inside Splunk Enterprise. For more information about the health dashboard, see [Monitor database connection health](#).

You can also see whether DB Connect is generating any internal errors, using a search with the following parameters:

```
index=_internal sourcetype="dbx*" error
```

### Health dashboard shows "Permission denied" error message

If the [health dashboard](#) displays a "Permission denied" error message instead of any data, the problem is likely a permissions issue with the logged-on user.

A logged-on user must have an assigned role that has access to both the `_internal` index and `"dbx_health"` source type in order to see the health dashboard. If none of the roles you assign a user to has permission to either search `_internal` or view data with `sourcetype="dbx_health"`, the dashboard displays the "Permission denied" error.

## DB Connect logging

Splunk DB Connect has extensive logging options, which you can configure in [Settings](#). Before contacting Splunk support, you might want to enable debug logging, in case you need to provide Splunk support with DB Connect debug logs. Splunk logs DB Connect activity to files in `$SPLUNK_HOME/var/log/splunk` and automatically indexed to `_internal`. The relevant log files for DB Connect are

- `splunk_app_db_connect_server.log`
- `splunk_app_db_connect_server_access.log`
- `splunk_app_db_connect_job_metrics.log`
- `splunk_app_db_connect_health_metrics.log`
- `splunk_app_db_connect_dbx.log`
- `splunk_app_db_connect_audit_server.log`
- `splunk_app_db_connect_audit_command.*.log`

To view DB Connect logging activity, use a search command such as the following:

```
index=_internal sourcetype=dbx*
```

You can control access to logged events by limiting access to the `_internal` index using Splunk Enterprise roles. For example, by default non administrators can't access `_internal`. Database and Splunk Enterprise administrators work together to determine the optimal logging setup for their environment and decide how to handle special scenarios like troubleshooting. For more information about what Splunk Enterprise logs about itself, see "What Splunk logs about itself" in the *Splunk Enterprise Troubleshooting Manual*.

By default, DB Connect logs all executed SQL queries at INFO level. Logged along with each SQL query is the user that ran the query, the parameters, the number of results returned, and the name of the input, output, or lookup that Splunk ran.

## Troubleshoot driver connections

If you're having trouble connecting to your database using Splunk DB Connect or loading your database's JDBC driver, run the following checks before contacting Splunk support:

- Is DB Connect running in Splunk Enterprise?
- Is Splunk Enterprise working correctly?
- Is the database working correctly?
- Verify you've installed the driver correctly by repeating the steps in "[Install database drivers](#)".
- Verify that Splunk supports the driver and the driver version by looking for them in [Supported databases](#). If necessary, download a newer version of the driver.
- Are you loading the correct JDBC driver? To find out, in DB Connect, select **Configuration** in the top navigation bar, **Settings**, and **Drivers** to view the driver status screen. You can also search the `_internal` index for your specific driver's .JAR file. For example: `index=_internal sourcetype=dbx_server mysql-*.jar`
- Can you access the database with your browser or database client?
  - ◆ To test the connection to an Oracle database, use the Toad Java viewer.
  - ◆ To talk to a Postgres database, use the Postgres tool.
  - ◆ To talk to other types of databases, try DBVisualizer.

Use one of these tools to connect to the database and ensure that connectivity is good, host and port are correct, and that your credentials work. Then, copy those settings to DB Connect and try again.

## Issues with bad line breaking/line merging

Splunk line break heuristics is what causes this problem. Typically, log file data includes event timestamps, which Splunk understands. If you have timestamps in your database rows, there won't be any line break issues. Be sure to set output timestamp and confirm that the timestamp column is the actual timestamp column.

### *If you don't have timestamps in your database rows*

If you don't have timestamps in your database rows, you have to set the [timestamp option](#) to **Current Index Time** when creating an input.

### *If your timestamp is not of type datetime/timestamp*

Splunk DB Connect expects the timestamp column in your database to be of type **datetime/timestamp**. If it is not (for example, it is in format **char/varchar/etc.**), you can first try to convert the SQL statement into the correct type using the `CAST` or `CONVERT` functions. If this method doesn't work, you can use the following workaround:

In the [Set timestamp](#) step when creating or editing an input, next to **Timestamp**, select the **Choose Column** setting. Then, select the column from the drop down menu. Next, select the **Datetime Format** option, and then select the timestamp using a Java **DateTimeFormatter** pattern so DB Connect can obey the timestamp output format setting. For example, if the database column `EVENT_TIME` which is the 5th columns from the result contains strings, such as `CHAR`, `VARCHAR`, or `VARCHAR2`, with values like `01/26/2013 03:03:25.255`, you must enter the parse format in the appropriate copy of `db_inputs.conf`.

```
index_time_mode = dbColumn
```

```
input_timestamp_column_number = 5
input_timestamp_format = MM/dd/yyyy HH:mm:ss.SSS
```

## Unexpected session key expiration

A system clock change or suspend/resume cycle can cause unexpected session key expiration. To remedy the problem, restart the Splunk Enterprise system using DB Connect.

## Cannot connect to IBM DB2

Splunk supports IBM DB2 only when the database is running on Linux. Splunk doesn't test or support DB Connect with DB2 on AS/400 or on Microsoft Windows.

## Cannot connect to Microsoft SQL server

If you cannot connect to a Microsoft SQL server, ensure you've correctly followed the steps listed in [Microsoft SQL Server](#). Specifically, verify that you are using the correct driver, host, and port:

- **Host:** To enter a host for Microsoft SQL, use a fully qualified domain name, a short name, or an IP address. Do not use the Microsoft SQL convention of <SERVERNAME\DATABASE> for the host field.
- **Port:** Many Microsoft SQL Servers use dynamic ports instead of TCP/1433. Work with your database administrator to identify the correct port, or see "Verifying the port configuration of an instance of SQL Server" on the Microsoft website.

For more information about using Windows authentication with DB Connect and Microsoft SQL Server, see "[Connecting to SQL Server](#)" in the [Install database drivers](#) topic.

## Cannot connect to Oracle SQL Server

If you cannot connect to a Oracle database, first ensure you've correctly followed the steps listed in [Oracle database](#).

### *Connect to Oracle using SSL (for encryption only)*

If you're having trouble connecting to Oracle using SSL for encryption:

1. First verify whether the connection works using an external tool that uses JDBC to connect to Oracle, such as DBVisualizer, with the following JDBC URL. Replace the <host>, <port>, and <database> placeholders with their actual values in your setup.  

```
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps) (HOST=<host>) (PORT=<port>)) (CONNECT_DATA=(SERVICE_NAME=<database>)))
```

 If the connection does not work, work with your database admin to correctly configure SSL for encryption.
2. If the connection works, select the **Edit JDBC URL** button and enter the JDBC URL in the **JDBC URL** field. See more details about the JDBC URL settings, see [create a database connection](#).
3. Restart Splunk Enterprise.

### **Oracle Error Codes**

If you receive an error attempting to connect to an Oracle database, try the following. The most common error codes are:

#### **ORA-12504: TNS:listener was not given the SID in CONNECT\_DATA**

This error means that the SID was missing from the CONNECT\_DATA configuration. To troubleshoot, check that the connect descriptor corresponding to the service name in **TNSNAMES.ORA** also has an SID component in the CONNECT\_DATA stanza.

#### **ORA-12505: TNS:listener does not currently know of SID given in connect descriptor**

You are receiving this error because the listener received a request to establish a connection to the Oracle database, but the SID for the instance either has not yet dynamically registered with the listener or has not been statically configured for the listener. Typically, this is a temporary condition that occurs after the listener has started, but before the database instance has registered with the listener.

To troubleshoot, try waiting a few moments and try the connection again. Confirm which instances are currently known by the listener by executing: `lsnrctl services <listener name>`

#### **ORA-12514: TNS:listener does not currently know of service requested in connect descriptor**

This error occurs because the listener received a request to establish a connection to the database. The connection descriptor received by the listener specified a service name for a service that either has not yet dynamically registered with the listener or has not been statically configured for the listener.

To troubleshoot, try waiting a few moments and try the connection again. Confirm which instances are currently known by the listener by executing: `lsnrctl services <listener name>`

### ***Explanation of Oracle TNS Listener and Service Names***

TNS is a proprietary protocol developed by Oracle. It provides a common interface for all industry-standard protocols and enables peer-to-peer application connectivity without the need for any intermediary devices.

DB Connect uses Java (through JDBC driver) to connect Splunk Enterprise to a TNS Listener, which in turn connects to the Oracle Database. You can configure DB Connect to connect by using the Service Name or the Oracle SID. Typically, most connectivity issues with DB Connect and Oracle Databases come from a misconfiguration of the TNS Listener.

### **Splunk does not honor settings in local .conf files**

If you notice that Splunk is not honoring the custom settings you've specified in .conf files in your DB Connect **/local** directory, here are a few things to try:

- First, be aware that settings specified within a stanza in a .conf file that resides in the **/local** DB Connect directory take precedence over the corresponding entry in the same .conf file in the **/default** DB Connect directory. For more information, see Configuration file precedence.
- Stanza names and .conf file names must match *exactly* in both the **/default** and **/local** directories. The most common cause of this problem is misspellings in stanza names.

### **Missing instances of Microsoft SQL Server**

If you have multiple instances of Microsoft SQL Server installed on your server, edit the JDBC connection string to add a parameter that explicitly references the instance you want to contact.

1. Follow the instructions in "[Override db\\_connection\\_types.conf](#)" to make a copy of the **db\_connection\_types.conf** file in the **local** directory and copy the stanza for the Microsoft SQL Server driver

you're using into the file.

2. Edit the `jdbcUrlFormat` or, if you're connecting using SSL, `jdbcUrlSSLFormat` setting by appending it with the following: `;instanceName=`
3. Set the `instanceName` parameter to the name of the instance you want to connect to. For example:  
`jdbc:sqlserver://dbx-sqlserver.mydomain.com:1433;databaseName=master;instanceName=test`
4. Save and close the file, and then restart Splunk Enterprise.

## Splunk extracts incomplete field values when the value contains double quotes

When DB Connect encounters a column value during search-time field extraction that contains double quotation marks, it extracts the value, but stops extracting at the first quote. For example, consider this value named `string`:

```
string="This is "an extraordinary" event, not to be missed".
```

DB Connect extracts this as follows:

```
string="This is "
```

Splunk ignores the rest of the value.

This occurs because auto-kv extraction doesn't handle quotation marks inside fields. If you have quotation marks inside fields, you need to create field extractions for the datasource. For more information about search-time field extractions, see [Create and maintain search-time field extractions through configuration files](#).

## Splunk skips rows with invalid timestamp value during indexing

You might encounter two scenarios in which the input fails because of invalid timestamp values:

- The data type of the timestamp column is `DATE`, `TIME` or `TIMESTAMP` and it contains `NULL`.
- The data type of the timestamp column is not `DATE`, `TIME` or `TIMESTAMP`, and it returns invalid value after you convert it to Java date format.

The following screen shot is of the Configure Timestamp Column window from [the third step of database input setup](#). The data format of the highlighted **last\_update** column is `VARCHAR`. You need to set the data format in **Datetime format** under the table.

Configure Timestamp Column

×

Select the timestamp column below:

10 per Page ▾

« prev 1 2 3 4 5 6 7 8 9 10 next »

	actor_id (SMALLINT UNSIGNED) ▾	first_name (VARCHAR) ▾	last_name (VARCHAR) ▾	last_update (VARCHAR) ▾
1	1	PENELOPE	GUINNESS	2006-02-15 04:34:33.0
2	2	NICK	WAHLBERG	2006-02-15 04:34:33.0
3	3	ED	CHASE	2006-02-15 04:34:33.0
4	4	JENNIFER	DAVIS	2006-02-15 04:34:33.0
5	5	JOHNNY	LOLLOBRIGIDA	2006-02-15 04:34:33.0
6	6	BETTE	NICHOLSON	2006-02-15 04:34:33.0
7	7	GRACE	MOSTEL	2006-02-15 04:34:33.0
8	8	MATTHEW	JOHANSSON	2006-02-15 04:34:33.0
9	9	JOE	SWANK	2006-02-15 04:34:33.0
10	10	CHRISTIAN	GABLE	2006-02-15 04:34:33.0

Datetime format

eg: dd/MM/yyyy HH:mm:ss.SSS

As selected column is not one of native datetime types (DATE, TIME, TIMESTAMP etc.), you must either specify a format string (in the style of Java's SimpleDateFormat) or cast the column to timestamp manually in your SQL.

Cancel

Set

==Upgrade on Windows fails

If during upgrade of DBX on Windows errors like "The process cannot access the file because it is being used by another process", disable the application first and then run the upgrade.

## Unicode decode errors

If while using Splunk DB Connect you see Unicode decoding errors, you might not have set up the database to return results encoded in UTF-8. For example, you might see errors in Splunk Enterprise that include the following line if your database is returning results that are not UTF-8-encoded:

```
UnicodeDecodeError: 'utf8' codec can't decode byte 0xe4 in position 30: invalid continuation byte
```

Splunk DB Connect requires you to set up your database connection to return results encoded in UTF-8. Consult your database vendor's documentation for instructions about how to do this.

## DB Connect shows a task server communication error

When you open DB Connect, the RPC service attempts to bind to port 9998 by default. If port 9998 is already in use by another service, Splunk displays an error to load your configurations.

To work around this, you can change the port number to which the RPC service binds. To do this:

1. Go to the settings page from the configuration menu item
2. Edit the Task server port field to set an available port
3. Save your settings

4. The server restarts using this new port setting

## Web interface timeouts

If you are experiencing timeouts while using the Splunk DB Connect interface, and hardware and network performance are not an issue, consider increasing the user session timeout values as described in *Configure user session timeouts* in the *Splunk Enterprise Admin Manual*.

## DB Connect opens a lot of DB connections

Scheduled jobs such as inputs and outputs use and reuse connections from a connection pool (this pool size is 8 by default, see `maxTotalConn` and `useConnectionPool` in [db\\_connections.conf](#) to change connection pooling behavior for scheduled jobs). Interactive DBX search commands (`dbxquery`, `dbxlookup`, `dbxoutput`) support the use of the connection pool mechanism.

## Debug HTTP Event Collector port issues

In DB Connect 3.0.0, the architecture changed so that the JDBC driver returns the results to a Java thread. The Java thread calls the HTTP Event Collector (HEC) which then sends the results to `splunkd` and populates the indexers.

As a result of this change, you might find that the HEC is not working because of port issues. The following are examples of indicators that HEC is not working:

- **Normal Query** (in the **Data Lab > SQL Explorer** menu) returns results, but the DB Input does not populate the indexer.
- In the **Health > DB Input** menu, you see an error.

To validate and resolve the HEC port issues:

1. Navigate to your Splunk search bar and enter `index=_internal 8088`, replacing 8088 with whatever your HEC port is.
2. The search returns one of these two messages:

```
◇ FATAL HTTPServer - Could not bind to port 8088 This error comes from splunkd.log
◇ [QuartzScheduler_Worker-1] ERROR .. org.apache.http.conn.HttpHostConnectException: Connect to
  127.0.0.1:8088.. This errors comes from splunk_app_db_connect_server.log
```

3. To fix HEC port issues, go to **Settings > Data Input > HEC > Global Settings** and change the port.
4. Restart your instance of Splunk Enterprise.

## The performance is slow when output data events from DB Connect to MySQL database

If you encounter issues of poor performance when exporting data events to MySQL databases, you need to check your network status. If the issue is due to network latency, you can work around this by setting `rewriteBatchedStatements=true` when you [edit JDBC URL](#). See more details about this workaround on [JDBC batch insert performance](#). Using this workaround might have some potential issues, see more on [MySQL Configuration Properties for Connector](#)



## Oracle driver delaying database connection in Linux environments

When deployed in Linux environments, the Oracle driver sometimes causes connection delays. Program your Java Virtual Machine (JVM) to use `/dev/urandom` instead of `/dev/random` using one of the following options:

### Option 1

1. On your JVM, navigate to **Configuration > Settings**.
2. Edit the JVM options to override the `securerandom.source` setting in your Java environment with `/dev/urandom`:

```
-Djava.security.egd=file:/dev/./urandom
```

3. Navigate to `$SPLUNK_HOME/etc/apps/splunk_app_db_connect/local/`.
4. Open `commands.conf` and add an additional argument to `dbxquery`, `dbxlookup`, `dbxoutput` commands. For example:

```
[dbxquery]
command.arg.1 = -Dlogback.configurationFile=../config/command_logback.xml
command.arg.2 = -Djava.security.egd=file:/dev/./urandom
command.arg.3 = -DDBX_COMMAND_LOG_LEVEL=INFO
command.arg.4 = -cp
command.arg.5 = ../jars/command.jar
command.arg.6 = com.splunk.dbx.command.DbQueryCommand
```

```
[dbxoutput]
command.arg.1 = -Dlogback.configurationFile=../config/command_logback.xml
command.arg.2 = -Djava.security.egd=file:/dev/./urandom
command.arg.3 = -DDBX_COMMAND_LOG_LEVEL=INFO
command.arg.4 = -cp
command.arg.5 = ../jars/command.jar
command.arg.6 = com.splunk.dbx.command.DbOutputCommand
```

```
[dbxlookup]
command.arg.1 = -Dlogback.configurationFile=../config/command_logback.xml
command.arg.2 = -Djava.security.egd=file:/dev/./urandom
command.arg.3 = -DDBX_COMMAND_LOG_LEVEL=INFO
command.arg.4 = -cp
command.arg.5 = ../jars/command.jar
command.arg.6 = com.splunk.dbx.command.DbLookupCommand
```

5. Save your changes.

### Option 2

1. Navigate to `$JAVA_HOME/jre/lib/security/`, and open the `java.security` file in a text editor.
2. Add the following line: `securerandom.source=file:/dev/./urandom`
3. Save your changes.

## Connect Splunk DB Connect to Oracle Wallet environments using ojdbc8

1. Download the full ojdbc8 package.
  1. Unpack the package.
  2. In `$SPLUNK_HOME/etc/apps/splunk_app_db_connect/drivers`, create a directory named `ojdbc8-libs`.
  3. Copy over all other jars in the package to `$SPLUNK_HOME/etc/apps/splunk_app_db_connect/drivers/ojdbc8-libs/` except `ojdbc*.jar`
  4. Copy over `ojdbc*.jar` to `$SPLUNK_HOME/etc/apps/splunk_app_db_connect/drivers/`
2. In Splunk DB Connect, navigate to **Configuration > Settings > JVM Options**, and prepend the following settings to the JVM options. Replace `<wallet-directory>` according to your environment.

```
-Doracle.net.wallet_location=(SOURCE=(METHOD=FILE) (METHOD_DATA=(DIRECTORY=<wallet-directory>)))
```

```
-Doracle.net.ssl_server_dn_match=false
```

3. In Splunk DB Connect, navigate to **Configuration > Databases > Identities** and create your Oracle database identity, if you have not done so already.
4. In Splunk DB Connect, navigate to **Configuration > Databases > Connections** and create a connection to the Oracle database.
5. Select **Oracle** as the connection type
6. Select the **Edit JDBC URL** check box, and update the text box with the following URL.

```
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCPS) (HOST=<oracle-database-hostname>) (PORT=2494)) (CONNECT_DATA=(SERVER=DEDICATED) (SERVICE_NAME=<oracle-database-service-name>)))
```

Replace oracle-database-host and oracle-database-service-name according to your environment.

7. In Oracle Wallet, create a TLS connection.
8. Navigate to `$SPLUNK_HOME/etc/apps/splunk_app_db_connect/<os-arch>/bin/command.sh`, and make the following updates. For example, `<os-arch>` is `linux_x86_64`.

```
#!/usr/bin/env sh
```

```
SCRIPT=$(readlink -f "$0")
```

```
JAVA_PATH_FILE=$(dirname "$SCRIPT")/customized.java.path
```

```
JVMOPTS="-Doracle.net.wallet_location=(SOURCE=(METHOD=FILE) (METHOD_DATA=(DIRECTORY=<wallet-directory>)))  
-Doracle.net.ssl_server_dn_match=false"
```

```
if [ -f $JAVA_PATH_FILE ]; then  
    JAVA_CMD=`cat $JAVA_PATH_FILE`  
elif [ ! -z "$JAVA_HOME" ];then  
    JAVA_CMD="$JAVA_HOME/bin/java"  
else  
    JAVA_CMD="java"  
fi
```

```
exec $JAVA_CMD $JVMOPTS $@
```

9. Save your changes.
10. Create a data input, using the connection you created.

## DB Connect search stuck at 0% if user does not have Splunk Admin role

Verify that your Splunk license is valid.

## Missing events in Splunk even though there are no errors in logs

Verify your input count, limit congestion and try scaling horizontally on other instances.

## DB Connect can't communicate with task server

Verify your firewall, antivirus and proxy settings are correct and run these diagnostics:

```
curl -k "https://<ip_address>:<mgmt_port>/servicesNS/nobody/splunk_app_db_connect/db_connect/java" -v -u  
<Splunk user>  
ps -aux | grep java OR ps -ef | grep java  
java -jar $SPLUNK_HOME/etc/apps/splunk_app_db_connect/jars/server.jar --scheme
```

## Unable to process JSON from the Splunk user interface

Navigate to `$SPLUNK_HOME/etc/apps/splunk_app_db_connect/config/dbx_task_server.yaml` and update the yaml file by adding this line: `"org.glassfish.jersey": TRACE`

You can add it below this line: `"com.splunk.dbx.connector": ${CONNECTOR_LOG_LEVEL:-INFO}`

## Unable to find valid certification path to requested target

Verify you imported the public certificate of the target instance using the certificate input on the connection edit page. Verify that the certificate has not expired. From the Configuration > Settings > General page, add the `-Djavax.net.debug=all` option to Task Server JVM Options and Query Server JVM Options to enable more detailed logging of the validation process.

## Differences between Time and Event dates during the searches

If the Time and Event dates from the search are different than you expect, it might be because DB Connect is running on a server where daylight savings time is active.

**JVM extracts the time zone from the operating system based on the following:**

1. JVM uses the environment variable TZ if it is set.
2. If TZ is not set, then JVM looks for the file `/etc/sysconfig/clock` and finds the ZONE entry.
3. If neither TZ nor ZONE is set, JVM compares the contents of `/etc/localtime` to the files in `/usr/share/zoneinfo` looking for a match. The matching path and filename under `/usr/share/zoneinfo` provides the time zone.

### *How to solve it*

For Unix systems, to apply time zone rules correctly, such as daylight savings time for Java applications ( like DB Connect), make sure to apply one of the following options:

#### **Option 1**

Set the TZ environment variable to the proper time zone: `$ export TZ=<Timezone>`

**Note:** You need to restart Splunk or DB Connect after the timezone change.

#### **Option 2**

Specify the time zone for the Java application (in this case, DB Connect), by adding a system property:

`-Duser.timezone=<Timezone>` For DB Connect, access the menu from **Configuration > Settings > General**, and add `-Duser.timezone=<Timezone>` to **Task Server JVM Options**.

## SQL tips and tricks

If you are having trouble configuring Splunk DB Connect database inputs, or if you have written overly complex Splunk Processing Language (SPL) queries to use with your DB Connect inputs, consider instead refining your SQL queries.

- Use the `dbxquery` search command to write SQL queries directly in the Splunk Enterprise Search & Reporting app.
- When you set up a new database input, use Advanced Query Mode to enter your custom SQL query. In [step 2 of the input creation process](#), you specify a query to run to fetch data from your database. Instead of the default **Simple Query Mode**, which limits you to simply choosing catalog, schema, and table, choose **Advanced Query Mode**, then enter your SQL query into the field. You can preview your query results before saving the input.

## About NULL values

Splunk does not distinguish `NULL` and empty values. In other words, for Splunk a `NULL` value is equivalent to an empty string. If you want to replace `NULL` value by a well identified value you can use `fillnull` or `eval` commands. `NULL` values can also be replaced when writing your query by using `COALESCE` function. You can consult your database's documentation about this function for details.

It is also important to note that DB Connect cannot accept `NULL` values for a rising column or a column used as index time in the inputs definition. Such values will cause input jobs to fail.

## Use CAST or CONVERT to handle certain column types

Splunk Enterprise supports ASCII text, and cannot directly index binary fields like Binary Large Object (BLOB) and Character Large Object (CLOB). If Splunk Enterprise does not support a column type in your database, or you want to change a column type to a specific type before sending the data to Splunk Enterprise, use the `CAST` or `CONVERT` SQL functions to change the type of the column.

The `CAST` and `CONVERT` functions convert a value from one data type to another. `CONVERT` provides additional style functionality and more complex conversions, but Oracle discourages its use in current Oracle database releases.

## Use CAST or CONVERT to improve date and time handling

Splunk Enterprise assigns timestamps to indexed event data at index time. You can also specify a column for timestamp extraction when you configure an input. If the column is `DATETIME`, no additional configuration is required.

If they are not in `DATETIME` columns, you need to help Splunk recognize the timestamps in your database. If the column is a `VARCHAR` or string, then you can pre-process the data using Java `DateTimeFormatter`. Alternatively, you can convert the timestamp to the correct data type using a custom SQL statement with `CAST`, `CONVERT`, or `TO_TIMESTAMP` functions. Use SQL to change your string column type into a `DATETIME` column to set the index time value.

Many data sources may contain multiple date or time columns that you do not need to set the index time value for. However, you do need to present the data in readable, friendly formats. In this scenario, use a custom SQL statement with `CAST` or `CONVERT` functions to turn epoch values into a locale-oriented date and time, shift a local time to UTC, or trim a detailed timestamp to a broader date value.

## Use REPLACE or SUBSTRING to modify value quoting

If you have problems involving quotation mark processing during search-time field extraction or indexing (such as the ["Incomplete field values are extracted when the value contains double quotes"](#) troubleshooting issue), modify your quotation marks. Modify quotation marks, or any punctuation mark, in event data before Splunk Enterprise processes the events by using either the `REPLACE` or `SUBSTR` SQL functions.

Use the `REPLACE` function to replace every occurrence of a quotation mark (") or other character with another character or string. Use the `SUBSTR` function to return a portion of the string, beginning at a position in the string that you specify.

Be aware that these SQL functions are not equivalent to the `replace(X,Y,Z)` and `substr(X,Y,Z)` evaluation functions that you can use in Splunk Enterprise with the `eval`, `fieldformat`, and `where` search commands. The `REPLACE` or `SUBSTR` SQL functions execute before Splunk Enterprise receives data, while the `replace(X,Y,Z)` or `substr(X,Y,Z)` evaluation functions execute at search time in Splunk Enterprise. Use the former to ensure that Splunk Enterprise properly indexes values with spaces or other delimiter characters.

## Use AS to change column names

To change the names of columns before either indexing or running lookups, you can use the `AS` keyword to create an **alias**. You can use aliases to create a temporary name for columns, making the column headings easier to read when you are retrieving results.

The basic syntax to alias a column in SQL is as follows:

```
<column_name> AS <alias_name>
```

In this example, `<column_name>` is the name of the column that you want to alias, and `<alias_name>` is the alias you want to assign to the column.

In the following example, you rename the `MIN(price)` field as `cheapest`. Therefore, `cheapest` is the name of the second column that Splunk Enterprise returns to DB Connect in the result set.

```
SELECT category, MIN(price) AS cheapest
FROM items
GROUP BY category
```

## Use CASE, COALESCE, or CONCAT to compare and combine two fields

You have several options to compare and combine two fields in your SQL data. The following examples describe situations in which you can use `CASE`, `COALESCE()`, or `CONCAT()` to compare and combine two column values. Use either [query wrapping \(inline views\)](#) or [an advanced mode database input](#) to use the resulting columns as a rising column. Inline views are enabled by default. For more information, see [Use inline views \(query wrapping\)](#) and [Use an advanced mode database input](#), in this topic.

### CASE

Use the `CASE` expression to take actions based on the result of a conditional expression. For example, use a SQL statement like the following if you have two fields that you want to conditionally merge into one.

```
CASE WHEN first_name IS NOT NULL THEN last_name AS full_name
```

### COALESCE()

The `COALESCE()` function is shorthand for the `CASE` expression, and returns the first non-null expression that it finds within its arguments. For example, use a SQL statement like the following if you have several fields that you want to conditionally merge into one. This returns the first existing field.

```
COALESCE(first_name,middle_name,last_name) AS full_name
```

### CONCAT()

Use the `CONCAT()` function to concatenate, or combine, fields. It is equivalent to the `||` operator. For example, use a SQL statement like the following if you have two fields that have values that you want to merge into one:

```
CONCAT(first_name,last_name) AS full_name
```

The following example is only possible with an advanced mode database input, which replaces the question mark (?) character with a checkpoint value. For more information about advanced mode, see [Advanced](#) in the [Create and manage database inputs](#) topic.

```
SELECT CONCAT(last_name, first_name) as NAME, ACTOR_ID, FIRST_NAME, LAST_NAME FROM actor WHERE  
CONCAT(last_name, first_name) > ? ORDER BY CONCAT(last_name, first_name))
```

# Terminology

## Splunk DB Connect terminology

### access controls

Also known as **role-based access controls (RBAC)**, **access controls** comprise a system for defining access policy based on the system rather than the user. Splunk DB Connect uses the RBAC system in Splunk Enterprise, in which you define roles and assign users to them. Users can belong to multiple roles. A user's access to a given resource or action reflects what's specified in the permissive role to which that user belongs. For more information, see [Configure security and access controls](#).

### batch input

In **batch input** mode, the app invokes the same query each time and returns all results. Batch input mode is ideal for use with historical data that will not change and will be consumed and indexed once. For more information, see [Create and manage database inputs](#).

### connection

A database **connection** object contains the necessary information for connecting to a remote database. For more information about connection objects, see [Create and manage database connections](#).

### database input

A **database input** object lets you fetch and index data from a database. Database inputs are what enable Splunk Enterprise to query your database, identify the data to consume, and then tag and index the data. Once you've set up a database input, you can use that input just as you do any other data input you have defined in Splunk Enterprise. For more information, see [Create and manage database inputs](#).

### database lookup

A **database lookup** object enables you to enrich and extend the usefulness of your Splunk Enterprise data by fetching additional data from your external database. For more information, see [Create and manage database lookups](#).

### database output

A **database output** object lets you define how to send data from Splunk Enterprise to a database on a recurring basis. Define database outputs to store your indexed Splunk Enterprise data in a relational database. For more information, see [Create and manage database outputs](#).

### identity

An **identity** object contains database credentials. It comprises the username and obfuscated password that DB Connect uses to access your database. For more information about identity objects, see [Create and manage identities](#).

## permissions

**Permissions** indicate the level of access you assigned to a role. The access specifies how a user with that role can interact with knowledge objects such as identities and connections in Splunk Enterprise. **Read access** to an object means that Splunk Enterprise roles are able to use the object. **Read-write access** to an object means that Splunk Enterprise roles are able to use and modify the object. For more information, see [Configure security and access controls](#).

## rising column

A **rising column** input finds the new records you want and returns only those records with each query. When you create a rising column input type, you must specify the rising column. You can specify as a rising column any column whose value increases over time, such as a timestamp or sequential ID. The rising column is how DB Connect keeps track of which records are new. For more information, see [Create a database input](#).

## role

A **role** is a collection of permissions and capabilities. You assign roles to users in Splunk Enterprise. The roles a user belongs to determine which identity, connection, input, output, and lookup objects that user is able to see and interact with in Splunk DB Connect. For more information, see [Configure security and access controls](#).

## task server

A **task server** enables Splunk DB Connect to send and receive data via JDBC. The task server is an instance of a web server that listens on a dedicated port (9998 by default) for instructions to connect to a database.

## table

A database **table** is a collection of data organized by intersecting vertical columns and horizontal rows. Rows and columns intersect in cells.