ThoughtSpot

ThoughtSpot Data Integration Guide

Release 6.0 December 13, 2019

910 Hermosa Court, Sunnyvale, California 94085

Table of Contents

ThoughtSpot mobile overview

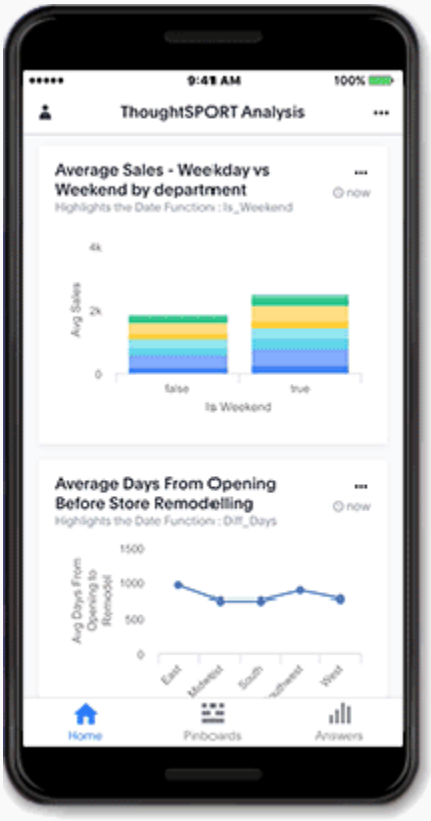
ThoughtSpot mobile app allows you to discover insights in seconds from billions of rows of data, right from the palm of your hand. You can access your ThoughtSpot cluster to search answers and pinboards. You can also create pinboards using existing answers and use pinboard filters.

Version 1.1.2 now supports auto-redirect Single Sign-On (SSO) for clusters with SSO enabled.

Requirements

- User account on a ThoughtSpot cluster running release 5.1 or later
- Apple iOS version 9.0 or later (iPad and iPhone only)

Features



Home Pinboard
Add charts and tables to a customizable Home Pinboard.

Quick Share
Share KPIs and charts using iMessage, email, and Slack.

Offline Exploration
Access your Home Pinboard even when you are offline.

Responsive & Interactive Experience
Tap and swipe to see chart details. Filter to pinpoint relevant insights.

Favorites
Tag your favorite Pinboards and Answers for quick access.

Getting Started

For administrators:

- To deploy the app to users in your company, see Deploy mobile app.
- To try the app before deploying it, see Try mobile app.

For users:

- To install and set up the app, see Install and set up mobile app.

Introduction to Embedding

ThoughtSpot Extended Enterprise Edition lets you give people outside of your company access to some ThoughtSpot capabilities within the context of an external application, website, or portal. We call these people “external users”. For example, you could give external users access to data, search, search results, data visualizations, and/or pinboards.

External users cannot be granted administrative privileges or receive technical support from ThoughtSpot. All first-line support must be handled through your company or organization.

ThoughtSpot Extended Enterprise Edition includes these capabilities:

- Full Application Embedding
- Embedded Charts and Pinboards
- Data REST API
- Runtime Filters
- Metadata API

Rights and Obligations

When you buy ThoughtSpot Extended Enterprise, the following rights and obligations apply:

1. External users may only access those elements that are exposed through ThoughtSpot public APIs. These include search, search results and data visualizations, saved pinboard and answers, SearchIQ (**Beta**), and SpotIQ.
2. External users may not be granted Administrator privileges such as the ability to create and modify users and groups.
3. External users are not permitted to copy or download the ThoughtSpot software.
4. You may not include external users in a group that has access to these privileges:
    - Can administer ThoughtSpot
    - Can administer and bypass RLS
5. The license for Extended Enterprise does not enable you to act as an MSP (Managed Service Provider). This means that you must not offer managed services to third parties that are based on the ThoughtSpot software.
6. As a company offering access to ThoughtSpot Extended Enterprise to external users, you are responsible for your own data policy and for complying with local laws and regulations

concerning data privacy, such as GDPR and HIPAA.

7. If you are using either ***Application Embedding*** or ***Embedded Search***, an NPS (Net Promoter Score) survey will be offered to your external users. You may not disable this survey.
   If you are using either the ***Visualization Embedding*** or ***Pinboard Embedding*** in your application, it is okay to disable the NPS survey.

8. Deployments of ThoughtSpot Extended Enterprise must conform to logo restrictions.

   These are the branding elements you can change in ThoughtSpot Extended Enterprise:

   - Logos & Favicon
   - Color Palettes
   - Background Color
   - Chart and Table Fonts
   - Footer Text

9. You may not remove the *Powered by ThoughtSpot* logo that appears at the bottom right of the embedded application when using ThoughtSpot full application embed, or when embedding a pinboard. It is okay to remove the logo if you are embedding a single visualization.

Section Contents

Here are the main topics for Embedding with Extended Enterprise:

- Log into the Linux shell using SSH
- Login credentials
- Using the JavaScript API
- SAML
- Data REST API
- Embed ThoughtSpot
- Runtime Filters
- Style Customization
- API Reference

Log in to the Linux shell using SSH

To perform basic administration such as checking network connectivity, starting and stopping services, and setting up email, log in remotely as the Linux administrator user “admin”. To log in with SSH from a client machine, you can use the command shell or a utility like Putty.

In the following procedure, replace `<hostname_or_IP>` with the hostname or IP address of a node in ThoughtSpot. The default SSH port (22) will be used.

1. Log in to a client machine and open a command prompt.
2. Issue the SSH command, specifying the IP address or hostname of the ThoughtSpot instance:

   ```
   ssh admin@<hostname_or_IP>
   ```

3. Enter the password for the admin user.

Log in credentials

You can access ThoughtSpot through SSH at the command prompt, and from a Web browser.

Administrative access

Each ThoughtSpot cluster has three default users. Contact your ThoughtSpot support team to get the passwords.

Type	Username	Description
Shell user	`admin`	For work that requires `sudo` or `root` privileges Not for application login Logs for this user are in `/usr/local/scaligent/logs` directory
Shell user	`thoughtspot`	For command-line work that does not `sudo` or `root` privileges Can use `tsload`, `tql`, and check the cluster status Not for application login Logs for this user are in `/tmp` directory
Application user	`tsadmin`	Access through a Web browser

Both the `admin` and `thoughtspot` user can SSH into the cluster. After authenticating, either user can use and and all of the following utilities:

- tscli; `thoughtspot` user cannot use commands that require `sudo` or `root` privileges
- tsload
- tql

SSH to the appliance

To perform basic administration such as checking network connectivity, starting and stopping services, and setting up email, log in remotely as the Linux administrator user “admin”. To log in with SSH from any machine, you can use the command shell or a utility like Putty.

In the following procedure, replace `<hostname_or_IP>` with the hostname or IP address of a node in ThoughtSpot. The default SSH port (22) will be used.

1. Log in to a client machine and open a command prompt.
2. Issue the SSH command, specifying the IP address or hostname of the ThoughtSpot instance:

   ```
   ssh admin@<hostname_or_IP>
   ```

3. Enter the password for the admin user.

Log in to the ThoughtSpot application

To set up and explore your data, access the ThoughtSpot application from a standard Web browser, using a username and password.

Before accessing ThoughtSpot, you need the following:

- ❏ The Web address (IP address or server name) for ThoughtSpot
- ❏ A network connection
- ❏ A Web browser
- ❏ A username and password for ThoughtSpot

ThoughtSpot supports the following Web browsers:

Firefox

68.x, 69.x, and later

Chrome

76.x, 77.x, and later

Internet Explorer

11.x, and later

Edge

44.x, and later

Safari

13.x, and later

☑ **Tip:** We support, but do not recommend, the use of the Internet Explorer.

Depending on your environment, you can experience performance or UI issues.

To sign in to ThoughtSpot from a browser, follow these steps:

1. Open the browser and type in the Web address for ThoughtSpot:

   `http://<hostname_or_IP>`
2. Enter your username and password, and click **Sign in**.

Using the JavaScript API

The ThoughtSpot JavaScript API (JS API) enables you to use ThoughtSpot within your own Web application and to perform the following tasks:

- Authenticate users to ThoughtSpot
- Embed ThoughtSpot visualizations in your Web page using the `<iframe>` HTML tag
- Supply ThoughtSpot data to your Web page through ThoughtSpot's REST APIs

You can download the ThoughtSpot JavaScript library from our secure storage server.

> **Note:** To use the JS API in your Web page, you must have the access and permissions to update the code of your Web page or application.

Browser Support

The JS API works in the following browsers:

Firefox

68.x, 69.x, and later

Chrome

76.x, 77.x, and later

Internet Explorer

11.x, and later

Edge

44.x, and later

Safari

13.x, and later

Internet Explorer 10

Microsoft introduced a compatibility mode in Internet Explorer 10, which displays the page using the version of Internet Explorer that is most compatible with that page. Because we do not support any version earlier than 11, this feature may break the code.

There are two approaches for forcing the Internet Explorer to emulate the most recent version:

- **Add a Custom Response Header** We recommend this approach because it is more robust, offers more control, and has a lower risk of introducing a bug to your code. in general, you must set the response header to match the server and the technology.
    - set the header name to “X-UA-Compatible”
    - set the value to “IE=Edge”

- **Add a Meta Tag** Add this meta tag as the *first* tag in the header section of the page:

```
<meta http-equiv="X-UA-Compatible" content="IE=Edge" \>
```

Cross-Origin HTTP Requests (CORS)

Collecting user credentials from one application (domain) and sending them to another (such as ThoughtSpot) can present security vulnerabilities such as a phishing attack. Cross-origin or cross-domain verification closes this vulnerability.

When you use the JavaScript API, your client calls ThoughtSpot from your Web page, portal, or application. Because your client and ThoughtSpot are on different domains, you must enable cross-origin HTTP requests from your client application to the ThoughtSpot application. This protects your data by preventing another actor from using the same URL to embed the visualization in its own Web pages.

Your cluster’s CORS configuration controls which domains can use your client code to authorize users. It also prevents code copying and deployment on unauthorized sites. For example, if your Web site is hosted on the domain `example.com`, you must enable CORS for that domain. Similarly, to test your code locally, you must also add the domain for your local server, such as `http://localhost:8080`. We recommend that you disable the `localhost` access after you finish testing.

To enable CORS between your client applications and your ThoughtSpot instance, you must work with ThoughtSpot Support.

About SAML

ThoughtSpot can be set up with Security Assertion Markup Language (SAML) to enable Single Sign On (SSO). SAML can be configured in several ways, including with CA SiteMinder.

For basic instructions on configuring SAML, use one of these procedures:

- Configure SAML, for instructions to configure SAML in ThoughtSpot.
- Configure SAML with CA SiteMinder, for configuring SAML specifically with CA SiteMinder.

Configure SAML

ThoughtSpot can use Security Assertion Markup Language (SAML) to authenticate users. You can set up SAML through the shell on the ThoughtSpot instance using a `tscli` based configurator.

Before configuring SAML, you need this information:

- Domain name for ThoughtSpot service (E.g. - `thoughtspot.ts-customer.com`).
- Port of the server where your ThoughtSpot instance is running (E.g. - `443`).
- Protocol, or the authentication mechanism for ThoughtSpot (E.g. - `http` or `https`)
- Unique service name that is used as the unique key by IDP to identify the client (E.g. - `urn:thoughtspot:callosum:saml`)
- Allowed skew time, which is the time after authentication response is rejected and sent back from the IDP. `86400` is a popular choice.
- The absolute path to identity provider's metadata file. Typically called `idp-meta.xml` or similar. This is needed so that the configuration persists over upgrades. Best to set it up on persistent/HA storage (NAS volumes) else in the same absolute path on all nodes in the cluster.
- This configurator also checks with the user if internal authentication needs to be set or not. This internal authentication mechanism is used to authenticate `tsadmin` and other ThoughtSpot local users. Set it to true by default to let local system/admin users in via the frontend.

Use this procedure to set up SAML on ThoughtSpot for user authentication. Note that this configuration persists across software updates, so you do not have to reapply it if you update to a newer release of ThoughtSpot.

1. Log in to the Linux shell using SSH.
2. Execute the command to launch the interactive SAML configuration:

```
tscli saml configure
```

3. Complete the configurator prompts with the information you gathered above.
4. When the configuration is complete, open a Web browser and go to the ThoughtSpot login page. It should now show the Single Sign On option.

Configure CA SiteMinder

Summary: CA SiteMinder can be used as an Identity Provider for single sign on to ThoughtSpot.

Before configuring CA SiteMinder, you must configure SAML in ThoughtSpot. Use this procedure to set up CA SiteMinder for use with ThoughtSpot:

1. Configure the Local Identity Provider Entity as follows:

Section	Entry
Entity Location	Local
Entity Type	SAML2 IDP
Entity ID	Any (Relevant ID)
Entity Name	Any (Relevant name)
Description	Any (Relevant description)
Base URL	https://<FWS_FQDN> where FWS_FQDN is the fully-qualified domain name for the host serving SiteMinder Federation Web Services
Signing Private Key Alias	Select the correct private key alias or import one if not done already
Signed Authentication Requests Required	No
Supported NameID format	Optional

2. Create the Remote SP Entity, either through a metadata import or manually. To configure the Remote SP entity manually, select **Create Entity**. Create ThoughtSpot as a Remote Entity with following details:

Section	Entry
Entity Location	Remote
New Entity Type	SAML2 SP
Entity ID	Your cluster
Entity Name	Any (relevant name)
Description	Any (relevant description)
Assertion Consumer Service URL	(Relevant URL)
Verification Certificate Alias	Select the correct certificate or import one if not done already. This is used to verify the signature in incoming requests
Supported NameID Format	Optional

3. You will now configure the Federation Partnership between CA SiteMinder (the IDP) and ThoughtSpot (the Remote SP) in CA SiteMinder. Log in to CA SiteMinder.
4. Navigate to **Federation** -> **Partnership Federation -> Create Partnership** (**SAML 2 IDP -> SP**).
5. Click **Configure Partnership** and fill in the following values:

Section	Entry
Add Partnership Name	Any (relevant name)
Description	Any (relevant description)
Local IDP ID	Select Local IDP ID
Remote SP ID	Select Remote SP ID
Base URL	Will be pre-populated
Skew Time	Any per environment requirement
User Directories and Search Order	Select required Directories in required search order

6. Click **Configure Assertion** and fill in the following values:

Section	Entry
Name ID Format	Optional
Name ID Type	User Attribute
Value	Should be the name of the user attribute containing the email address or user identifier. For example, ‘mail’

7. Click **Configure SSO and SLO** and fill in the following values:

Section	Entry
Add Authentication URL	This should be the URL that is protected by SiteMinder
SSO Binding	Select SSO Binding supported by the SP, typically the HTTP-Post
Audience	(Relevant audience)
Transaction Allowed	Optional
Assertion Consumer Service URL	This should be pre-populated using the information from the SP entity

8. Continue to **Partnership Activation** and select **Activate**.

Configure Active Directory Federated Services

You can configure Active Directory Federated Services (AD FS) to work with ThoughtSpot. This procedure outlines the basic prerequisites and steps to set up AD FS.

- Configure SAML in ThoughtSpot.
- Install AD FS 2.0.
- Make sure you can run AD FS 2.0 Federation Server Configuration Wizard from the AD FS 2.0 Management Console.
- Make sure that DNS name of your Windows Server is available at your service provider (SP) and vice versa. You can do this by running the command `nslookup` on both machines, supplying the DNS of the other server.

AD FS 2.0 supports SAML 2.0 in IdP (Identity Provider) mode and can be easily integrated with the SAML Extension for both SSO (Single Sign-On) and SLO (Single Log Out).

After completing the prerequisites, use these procedures to configure AD FS for use with ThoughtSpot.

1. Initialize IdP metadata.
2. Initialize the Service Provider metadata.
3. Test your ADFS integration.

Initialize the Identity Provider Metadata

Summary: This procedure shows how to initialize the Identity Provider (IdP) metadata for AD FS.

This is one part of the configuration procedure for setting up ThoughtSpot to work with AD FS for authentication. You should also refer to the overview of the entire process of integrating with AD FS.

To initialize the IdP metadata on AD FS:

1. Download the AD FS 2.0 IdP metadata from the AD FS server. You can reference this file by its URL, which looks like:

   ```
   https://<adfsserver>/FederationMetadata/2007-06/FederationMetadata.xml
   ```

2. Log in to the Linux shell using SSH.
3. Change directories to the SAML directory:

   ```
   $ cd /usr/local/scaligent/release/production/orion/tomcat/callosum/saml
   ```

4. Replace the contents of the file `idp-meta.xml` with the metadata of the IdP that you downloaded. Do not change the name of the file.
5. Contact ThoughtSpot support for help restarting ThoughtSpot's Tomcat instance.
6. Next, Initialize the Service Provider Metadata.

Initialize the Service Provider Metadata

Summary: This procedure shows how to initialize the Service Provider (SP) metadata for AD FS.

This is the second part of the configuration procedure for setting up ThoughtSpot to work with AD FS for authentication. You should also refer to the overview of the entire process of integrating with AD FS.

To initialize the Service Provider metadata on AD FS:

1. Open the AD FS 2.0 Management Console.
2. Select **Add Relying Party Trust**.
3. Select **Import data about the relying party from a file**.
4. Upload the metadata.xml file that you downloaded from ThoughtSpot earlier.
5. Select **Next.** The wizard may complain that some of the content of the metadata is not supported. You can safely ignore this warning.
6. In the **Ready to Add Trust** section, make sure that the tab endpoints contains multiple endpoint values. If not, verify that your metadata was generated with the HTTPS protocol URLs.
7. Leave the **Open the Edit Claim Rules dialog** checkbox checked. Click **Next**.
8. Select **Add Rule**.
9. Choose **Send LDAP Attributes as Claims** and click **Next.**
10. For **NameID** enter “Claim rule name”
11. For **Attribute store**, choose “Active Directory”.
12. For **LDAP Attribute** choose “SAM-Account-Name”.
13. For **Outgoing claim type**, choose “Name ID”.
    a. If you are using ADFS 3.0, you might need to configure the Name ID as a Pass Through claim.
14. Finish the wizard and confirm the claim rules window.
15. Open the provider by double-clicking it.
16. Select the **Advanced** tab and change **Secure hash algorithm** to “SHA-1”.
17. Your Service Provider is now registered.
18. Test the ADFS Integration.

Test the ADFS Integration

After setting up the AD FS integration, test to make sure it is working properly. To test your AD FS integration, go to ThoughtSpot login page using a Web browser and try to login with SAML.

About the Data REST API

> **Summary:** The purpose of the REST API is to get data out of ThoughtSpot so you can use it in a Web page, portal, or application.

When using the Data REST API, authentication is achieved through SAML. After authentication, use the POST method to call a URL for the desired visualization or pinboard. A JSON (JavaScript Object Notation) representation of the data will be returned.

Authentication

Before you can use the Data REST API, you must authenticate to ThoughtSpot using SAML with the JavaScript API.

Cross Domain Verification

You can enable cross-domain verification when using the Data REST API. This protects your data, so that another website cannot use a URL to get data from ThoughtSpot. The procedure for enabling the JavaScript API includes information on how to enable this.

Data REST API capabilities

Use a POST method to access the URL, which calls the REST API. The data is returned as a JSON string. When using this method, you must extract the data from the JSON file and render it on your Web page, portal, or application.

You can use the REST API to do things like:

- Generate dynamic picklists on your Web page.
- Display a single value.
- Retrieve the data to populate a visualization drawn by your own renderer.
- Pull data directly from ThoughtSpot

Remember that the data you retrieve from ThoughtSpot is live data, so whenever the Web page is rendered, the current value(s) will be shown.

Direct Search-to-Embed API

The Direct Search-to-Embed API enables searching directly from an external application or web page to pull data from ThoughtSpot. This feature was introduced in ThoughtSpot 5.0. When using it, you can access data stored in ThoughtSpot directly. You do not have to save a search result to a pinboard and then reference it using the visualization's URL.

Public API reference

You can find more information on our public APIs in the Reference guide.

Related information

- API Reference guide.
- Direct Search-to-Embed API.

Calling the Data REST API

To call the Data REST API, you can specify a URL using the POST method, passing the ID numbers of the objects from which you want to obtain data.

Specify the pinboard or visualization example

For a pinboard, you can append the ID of your pinboard as a parameter, like this example:

```
https://<thoughtspot_server>/callosum/v1/tspublic/v1/pinboarddata?id=7752fa9e-db22-415e-bf34-e082c4bc41c3
```

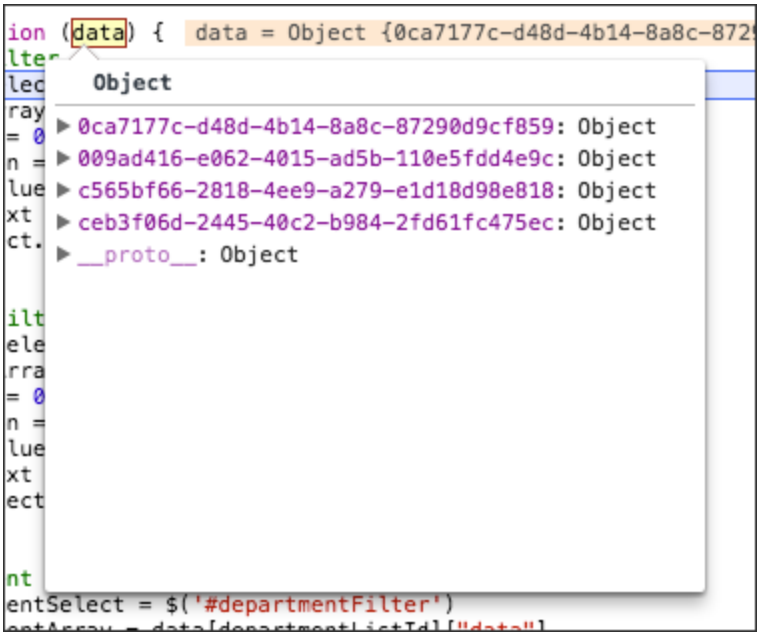
To retrieve data from a specific visualization within a pinboard, you would append the ID number of the visualization using the vizid parameter:

```
https://<thoughtspot_server>/callosum/v1/tspublic/v1/pinboarddata?id=7752fa9e-db22-415e-bf34-e082c4bc41c3&vizid=%5B1e99d70f-c1dc-4a52-9980-cfd4d14ba6d6%5D
```

Remember: You must add brackets around the vizid parameter. The URL encoding for open bracket is `%5B`, and the URL encoding for close bracket is `%5D`.

Object Format for Returned Data

When you parse the returned JSON data you can see that there is one object for every viz on the pinboard. The objects are named according to the corresponding vizid.



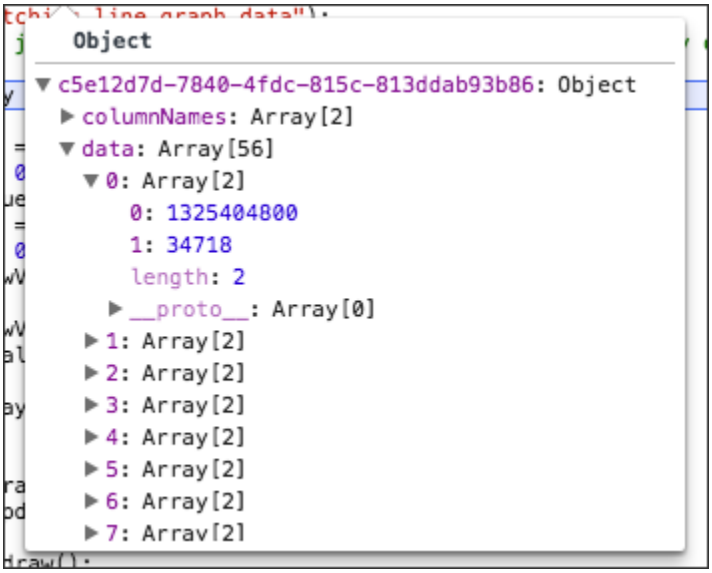
If you make a call to a specific viz on a pinboard, it will return just one object. The JSON object format for the data that is returned from ThoughtSpot is:

```
{
vizId1 : {
         name: “Viz name”,
         :[[2-d array of data values], [], [] …..[]],
         columnNames:[col1, col2, …. ],
         samplingRatio: n
         },
vizId2 : {
         .
         }
}
```

Each object contains four components:

1. An array of column headers.
2. An array of data.
3. The name given to the specific viz.
4. And a sampling ratio. The sampling ratio tells you the percentage of total data returned. `1` would mean all data in the viz was returned in the API call.

The `columnNames` array contains a list of all column headers. And the `data` array contains a list of other arrays. Each sub array represents a new row of data.



The REST API supports filtering the data returned through parameters that you pass within the URL. These are called Runtime Filters.

Example

The following example shows a JavaScript function that calls the REST API, gets the results back, and retrieves a single value from the JSON results:

```
/**
 * Generates headline by making a data API call.
 *
 * @param void
 * @return void
 */
function generateHeadline(filters) {
    var pinboardId = "0aa0839f-5d36-419d-b0db-10102131dc37";
    var vizId = "67db30e8-06b0-4159-a748-680811d77ceb";
    var myURL = "";

    if (filters === void 0) {
        myURL = "http://192.168.2.55:443/callosum/v1/tspublic/v
1/" +
                "pinboarddata?id=" + pinboardId + "&" +
                "vizid=%5B" + vizId + "%5D";
    } else {
        var query = getQueryString(filters);
        myURL = "http://192.168.2.55:443/callosum/v1/tspublic/v
1/" +
                "pinboarddata?id=" + pinboardId + "&" + +
                "vizid=%5B" + vizId + "%5D&" + query;
    }

    var jsonData = null;
    var xhr = new XMLHttpRequest();
    xhr.open("POST", myURL, true);
    xhr.withCredentials = true;
    xhr.onreadystatechange = function() {
        var headline = document.getElementById("embeded-headlin
e");
        if (xhr.readyState == 4 && xhr.status == 200) {
            jsonData = JSON.parse(xhr.responseText);
            headline.innerHTML = jsonData[vizId].data[0][0];
        } else {
            headline.innerHTML = "Error in getting data !!!";
        }
    };
    xhr.send();
}
```

Data REST API pagination

> **Summary:** You can paginate the JSON response that is called from the REST API. The order of the data is retained from page to page.

Given the ability to paginate, you can quickly populate tables and make new REST calls every time you go to the next page of the data on the table. There is significant load time if you want to populate the data table with many rows (greater than 1000) from the Data REST API.

To paginate results in your API response, you must add new parameters to the query:

`PageSize` determines the number of rows to be included.

```
{
    "name": "pagesize",
    "description": PageSize: The number of rows.",
    "defaultValue": "-1",
    "type": "integer"
}
```

`Offset` determines the starting point.

```
{
    "name": "offset",
    "description": Offset: The starting point",
    "defaultValue": "-1",
    "type": "integer"
}
```

`PageNumber` is an alternate way to determine the offset. You must make a call with `pageNumber = 1` first. Then you can access any page. Calling with `pageNumber != 1` as the initial call will fail. `pageNumber = 0` is not a valid value.

```
{
    "name": "pagenumber",
    "description": PageNumber: This is an alternate way to set
offset. This is 1-based
                   indexing. Offset = (pageNumber - 1) * pageSi
ze.
    "defaultValue": "-1",
    "type": "integer"
}
```

`FormatType` is the JSON format type.

```
{
    "name": "formattype",
    "description": FormatType: This sets the JSON format type.
Values that are allowed are
                   FULL and COMPACT.
    "defaultValue": "COMPACT",
    "type": "string"
}
```

`COMPACT` is the default type, and is formatted as follows: `['col1', 'col2'] [1, 'a']`. While `FULL` is formatted like this: `{'col1': 1 'col2': 'a'}`

Example

The following example shows ThoughtSpot data that is being populated in a table:

```
 /**
 * Sample response for Page-1.
 */
{
    "totalRowCount": 1500,
    "pageSize": 100,
    "pageNumber": 1
        "data":
    [
       {
            "key1": "value1",
            "key2": "value2",
        },
        {
            "key1": "value1",
            "key2": "value2",
        },
    ]
}
```

Use the Data REST API to get data

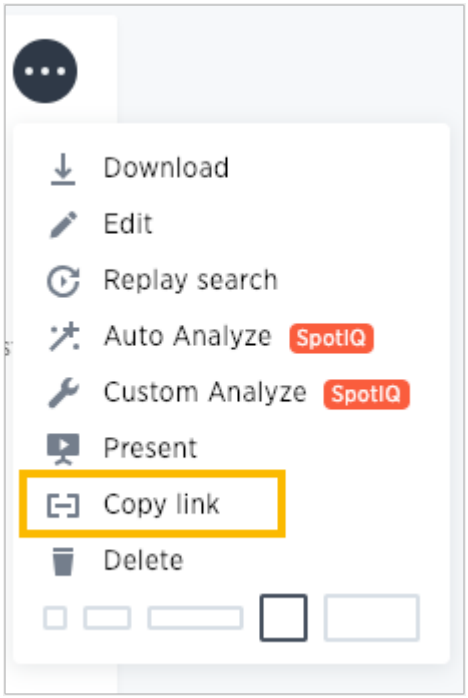
Summary: This procedure shows how to use the REST API to get data out of ThoughtSpot, so you can use it in a Web page, portal, or application.

Data retrieved using the Data REST API is returned as JSON (JavaScript Object Notation).

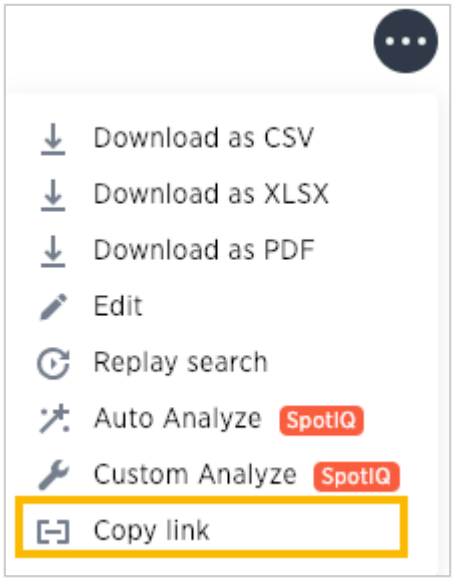
Before you can use the Data REST API, you need to enable the JavaScript API (JS API) and authenticate to ThoughtSpot.

Use this procedure to construct the URL you will use to call the Data REST API:

1. Log in to ThoughtSpot from a browser.
2. Navigate to the pinboard from which you want to get data. If it doesn't exist yet, create it now.
3. Find the ID number of the object you want to get the data from. If the object is:
   - A pinboard, click ellipses icon ••• the and select **Copy Link**.



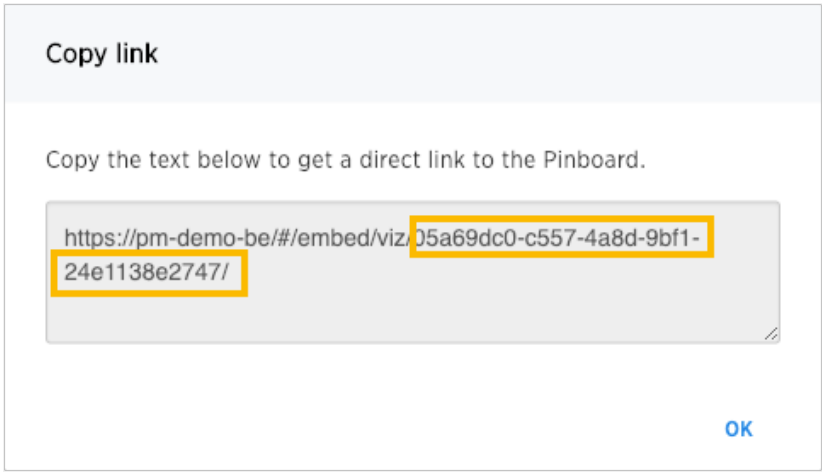
- A visualization, click the **Copy Link** icon in the upper right corner of the table or chart.



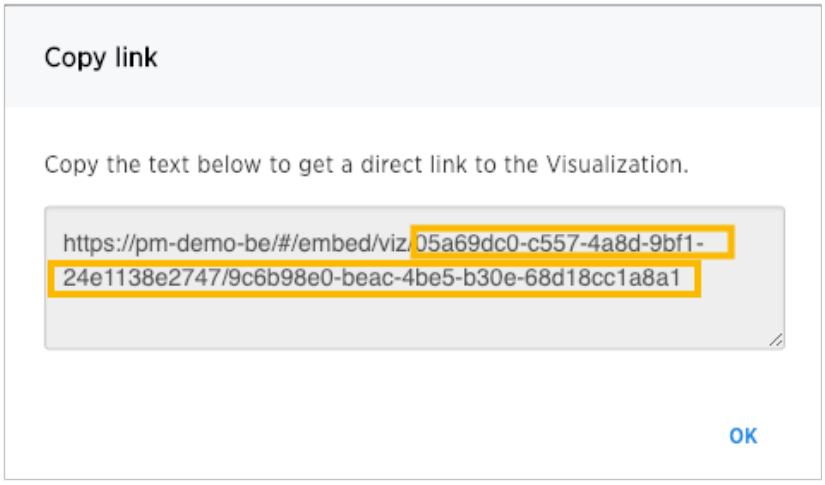
4. Copy the ID number from the link shown. Paste it somewhere so that you can use it later to construct the URL to use when calling the REST API.

   If the object is:

   - A pinboard, copy the identifier that appears after “viz/”. Omit the trailing “/”.



- A visualization (table or chart), copy the identifier that appears after “viz/”. This is the visualization ID.



5. Construct the URL as follows: For a pinboard, the URL takes the form:

```
 https://<thoughtspot_server>/callosum/v1/tspublic/v1/pinboarddata?id=<pinboard_id>
```

For a visualization, the URL takes the form:

```
 https://<thoughtspot_server>/callosum/v1/tspublic/v1/pinboarddata?id=<pinboard_id>&vizid=%5B<visualization_id>%5D
```

6. If you want to apply any filters to the data that will be returned, apply Runtime Filters.

7. Now your URL is complete, and you can use it to access the data directly through the HTTP POST method. The Data REST API returns the data formatted as JSON.

8. Retrieve the data from the JSON and display it in your Web page, Web portal, or application.

Use the Embedded Search API to pull data from ThoughtSpot

Summary: This procedure shows how to use the Embedded Search API to get data from ThoughtSpot

The Embedded Search API enables searching directly from an external application or web page to pull data from ThoughtSpot. This feature was introduced in ThoughtSpot 5.0. When using it, you can access data stored in ThoughtSpot directly. You do not have to save a search result to a pinboard and then reference it using the visualization's URL.

This embedded search is useful when you want to allow an application to pull data directly from ThoughtSpot in an ad hoc fashion.

To have the Embedded Search API functionality turned on, contact ThoughtSpot Support.

Data retrieved using the Embedded Search API is returned as JSON (JavaScript Object Notation). You must parse the JSON to get the data values you need, generally using JavaScript in the receiving application.

Use this procedure to construct the call to the Embedded Search API:

1. Enable the JavaScript API (JS API) on the receiving page of the target application.
2. Authenticate to ThoughtSpot on the receiving page of the target application.
3. Embed the ThoughtSpot application in your own web page or application.
4. To subscribe to results for all the searches the user does in the embedded ThoughtSpot application, use the API JavaScript function subscribeToData( ). This will allow your page to listen for data coming from ThoughtSpot.

   Now when a user searches, the iFrame will send data to the subscription. The parent web page or application receives the data as JSON, and can do whatever you want with it.
5. You can set up your web page or application to display or otherwise act on the data it receives from the subscription.

6. To test it out, do a search in the embedded ThoughtSpot application to retrieve the data. Your application should act on the data in the way you set it up to do so.

Use the Data Push API

Summary: This procedure shows how to use the Data Push API to send data from ThoughtSpot to another application

The Data Push API allows you to open a web page in the context of the ThoughtSpot application. This third party web page will then have access to the results of the ThoughtSpot search from which it was invoked. This is useful when you want to initiate an action in another application based on the result of a search in ThoughtSpot. The Data Push API was introduced in ThoughtSpot 5.0.

An example of pushing data to another system to trigger an action would be where you do a search to find customers who are coming due for renewal of their contract in the next month. You could then trigger an action that brings up a web page from an external billing system. The billing system could be set up to read the data (list of names, emails, products, and renewal dates) from ThoughtSpot. The billing system might then add the price, generate an invoice for each customer, and send it by email.

To have the Data Push API functionality turned on, contact ThoughtSpot Support.

The DataPush API makes the data available to the external application formatted as JSON (JavaScript Object Notation). You must parse the JSON to get the data values you need using JavaScript in the receiving application.

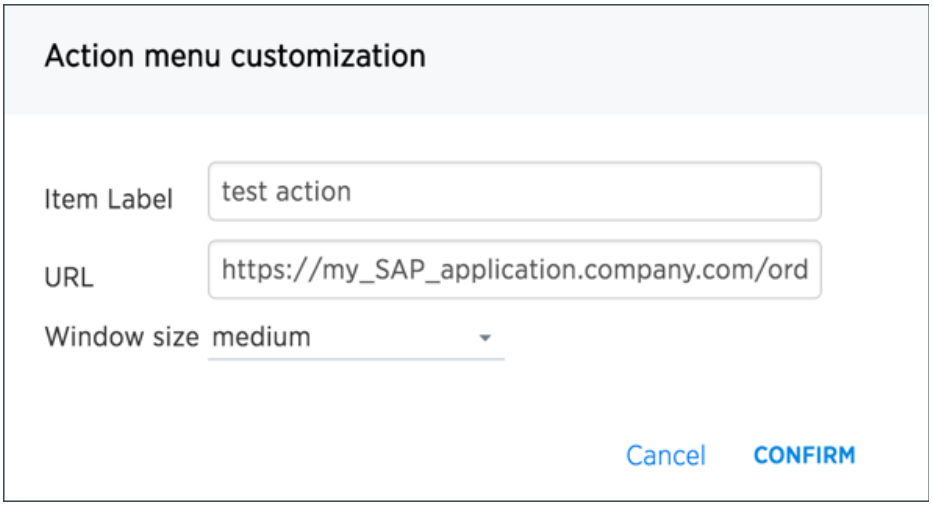
Create an Custom Action

To create a Custom Action, you must have the **Can administrator ThoughtSpot** privilege.

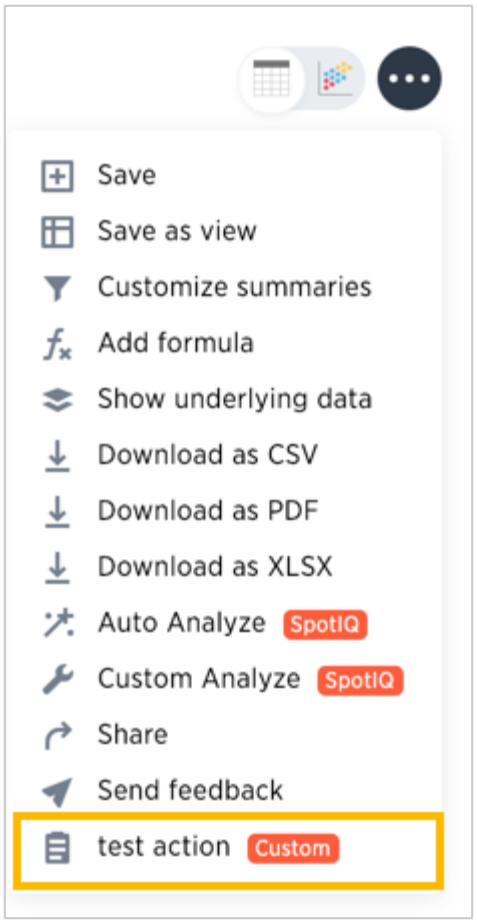
Use this procedure to create an Custom Action in ThoughtSpot:

1. Log in to ThoughtSpot from a browser.
2. Choose **Admin** and then **Action Customization**.
3. Click the **Add custom action item** button.
4. Fill in the details for your custom action:

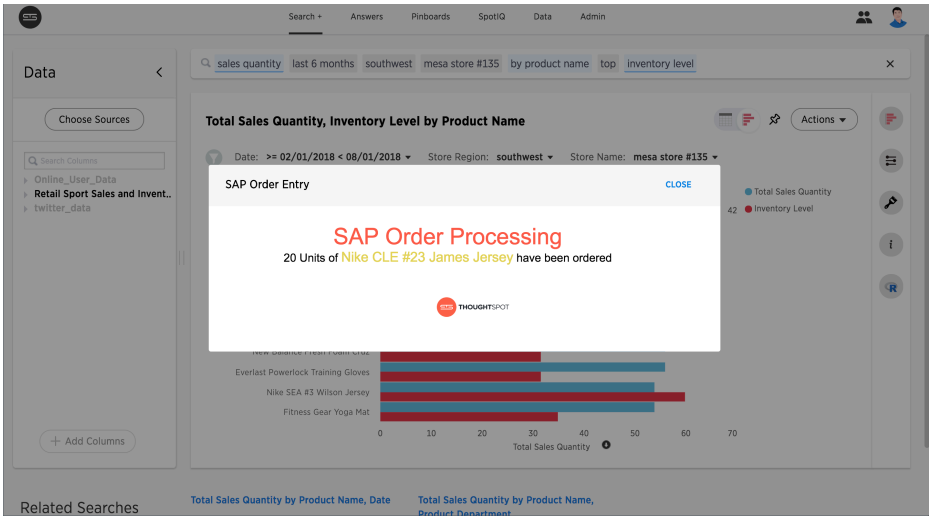
- Item Label: Clicking the menu item with the name you provide here will initiate the data push to the other system. This menu item will appear under the three dot menu of a search result.
- URL: The URL of the target page in the external web page or application.
- Window size: The size of the window that will display the external web page or application in ThoughtSpot.



5. Now when a user is viewing a search result, they'll have the option to use the Custom Action you created. To initiate the action, they'll click the ellipses icon , and select **Your Action Name**. Notice a **Custom** tag next to your action name; it indicates that this is something custom built, and not a ThoughtSpot action.



6. When a user clicks your action, they'll see the web page you entered as the URL for your custom action.



Note: In order for your action to work correctly, the answer from which the user selected the action needs to have the correct search terms which your application or web page is expecting to receive. There is no way to guarantee this, other to train your users on the purpose of your action, and what's required for it to run.

Sample application

Here is a sample application you can use to try out the Data Push API:

```
<!doctype html>
<html lang="en">
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/
1.6.9/angular.min.js"></script>
<script type="text/javascript" src="api/api.js"></script>
<body>
    <script>

        var app = angular.module("latestData", []);
        app.controller("dataCtrl", ['$scope', '$window', functi
on($scope, $window) {

            $scope.currentData=undefined;
            $scope.showData=false;
            $scope.displayData = function() {
                $scope.showData = true;
            };
            function currentDataCallback(event) {
                $scope.currentData = event;
            }
            $window.onload = function(){
                $window.thoughtspot.getCurrentData(currentDataC
allback);
            };

        }]);
    </script>
    <div ng-app ="latestData" ng-controller="dataCtrl">
        <button class="get-data" ng-click="displayData()">Clic
k here for latest exported data</button>
        <div class="display-data" ng-if="showData"> </div>
    </div>
</body>
</html>
```

Understand embedding

Embedding allows you to embed all or part of ThoughtSpot in another client application. This page provides an explanation of what you must consider when embedding ThoughtSpot

Decide what to embed and where

The type of embedding your company requires can help you determine what type of embedding to use. For example, you may simply need a single chart displayed as a wallboard or you may want your customers to access reports on their own data. The first example could require modifying a single HTML page while the later example may require working with a development team and several different workflows in a browser application.

Regardless of the simplicity or complexity of your client application, its infrastructure must allow for loading and calling the ThoughtSpot JS library. This library allows you to authenticate to ThoughtSpot and load specific objects.

There are different methods for embedding ThoughtSpot into a client application:

Type	Description
Full	Embeds the entire ThoughtSpot application including menu bars. Full navigation is supported.
Page-level	Embeds pages without the menus bars or page-level navigation. This is useful where you want to limit the inclusion to a portion of ThoughtSpot. For example, you may only embed the **Search** or the **Answers** page.
Object-level	Embed a single visualization in your application. Content is created in ThoughtSpot and then that content is embedded. The content is rendered within an `iframe` . This returns a JSON object that includes the underlying data.

You can also use the ThoughtSpot data APIs to request data from ThoughtSpot.

Configuration requirements for embedding

Only Extended Enterprise installation can use ThoughtSpot's embed functionality. ThoughtSpot Enterprise installations must also work with ThoughtSpot Support to enable embed before using this functionality.

Optional settings for embedding

There are some settings that apply to embedding which ThoughtSpot Support or your other ThoughtSpot technical contact can make for you.

One of these involves what happens when a user clicks on a link within the data. When your data includes URLs, they display as clickable links in ThoughtSpot tables. By default, clicking on a link opens the URL in a separate tab. But there is a system-wide setting that can be changed to open the links within the context in which they appear.

Changing this setting opens the links:

Link type	Opens in
Link in search result table in ThoughtSpot	Same browser tab as ThoughtSpot application
Link in table embedded in an iFrame	Same iFrame that contains the table
Link in full ThoughtSpot application embedded in an iFrame	Same iFrame that contains the application

Choose an authentication methodology

You can control which type of authentication you use between your client application and ThoughtSpot.

No Authentication

You can simply not set up authentication. This would require the user to *already be logged into ThoughtSpot* before interacting with your client application. This is typically only useful when testing your client. You would not use this in your production environment.

SAML

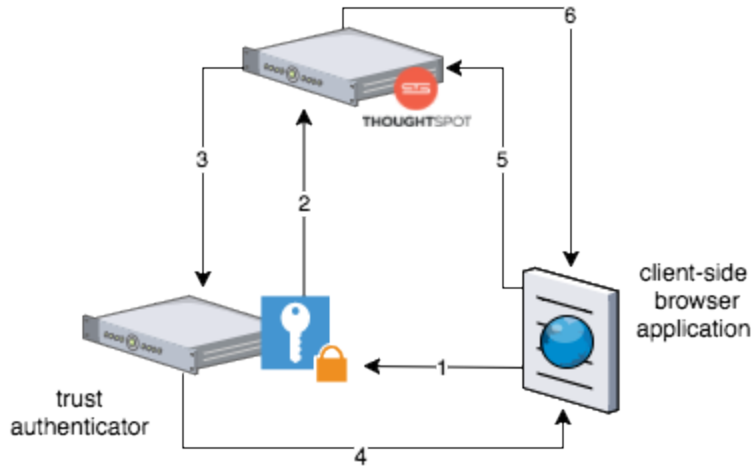
Before you can embed all or part of ThoughtSpot, you must authenticate to ThoughtSpot using SAML with the the public REST API call. After authentication, a URL is provided to call the desired visualization and populate it into an `iframe`.

You must configure SAML on your ThoughtSpot instance before using this method.

Trusted authentication service

A ThoughtSpot installation can enable support for token-based authentication service. This allows an installation to use a central authentication service rather than using ThoughtSpot to authenticate. In this architecture, ThoughtSpot provides the service with a token that allows it to authenticate on behalf of users.

A trusted authenticator application or service obtains a token from ThoughtSpot. This token is used to obtain trust from other, third-party client applications that need access to ThoughtSpot. In the following scenario, the trust authenticator forwards requests for ThoughtSpot data from client applications to ThoughtSpot.



A user already logged into client-application interacts with a ThoughtSpot embed point which causes the following processes:

1. The client-side application requests a user token from the trusted authenticator.
2. The trusted authenticator requests user token from ThoughtSpot.
3. ThoughtSpot verifies the authenticator and returns a user token.
4. The authenticator returns the user token to the client.
5. The client forwards the user token to ThoughtSpot.
6. ThoughtSpot validates the token and returns information commensurate with that authenticated user's authorization.

Plan for Cross-Origin HTTP Requests (CORS)

Collecting user credentials from one application (domain) and sending them to another (such as ThoughtSpot) can present security vulnerabilities such as a phishing attack. Cross-origin or cross-domain verification closes this vulnerability.

When embedding, you must enable CORS between your client application domain and the ThoughtSpot domain. This protects your data, so that another actor cannot use the same URL to embed the visualization in its own Web pages.

Decide if you need to change the feedback email

ThoughtSpot has an automated feature that collects feedback from users and sends it to `support@thoughtspot.com`. Depending on what and how you embed, user actions with your embedded application can trigger feedback. You can continue to forward feedback in this manner or direct the feedback to another email. To learn how to change the feedback email, see Manage the feedback contact.

Remove the ThoughtSpot branded footer

The ThoughtSpot footer appears by default in the ThoughtSpot application. It also appears with an embed application that encompasses an individual pinboard or a full application. In embed applications that are have a single visualization, you can ask your ThoughtSpot support engineer to disable the footer.

Embed pinboard or visualization

This page explains, through an example, how to embed a visualization (table or chart) or pinboard from ThoughtSpot in your own static Web page, portal, or application.

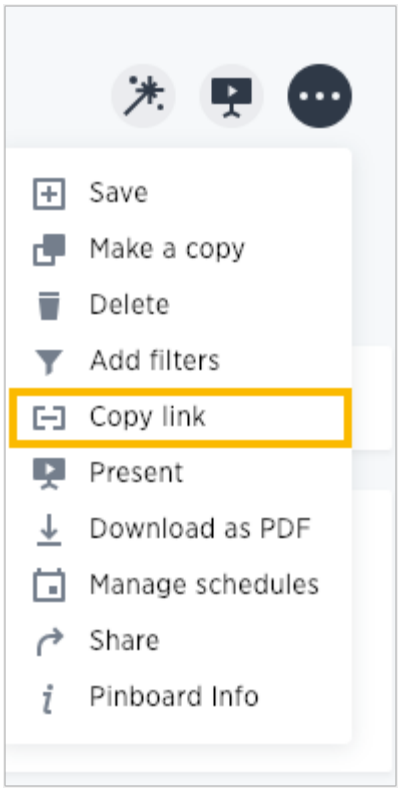
To build this sample, you need to have access to a text editor and a ThoughtSpot instance with a visualization. You should also have some experience working with Javascript.

Get the link for an entire pinboard or single visualization

This procedure assumes the pinboard with the visualization you want to embed already exists. If the pinboard or visualization doesn't exist, create it now before continuing.

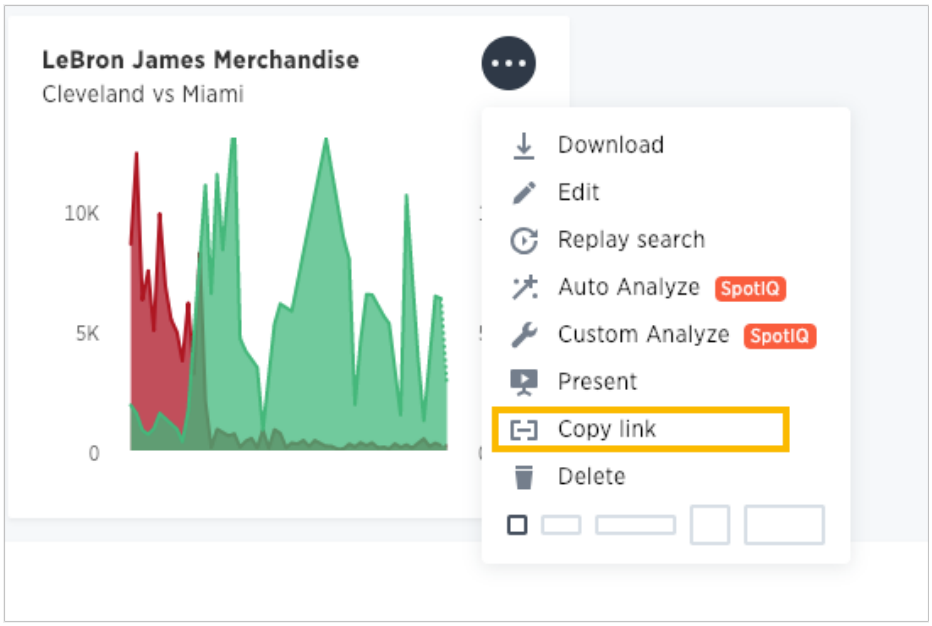
1. Log in to ThoughtSpot from a browser.
2. Navigate to a visualization on the **Pinboard** tab.
3. Open a pinboard.
4. Copy the URL for the entire pinboard and for a single visualization.

   If the object is a pinboard, click the ellipses icon > **Copy Link**.



The format for the link is: `<protocol>:<host>:<port>/#/embed/viz/<pinboardID>`

For a vizualization in a pinboard, click the ellipses icon > **Copy Link**.



The format for the link is: `<protocol>:<host>:<port>/#/embed/viz/<pinboardID>/<vizualizationId>`

Edit the test.html

You must edit the page in your application or web page where you want to embed a ThoughtSpot pinboard or visualization. For this example, you can get a copy of the `test.html` file.

1. Create an empty directory called `test`.
2. Save the `test.html` file to the `test` directory.
3. Download the ThoughtSpot JavaScript library.
4. Place the Javascript library in an `api` directory co-located with the `test.html` file.
5. Edit the `test.html` file in your favorite editor.
6. Scroll down to the `Variables` section (about line 37).

   Here are the fields in the `test.html` file you need to edit.

   ```
   var protocol = "THOUGHTSPOT_PROTOCOL";
   var hostPort = "HOST_PORT";

   var pinboardId = "PINBOARD_ID";
   var vizualizationId = "VIZUALIZATON_ID";
   ```

7. Edit each variable in the section and replace it with the IDs you copied from the pinboard.

   For example, your URL may look similar to the following:

   `http://172.18.202.35:8088/#/embed/viz/061457a2-27bc-43a9-9754-0cd873691bf0/9985fccf-b28d-4262-b54b-29619a38348e`

   This is a link copied from an individual visualization, the result in the file is:

```
 var protocol = "http";
 var hostPort = "172.18.202.35:8088";

 var pinboardId = "061457a2-27bc-43a9-9754-0cd873691bf
0";
 var vizualizationId = "9985fccf-b28d-4262-b54b-29619a38
348e";
```

The protocol (`http` or `https`) of your client and your ThoughtSpot instance should match. You can use this identifier in the next part.

8. Save your changes and close the `test.html` file.

Enable CORS for your client domain

You must work with ThoughtSpot support to enable CORS between your client application domain and the ThoughtSpot domain. If you don't do this, you will receive an error message when `test.html` attempts to load the embedded objects.

The test infrastructure uses Python's `simplehttpserver` , which runs by default as `localhost:8000` . ThoughtSpot support must have this information. You can also copy the `test` directory to an existing web server. If you do this, you must DNS for the server when you contact Support.

Test the example page

You are almost ready to view your embedded pinboard and visualization. The fastest way to run a webserver and test the code is using Python's `simplehttpserver` . If you have Python on your system you already have the `simplehttpserver` .

1. Log into ThoughtSpot.

   In production, you would have added authentication code to your client. You haven't done that with this system. So, before you test, you must login to the ThoughtSpot. Successfully logging in causes the system to create a session and an authentication key. Your browser has this information and so when you load the `test.html` page in another tab, you won't need to authenticate again.

2. Change to your `test` directory.

3. Start the `simplehttpserver` web server.

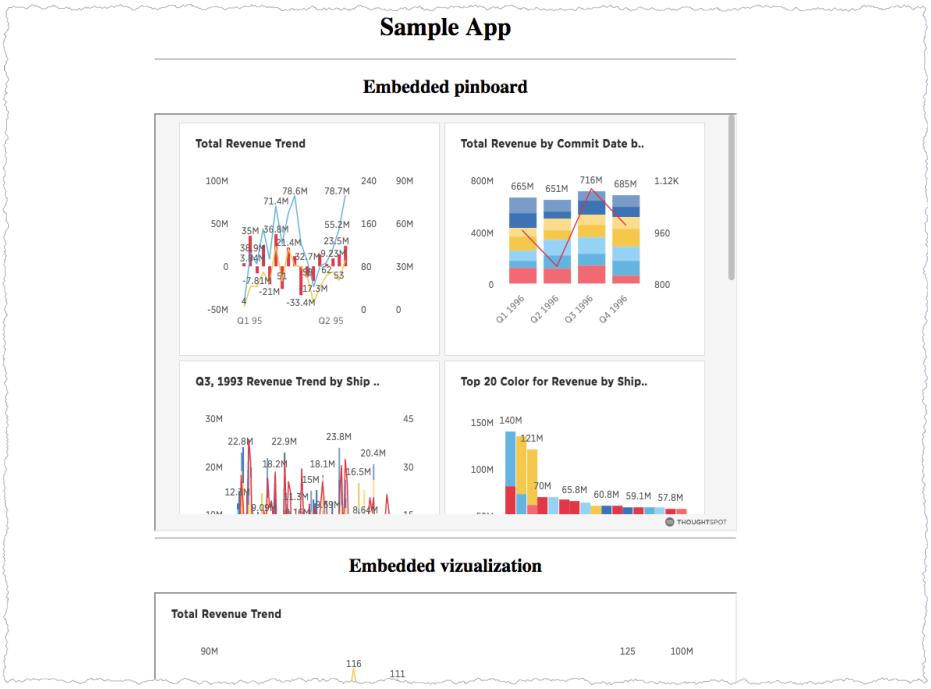
```
python -m SimpleHTTPServer 8000
```

4. Open your browser's **Developer** tools.

5. Navigate to the test page in your browser.

   `http://localhost:8000/test.html`

   You should see something similar to the following:



6. Check the browser console.

   Success is appears in the console with a message similar to this:

```
 test.html:60 Initialization successful.
 test.html:113 http://172.18.202.35:8088/#/embed/viz/061
457a2-27bc-43a9-9754-0cd873691bf0
 test.html:129 http://172.18.202.35:8088/#/embed/viz/061
457a2-27bc-43a9-9754-0cd873691bf0/9985fccf-b28d-4262-b54
b-29619a38348e
```

Troubleshooting embeds

If your embeds don't load, open the developer tools on your browser. Look for errors in the page loading, usually on the **Console** tab. If you see an error similar to:

```
No 'Access-Control-Allow-Origin' header is present on the reque
sted resource.
```

Typically you see this if the cross domain (CORS) setting was not completed correctly on your ThoughtSpot cluster. Contact support@thoughtspot.com for more help.

Authentication flow with embed

If your ThoughtSpot system is configured for Security Assertion Markup Language (SAML) you can enable Single Sign On (SSO) for your embed application.

Place the JS API library in the `<head>` section of the HTML on your Web page. Ensure that the JS API script tag is the first script loaded in the page. You can see examples of this

Authenticate when the window is initialized

Your web page needs to authenticate by calling `window.thoughtspot.initialize` and waiting for the `onInitializationCallback` to be called before embedding any ThoughtSpot visualizations or making any ThoughtSpot REST API calls.

The JS API call `window.thoughtspot.initialize` can cause the entire Web page to be re-directed to your Identity Provider (IDP). This order implies that you may not execute any of your application logic before `window.thoughtspot.initialize` has called your callback.

Any redirection could interfere with your application logic. So, don't embed any static ThoughtSpot visualizations in your HTML. In other words, you should generate the ThoughtSpot visualizations dynamically after `window.thoughtspot.initialize` has called your callback.

The `onAuthExpiration` is only available if you have at least one ThoughtSpot visualization iframe in your web page.

Example of code flow

To authenticate with SSO.

1. Download the ThoughtSpot JavaScript library.
2. Include the library file into your web page's `<head>` section:

```
 <head>
 <script type="text/javascript" src="<protocol><your.tho
ughtspot.domain>/js/api/api.min.js">
 ...
 </head>
```

3. From your application code, authenticate to ThoughtSpot by calling to the `window.thoughtspot.initialize` method.

   For example:

```
 <script type="text/javascript">
     thoughtspotHost = <hostname_or_ip_w/o_http>
     function setUpThoughtSpotAPI() {
         window.thoughtspot.initialize(
             function(isUserAuthenticatedToThoughtSpot)
{
                     if (isUserAuthenticatedToThoughtSpo
t) {
                         // load an embedded ThoughtSpot
visualization or
                         // make a ThoughtSpot data API c
all
                     } else {
                         // the current user into your sy
stem is not authenticated
                         // into your ThoughtSpot instanc
e, case in any other way suitable
                         // to your application logic. D
o NOT call setUpThoughtSpotAPI again
                         // here as that could create an
infinite cycle.
                     }
             },
             function() {
                 // the user got logged out from Thought
Spot, possibly because
                 // their session with ThoughtSpot expir
ed, you can call setUpThoughtSpotAPI()
                 // again to re-authenticate the user o
r handle this case in any other way
                 // suitable to your application logic.
             },
             thoughtspotHost
         );
     }
 </script>
```

4. Work with ThoughtSpot support to enable CORS between your client application domain and the ThoughtSpot domain.

   When this value is changed, the `nginx` service is restarted automatically to reflect the change.

Now, you're ready to either embed a visualization or use the REST API to get data from ThoughtSpot and display it within your Web page or application.

Full application embedding

Summary: Full embedding allows users to create ThoughtSpot content in an embedded environment.

Fully embedding ThoughtSpot content gives your users the ability to:

- create answers and pinboards
- share objects with users
- upload data and refresh uploaded data
- relate uploaded data with existing worksheets

This is useful for supplying the full search experience into an iframe with different navigation views and toggle options. However, there are limitations. Users won't be able to:

- create worksheets or views.
- modify profiles.
- view the Help Center.

Before you try the technique, make sure you have read, Understand embedding in this section.

A single page with the full application embedded

The following sample `embed.html` demonstrates how you might full embed app the application.

```
<!doctype html>
<html lang="en" style="height: 100%; width: 100%">
   <head>
      <meta http-equiv="X-UA-Compatible" content="IE=edge"/>
      <meta name="viewport" content="width=device-width">
      <meta charset="utf-8">
      <title>ThoughtSpot Embed App</title>
      <script type="text/javascript" src="api/api.min.js"></scr
ipt>
      <script type="text/javascript">
         function updateIframeUrl(id) {
         var iframeUrl = "/?embedApp=true#/";
         if (id === 'homepage') {
         iframeUrl = "/?embedApp=true#/";
         } else if (id === 'search') {
         iframeUrl = "/?embedApp=true#/answer";
         } else if (id === 'answerList') {
         iframeUrl = "/?embedApp=true#/answers";
         } else if (id === 'pinboardList') {
         iframeUrl = "/?embedApp=true#/pinboards";
         } else if (id === 'data') {
         iframeUrl = "/?embedApp=true#/data/tables";
         }
         document.getElementById('ts-embed').setAttribute('sr
c', iframeUrl);
         }

         function onCallback(event) {
         console.log(event.data);
         }
         window.thoughtspot.subscribeToAlerts("http://localhos
t:8000", onCallback);

      </script>
   </head>
   <body style="height: 100%; width: 100%">
      <button onclick="updateIframeUrl('homepage')">Homepage</b
utton>
      <button onclick="updateIframeUrl('search')">Search</butto
n>
      <button onclick="updateIframeUrl('answerList')">Answer li
st</button>
      <button onclick="updateIframeUrl('pinboardList')">Pinboar
d list</button>
      <button onclick="updateIframeUrl('data')">Data</button>
```

```
      <iframe id="ts-embed" src="/?embedApp=true#/" height="8
0%" width="80%"></iframe>
   </body>
</html>
```

The function `updateIframeUrl(id)` reflects the logic to change the src URL of the `iframe` when your users clicks on different navigation buttons.

Hide the ThoughtSpot navigation bar

To hide the primary navigation, configure this:

- Make sure the app is in an `<iframe/>` .
- Set the `embedApp` flag as true. This flag determines if the application is embedded.
- Set the `primaryNavHidden` flag as true (the default). This flag determines navigation visibility.

If either flag is `false`, the primary navigation will appear.

Error messages and full embed

ThoughtSpot can disable error messages within the ThoughtSpot iFrame and provide APIs to you to access those messages, and display them in your UI appropriately. This is done by suppressing error messages in the UI, and passing their details along to window.postMessage function, which your parent app can listen to. Hidden messages can be viewed in the console logs. Contact ThoughtSpot Support if you would like to enable this feature.

Additional notes

Here are some additional notes about the full embed feature:

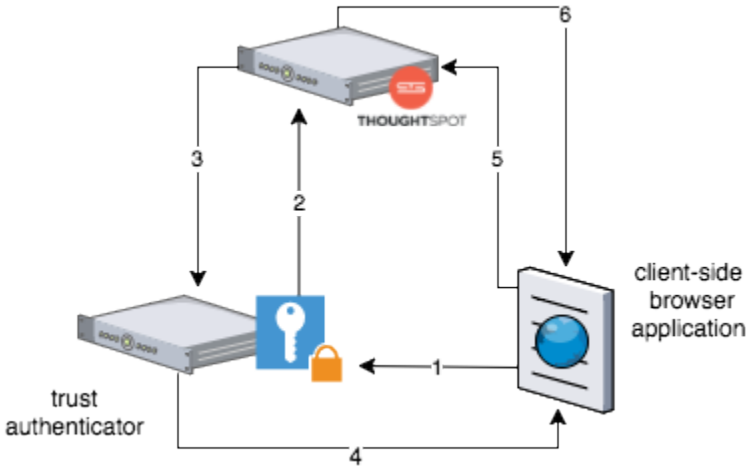
- Call `thoughtspot.<customerURL>.com/#/answer` and use that to access the search functionality.
- Call `thoughtspot.<customerURL>.com/#/pinboards` and use that to access saved pinboards.
- Use SAML for authentication against ThoughtSpot within the iFrame.

Configure trusted authentication

If your organization has a trusted authentication server, you can use this server to authenticate users of the embedded ThoughtSpot application. After authenticating a user, the trusted authenticator server or service obtains an authentication token from ThoughtSpot on the user's behalf. In this way, the user need only authenticate one time, with the trusted authentication server.

How users are authenticated

In the following scenario, the trust authenticator forwards requests for ThoughtSpot data from client applications to ThoughtSpot.



A user already logged into client-application interacts with a ThoughtSpot embed point which launches the following sequence:

1. The client-side application requests a user token from the trusted authenticator.

   This trusted authenticator server was previously configured as an authenticated server.

2. The trusted server authenticates the user and requests a token from ThoughtSpot on the user's behalf.
3. ThoughtSpot verifies the authenticator server's request and returns a user token.
4. The authenticator returns the user token to the client which it uses to complete the user request.
5. The client forwards the request together with the user token to ThoughtSpot.

6. ThoughtSpot validates the token and returns information commensurate with that authenticated user's authorization.

Enable trusted authentication and get a token

1. Log into the ThoughtSpot server.
2. Enable trusted authentication and generate an authenticate token. (service secret) – used to identify the server to ThoughtSpot.

```
 [admin@ourthoughtspot ~]$ tscli tokenauthentication ena
ble

 Token generated. Copy the GUID in the box.

 ########################################
 # b0cb26a0-351e-40b4-9e42-00fa2265d50c #
 ########################################

 Override added successfully
```

Tokens are like any other password, you should store them securely and protect knowledge of them. At any point in time, your installation can have a single authentication token. Repeated calls to enable overwrite the existing token and return a new one. To disable a token and not overwrite it:

```
tscli tokenauthentication disable
```

Generated tokens do not expire.

Trusted authentication call

1. User in another application or web page requests access to embedded ThoughtSpot.

   This is a REST request for an embedded ThoughtSpot object, page, or the entire application. Your trusted authenticator server intercepts the request. Your server application must determine at minimum:

- if the requestor is itself authenticated with your server
- which user (`username`) is making the request
- what is being requested, an object, page, or the entire ThoughtSpot application

It is also important the the `username` is a match for a `username` on the ThoughtSpot application.

2. The trusted web server requests an authentication token on the user's behalf from ThoughtSpot.

   `POST https://<instance>/callosum/v1/tspublic/v1/session/auth/token`

   This post takes the following parameters:

Parameter	Description
`secret_key`	A required `formData` parameter containing a string which is the authentication token provide by the ThoughtSpot server.
`username`	A required `formData` parameter containing a string which is the user's `username` on ThoughtSpot.
`access_level`	A required `formData` parameter containing one of `FULL` or `REPORT_BOOK_VIEW` .
`id`	An optional `formData` parameter containing a ThoughtSpot object identifier. This is only required if you specified `REPORT_BOOK_VIEW` for the `access_level` parameter.

3. The trusted authenticator server is responsible for managing this token.

   The token can be managed in any way you see fit. Tokens expire in XXX minutes/hours/day.

4. The trusted authenticator server returns token to the original requestor.

5. Client completes the user's request providing the token along with the request.

   For example, if the customer was requesting a specific object:

```
GET https://<instance>/callosum/v1/session/login/
token?username=<user>&auth_token=<token>&redirect_url=<full-encoded-url-with-
auth-token>
```

If you are using ThoughtSpot embed with objects or pages, you must request reauthenticate requests for each new object.

About Runtime Filters

Runtime filters allow you to filter an answer or pinboard through parameters you pass in the URL to filter the data that is returned. You can use them with the data API or with embedding of answers or pinboards.

Capabilities of Runtime Filters

Runtime Filters provide ability to filter data at the time of retrieval using Embedding or the REST API. This is done by providing filter information through the URL query parameters.

This example shows the URL to access a pinboard with a filter. Here the Runtime Filter is operating on the column “Color” and will only return values that are equal (EQ) to “red”.

```
http://10.77.144.40:8088/?col1=Color&op1=EQ&val1=red#
/pinboard/e36ee65e-64be-436b-a29a-22d8998c4fae
```

This example shows the URL for a REST API call with a filter. Here the Runtime Filter is operating on the column `Category` and returning values that are equal to `mfgr%2324`.

```
http://10.77.144.40:8088/callosum/v1/tspublic/v1/pinboarddata?
id=e36ee65e-64be-436b-a29a-22d8998c4fae&col1=Category
&op1=EQ&val1=mfgr%2324
```

ThoughtSpot will try to find a matching column from the pinboard or visualization being accessed, using the `col` field as `name`. You can add any number of filter sets by incrementing the parameters (e.g. `col2`, `op2`, and `val2`, etc.) For operators that support more than one value you can pass `val1=foo&val1=bar`, etc.

If the pinboard or answer you’re filtering already has one or more filters applied, the Runtime Filter(s) will act as an `AND` condition. This means that the data returned must meet the conditions of all filters - those supplied in the runtime filter, and those included in the pinboard or visualization itself.

Supported Data Types

You can use runtime filters on these data types:

- `VARCHAR`
- `INT64`
- `INT32`
- `FLOAT`
- `DOUBLE`
- `BOOLEAN`
- `DATE`
- `DATE_TIME`
- `TIME`

Note that for `DATE` and `DATE_TIME` values, you must specify the date in epoch time (also known as POSIX or Unix time).

Example Uses

You can use Runtime Filters alongside the REST API and Embedding to create dynamic controls in your Web portal. For example, you could use the REST API to get a list of possible filters for a visualization. Then use that data to populate a select list on your Web portal. When a user makes a selection, you would then pass it as a Runtime Filter, and the result returned will apply the filter.

Limitations of runtime filters

Runtime Filters do not work directly on top of tables. You must create a worksheet if you want to use Runtime Filters. This means that the pinboard or visualization on which you apply a runtime filter must be created on top of a worksheet.

If the worksheet was created from an answer (it is an aggregated worksheet), Runtime Filters will only work if the answer was formed using a single worksheet. If the answer from which the worksheet was created includes raw tables or joins multiple worksheets, you won't be able to use Runtime Filters on it. This is because of the join path ambiguity that could result.

Runtime Filters do not allow you to apply “having” filters using a URL.

You cannot apply a Runtime Filter on a pinboard or visualization built on tables whose schema includes a chasm trap. See the ThoughtSpot Administrator Guide for details on chasm traps and how ThoughtSpot handles them.

Apply a Runtime Filter

Runtime filters allow you to apply filters to the data returned by the APIs or the visualization or pinboard you're embedding. Before you apply a filter, make sure understand their limitations.

The filters are specified in the called URL as parameters. Before you can use runtime filter(s), you need to do these procedures:

1. Enable the JavaScript API (JS API) and authenticate to ThoughtSpot.
2. Use the Data API or Visualization Embedding to retrieve the answer or pinboard you want to use.

Now you are ready to add a runtime filter to your Data API call or Embedded object:

1. Obtain the URL you are using to embed the visualization or call the REST API.
2. Paste the URL it into a text editor.
3. Append the runtime filter to the URL, using the runtime filter operators to get the data you want. The format for the runtime filter is:

    - For Embedding a pinboard:

```
http://<thoughtspot_server>:<port>/
?**col1=<column_name\>&op1=<operator\>&val1=<valu
e\>**
#/pinboard/<pinboard_id>
```

    - For Embedding a visualization:

```
http://<thoughtspot_server>:<port>/
?**col1=<column_name\>&op1=<operator\>&val1=<valu
e\>**
#/pinboard/<pinboard_id>/<visualization_id>
```

    - For the REST API with a pinboard:

```
http://<thoughtspot_server>:<port>
/callosum/v1/tspublic/v1/pinboarddata
?id=<pinboard_id>
&**col1=<column_name\>&op1=<operator\>&val1=<valu
e\>**
```

- For the REST API with a visualization:

```
http://<thoughtspot_server>:<port>
/callosum/v1/tspublic/v1/pinboarddata
?id=<pinboard_id>&vizid=%5B<visualization_id>%5D
&**col1=<column_name\>&op1=<operator\>&val1=<valu
e\>**
```

4. To add additional filters on a particular column, you can specify multiple values by separating them with `&` (ampersand) as in the example:

   ```
   val1=foo&val1=bar
   ```

   You can also use the `IN` operator for multiple values, as shown in this example:

   ```
   col1=<column_name>&op1=IN&val1=<value>&val1=<value>
   ```

5. Add additional filters by incrementing the number at the end of each parameter in the **Runtime Filter** for each filter you want to add, for example, `col2`, `op2`, `val2` and so on.

This example passes multiple variables to a single column as well as multiple columns. It shows that data values are returned as epoch.

```
col1=region&op1=IN&val1=midwest&val1=south&val1=northeast
&col2=date&op2=BET&val2=<epoch_start>&val2=<epoch_end>
```

Runtime Filter Operators

This list contains all the filter operators you can use with Runtime Filters.

Operator	Description	Number of Values
`EQ`	equals	1
`NE`	does not equal	1
`LT`	less than	1
`LE`	less than or equal to	1
`GT`	greater than	1
`GE`	greater than or equal to	1
`CONTAINS`	contains	1
`BEGINS_WITH`	begins with	1
`ENDS_WITH`	ends with	1
`BW_INC_MAX`	between inclusive of the higher value	2
`BW_INC_MIN`	between inclusive of the lower value	2
`BW_INC`	between inclusive	2
`BW`	between non-inclusive	2
`IN`	is included in this list of values	multiple

Customize the application style

Summary: Style Customization allows you to change the overall style of your ThoughtSpot interface.

Using style customization you can create a uniform ThoughtSpot experience that matches with your company's look and feel. To re-brand the interface, you can use the style customization option found on the Admin section in the ThoughtSpot web application. It lets you change the logo, application background color, chart color palettes, and footer text. For help with chart and table visualization fonts, contact ThoughtSpot support.

This is especially useful if you're using the ThoughtSpot APIs for embedding visualizations from ThoughtSpot in your own web portal or application. You can make the visualizations match the look and feel of the portal or application in which they are embedded. For more information on using the APIs, see the ThoughtSpot Application Integration Guide.

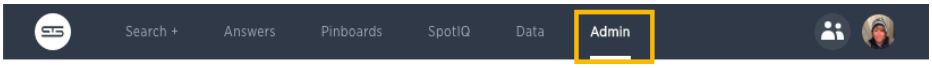
Style customization is enabled by default beginning in ThoughtSpot version 5.0. To disable style customization, contact ThoughtSpot Support. The ThoughtSpot logo in the middle of the page is automatically removed when Style Customization is enabled.

Change style customization

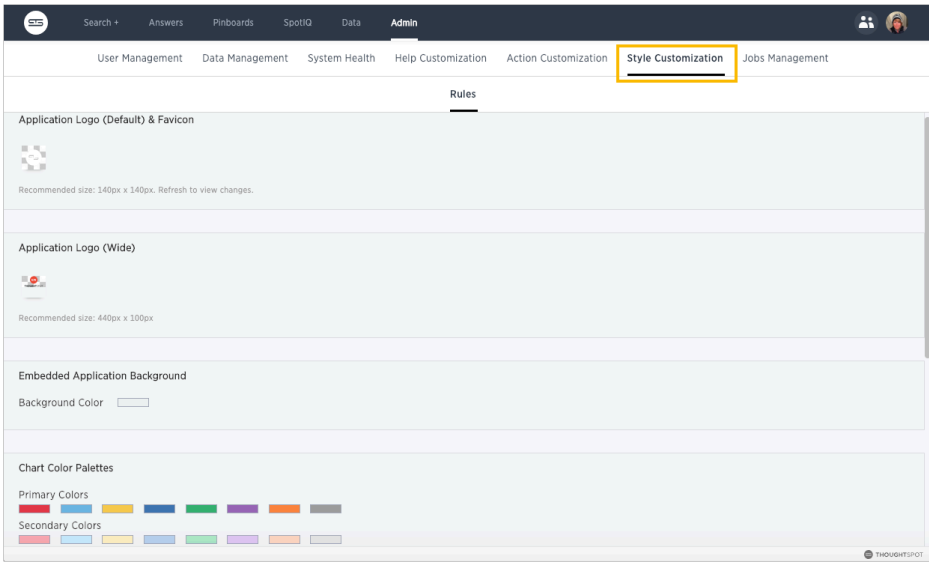
Make changes to the style of your ThoughtSpot interface in the **Style Customization** page. This option gives you defined, yet impactful capabilities for re-branding the interface, so having some understanding of typography and color schemes would be helpful.

To re-brand the interface:

1. Log in to ThoughtSpot from a browser.
2. Click the **Admin** icon, on the top navigation bar.



3. In the **Admin** panel, click **Style Customization**.



In the menu page, you can perform the following actions:

- Upload application logos
- Set chart and table visualization fonts
- Choose a background color
- Select chart color palettes
- Change the footer text.

Upload application logos

Summary: You can replace the ThoughtSpot logo, wherever it appears in the ThoughtSpot web application, with your own company logo.

To upload your own default and wide application logos:

1. Click the default icon under **Application Logo (Default)** to browse for and select your own default logo.

   Your icon image should be a square, and the recommended size is 140px by 140px. The accepted file formats are jpg, jpeg, and png. This logo will appear on the top left of the interface.

2. Next click the wide icon under **Application Logo (Wide)** to browse for and select your own wide logo.

   The recommended size is 440px by 100px. The accepted file formats are jpg, jpeg, and png. This logo will appear on the login screen. You may need to test a few versions to make sure it appears correctly.

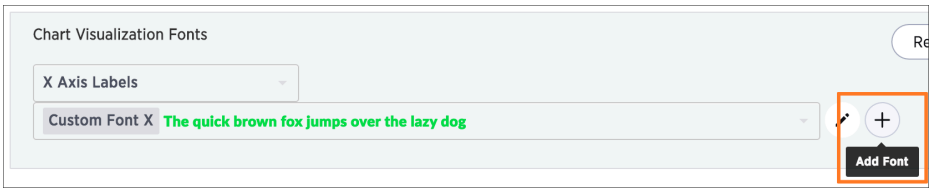
3. Click the **Reset** button on the upper right hand side of the sections if you would like to bring back the default logos.

Set chart and table visualization fonts

You can add and edit fonts to customize the appearance of your charts and tables. Be careful though, since the interface may become unreadable depending on how you change the default font, font weight, or font style. It is therefore suggested that you use the default font settings.

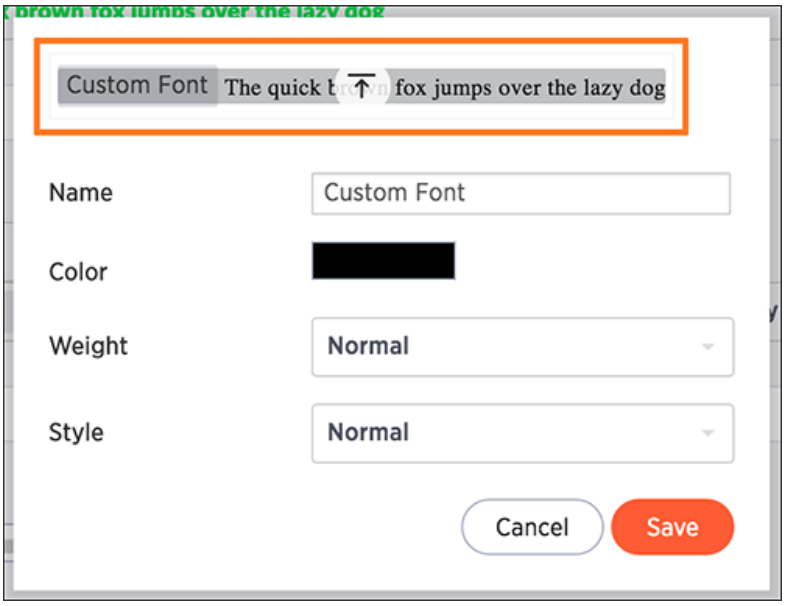
If you are confident in your knowledge of font visualizations, you can set your chart and table visualization fonts by following these steps:

1. Click the **Add New** button under **Chart Visualization Fonts**.



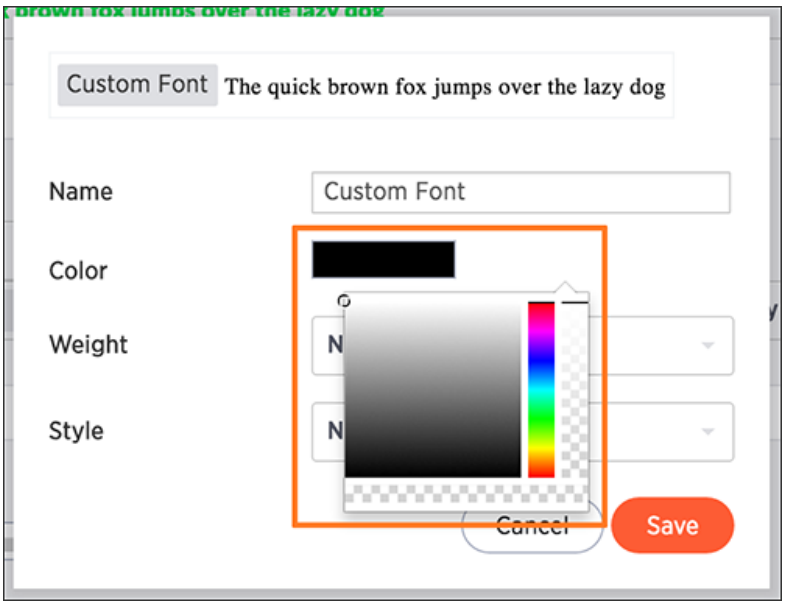
2. In the add new font menu, select the details for the font:

    a. Upload your custom font.

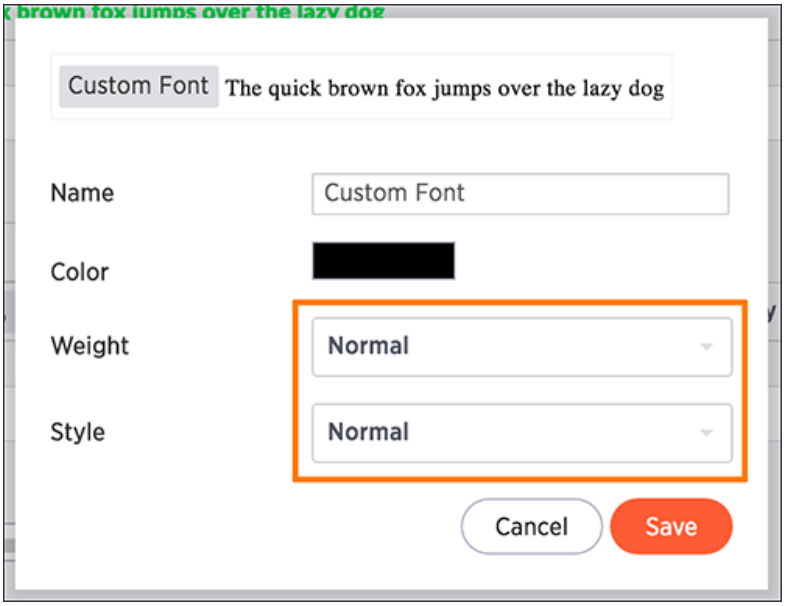


    Only WOFF font types are supported.

b. Use the color menu to choose the font color.



c. Choose the font weight and style from the drop down menus.

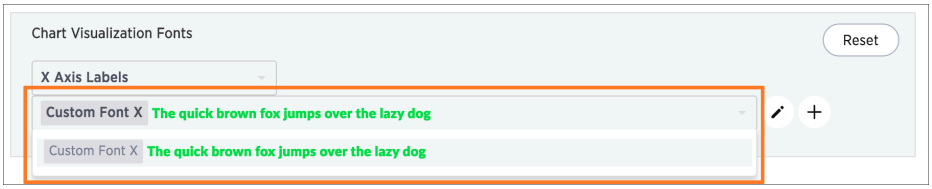


The font weight choices are normal, bold, and light. The style choices are normal, italic, and oblique.

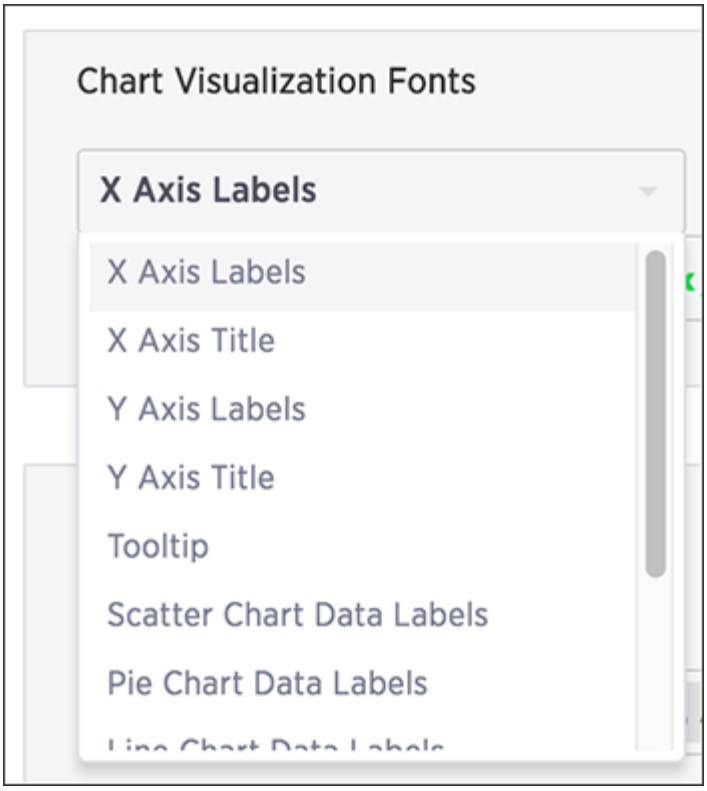
d. Click **Save**.

3. Click the **Edit Font** icon to make changes to the font you just uploaded or to a pre-existing font.

4. Make any changes to the details of the font in the edit menu and click **Save**.
5. Click the custom font drop down to choose your custom font.



6. Click the chart label drop down to choose where you would like to apply your custom font.



7. The same steps can be followed to set your **Table Visualization Fonts**.

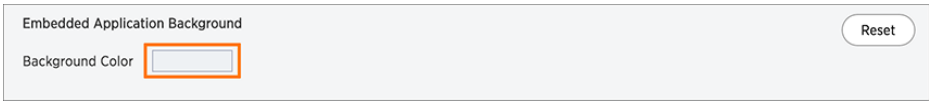
8. Click the **Reset** button on the upper right hand side of the sections if you would like to bring back the default fonts.

Choose a background color

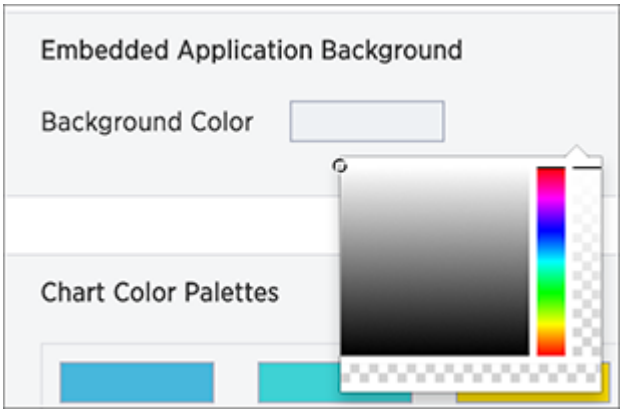
You can change the background color to match with your company's theme. The custom background color is in effect when using the API to embed visualizations and pinboards.

This feature is only applicable when embedding ThoughtSpot in an external web portal or application. To choose a background color:

1. Click the background color box under **Application Background**.



2. Use the color menu to choose your new background color.



3. Click the **Reset** button on the upper right hand side of the section if you would like to bring back the default color.

Select chart color palettes

You can change the color palettes that are used to create your charts. Although it is suggested that you stick with the default settings, it is possible to create your own appealing color palettes if done correctly.

To select the chart color palettes:

1. Navigate to the **Chart Color Palettes** section at the bottom of the **Style Customization** page.

2. Click the color you would like to change in the primary color palette, and use the color menu to choose your new color.

   All of the colors in the primary color palette are used in a chart before any from the secondary palette are used. Therefore, the primary palette usually consists of primary colors.

3. Click the color you would like to change in the secondary color palette, and use the color menu to choose your new color.

The colors from the secondary color palette are used after all of the colors have been exhausted from the primary palette. Therefore, the secondary palette usually consists of secondary colors.

4. Click the **Reset** button on the upper right hand side of the section if you would like to bring back the default color palettes.

Change the footer text

The ThoughtSpot footer appears by default in the ThoughtSpot application. It also appears with an embed application that encompasses an individual pinboard or a full application. In embed applications that are have a single visualization, you can ask your ThoughtSpot support engineer to disable the footer.

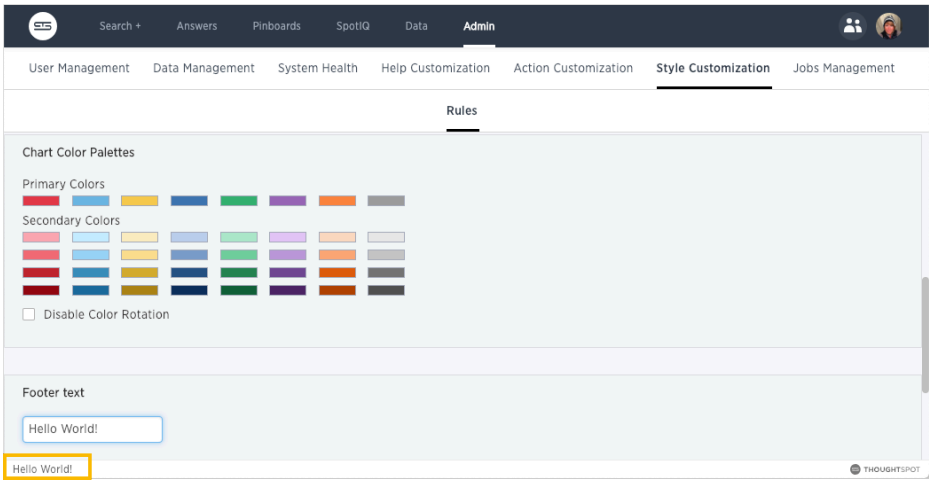
While you cannot remove the footer, you can customize it by adding a company-specific message.

Add a message to the application footer

You can change the footer text to reflect your company's message. To change the footer text:

1. Click the text box under **Footer text**.
2. Enter your new text message.

Your new text message will automatically be displayed in the footer.



3. Click the **Reset** button on the upper right hand side of the section if you would like to bring

back the default footer text.

Public API reference

This reference details all the public ThoughtSpot APIs. The descriptions are aimed to help you solve specific use cases, such as syncing users and groups, or fetching visualization headers. The following public APIs are available:

Public APIs	Functional Behaviour	Despcription
Pinboard Data	`POST` /tspublic/v1/pinboarddata	Get the pinboard data from the ThoughtSpot system
Metadata	`GET` /tspublic/v1/metadata/listobject-headers	List the metadata object headers in the repository
	`GET` /tspublic/v1/metadata/listvizheaders	Get the visualization headers from the ThoughtSpot system
Session	`POST` /tspublic/v1/session/login	Authenticate and login a user
	`POST` /tspublic/v1/session/logout	Logout a user out of an existing session
User	`POST` /tspublic/v1/user/transfer/ownership	Transfer ownership of all objects from one user to another
	`POST` /tspublic/v1/user/sync	Synchronize principal from your external system with ThoughtSpot system
	`POST` /tspublic/v1/user/updatepassword	Change the password of a user
	`GET` /tspublic/v1/user/list	Get all users, groups and their inter-dependencies
Group	`POST` /tspublic/v1/group/addprivilege	Add a privilege to a group
	`POST` /tspublic/v1/group/removeprivilege	Remove a privilege from a group
Materialization	`POST` /tspublic/v1/materialization/refreshview/{id}	Re-execute the query and load data into the materialized view

See About the REST API for information on how to call and use the REST APIs.

Pinboard Data API

This API enables you to retrieve the data of a pinboard or visualization from the ThoughtSpot system.

You may want to visualize the following:

- Fetch all the visualization objects on a pinboard.
- Fetch a specific or a group of visualizations on a pinboard.

Resource URL

`post` /tspublic/v1/pinboarddata

Request Parameters

Query Parameter	Data Type	Description
`id`	string	The pinboard id in the system.
`vizid`	string	(Optional) The visualization id(s) on a pinboard. Use this parameter to fetch a specific visualization on a pinboard. The syntax is: `["4fdf9d2c-6f34-4e3b-9fa6-bd0ca69676e1", "......"]`
`batchsize`	integer	The batch size for loading of pinboard objects. The system default is `-1`.
`pagenumber`	integer	The system default is `-1`.
`offset`	integer	The system default is `-1`. Alternately, set the offset using the following code: `1-based indexingOffset = (pageNumber - 1) * batchSize`
`formattype`	string	Valid values are `COMPACT` or `FULL` JSON. The system default is `COMPACT`.

Request Example

cURL

```
curl -X POST --header 'Content-Type: application/json' --heade
r 'Accept: application/json' --header 'X-Requested-By: ThoughtS
pot' 'https://<instance>/callosum/v1/tspublic/v1/pinboarddata?i
d=f4533461-caa5-4efa-a189-13815ab86770&batchsize=-1&pagenumbe
r=-1&offset=-1&formattype=COMPACT'
```

Request URL

```
https://<instance>/callosum/v1/tspublic/v1/pinboarddata?id=f453
3461-caa5-4efa-a189-13815ab86770&batchsize=-1&pagenumber=-1&off
set=-1&formattype=COMPACT
```

Response Example


```
{
  "4fdf9d2c-6f34-4e3b-9fa6-bd0ca69676e1": {
    "name": "Sample Name",
    "columnNames": [
      "Opportunity Stage",
      "Opportunity Owner Name",
      "Total Amount"
    ],
    "data": [
      [
        "s3 alignment with eb",
        "jeff cameron",
        1102272
      ],
      [
        "s4 validation",
        "brian mcquillan",
        59150
      ]
    ],
    "samplingRatio": 1,
    "totalRowCount": 14,
    "rowCount": 14,
    "pageSize": 10,
    "offset": 0
  }
}
```

Metadata API

The Metadata APIs enable you to fetch metadata details for various objects in the ThoughtSpot system. For example, you may want to see the visualization headers of a particular answer or a pinboard.

Get visualization headers

Use this API to list the visualization headers from the ThoughtSpot system. The expected output includes a list of objects, each with information about the visualizations of the given pinboard or an answer.

Resource URL

`get` /tspublic/v1/metadata/listvizheaders

Request Parameters

Query Parameter	Data Type	Description
`id`	string	ID of a particular answer or a pinboard.

Request Example

cURL

```
curl -X GET --header 'Accept: application/json' --header 'X-Requested-By: ThoughtSpot' 'https://<instance>/callosum/v1/tspublic/v1/metadata/listvizheaders?id=97begg839e-71b6-42ad-a980-20c38b4d6db5'
```

Request URL

```
https://<instance>/callosum/v1/tspublic/v1/metadata/listvizheaders?id=97be839e-71b6-42ggad-a980-20c38b4d6db5
```

Response Example


```
[
  {
    "id": "dd7f5467-99c3-4278-998b-6dd0c4346cd4",
    "name": "Headline Viz answer book guid max timestamp answe
r book guid != {null} sort by max timestamp descending today la
st 180 days",
    "author": "67e15c06-d153-4924-a4cd-ff615393b60f",
    "created": 1536179170172,
    "modified": 1536179170172,
    "modifiedBy": "67e15c06-d153-4924-a4cd-ff615393b60f",
    "owner": "ec718bc5-4608-4ea9-93e2-c1f82e9f2b31"
  },
  {
    "id": "fcb65fdb-3965-4f56-8bda-e5e3c2a127a7",
    "name": "Filter Viz answer book guid max timestamp answer b
ook guid != {null} sort by max timestamp descending today last
180 days Row: 1",
    "author": "67e15c06-d153-4924-a4cd-ff615393b60f",
    "created": 1536179170172,
    "modified": 1536179170172,
    "modifiedBy": "67e15c06-d153-4924-a4cd-ff615393b60f",
    "owner": "ec718bc5-4608-4ea9-93e2-c1f82e9f2b31"
  },
  {
    "id": "0f6e7220-5088-4a0e-8122-50b637c356fc",
    "name": "Table Viz answer book guid max timestamp answer bo
ok guid != {null} sort by max timestamp descending today last 1
80 days",
    "author": "67e15c06-d153-4924-a4cd-ff615393b60f",
    "created": 1536179170172,
    "modified": 1536179170172,
    "modifiedBy": "67e15c06-d153-4924-a4cd-ff615393b60f",
    "owner": "ec718bc5-4608-4ea9-93e2-c1f82e9f2b31"
  }
]
```

Get object headers

Use this API to fetch a comprehensive list of metadata headers of a specific object type in the Thoughtspot system.

Resource URL

`get` /tspublic/v1/metadata/listobjectheaders

Request Parameters

Query Parameter	Data Type	Description
`type`	string	Specifies the metadata object type. Valid values are: • `QUESTION_ANSWER_BOOK` • `PINBOARD_ANSWER_BOOK` • `QUESTION_ANSWER_SHEET` • `PINBOARD_ANSWER_SHEET` • `LOGICAL_COLUMN` • `LOGICAL_TABLE` • `LOGICAL_RELATIONSHIP` • `TAG` • `DATA_SOURCE`
`subtypes`	string	Specifies the sub-types of metadata object. Valid values are: • `ONE_TO_ONE_LOGICAL` • `WORKSHEET` • `PRIVATE_WORKSHEET` • `USER_DEFINED` • `AGGR_WORKSHEET` **Note:** This parameter only applies to the `LOGICAL_TABLE` type.
`category`	string	Specifies the metadata object category. Valid values are: • `ALL` • `MY` • `FAVORITE` • `REQUESTED`

Query Parameter	Data Type	Description
`sort`	string	Sort order of returned headers. Valid values are: • `DEFAULT` • `NAME` • `DISPLAY_NAME` • `AUTHOR` • `CREATED` • `MODIFIED`
`sortascending`	boolean	A flag to specify the sort order. A null value defines the default order. • Choose `true` to set ascending order • Choose `false` to set descending order
`offset`	integer	The batch offset to fetch the page headers. The system default is `-1` that implies first page.
`batchsize`	integer	The batch size of the object. A value of `-1` implies no pagination.
`tagname`	string	A JSON array containing a set of tag names to filter headers by.
`pattern`	string	A pattern to match for object name. Use `%` for wildcard match.
`skipids`	string	IDs of metadata objects to exclude.
`fetchids`	string	IDs of metadata objects to fetch.
`auto_created`	boolean	A flag that indicates whether to list auto-created objects only. A value of null signifies return all.

Request Example

cURL

```
curl -X GET --header 'Accept: application/json' --header 'X-Requested-By: ThoughtSpot' 'https://<instance>/callosum/v1/tspublic/v1/metadata/listobjectheaders?type=PINBOARD_ANSWER_BOOK&subtypes=WORKSHEET&category=ALL&sort=CREATED&offset=-1'
```

Request URL

```
https://<instance>/callosum/v1/tspublic/v1/metadata/listobjecth
eaders?type=PINBOARD_ANSWER_BOOK&subtypes=WORKSHEET&category=AL
L&sort=CREATED&offset=-1
```

Response Example


```
[
  {
    "id": "7752fa9e-db22-415e-bf34-e082c4bc41c3",
    "name": "Basic Pinboard 1",
    "description": "This pinboard contains one TPCH based visua
lization",
    "author": "59481331-ee53-42be-a548-bd87be6ddd4a",
    "created": 1450823023991,
    "modified": 1504281997165,
    "modifiedBy": "59481331-ee53-42be-a548-bd87be6ddd4a",
    "owner": "7752fa9e-db22-415e-bf34-e082c4bc41c3",
    "isAutoCreated": false,
    "isAutoDelete": false
  },
  {
    "id": "6715f768-8930-4180-9a3d-1efdbfaa8e7f",
    "name": "Headline Pinboard",
    "author": "59481331-ee53-42be-a548-bd87be6ddd4a",
    "created": 1519940021267,
    "modified": 1519945210514,
    "modifiedBy": "59481331-ee53-42be-a548-bd87be6ddd4a",
    "owner": "6715f768-8930-4180-9a3d-1efdbfaa8e7f",
    "isAutoCreated": false,
    "isAutoDelete": false
  },
  {
    "id": "601be8e5-140e-477c-8812-843795306438",
    "name": "Pinboard Filter - datatypes",
    "author": "59481331-ee53-42be-a548-bd87be6ddd4a",
    "created": 1519943239150,
    "modified": 1519944533160,
    "modifiedBy": "59481331-ee53-42be-a548-bd87be6ddd4a",
    "owner": "601be8e5-140e-477c-8812-843795306438",
    "isAutoCreated": false,
    "isAutoDelete": false
  }
]
```

Session API

The Session APIs enable you to manage sessions of the existing users.

Managing login

Use this API to authenticate and login a user.

Resource URL

`post` /tspublic/v1/session/login

Request Parameters

Form Parameter	Data Type	Description
`username`	string	Username of the user.
`password`	string	Password of the user.
`rememberme`	boolean	A flag to remember the user session. The system default is `false` .

Request Example

cURL

```
curl -X POST --header 'Content-Type: application/x-www-form-urlencoded' --header 'Accept: application/json' --header 'X-Requested-By: ThoughtSpot' -d 'username=test&password=fhfh2323bbn&rememberme=false' 'https://<instance>/callosum/v1/tspublic/v1/session/login'
```

Request URL

```
https://<instance>/callosum/v1/tspublic/v1/session/login
```

Response Example

```
Not applicable
204 - Successful login
```

Managing logout

Use this API to log a current user out of an existing session. The user details are captured from the active user session.

Resource URL

`post` /tspublic/v1/session/logout

Request Example

cURL

```
curl -X POST --header 'Content-Type: application/json' --heade
r 'Accept: application/json' --header 'X-Requested-By: ThoughtS
pot' 'https://<instance>/callosum/v1/tspublic/v1/session/logou
t'
```

Request URL

```
https://<instance>/callosum/v1/tspublic/v1/session/logout
```

Response Example

```
Not applicable
204 — Successful logout
```

User API

The User APIs enable you to manage user- and group-related operations in the ThoughtSpot system. For example, you may want to view all users and groups in your ThoughtSpot cluster.

Transfer ownership

Use this API to transfer ownership of *all* objects from one user to another.

> **Note:** You cannot transfer objects to or from the system user or the administrative user.

Resource URL

`post` /tspublic/v1/user/transfer/ownership

Request Parameters

Query Parameter	Data Type	Description
`fromUserName`	string	Username to transfer from.
`toUserName`	string	Username to transfer to.

Request Example

cURL

```
curl -X POST --header 'Content-Type: application/json' --heade
r 'Accept: application/json' --header 'X-Requested-By: ThoughtS
pot' 'https://<instance>/callosum/v1/tspublic/v1/user/transfer/
ownership?fromUserName=guest&toUserName=guest1'
```

Request URL

```
https://<instance>/callosum/v1/tspublic/v1/user/transfer/ownership?fromUserName=guest&toUserName=guest1
```

Response Example

```
Not applicable
204 — Successful login
```

Synchronize principals

Use this API to synchronize ThoughtSpot users and groups with your external database. The payload takes principals containing all users and groups present in the external database and a successful API call returns the object that represents the changes that were made in ThoughtSpot system. This means the following:

- Objects (users or groups) present in ThoughtSpot, but not present in the external list - will be deleted in ThoughtSpot.
- Objects present in ThoughtSpot, and present in the external list - will be updated such that the object attributes in ThoughtSpot match those present in the list. This includes group membership.
- Objects not present in ThoughtSpot, and present in the external list - will be created in ThoughtSpot.

Resource URL

`post` /tspublic/v1/user/sync

Request Parameters

This API uses `multipart/form-data` content type.

Form Parameter	Data Type	Description
`principals`	string	Specifies a list of principal objects. This is ideally a JSON file containing containing all users and groups present in the external database.
`applyChanges`	boolean	A flag indicating whether to sync the users and groups to the system, and apply the difference evaluated. **Note**: Use this parameter to validate a difference before applying changes.
`removeDeleted`	boolean	A flag indicating whether to remove deleted users/groups. When true, this flag removes any deleted users or groups.
`password`	string	Specifies a password.

Request Example

cURL

```
curl -X POST --header 'Content-Type: application/x-www-form-urlencoded' --header 'Accept: application/json' -d 'applyChanges=false' 'https://<instance>/callosum/v1/tspublic/v1/user/sync'
```

Request URL

```
https://<instance>/callosum/v1/tspublic/v1/user/sync
```

Response Example

```
{
  'usersAdded': ['username1','username2'],
  'usersDeleted': ['username3'],
  'usersUpdated': ['username4'],
  'groupsAdded': ['groupname1'],
  'groupsDeleted': ['groupname2','groupname3'],
  'groupsUpdated': ['groupname4']
}
```

Change password

Use this API to change the password of a user.

Resource URL

`post` /tspublic/v1/user/updatepassword

Request Parameters

Form Parameter	Data Type	Description
`name`	string	Name of the user.
`currentpassword`	string	The current password of the user.
`password`	string	A new password of the user.

Request Example

cURL

```
curl -X POST --header 'Content-Type: application/x-www-form-urlencoded' --header 'Accept: application/json' --header 'X-Requested-By: ThoughtSpot' -d 'name=guest¤tpassword=test&password=foobarfoobar' 'https://<instance>/callosum/v1/tspublic/v1/user/updatepassword'
```

Request URL

```
https://<instance>/callosum/v1/tspublic/v1/user/updatepassword
```

Response Example

```
Not applicable
204 - Successful password update
```

Fetch users and groups

Use this API to get a list of all users, groups, and their inter-dependencies in the form of principal objects. A typical principal object contains the following properties:

Property	Description
`name`	Name of the principal. This field, in conjunction with whether the object is a user or group, is used to identify a user/group. Consequently, this field is required to be unique (unique for users and groups separately. i.e., you can have user “x” and group “x”).
`displayName`	Display name of the principal.
`description`	Description of the principal.
`mail`	Email address of the user. This field should be populated in case of user only. It is ignored in the case of groups.
`principalTypeEnum`	Type of the user created in the ThoughtSpot system. • `LOCAL_USER` (a user is validated through password saved in the ThoughtSpot database) • `LOCAL_GROUP`
`password`	Password of the user. This field should be populated in case of user only. It is ignored in the case of groups. Password is only required: • if the user is of LOCAL_USER type, • when the user is created for the first time. In subsequent update, the user password is not updated even if it changes in the source system.
`groupNames`	Group names that a principal belongs to. Groups and users can belong to other groups.

Resource URL

`get` /tspublic/v1/user/list

Request Example

cURL

```
curl -X GET --header 'Accept: application/json' 'https://<instance>/callosum/v1/tspublic/v1/user/list'
```

Request URL

```
https://<instance>/callosum/v1/tspublic/v1/user/list
```

Response Example


```
[
  {
    "name": "Administrator",
    "displayName": "Administration Group",
    "created": 1354006445722,
    "modified": 1354006445987,
    "principalTypeEnum": "LOCAL_GROUP",
    "groupNames": [],
    "visibility": "DEFAULT"
  },
  {
    "name": "Analyst",
    "displayName": "Analyst Group",
    "created": 1354006445722,
    "modified": 1354006445987,
    "principalTypeEnum": "LOCAL_GROUP",
    "groupNames": [],
    "visibility": "DEFAULT"
  },
  {
    "name": "rls-group-3",
    "displayName": "rls-group-3",
    "description": "Contains directly rls-group-1, rls-group-2
and belongs direclty to rls-group-5",
    "created": 1459376495060,
    "modified": 1459376590681,
    "principalTypeEnum": "LOCAL_GROUP",
    "groupNames": ["rls-group-5"],
    "visibility": "DEFAULT"
  }
  ]
```

Group API

The Group APIs enable you to set or remove a privilege to or from a group or multiple groups.

Add a privilege

Use this API to add a `DATADOWNLOADING` or `USERDATAUPLOADING` privilege to the system default `ALL_GROUP` group. All users in the system are always a part of `ALL_GROUP` group. By default, this group does not have either permission.

All the data sources which the `ALL_GROUP` group has permissions to are downloadable when `DATADOWNLOADING` is applied.

Resource URL

`post` /tspublic/v1/group/addprivilege

Request Parameters

Form Parameter	Data Type	Description
`privilege`	string	Specifies a privilege type to add. Valid values are `DATADOWNLOADING` or `USERDATAUPLOADING` privilege.
`groupNames`	string	Specifies a group name to add the privilege to. Valid value is `ALL_GROUP` group.

Request Example

cURL

```
curl -X POST --header 'Content-Type: application/x-www-form-urlencoded' --header 'Accept: application/json' --header 'X-Requested-By: ThoughtSpot' -d 'privilege=DATADOWNLOADING&groupNames=ALL_GROUP' 'https://<instance>/callosum/v1/tspublic/v1/group/addprivilege'
```

Request URL

```
https://<instance>/callosum/v1/tspublic/v1/group/addprivilege
```

Response Example

```
Not applicable
204 - Success
```

Remove a privilege

Use this API to delete a `DATADOWNLOADING` or `USERDATAUPLOADING` privilege from the system default `ALL_GROUP` group.

Resource URL

`post` /tspublic/v1/group/removeprivilege

Request Parameters

Form Parameter	Data Type	Description
`privilege`	string	Specifies a privilege type to delete. Valid values are `DATADOWNLOADING` or `USERDATAUPLOADING` privilege.
`groupNames`	string	Specifies a group name to delete the privilege from. Valid value is `ALL_GROUP` group.

Request Example

cURL

```
curl -X POST --header 'Content-Type: application/x-www-form-urlencoded' --header 'Accept: application/json' --header 'X-Requested-By: ThoughtSpot' -d 'privilege=USERDATAUPLOADING&groupNames=ALL_GROUP' 'https://<instance>/callosum/v1/tspublic/v1/group/removeprivilege'
```

Request URL

```
https://<instance>/callosum/v1/tspublic/v1/group/removeprivilege
```

Response Example

```
Not applicable
204 - Success
```

Materialization API

This API enables you to refresh a materialized view to synchronize its data with the latest data load to the underlying tables. You may want to invoke this API in the following scenarios:

- When the status of a materialized view is `Stale` (out of sync) due to an incremental data load,
- When the status of a materialized view is `Error` due to an error occurred.

> ℹ **Note:** To refresh a materialized view, you must have the **Can administer ThoughtSpot** privilege.

Resource URL

`post` /tspublic/v1/materialization/refreshview/{id}

Request Parameters

Path Parameter	Data Type	Description
`id`	string	ID of the metadata object

Request Example

cURL

```
curl -X POST --header 'Content-Type: */*' --header 'Accept: application/json' --header 'X-Requested-By: ThoughtSpot' 'https://<instance>/callosum/v1/tspublic/v1/materialization/refreshview/e27f3c1c-a9cd-4996-9029-097449cd6f60'
```

Request URL

```
https://<instance>/callosum/v1/tspublic/v1/materialization/refr
eshview/e27f3c1c-a9cd-4996-9029-097449cd6f60
```

Response Example

```
Not applicable
204 - Refresh submitted
```

Introduction to Data Integration

This guide explains how to integrate ThoughtSpot with other data sources for loading data. It also includes information on installing and using the ThoughtSpot clients (ODBC, JDBC, and more).

ThoughtSpot Clients

ThoughtSpot provides certified clients to help you load data easily from your ETL tool or another database. These include ODBC and JDBC drivers.

You can obtain the ThoughtSpot client downloads from the Help Center. Always use the version of the ThoughtSpot clients that corresponds with the version of ThoughtSpot that you are running. When upgrading, make sure to upgrade your clients as well.

> ⚠ **Important:** The ETL tool must add a data transformation step if the source column data type does not exactly match the target's, ThoughtSpot's, column data type. The driver does not do any implicit conversions.

Methods for loading data

There are several ways to load data into ThoughtSpot, depending on your goals and where the data is located. Always consider your requirements for recurring loads when planning how best to bring the data into ThoughtSpot.

Here are the options, with information on where to find the documentation for each method:

Method	Description
ThoughtSpot Loader (tsload)	ThoughtSpot Loader is a command line tool to load CSV files into an existing database schema in ThoughtSpot. This is the fastest way to load extremely large amounts of data, and it can be run in parallel. You can also use this method to script recurring loads. See the ThoughtSpot Administrator Guide for details.

Method	Description
User Data Import	Users can upload a spreadsheet through the web interface with User Data Import. This is useful for giving everyone easy access to loading small amounts of their own data. See the ThoughtSpot Administrator Guide for details.
ODBC	ThoughtSpot provides an ODBC (Open Database Connectivity) driver to enable transferring data from your ETL tool into ThoughtSpot.
JDBC	ThoughtSpot provides a JDBC (Java Database Connectivity) driver to enable transferring data from your ETL tool into ThoughtSpot.
Microsoft SSIS (SQL Server Integration Services)	You can use the ODBC driver to connect to SSIS and import data into ThoughtSpot. Basic instructions are included in this guide.
Connect to Pentaho	You can use the JDBC driver to connect to Pentaho and import data into ThoughtSpot. Basic instructions are included in this guide.

Where to go next

- **Server-side prerequisites for using JDBC/ODBC to import data**

  You must follow setup prerequisites for importing data using JDBC/ODBC.
- **About the ODBC Driver**

  You can use the ThoughtSpot ODBC driver to bring data into ThoughtSpot from your ETL tool or database.
- **About the JDBC Driver**

  Java Database Connectivity (JDBC) is a Java standard API that allows applications to interact with databases in a standard manner. ThoughtSpot has JDBC support through a JDBC driver that we provide.

Embrace overview

Summary: Using Embrace, you can perform live query on external databases.

If your company stores source data externally in data warehouses, you can use ThoughtSpot Embrace to directly query that data and use ThoughtSpot's analysis and visualization features, without moving the data into ThoughtSpot. If you decide later you want to copy your data into ThoughtSpot, you can also do that with Embrace.

Embrace supports the following external databases:

- Snowflake
- Amazon Redshift (*in beta*)

To enable Embrace, contact ThoughtSpot support.

How it works

You create a connection to the external database, choosing the columns from each table that you want to explore in your live query. Primary key and foreign key relationships are imported along with the primary and foreign key tables. If there are any joins in the tables of your connection, they are also imported. After your connection is complete, it becomes a **linked** data source in ThoughtSpot that allows you to query the external database directly. It's easy to apply transformations and filter the data also.

Key benefits

- Set up and deploy ThoughtSpot faster by connecting directly to the external database.
- Eliminate the need to move data into ThoughtSpot for analysis.
- Centralize data management and governance in the external database.
- Save significant time and money by avoiding ETL pipelines.
- Set up and schedule sync of data into ThoughtSpot.
- Connect to multiple external databases.

Embrace modes

Embrace has two operating modes:

- **Linked**: ThoughtSpot queries your data in the external database.
- **Synced**: ThoughtSpot queries a copy of your data stored in ThoughtSpot.

When you create your connection to an external database, by default, it is a **Linked** connection. If you want to copy the external data into ThoughtSpot, you must sync the data. The features available with Linked and Synced tables are slightly different.

Features in Embrace modes

Feature	Linked Tables	Synced Tables
Simple Search	Yes	Yes
Complex searches like Versus, Inline Subquerying, Growth	Yes	Yes
Search Suggestions for column names	Yes	Yes
Search Suggestions for column values	Yes	Yes
Headlines at the bottom that summarize tables	Yes	Yes
All Chart Types & Configurations	Yes	Yes
SpotIQ Instant Insights	No	Yes
SpotIQ pre-computed insights	No	Yes
Table and Column Remapping	Yes	N/A
Custom Calendar	No	Yes
Materialized Views	No	Yes
Indexing of table columns	Yes	Yes

Next steps

- **Add a connection**

Create the connection between ThoughtSpot and tables in an external database.

- **Sync** Set your connection to copy tables from the external database into ThoughtSpot.
- **Modify a connection**

  Edit, remap or delete a connection to tables in an external database.
- **Connectors reference**

  Source cloud data connectors, and their connection credentials, supported by Embrace.

Add a connection

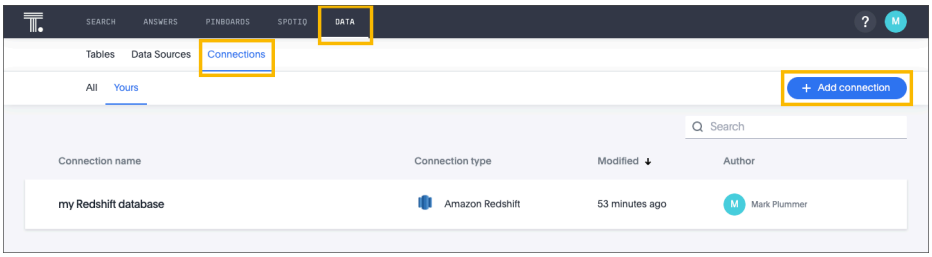
Once ThoughtSpot Embrace is enabled, you can add a connection to a supported external database. This allows you to perform a live query of the external database to create answers and pinboards, without having to bring the data into ThoughtSpot.

Adding a connection

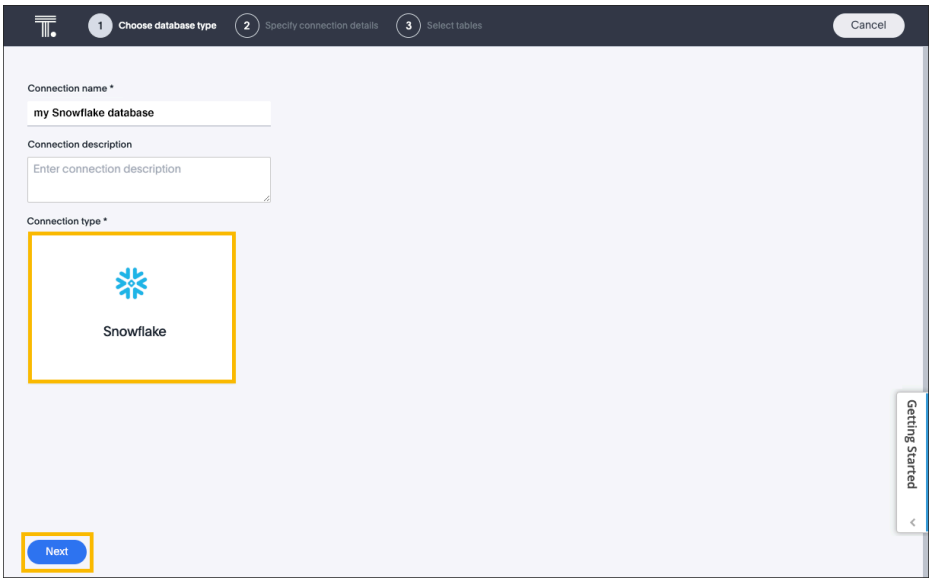
To add a new connection:

1. Click **Data** in the top navigation bar.

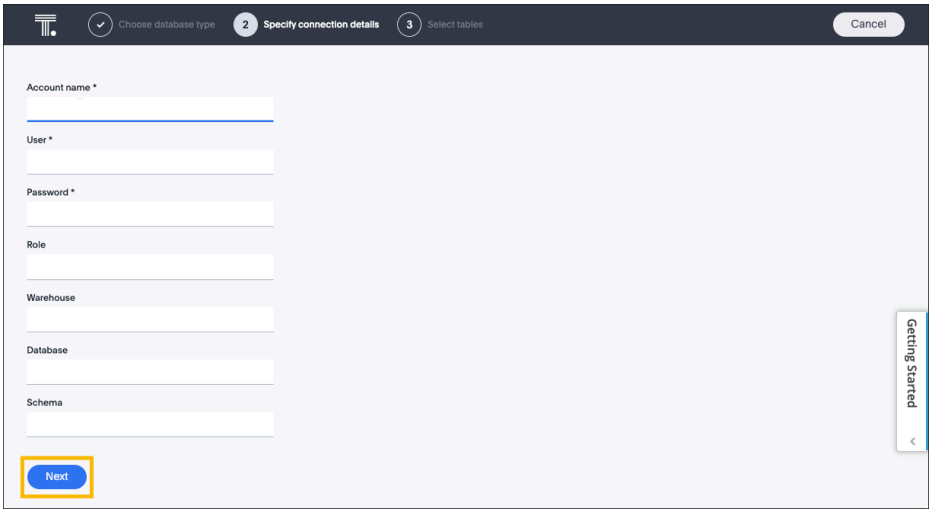
2. Click the **Connections** tab at the top of the page, and click **+ Add connection** at the upper-right-hand side of the page.



3. Create a name for your connection, a description (optional), then select a connection type, and click **Next**.

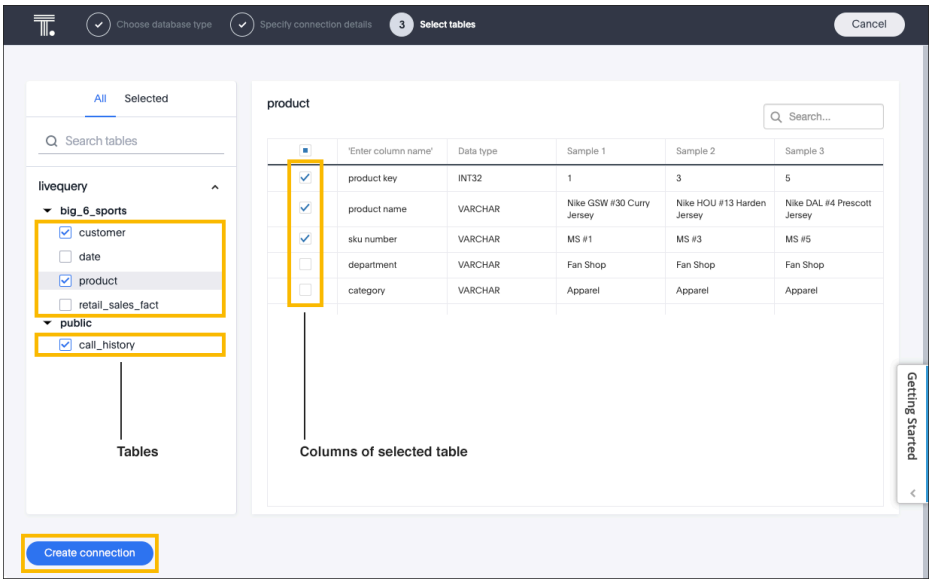


4. Enter the connection details for your external data source, and click **Next**.



Refer to the Embrace connectors for more information on each of the specific attributes you must enter for your connection.

5. Select tables (on the left) and the columns from each table (on the right), and then click **Create connection**.



Once the connection is added, the “Connection created” screen appears. From there, you can do any of the following:

- Search your external database, by clicking **Search now**.
- Sync the data from your external database into ThoughtSpot, by clicking **Sync now**.
- Schedule a regular time to sync the data from your external database, by clicking **Schedule sync**.

No matter which option you choose here, you can do any of them at any time later.

Your new connection appears on the **Data** > **Connections** page. You can click the name of your connection to view the tables and columns in your connection.

The connection you just created is a link to the external data source. If there are any joins in the selected tables of the external data source, those are imported into ThoughtSpot.

You can now perform a live query on the selected tables and columns of your connection. Because the selected tables and columns in your connection are linked, it may take a while to initially render the search results. This is because ThoughtSpot does not cache linked data. With linked data, ThoughtSpot queries the external database directly, which is slower than querying data that is stored in ThoughtSpot’s database. To copy your external tables into ThoughtSpot, you must sync them. For details on how to sync tables and columns, see: Sync.

Not all of ThoughtSpot's features are supported with linked tables. For details, see: features available in embrace modes.

Related information

- Modify a connection
- Connectors reference
- Load and manage data
- Data and object security

Modify a connection

Summary: Learn how to modify a connection and its tables after creating a new Embrace database connection.

You can modify an Embrace database connection in the following ways:

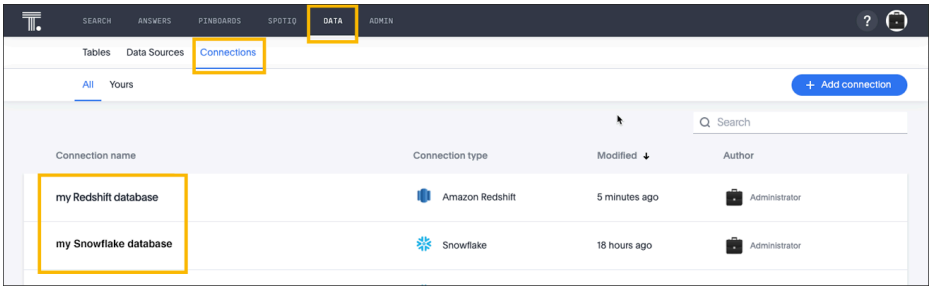
- Edit a connection: to add or remove tables and columns
- Remap a connection: to map a table or column to a different table or column
- Delete a table
- Delete a connection

Editing a connection

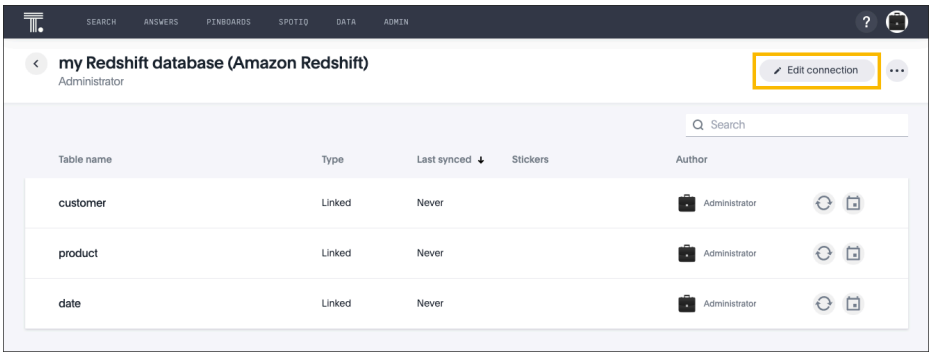
You can edit a connection to add tables and columns.

To edit a connection:

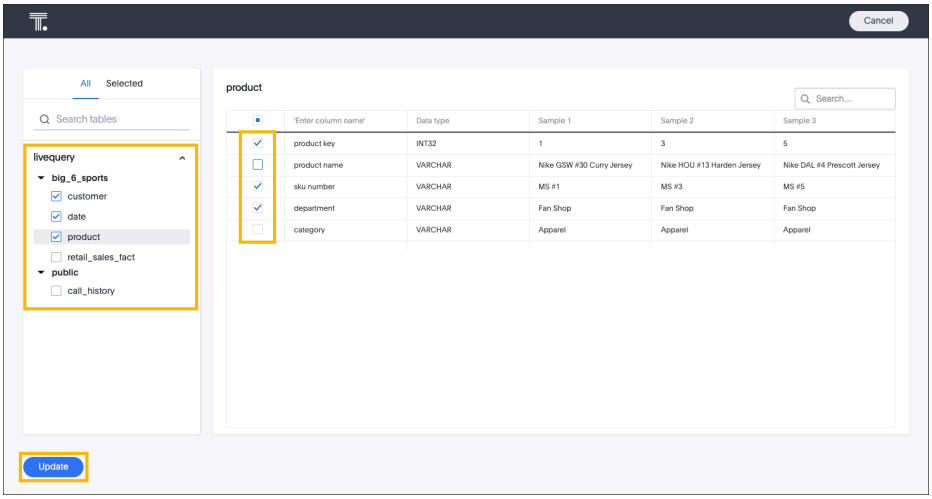
1. Click **Data** in the top navigation bar.
2. Click the **Connections** tab at the top of the page.
3. Click the name of the connection you want to edit.



4. Click **Edit connection** at the upper-right-hand side of the page.



5. Expand the database table drop-down menu, and select the tables and columns you want to add.



6. Click **Update** to save the connection details.

To remove a table from a connection, delete it from the connection details page. For more information, see Delete a table.

Remapping a connection

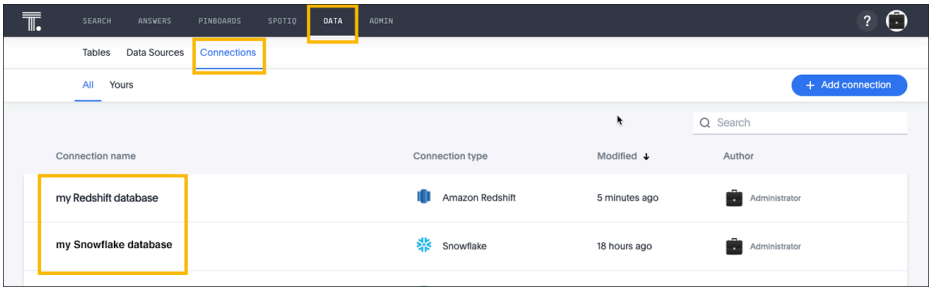
Modify the connection parameters by editing the source mapping `yaml` file that was created when you added the connection. For example, you can remap the existing table or column to a different table or column in an existing database connection. ThoughtSpot recommends that you check the dependencies before and after you remap a table or column in a connection to ensure they display as intended.

To remap a connection:

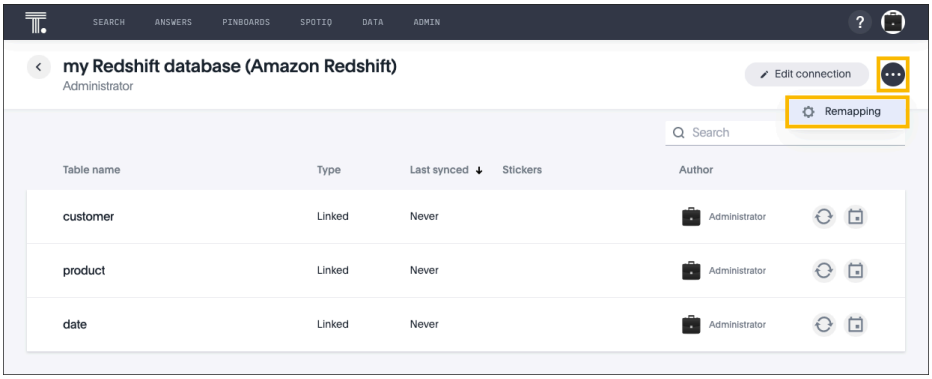
1. Click **Data** in the top navigation bar.

2. Click the **Connections** tab at the top of the page.

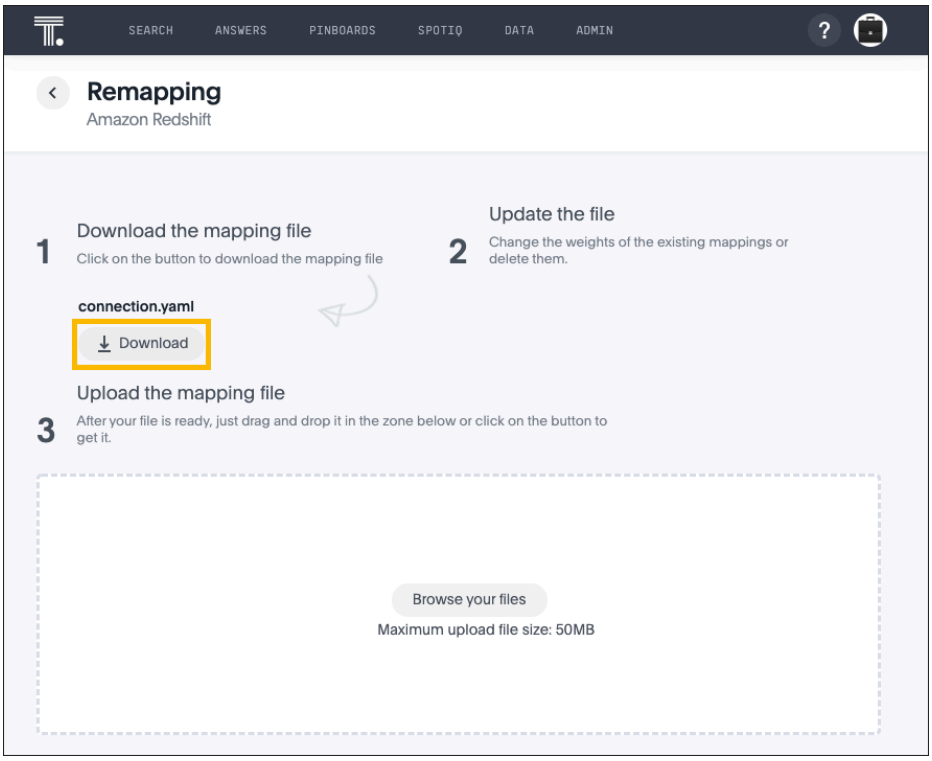
3. Click the name of the connection you want to remap.



4. Click the More Info icon ••• and select **Remapping** on the upper-right-hand side of the page.



5. Click **Download** to download the source mapping file.



6. Edit and update the file as required.

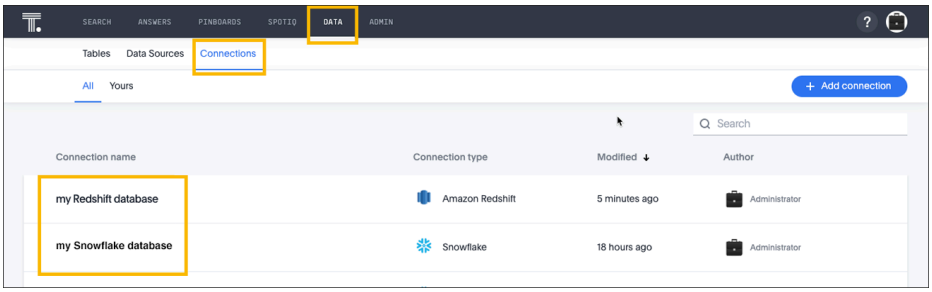
7. Finally, click **Browse your files**, and upload your edited mapping file to update the mapping of your connection.

Deleting a table

ThoughtSpot checks for dependencies whenever you try to remove a table in a connection. ThoughtSpot shows a list of dependent objects, and you can click them to delete them or remove the dependency. Then you can remove the table.

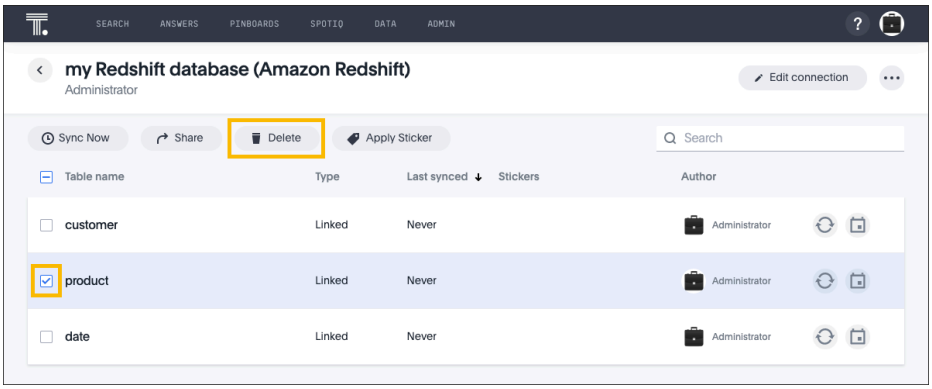
To delete a table:

1. Click **Data** in the top navigation bar.
2. Click the **Connections** tab at the top of the page.
3. Click the name of the connection that contains the table you want to delete.



4. Find the table you want to delete in the list, and check the box next to its name.

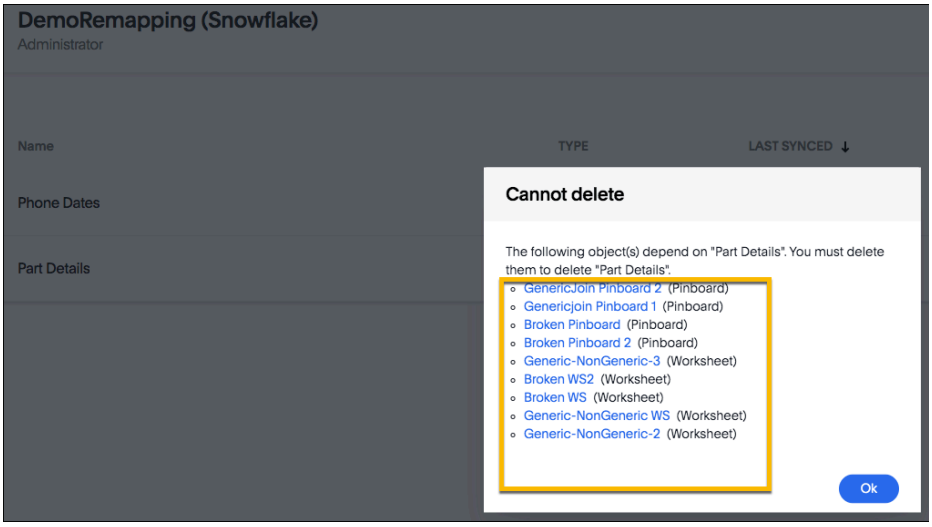
5. Click **Delete**.



> **Note:** If you attempt to delete a table with dependent objects, the operation is blocked. A warning appears, with a list of links to dependent objects.

6. Click the link for each object to modify or delete it.

   When all dependencies are removed, you can delete the table.



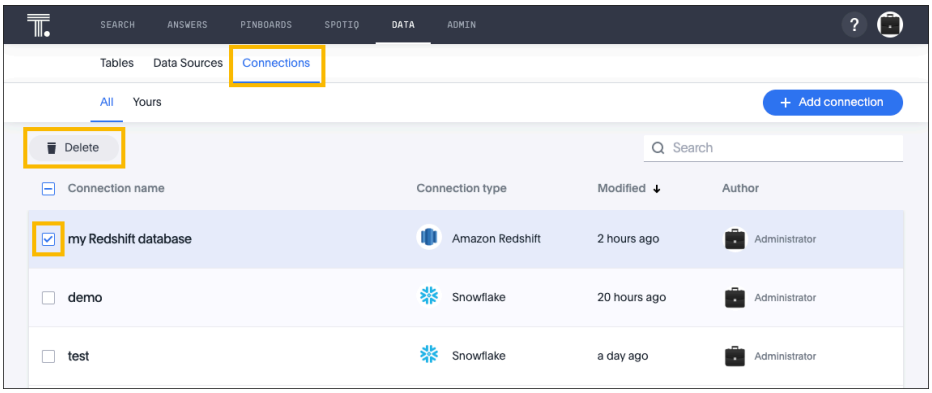
You can also click the name of a table and then click the linked objects to see a list of dependent objects with links. The list shows the names of the dependent objects (worksheets, pinboards or answers), and the columns they use from that table. You can use this information to determine the impact of changing the structure of the data source or to see how widely used it is. Click a dependent object to modify or delete it.

Deleting a connection

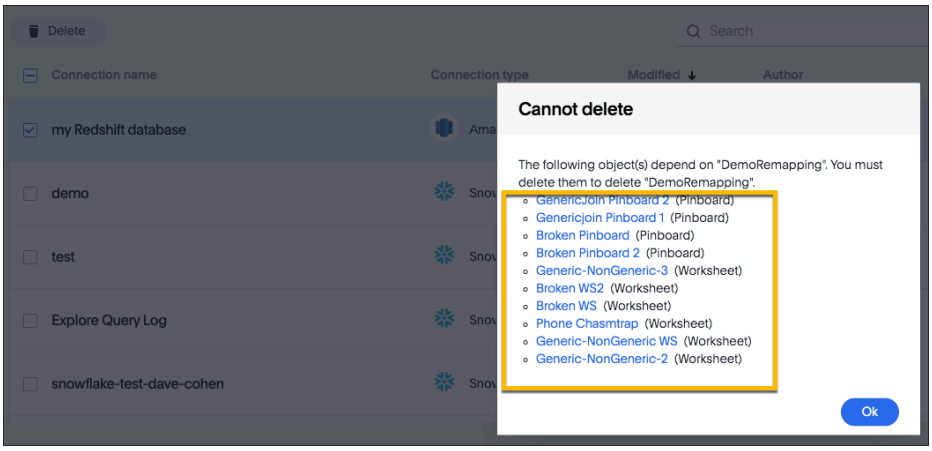
A connection can be used in multiple data sources or visualizations. Because of this, you must delete all of the sources and tasks that use that connection, before you can delete the connection.

To delete a connection:

1. Click **Data** in the top navigation bar.
2. Click the **Connections** tab at the top of the page.
3. Check the box next to the connection you want to delete.
4. Click **Delete**.



If you attempt to delete a connection with dependent objects, the operation is blocked, and a "Cannot delete" warning appears with a list of dependent objects with links.



5. If the "Cannot delete" warning appears, click the link for each object to delete it, and then click **Ok**. Otherwise, go to the next step.

6. When all its dependencies are removed, delete the connection by clicking **Delete**.

Sync Tables in Embrace

When you create a connection to Snowflake, the selected tables and columns in your connection are linked to the Snowflake database. With linked data, ThoughtSpot queries the Snowflake database directly, which is convenient because you don't have to import the data into ThoughtSpot, but search is slower and you don't have access to all of ThoughtSpot's features. With Sync, you get the search performance and all the features available of ThoughtSpot.

How sync works

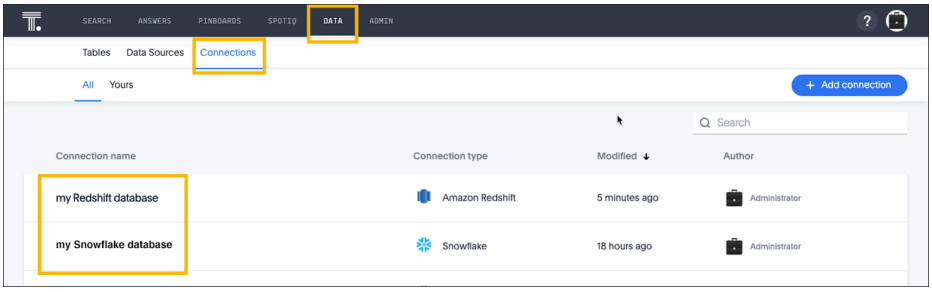
Sync copies selected tables or columns into ThoughtSpot. By syncing your tables and columns, your data is indexed, which improves search speed. You can sync manually at any time, and also schedule your sync.

Syncing manually

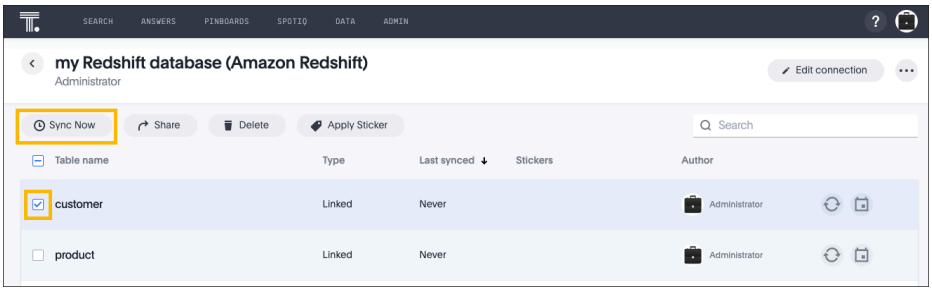
You can manually sync one or more tables in a connection.

To sync manually:

1. Click **Data** in the top navigation bar.
2. Click the **Connections** tab at the top of the page.
3. Click the name of your connection.



4. Check the box next one or more tables that you want to sync, and click **Sync Now**.



The status of the sync appears in the *Last synced* column and refreshes to indicate when the sync was completed.

Scheduling sync

To schedule sync:

1. Click **Data** in the top navigation bar.
2. Click the **Connections** tab at the top of the page.
3. Click the name of your connection that contains the table you want to sync.
4. Click the name of the table you want to sync.
5. Click the **Sync** tab.
6. Click **Schedule**.
7. In the *Sync table schedule* window, use the **Select interval** drop-down menu to set the following items:

   A. How often you want to sync (**Hourly**, **Daily**, **Weekly**, or **Monthly**).

   B. Select a sync mode:

   - **Append** adds to the current data.
   - **Overwrite** replaces the current data.

8. (Optional) Enter a filter condition to be added to the search query.
9. (Optional) If you want to shard the data as it is synced, click **Show advanced settings** and enter the **Primary key**, **Sharding key** and **Number of shards**.

10. Click **Schedule**

    The schedule is saved and runs automatically at the selected time.

    Every time the schedule is run, it is listed in the History list.

Updating or removing a schedule

To update a schedule:

1. From the sync tab of the table you want to update, click **Schedule**.
2. In the *Sync table schedule* window, make your changes and click **Update schedule**.

To remove a schedule:

- From the sync tab of the table you want to update, click the More Info icon and select **Delete schedule**.

Related information

- Modify a connection
- Connectors reference
- Load and manage data
- Data and object security

Connectors reference

Summary: Learn about the external database connectors necessary for ThoughtSpot Embrace.

Here is a list of all of the external database connectors, and their connection credentials, that you must save in ThoughtSpot Embrace. As you create a new external connection, you need specific connection information to establish seamless and secure connection with that database.

Snowflake

- **Connection name**: Mandatory. Enter a new Snowflake connection name.
- **Description**: Optional. Provide a short description about the connection.
- **Account Name**: Mandatory. Enter the account name associated with the Snowflake connection.
- **User**: Mandatory. Enter the Snowflake account username.
- **Password**: Mandatory. Enter the Snowflake account password.
- **Role**: Mandatory. Specify the privilege of the user.
- **Warehouse**: Mandatory. Specify the warehouse associated with the connection.
- **Database**: Optional. Specify the database associated with the account.
- **Schema**: Optional. Specify the schema associated with the database.

Related articles

- Add a connection
- Modify a connection

JDBC and ODBC setup prerequisites

Before you can use JDBC or ODBC to import data into ThoughtSpot, you must do the following server-side configuration:

1. Open up the ThoughtSpot firewall to allow incoming requests to Simba server.

   ```
   tscli firewall open-ports --ports 12345
   ```

2. Confirm that the `simba_server` process is up. Output of the command below should contain exactly one line, as shown below.

   ```
   ps -ef | grep simba_server | grep -v grep
   admin    26679 25672  0 Jul13 ?        00:01:49 simba_se
   rver_main --logbufsecs=0
   ```

   For assistance, contact ThoughtSpot Support.

Overview of the ODBC Driver

Summary: Use the ODBC driver to bring data in from your ETL tool or database.

ThoughtSpot comes packaged with an ODBC (Open Database Connectivity) driver, so that you can transfer data between ThoughtSpot and other databases. Basic knowledge of ODBC data source administration is helpful when setting up ODBC.

Supported operating systems for the ODBC driver are:

- Microsoft Windows 32-bit
- Microsoft Windows 64-bit
- Linux 32-bit
- Linux 64-bit

Version compatibility and connection parameters

To ensure compatibility, always use the ODBC driver with the same version number as the ThoughtSpot instance to which you are connecting. You can make a secure ODBC connection to the ThoughtSpot database by configuring a user and password combination with the driver. For detailed information about connection parameters, see the ODBC and JDBC configuration properties

Supported Data Types

The ODBC driver supports these data types:

- `INT`
- `BIGINT`
- `BOOLEAN`
- `DOUBLE`
- `FLOAT`
- `DATE`
- `TIME`
- `TIMESTAMP`
- `DATETIME`

- `CHAR`
- `VARCHAR`

Source and target data compatibility

By default, ThoughtSpot takes a permissive approach to data type compatibility between source and target data in ODBC. In this mode, ThoughtSpot *assumes* that the incoming data matches exactly with the target data types and loads the table as is.

Alternatively, you can explicitly require that ThoughtSpot match the source data types exactly and, if it can't find a match, it returns an error and the data load fails. In this mode, for example, if the target ThoughtSpot data type for a column is INT, the source data type for that column must be INT in order for the data load to succeed.

By toggling ***strict*** and ***permissive*** `true` and `false` options, you can configure settings along a scale of behavior between the permissive, automatic approach and the strictness of the "must match" approach.

		Strictness	
		true	false
Permissiveness	true	Data types are inferred and automatically converted. ThoughtSpot returns an error in cases where the data conversion is not possible. Data load fails in its entirety if any data contains mismatches. You must correct the problem in the source data and try the load again.	Data types are inferred and automatically converted. No error is thrown even if source and target data types don't match. Data load continues even when the source and target data types don't match. This means your data load may contain data types that you do not intend or that are not helpful. You are responsible for checking and validating the data in this case.

false	The source and target data types must match. If any data contains mismatches, ThoughtSpot returns an error to the client a data load fails in its entirety. You must correct the problem in the source data and try the load again. This is the strictest configuration.	No data types are inferred and conversion does not check for matches. This is the most permissive configuration.

Your customer support engineer can assist you in configuring custom ODBC behavior. Regardless of the configuration you choose, you must validate that the results of data loading *as they appear* in ThoughtSpot are what you require.

Data type conversion matrix

The following table describes the conversion matrix between SQL data types and ThoughtSpot data types.

Source SQL Data Types	BOOL	INT32	INT64	DOUBLE	FLOAT	CHAR	DATE	TIME	DATETIME
SQL_BIT	Y	Y	Y	Y	Y	Y	–	–	–
SQL_TINYINT	Y	Y	Y	Y	Y	Y	–	–	–
SQL_SMALLINT	Y	Y	Y	Y	Y	Y	–	–	–
SQL_INTEGER	Y	Y	Y	Y	Y	Y	–	–	–
SQL_BIGINT	Y	Y	Y	Y	Y	Y	–	–	–
SQL_CHAR	Y	Y	Y	Y	Y	Y	Y	Y	Y
SQL_VARCHAR	Y	Y	Y	Y	Y	Y	Y	Y	Y
SQL_LONGVARCHAR	Y	Y	Y	Y	Y	Y	Y	Y	Y
SQL_BINARY	–	–	–	–	–	Y	–	–	–
SQL_VARBINARY	–	–	–	–	–	Y	–	–	–
SQL_LONGVARBINARY	–	–	–	–	–	Y	–	–	–

Source SQL Data Types	BOOL	INT32	INT64	DOUBLE	FLOAT	CHAR	DATE	TIME	DATETIME
SQL_DOUBLE	Y	Y	Y	Y	Y	Y	–	–	–
SQL_REAL	Y	Y	Y	Y	Y	Y	–	–	–
SQL_FLOAT	Y	Y	Y	Y	Y	Y	–	–	–
SQL_NUMERIC	Y	Y	Y	Y	Y	Y	–	–	–
SQL_GUID	–	–	–	–	–	Y	–	–	–
SQL_INTERVAL_MINUTE_TO_SECOND	–	–	–	–	–	Y	–	–	–
SQL_INTERVAL_HOUR_TO_SECOND	–	–	–	–	–	Y	–	–	–
SQL_INTERVAL_HOUR_TO_MINUTE	–	–	–	–	–	Y	–	–	–
SQL_INTERVAL_DAY_TO_SECOND	–	–	–	–	–	Y	–	–	–
SQL_INTERVAL_DAY_TO_MINUTE	–	–	–	–	–	Y	–	–	–
SQL_INTERVAL_DAY_TO_HOUR	–	–	–	–	–	Y	–	–	–
SQL_INTERVAL_YEAR	–	Y	Y	–	–	Y	–	–	–
SQL_INTERVAL_MONTH	–	Y	Y	–	–	Y	–	–	–
SQL_INTERVAL_DAY	–	Y	Y	–	–	Y	–	–	–
SQL_INTERVAL_HOUR	–	Y	Y	–	–	Y	–	–	–
SQL_INTERVAL_MINUTE	–	Y	Y	–	–	Y	–	–	–
SQL_INTERVAL_SECOND	–	Y	Y	–	–	Y	–	–	–
SQL_TYPE_TIME	–	–	–	–	–	Y	–	Y	Y
SQL_TYPE_DATE	–	–	–	–	–	Y	Y	–	Y
SQL_TYPE_TIMESTAMP	–	–	–	–	–	Y	Y	Y	Y

If a conversion is not possible, an error is returned to the client to indicate conversion failure. The ETL tool must add a data transformation step if the source column data type does not exactly match the target's ThoughtSpot column data type. The driver does not do any implicit conversions.

Install the ODBC driver on Windows

Summary: Use this procedure to obtain the Microsoft Windows ODBC driver and install it.

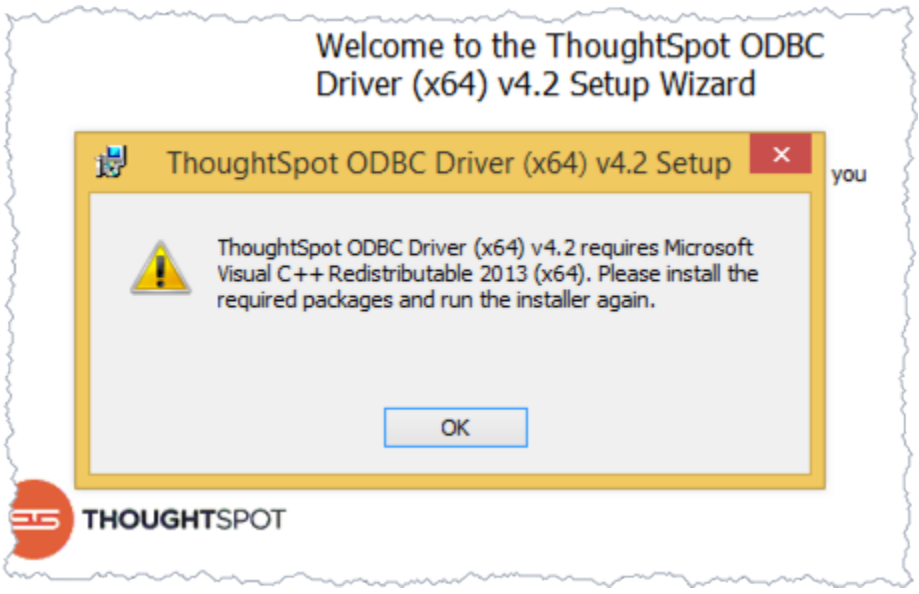
ThoughtSpot's ODBC connection relies on the SimbaEngine X SDK to connect through ODBC or JDBC to ThoughtSpot's remote data stores. The instructions on this page explain how to configure the Simba ODBC driver on a Windows workstation.

Make sure you have read the overview material in the ODBC driver overview. This workstation is the same machine where you plan to run your ETL activities.

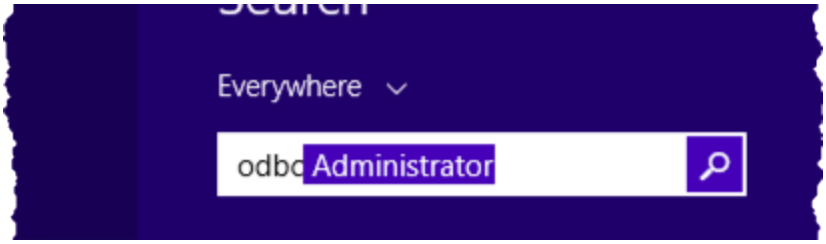
Prerequisites

These instructions include directions to use the `ssh` command. Make sure your Windows workstation is equipped with a tool such as Putty for making `ssh` connections to your ThoughtSpot server.

The ODBC driver for Windows requires Visual C++ Redistributable for Visual Studio 2013. You are prompted to install it during installation of the driver if it isn't already installed.



To check if this Microsoft tool is already installed, search for it on your workstation.



If it isn't installed, make sure you download and install it before continuing.

Check the ThoughtSpot IP and the simba_server status

Before you begin, you need to know the IP address or DNS name of the server you intend to connect your server to.

1. SSH as `admin` or the `thoughtspot` user to your ThoughtSpot node.
2. Verify the node IP(s).

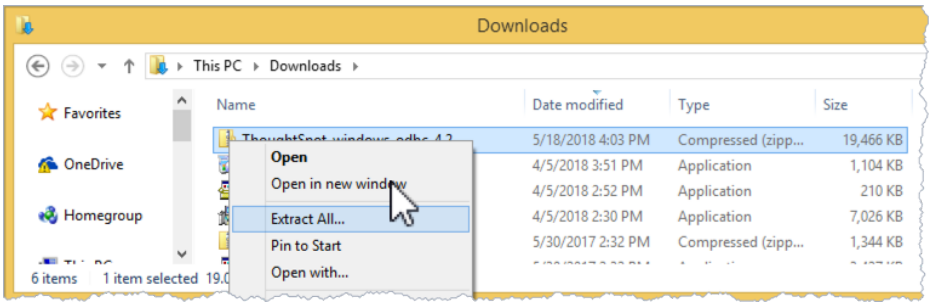
```
$ tscli node ls
172.18.231.17
172.18.231.18
```

3. Make a note of each IP; there may be more than one.
4. Configure the ThoughtSpot firewall to allow connections from your ETL client, by running the following command on any ThoughtSpot node: `tscli firewall open-ports --ports 12345`
5. Exit or close the shell.

Download the driver

On the workstation where you want to connect from, do the following:

1. Navigate to the **Downloads** page.
2. Download the **ODBC Driver for Windows**.
3. Unzip the file you downloaded at a convenient location on your workstation.



4. Take a moment to examine the contents of the new directory.

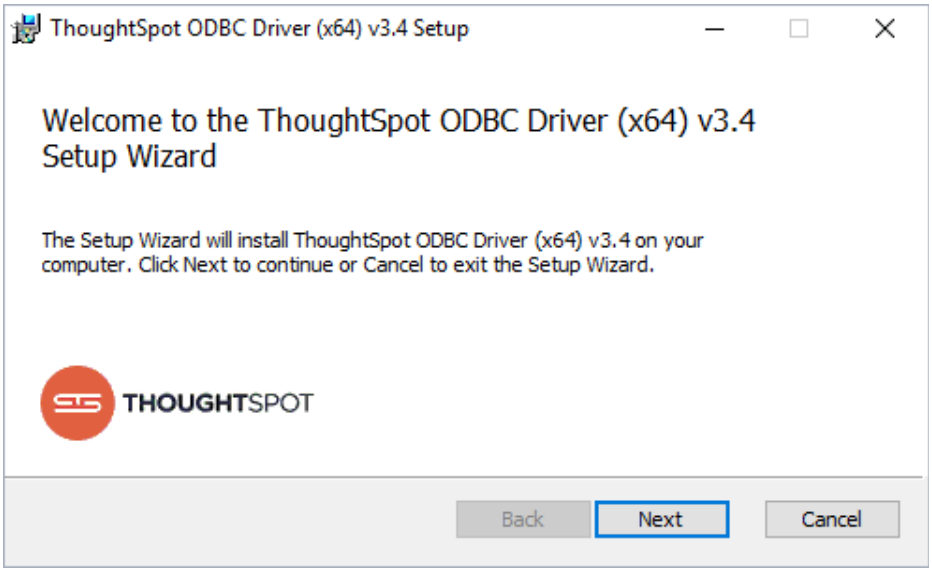
   There are two different Windows ODBC installers included in the file you downloaded.

   - ThoughtSpotODBC (x86).msi for Windows 32-bit
   - ThoughtSpotODBC (x64).msi for Windows 64-bit

Install the driver and supporting software

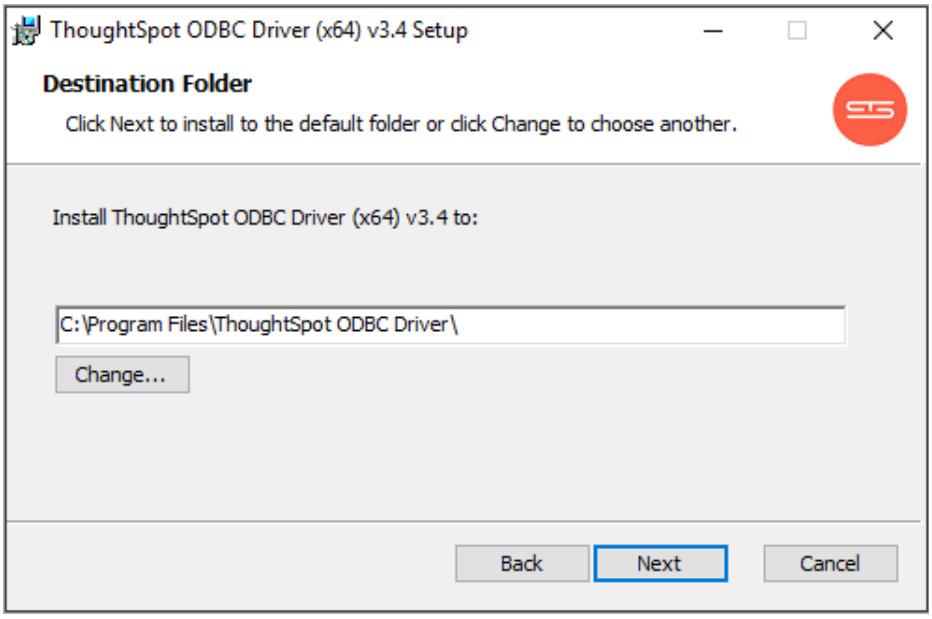
The installation process installs the Simba driver and adds the ODBC Administrator software to your workstation. You use this software to configure the driver.

1. Launch the installer for your version of Windows.
2. Click **Next** to continue.

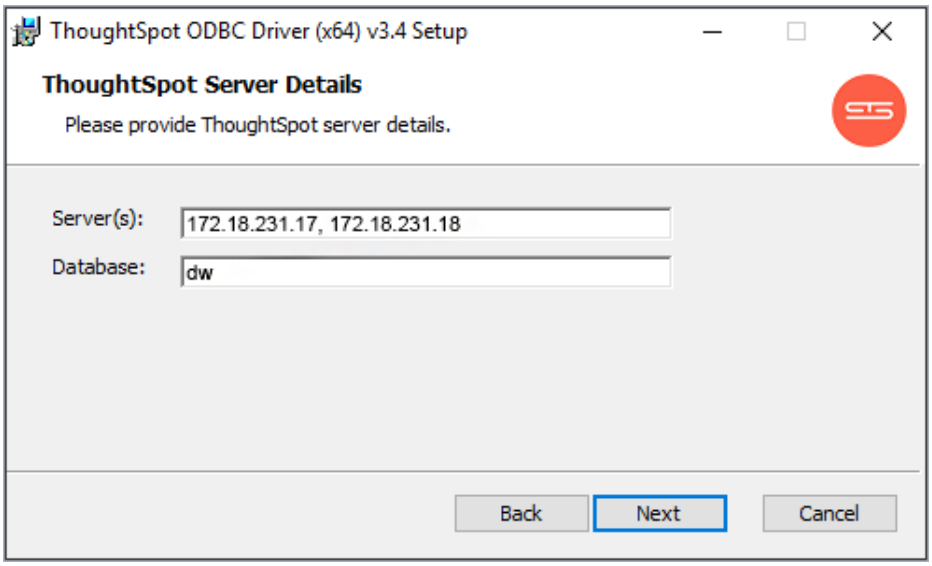


3. Accept the End User License Agreement (EULA), and click **Next**.

4. Specify the destination folder where the driver will be installed.



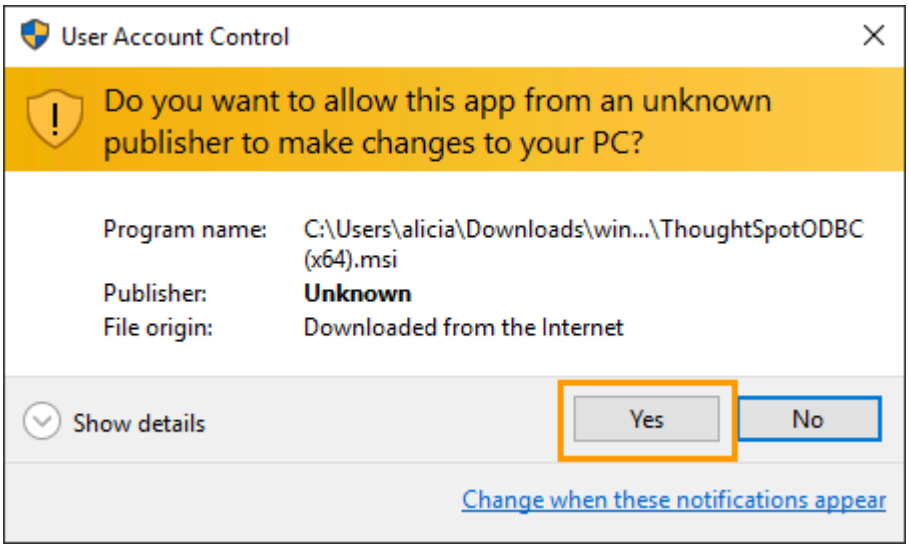
5. Enter the ThoughtSpot server details, and click **Next**.



- For **Server(s)**, provide a comma separated list of the IP addresses of each node on the ThoughtSpot instance.
- For **Database**, optionally specify the database to use. If you skip this entry, you must provide the database each time you connect using ODBC.

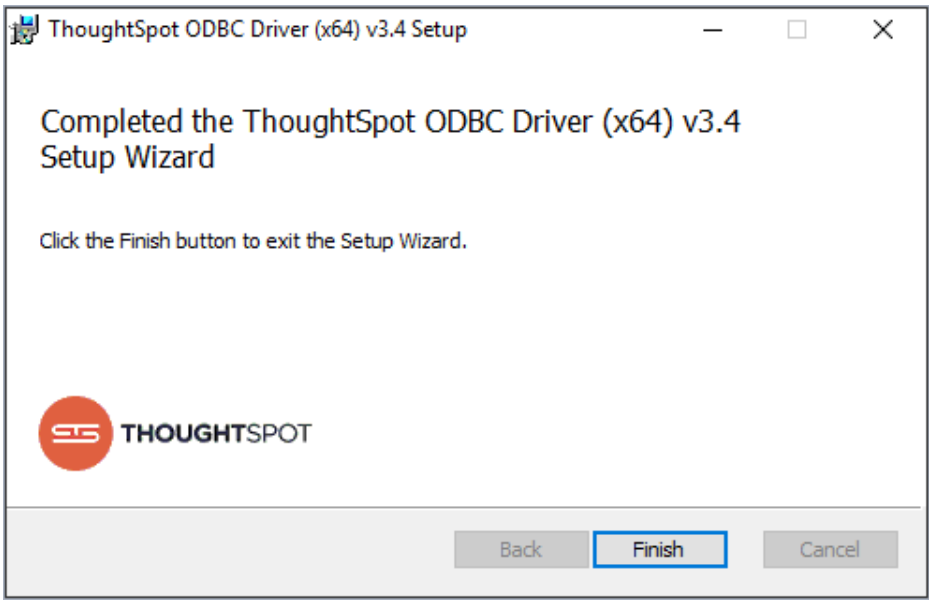
6. Confirm that the install can begin by clicking **Install**.

7. You may see a security warning.



8. Click **Yes** to continue.

   A confirmation message appears when the installation is complete.



9. Click **Finish**.

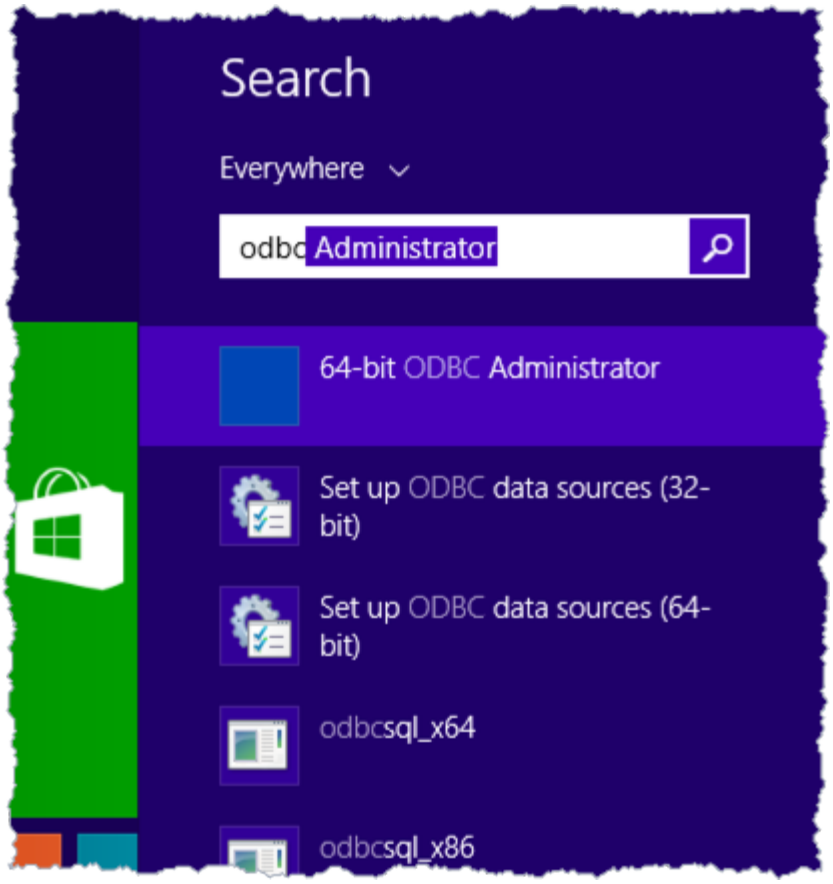
Configure the driver and test your connection

After installation completes, use the ODBC Administrator to configure the ODBC connection on your Windows workstation. For example, you may want to add a default schema or change the server IP address or the default database.

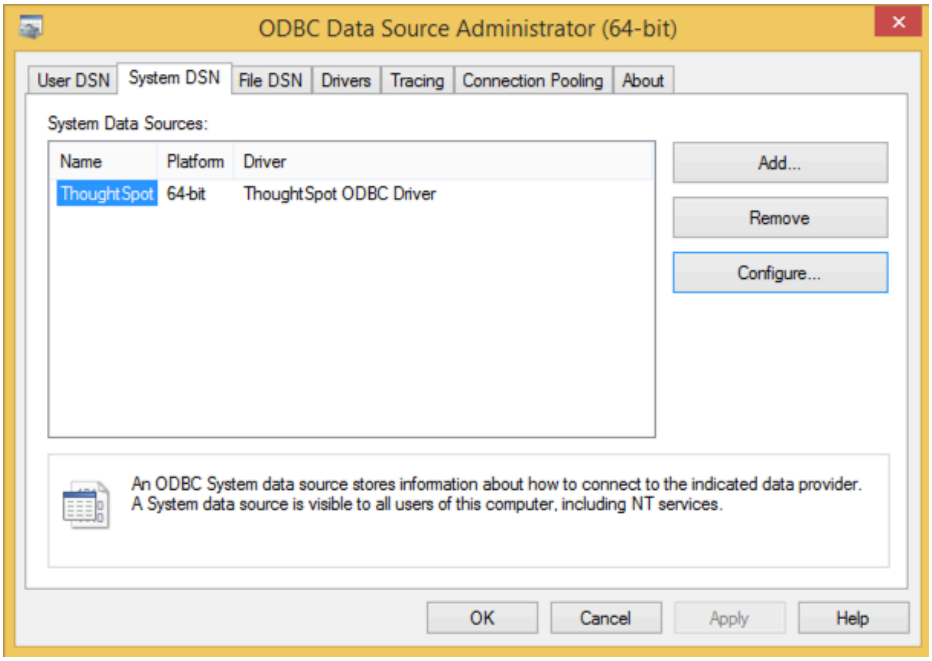
It is recommended to add a default schema. If you don't specify a default schema, you must supply it every time you use the ODBC driver.

At this point, you can test your ODBC connection to ThoughtSpot. It is important to recall that the username/password you use belongs to a ThoughtSpot application user. Typically, this user is a user with data management or administrative privileges on the application.

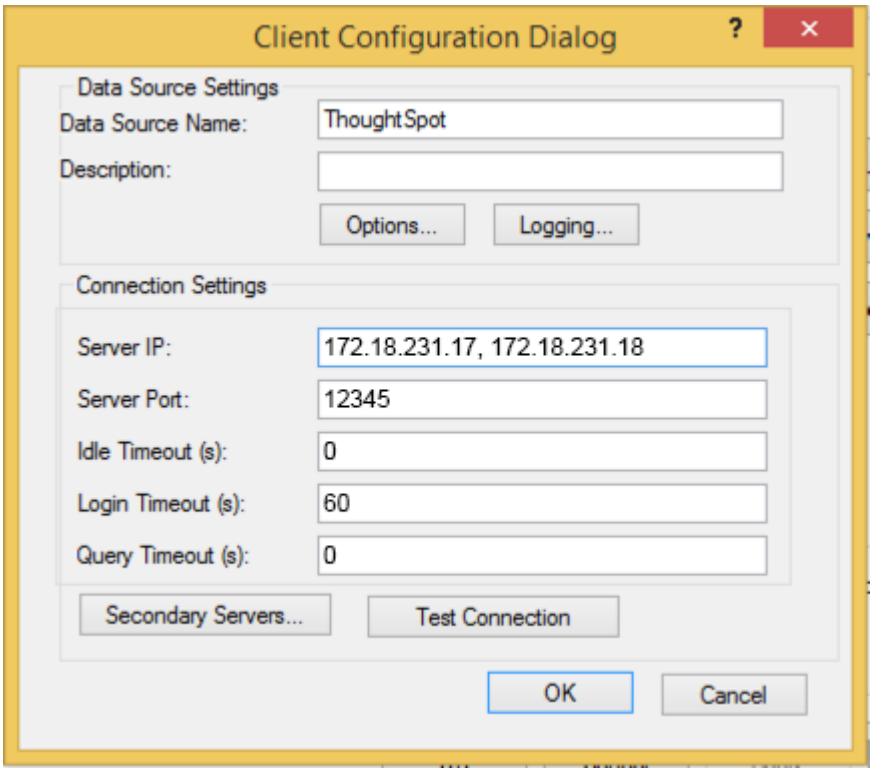
1. Before trying the ODBC connection, confirm a username/password that can log into the ThoughtSpot applications.
2. Click the **Data** tab, and confirm the user's privileges.
3. Return to your workstation.
4. Locate and open the **ODBC Data Source Administrator (64-bit)** application.



5. Click the **System DSN** tab.



6. Select **ThoughtSpot** and click **Configure…**



7. Click **Options…**

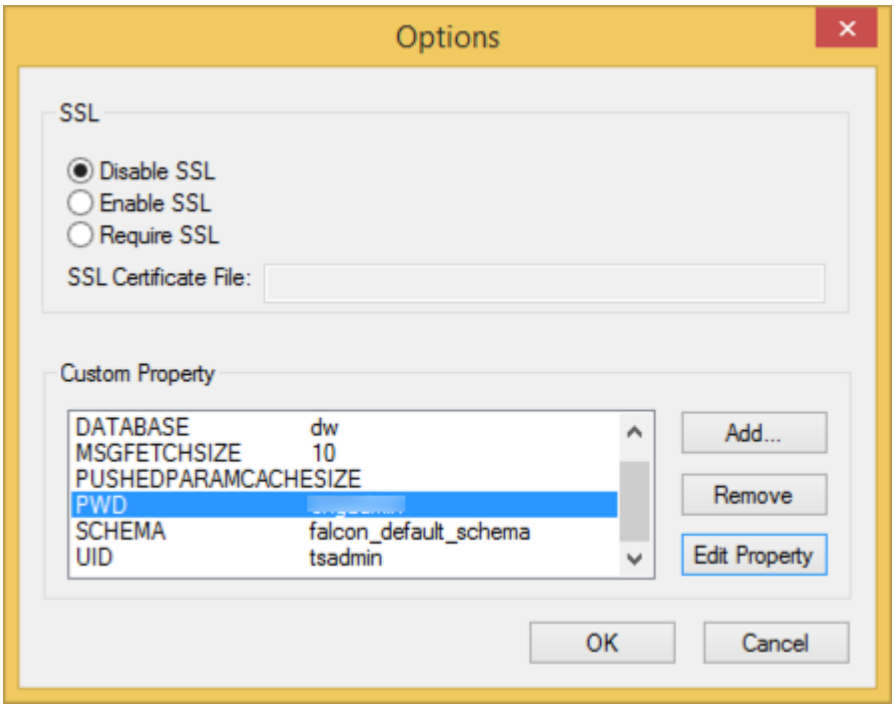
8. Ensure you have the following **Custom Property** values set:

Custom Property	Value
`SCHEMA`	`falcon_default_schema` is the default
`UID`	The username of a user with data management privilege.
`PWD`	The password for the username you specify.

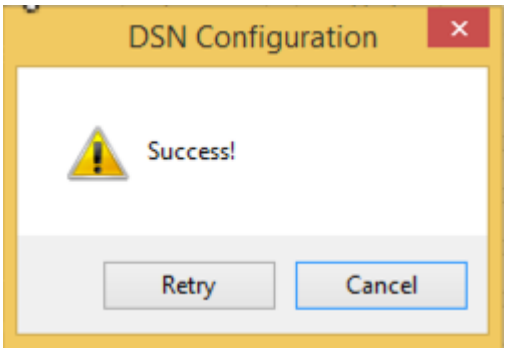
You don't have to use the ThoughtSpot default schema. You can specify your own. We recommend that you define a default schema. Otherwise, you must supply a schema every time you use the ODBC driver. Moreover, without a schema (or if the schema is not present), the ODBC driver returns an error that states that the schema could not be found.

Similarly, adding the `UID` and `PWD` properties are not required. If you don't add them, you are prompted to supply them each time you connect.

When you are done, your options should look similar to the following:



9. When you are done, click **OK** to save your new properties.
10. Click **Test Connection** to test your database connection.



11. Click **Cancel** to close the **DSN Configuration** dialog.
12. Click **OK** to close the **Client Configuration Dialog** the dialog.
13. Click **OK** to close the **ODBC Data Source Administrator (64-bit)** application.

Now, you are ready to begin using the connection you've configured.

Related information

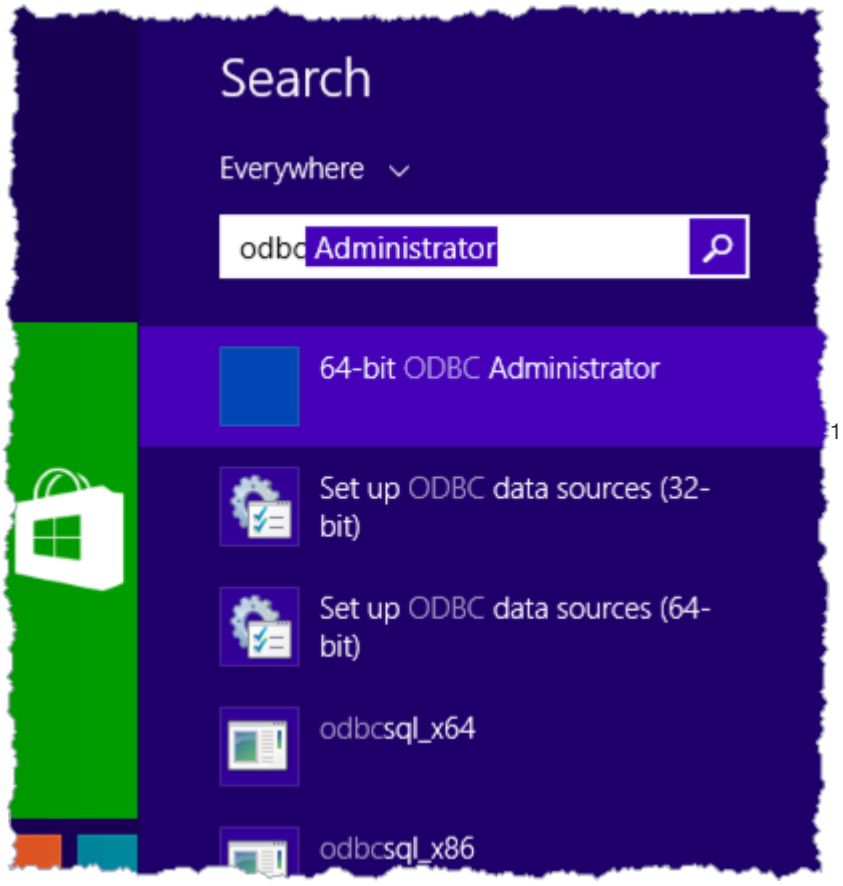
- Enable ODBC logs.
- Configure multiple connections on Windows.

Configure multiple connections on Windows

Summary: You can add multiple ODBC data sources.

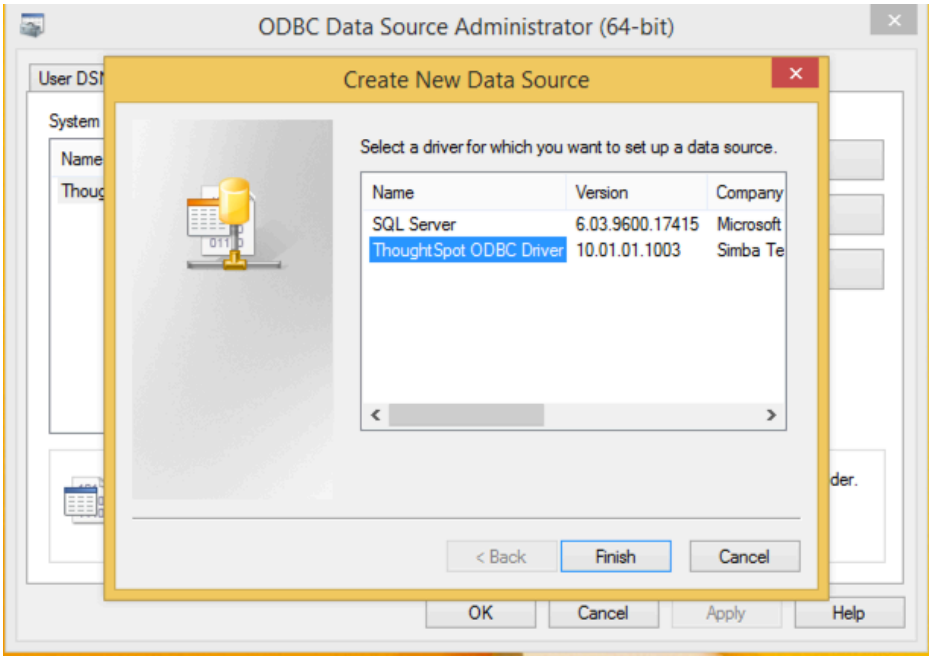
Use this procedure if you want to add an additional data source after creating a single source succeeds. The main reason for needing to set up multiple ThoughtSpot ODBC data sources is that you have a production cluster and a test or development cluster.

1. Locate and open the **ODBC Data Source Administrator (64-bit)** application.



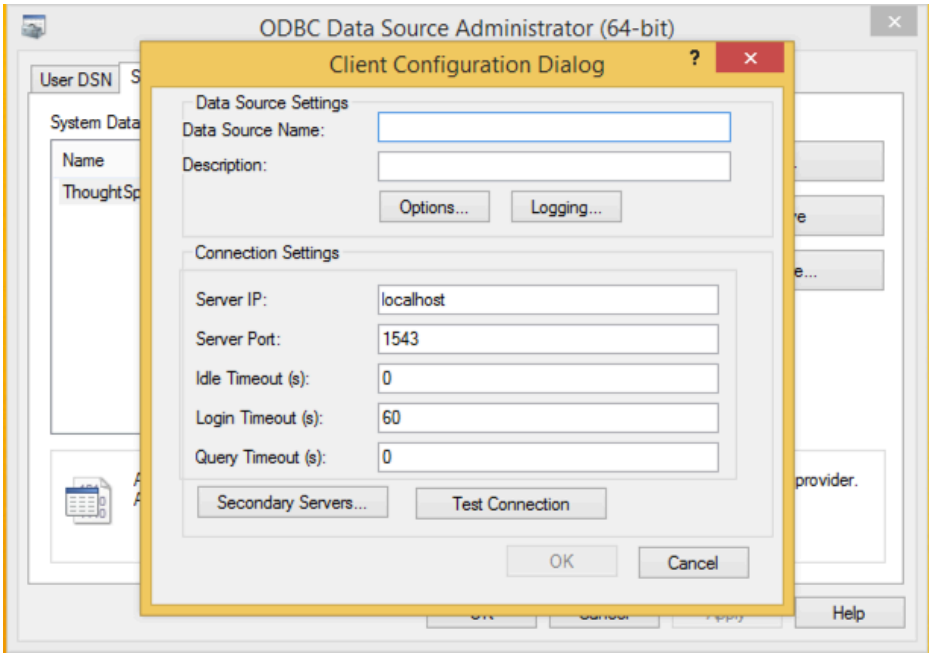
2. Click the **System DSN** tab.
3. Select **Add**.

   The system lists the available drivers.



4. Choose the **ThoughtSpot ODBC Driver** and click **Finish**.

   The system displays the **Client Configuration Dialog** dialog.

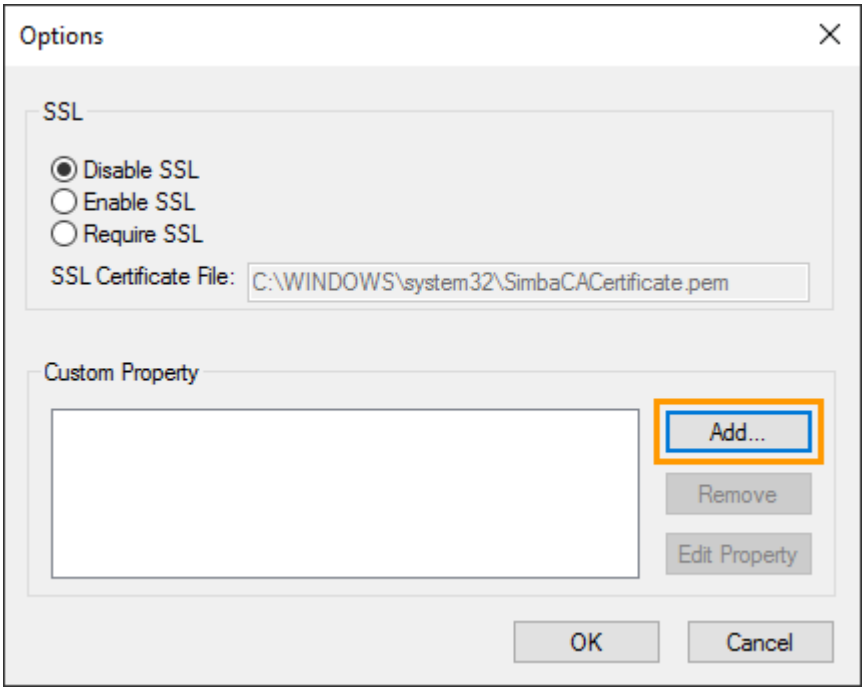


5. Enter the your data source configuration.

Configuration Property	Value
Data Source Name	The name you want to call the data source.
Description	A description of the data source.
Server IP	A list of the IP addresses for each node, separated by commas.
Server Port	12345
Idle Timeout	Time in seconds after which an idle ODBC connection times out.
Login Timeout	Time in seconds after which a login request times out.
Query Timeout	Time in seconds after which a query times out.

6. Configure custom properties by clicking **Options**.

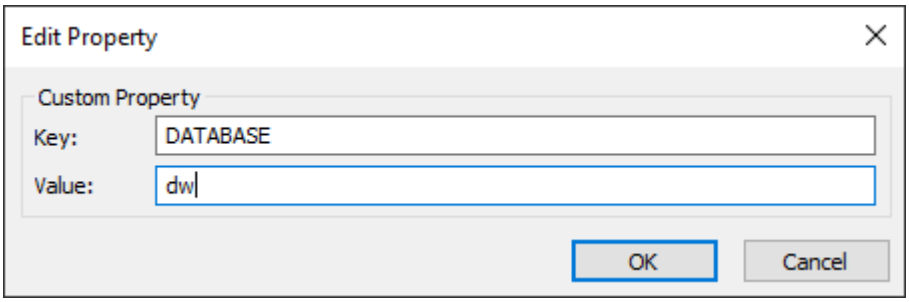
   The system displays the **Options** dialog.



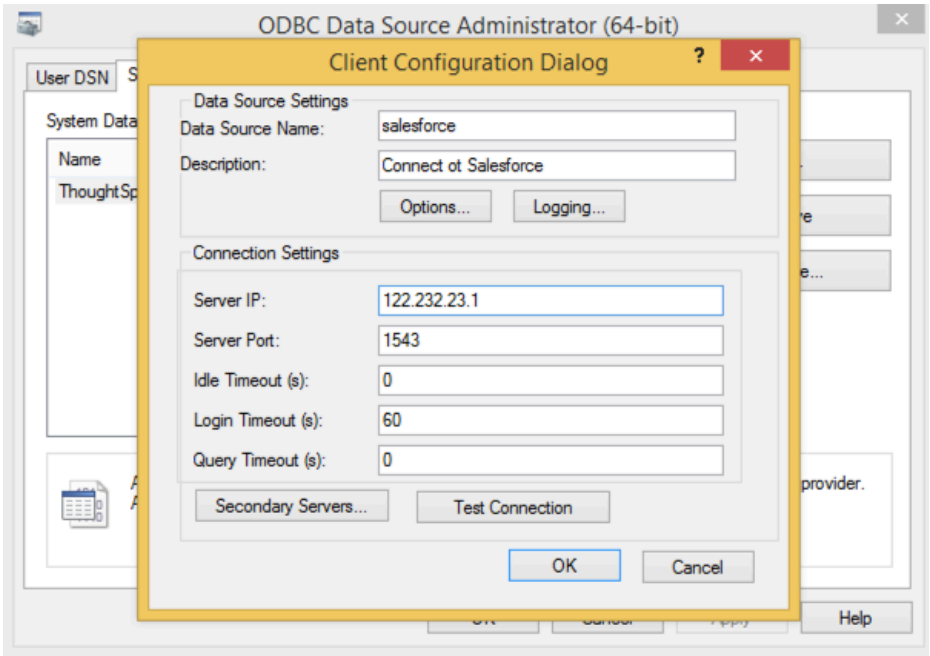
7. Add these properties using the **Add** to enter an option and click **OK** after to save an option.

Option	Value
DATABASE	The default database to connect to.
SCHEMA	The default schema to connect to. Use `falcon_default_schema` if you aren't sure.
CONNECTIONTIMEOUT	Optional. Seconds before an idle connection times out.

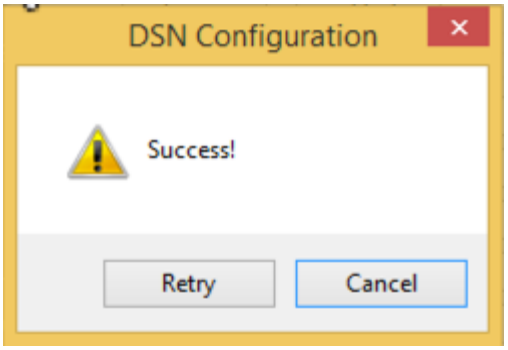
The key must be defined exactly as it appear here, using all capital letters. You can find other supported properties in ODBC and JDBC configuration properties.



8. When you are done, click **OK** to save your new configuration.



9. Click **Test Connection** to test your database connection.



If your test connection fails, enable ODBC logging to troubleshoot.

10. Click **Cancel** to close the **DSN Configuration** dialog.
11. Click **OK** to close the **Client Configuration Dialog** the dialog.
12. Click **OK** to close the **ODBC Data Source Administrator (64-bit)** application

Deploy SSL with ODBC on Windows

You can configure a secure ODBC connection between your ThoughtSpot cluster and a remote Windows Machine. This article explains the SSL resources and ODBC configuration options you need to enable SSL for an ODBC connection.

Prerequisites

Before configuring SSL over the ThoughtSpot ODBC connection, make sure that your system administrator has created and configured your network's Certificate Authority. Additionally, the system administrator should have available both the proper Private Key and Server Certificate.

Configure the ThoughtSpot cluster nodes

> ⚠ **Important:** Portions of this procedure require that you work with your ThoughtSpot Customer Service or Support Engineer.

The SimbaServer Configuration Properties reference includes full details on SSL Configuration Properties.

Before you change your ODBC configuration, decide on a path where you will store the Private Key and Server Certificate, for example, you could decide to use `/home/admin/Simba_SSL/` as the path.

Then, do the following on *every ThoughtSpot node* in your cluster.

1. Create the path on the node.
2. Copy the SSL certificate and private key to this path.
3. Edit the node's `/etc/thoughtspot/simba.ini` file (Simba server configuration) with your favorite editor.
4. Add the following lines:

```
 SslCertfile=/home/admin/Simba_SSL/Server-Certificate.pe
m
 SslKeyfile=/home/admin/Simba_SSL/Private-Key.pem
 UseSsl=Required
```

5. Restart the Simba service.

   You must work with your ThoughtSpot Customer Success or Support Engineer to do this.

Deploy the certificate on your windows workstation

Please note that the SSL settings on the server and client are interdependent.

The SimbaClient for ODBC Configuration Properties reference describes how to set parameters on the client to use SSL (scroll down to useSsl section at the end). The Simba documentation also provides a chart showing configuration properties for SSL where you can see how different combinations of SSL settings on client and server will behave. For example:

- Setting both server and client to `UseSsl=Enabled` provides the ability for clients to connect with or without SSL.
- Setting both server and client to `UseSsl=Required` requires that all clients use SSL.

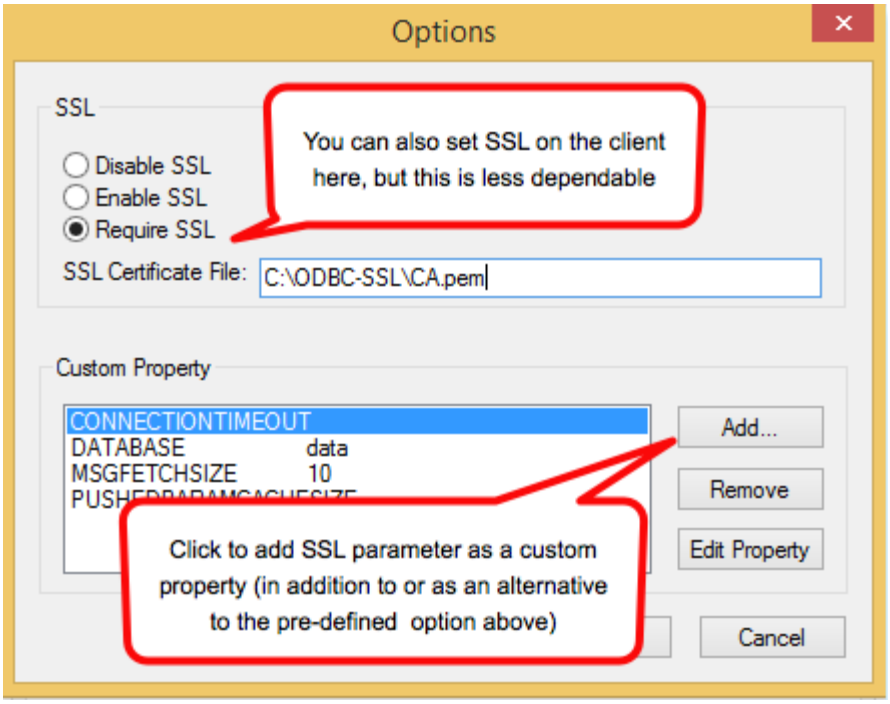
> ℹ **Note:** Note that the SSL and certificate parameters can be set through the pre-defined options on the options dialog, but customers have reported that these are not always reliable. In the following procedure, we recommend using custom properties to define these settings (either preemptively, or as a solution if the ODBC connection over SSL does not work with the pre-defined options). There is no harm in setting both. Example settings are: `UseSSL = Required` and `SslCACertfile = C:\ODBC-SSL\CA.pem`

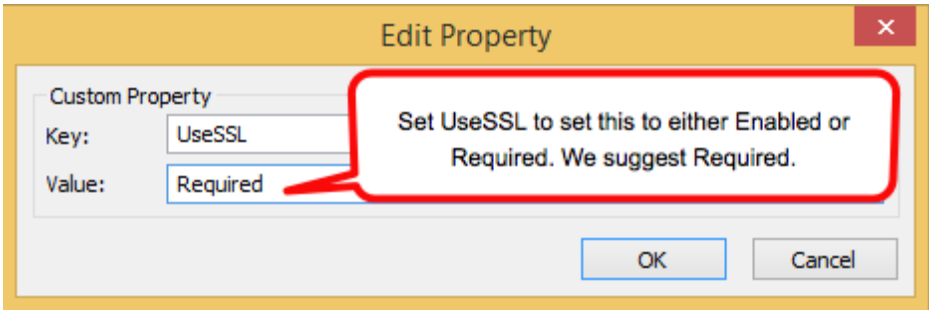
On the workstation you want to use for your ODBC connection, specify the level of SSL you want to use on the client along with the path to the CA certificate, and then test the connection.

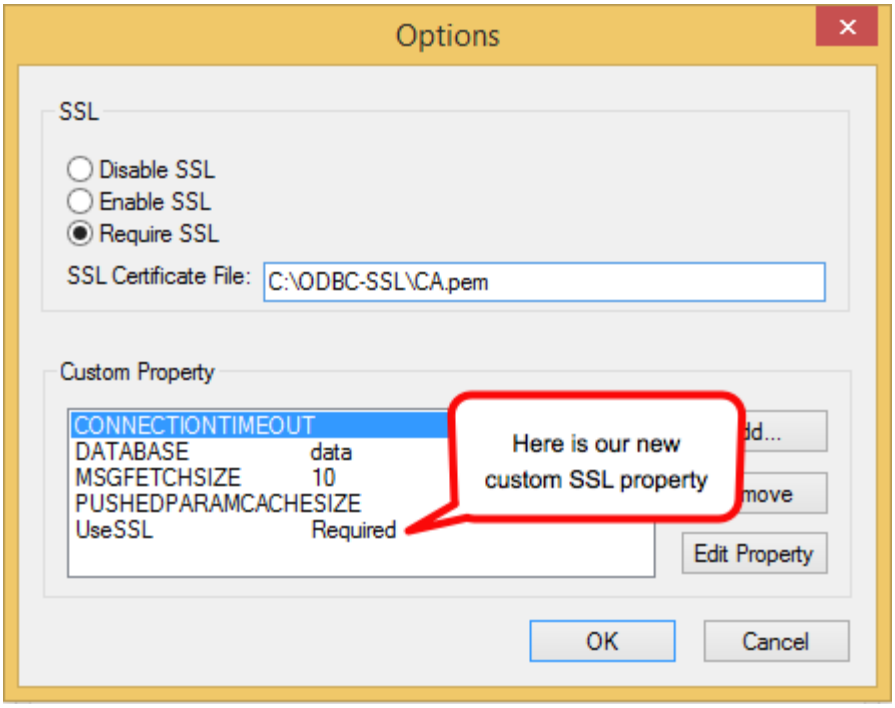
1. Save the CA certificate to a secure location on the workstation disk.

Choose a location where the certificate is unlikely to be deleted by mistake, for example, `C:\ODBC-SSL\CA.pem` is an example of a full path to such a location.

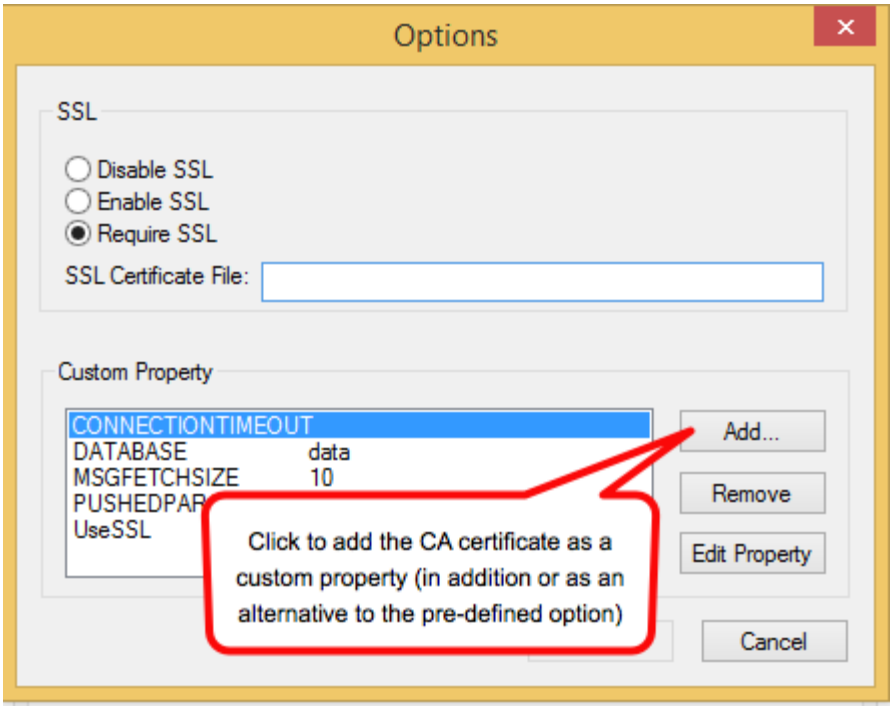
2. Open your ThoughtSpot ODBC connection configuration dialog.
3. Click **Options**.
4. Check the **Require SSL** option and/or add SSL as a custom property.

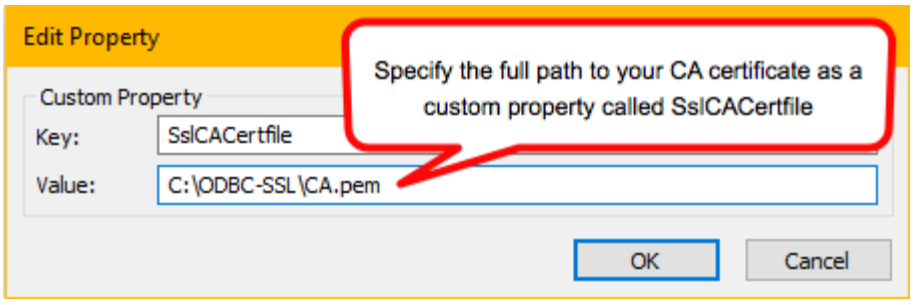

Options
SSL
Disable SSL
Enable SSL
Require SSL
You can also set SSL on the client here, but this is less dependable
SSL Certificate File:
C:\ODBC-SSL\CA.pem
Custom Property
CONNECTIONTIMEOUT
DATABASE data
MSGFETCHSIZE 10
Add...
Remove
Edit Property
Click to add SSL parameter as a custom property (in addition to or as an alternative to the pre-defined option above)
Cancel

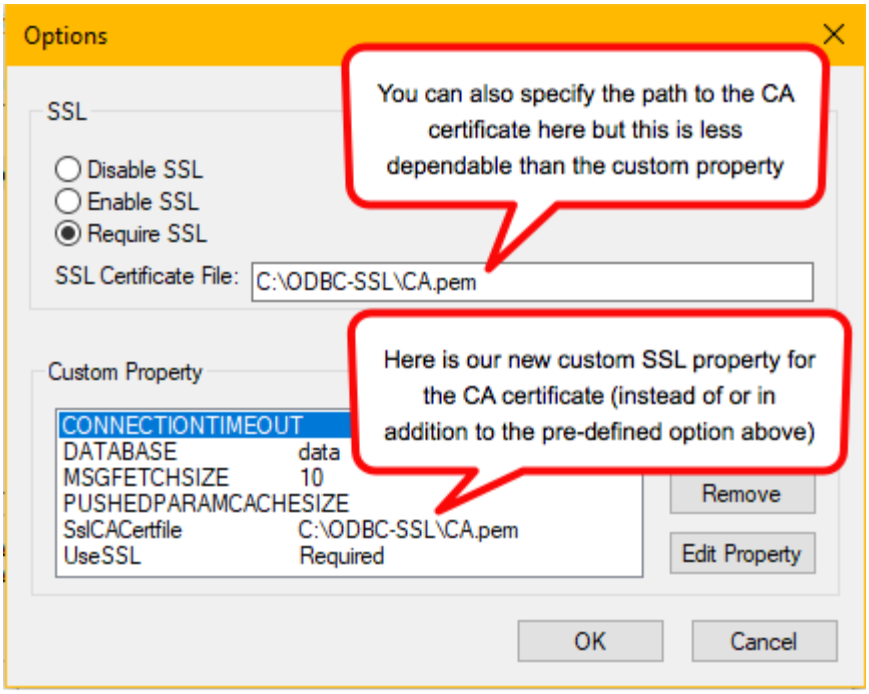

Edit Property
Custom Property
Key:
UseSSL
Value:
Required
Set UseSSL to set this to either Enabled or Required. We suggest Required.
OK
Cancel



5. Enter the location of the CA certificate in the **SSL Certificate File** field and/or add the CA certificate as a custom property. Be sure to provide the full path to the certificate (`{certificate_directory}\{CA_certificate}.pem`).

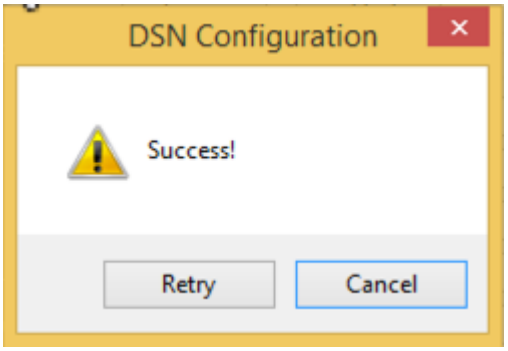

Options
SSL
Disable SSL
Enable SSL
Require SSL
SSL Certificate File:
Custom Property
CONNECTIONTIMEOUT
DATABASE data
MSGFETCHSIZE 10
PUSHEDPAR
UseSSL
Add...
Remove
Edit Property
Cancel
Click to add the CA certificate as a custom property (in addition or as an alternative to the pre-defined option)


Edit Property
Custom Property
Key:
SslCACertfile
Value:
C:\ODBC-SSL\CA.pem
OK
Cancel
Specify the full path to your CA certificate as a custom property called SslCACertfile



6. When you are done, click **OK** to save your new properties.

7. Click **Test Connection** to test your database connection.



8. Click **Cancel** to close the configuration dialog.
9. Click **OK** to close the **Client Configuration Dialog** the dialog.
10. Click **OK** to close the **ODBC Data Source Administrator (64-bit)** application.

Set up the ODBC Driver for SSIS

Summary: Use SSIS to set up the ODBC Driver.

Microsoft SSIS (SQL Server Integration Services) is a data integration and workflow applications platform you can use to connect to ThoughtSpot. The platform is a component of the Microsoft SQL Server database software.

You can use a SSIS connection to perform data migration tasks. Its data warehousing tool is useful for data ETL (extraction, transformation, and loading). The SSIS Import/Export Wizard creates packages that transfers data with no transformations. It can move data from a variety of source types to a variety of destination types, including text files and other SQL Server instances.

Use SSIS to set up the ODBC Driver by creating a connection manager. This manager connects an OLE DB Source and the ODBC Destination.

Prerequisites

On Windows 64-bit, you have to install both the 32-bit and 64-bit ThoughtSpot ODBC drivers. In addition, they must be named the same, such as ThoughtSpot. By default they are named ThoughtSpot-32 and ThoughtSpot-64. This is required because the 64-bit SSIS shows a list of 32-bit ODBC drivers when you configure an ODBC target. However, it executes the 64-bit driver. If the drivers aren't named the same, then you can get an error stating the driver doesn't exist.

Set up the driver

To set up the ODBC driver using SSIS:

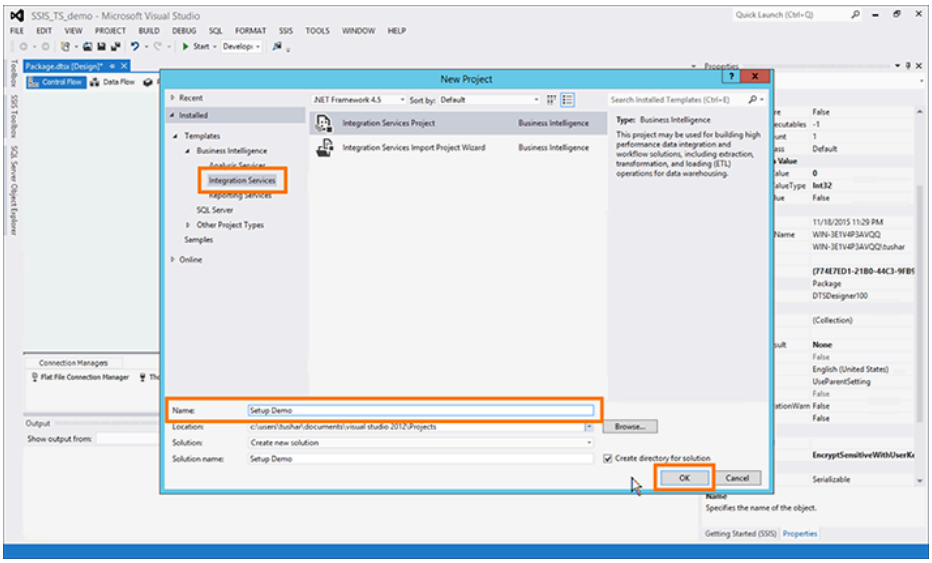
1. Open your SQL Server visual development tool that is based on Microsoft Visual Studio.
2. Select **OLE DB Source**, and click **New**.

   Where ODBC provides access only to relational databases, OLE DB provides access to data regardless of its format or location.
3. Add the server by name from the machine accessible list.

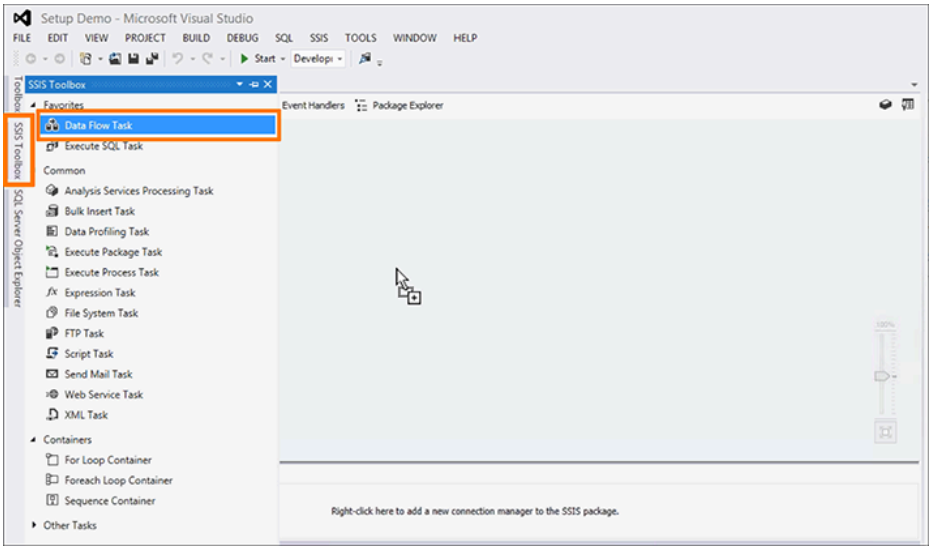
4. Enter the authentication information: db name, user name, password, and test connection.

   You can add the UID and password by clicking on **Options**.

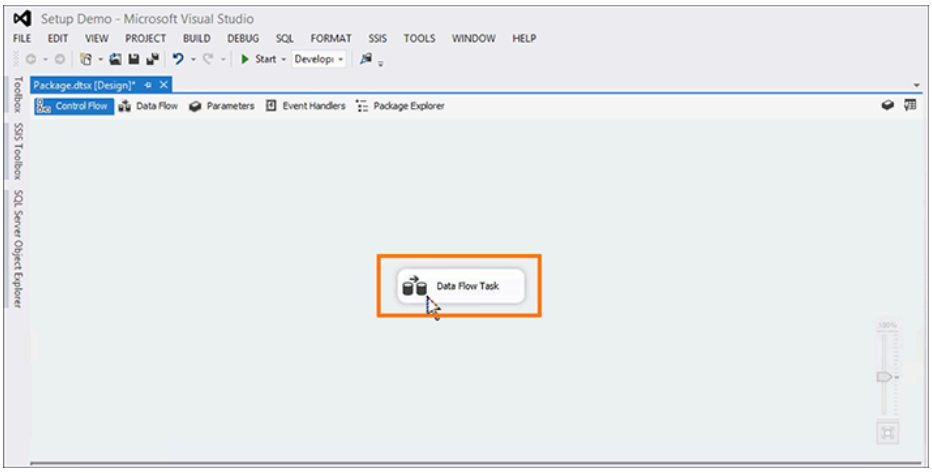
5. Click **File** and select **New**, then **Project**.
6. Select the **Integration Services** tab under **Installed > Templates > Business Intelligence**.
7. Enter a name in the **Name** field and click **OK**.



8. Select the **SSIS Toolbox** tab on the left hand side of the platform, and drag and drop **Data Flow Task** to the main window.



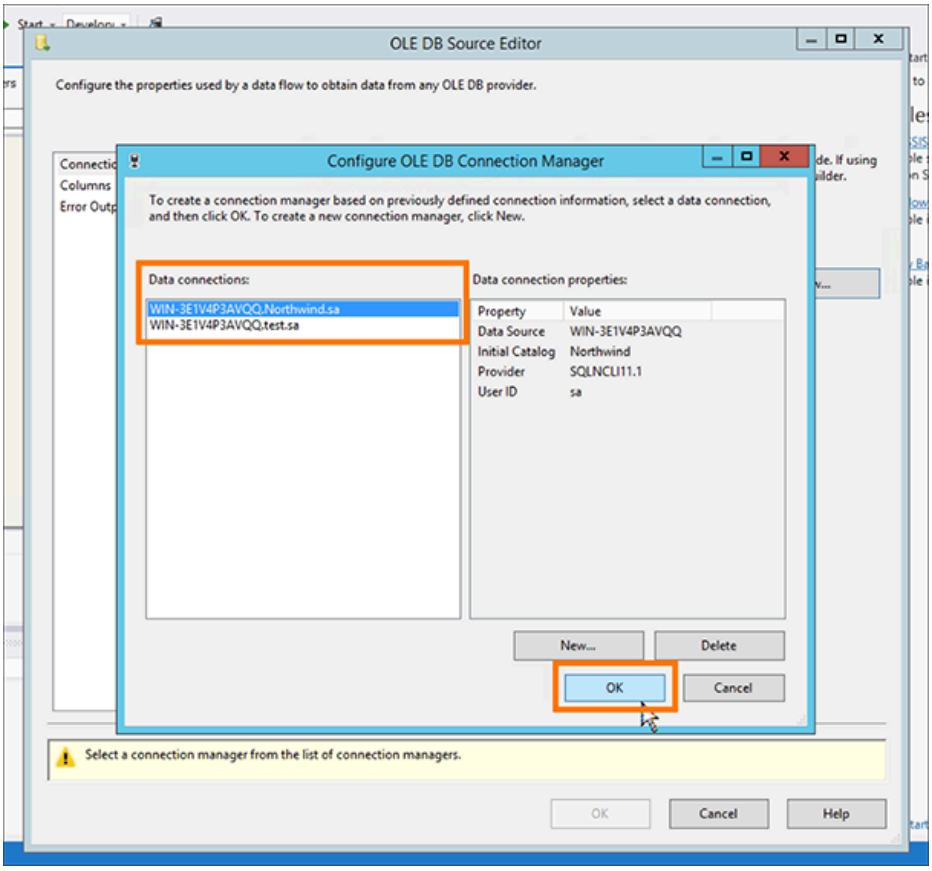
9. Double click the **Data Flow Task** icon when it appears in the center of the page.



10. Navigate back to the **SSIS Toolbox** tab. You now want to create sources and destinations.

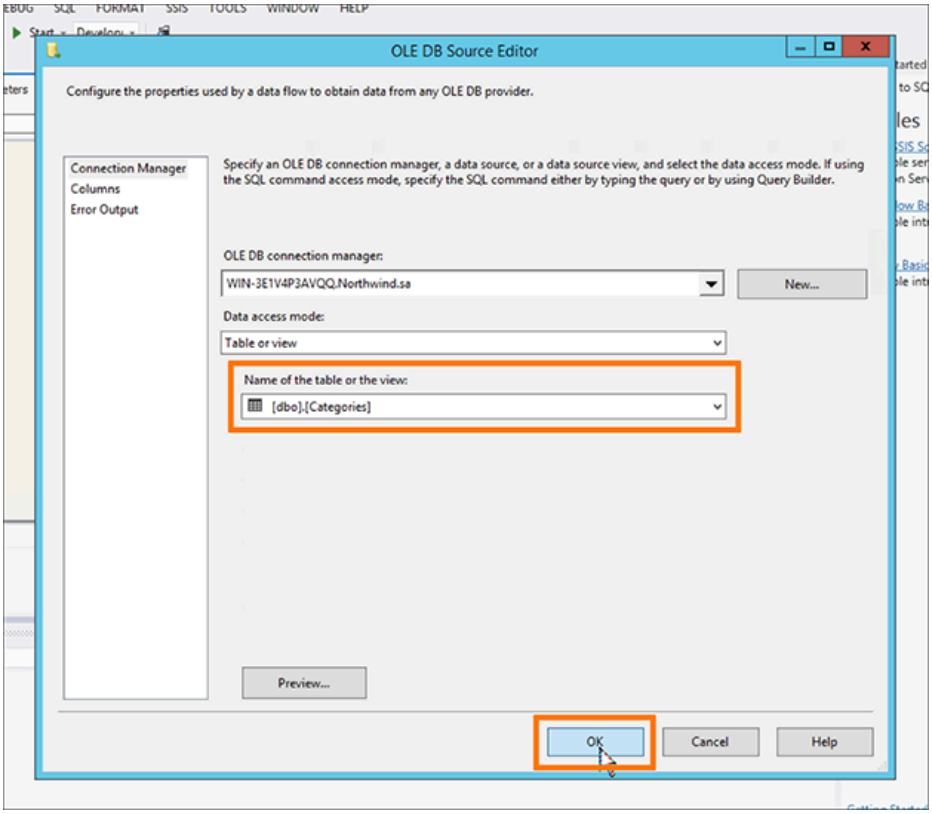
Create sources and destinations

1. Under **Other Sources**, find **OLE DB Source** and drag and drop it to the main window.
2. Double click the **OLE DB Source** icon when it appears in the center of the page to open the OLE DB Source Editor.
3. Select a new OLE DB connection manager by clicking **New**.
4. In the Configure OLE DB Connection Manager window, select your **Data connection** and click **OK**.



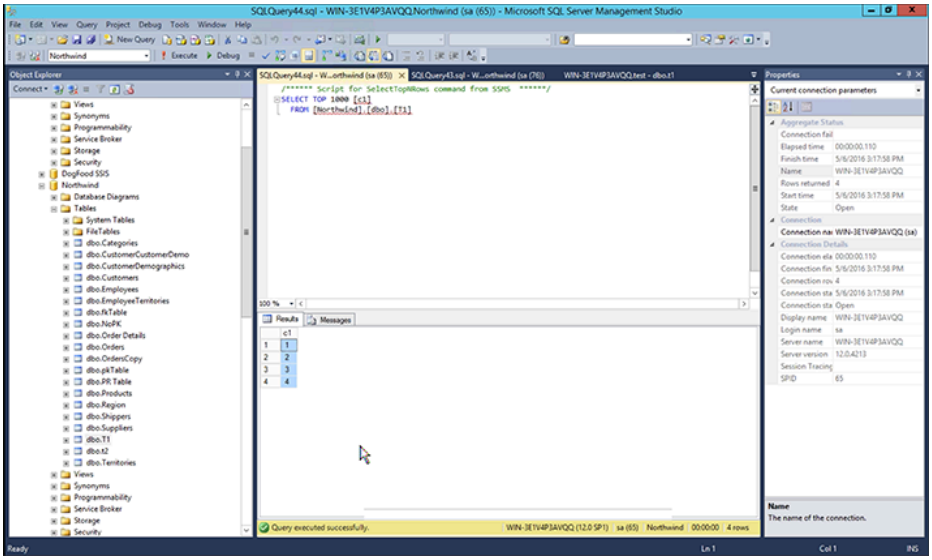
If you do not see your data connection, you will have to create a new one in the Connection Manager by clicking **New**.

5. Back in the OLE DB Source Editor, select the **Name of the table or the view**, and click **OK**.



6. Select the table, and see what columns are in it.

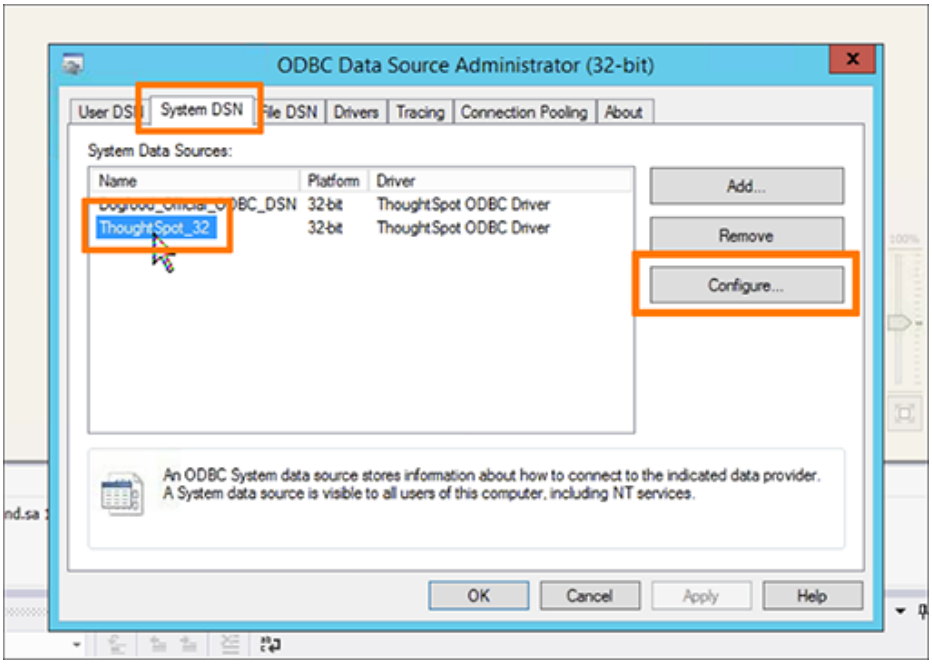
   In this example, a single column, `c1`, is selected.



Configure the ODBC Data Source Administrator

The ODBC Data Source Administrator has to be configured to connect to ThoughtSpot and bring the table in.

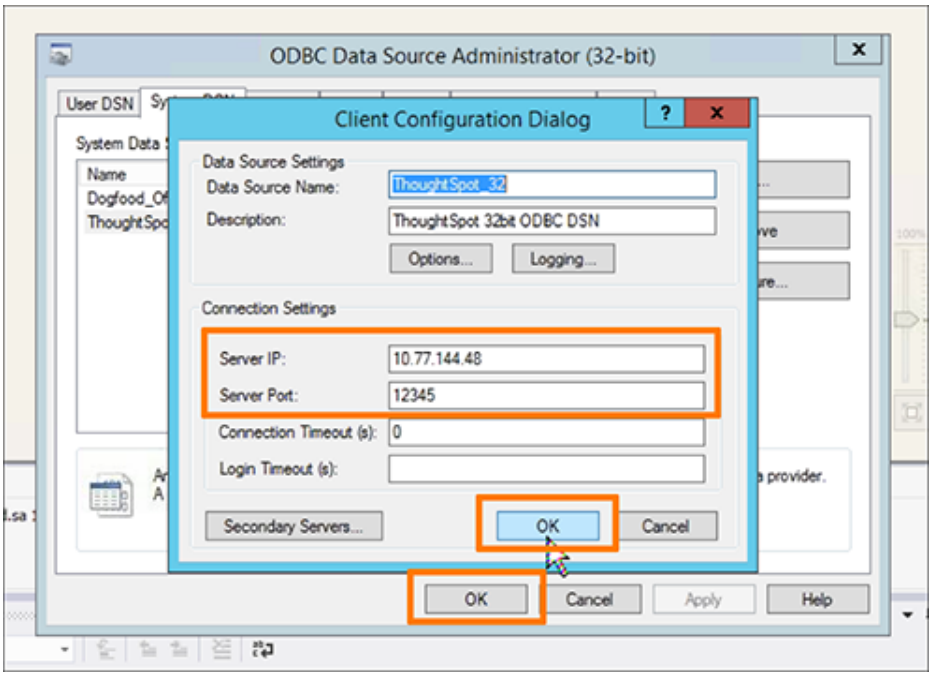
1. Search for and open your **ODBC Data Sources (32-bit)** program.
2. Click the **System DSN** tab and select **ThoughtSpot_32**.
3. Click **Configure**.



4. In the Client Configuration Dialog, enter the **Server IP** and **Server Port**.

   Enter any node IP that has Simba server running on it. In **Secondary Servers**, you must specify all node IPs, because ThoughtSpot must resolve to the server Simba runs on, and that server can change after an upgrade. Enter one server IP per line. The line return serves as a separator. Comma separated values are not supported.

5. Click **OK** twice to close the Client Configuration Dialog and the ODBC Data Source Administrator.

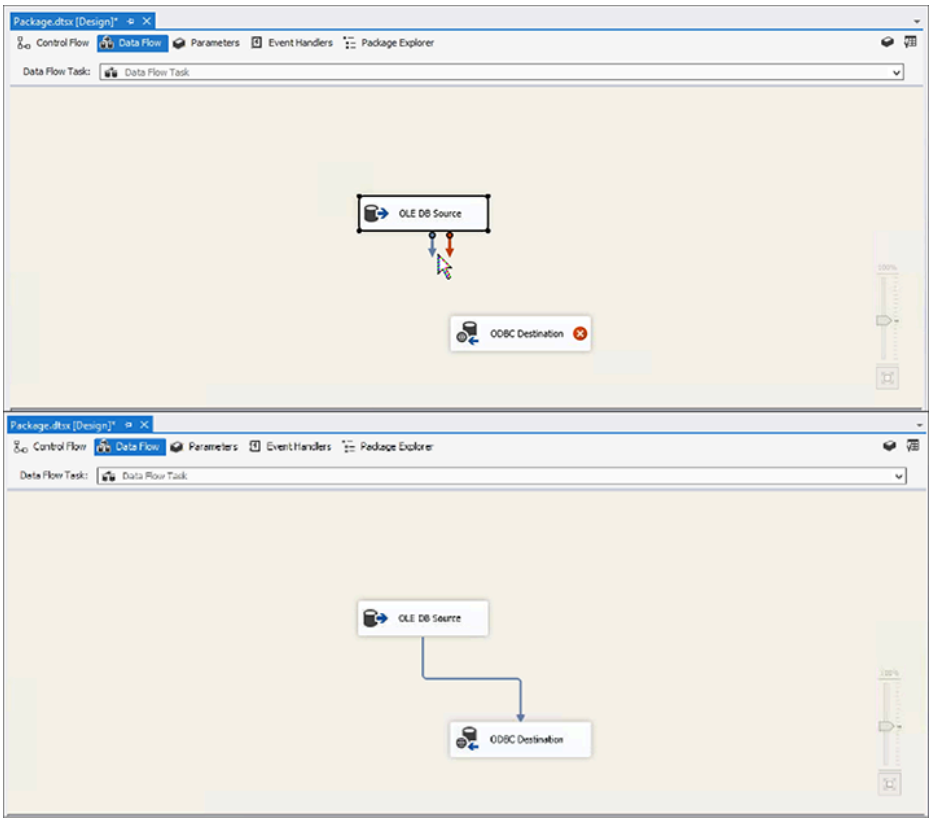


Create a file to take the feed

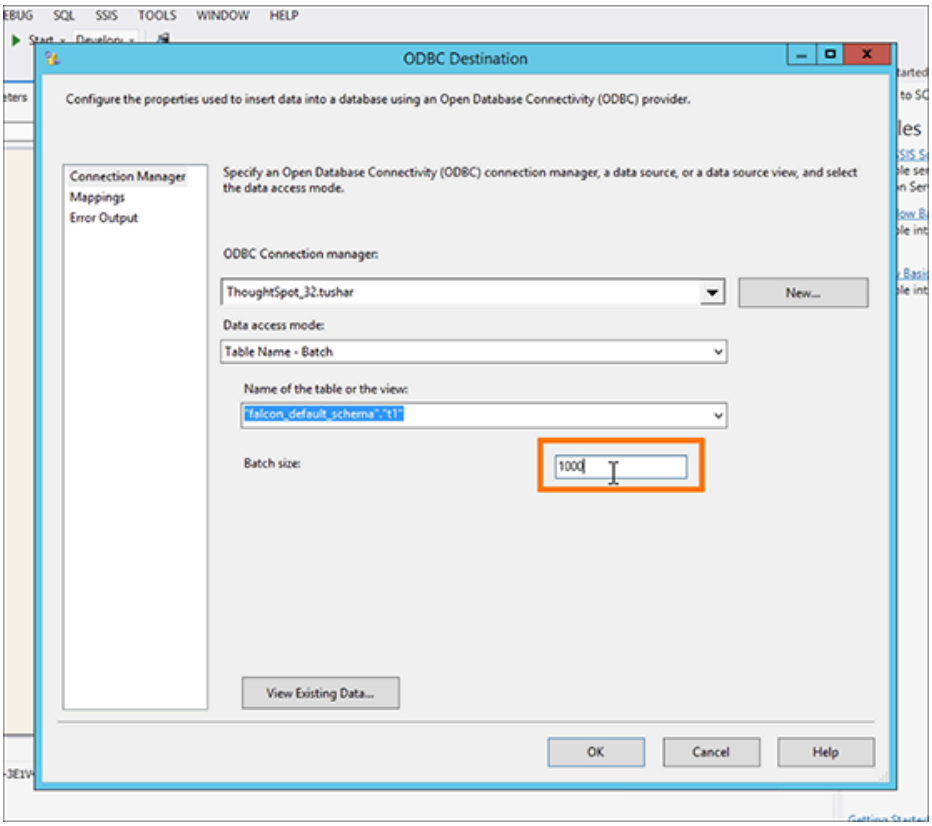
Now that you have set up your source, create the empty table in ThoughtSpot to take this feed. SSIS does not allow you to create the table in ThoughtSpot. You have to do this first in TQL. In Pentaho, it will create the table in ThoughtSpot, but not in SSIS.

Create the ODBC Destination. Use the one you created and named in the ODBC Data Source Administrator.

1. In the **SSIS Toolbox** tab, under **Other Destinations**, drag and drop **ODBC Destination** to the main window.
2. Drag the **blue arrow** to connect the OLE DB Source icon to the ODBC Destination icon.
3. Double click the **ODBC Destination** icon.



4. Use ODBC Destination to set the **Batch size** for the connection in the Connection Manager tab. You can set the size to be up to 10,000.



If the load fails, the entire batch will be lost, and you will have to start that load over again.

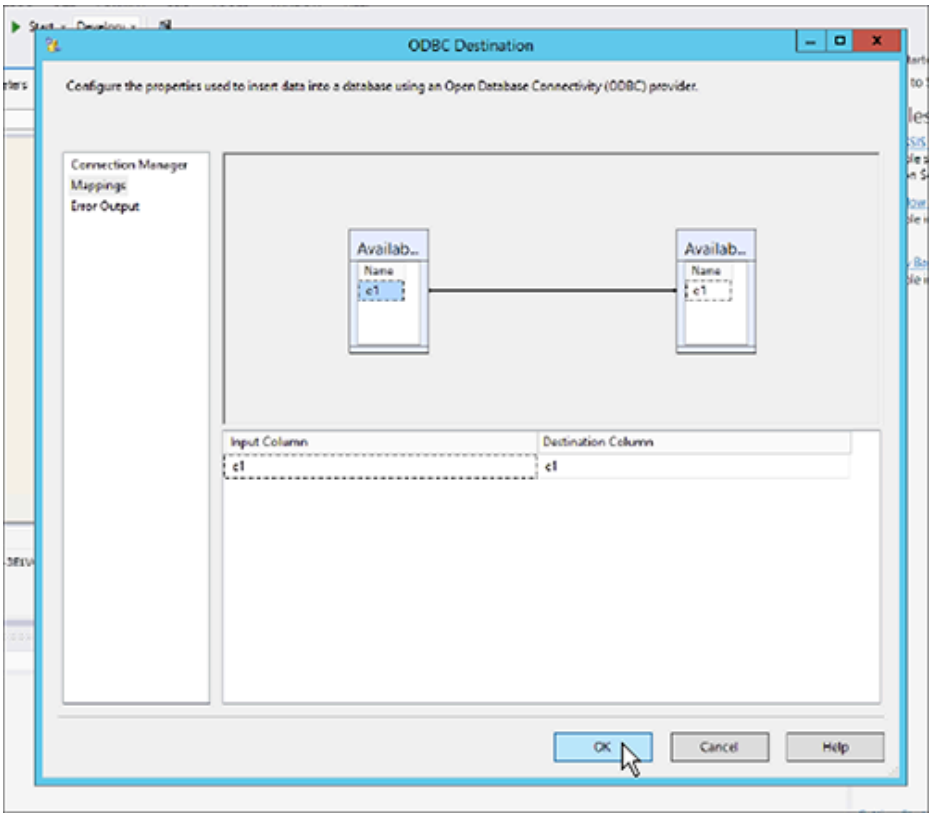
5. Set the **Transaction Size** to match the total number of rows that are expected to be loaded in the load cycle.

   Your transaction size can be quite large—even spanning a million rows. However, too many small batches can leave the cluster in a rough state. This is because each batch acts as a separate transaction and creates a separate commit. Too many of these will slow down our system since each transaction creates a “data version” in our system. In Pentaho, the transaction size setting is called Commit Size.

6. Set the **Transaction Option** attribute of the Data Flow Task to **Supported**.

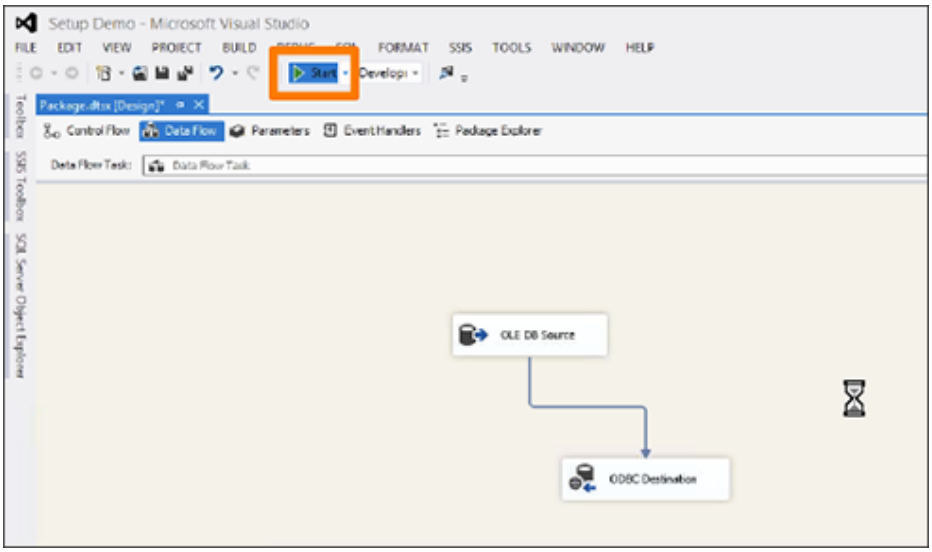
7. In the **Mappings** tab, validate the mapping or change it.

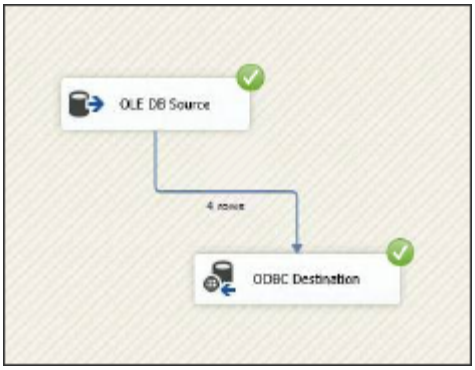
   You can have different column names in each database if you map them. Of course, they must be of the same or compatible datatype.



8. Start the import job by clicking the **Start** button.

   You should see an animation indicating that the data is transferring over. When the import is complete, the number of successfully transferred rows is displayed.





You can validate the import using TQL or from the **Data** screen.

Install the ODBC Driver on Linux

Summary: Use this procedure to obtain the Linux ODBC driver and install it.

ThoughtSpot's ODBC connection relies on the SimbaEngine X SDK to connect through ODBC or JDBC to ThoughtSpot's remote data stores. The instructions on this page explain how to configure the Simba ODBC driver on a Linux workstation.

Make sure you have read the overview material in the ODBC driver overview. This workstation is the same machine where you plan to run your ETL activities.

Check the ThoughtSpot IP and the simba_server status

Before you begin, you need to know the IP address or DNS name of the server you intend to connect your server to.

1. SSH as `admin` or the `thoughtspot` user to your ThoughtSpot node.
2. Verify the node IP(s).

```
$ tscli node ls
172.18.231.17
172.18.231.18
```

3. Make a note of each IP; there may be more than one.
4. Configure the ThoughtSpot firewall to allow connections from your ETL client, by running the following command on any ThoughtSpot node: `tscli firewall open-ports --ports 12345`
5. Exit or close the shell.

Install the Simba client

On your workstation, where you want to connect from, do the following to get the ODBC driver:

1. Open a browser on your workstation.

2. Navigate to the **Downloads** page.
3. Click **ODBC Driver for Linux** to download the driver.
4. Open a terminal on your workstation.
5. Change directory to the location where you downloaded the file.
6. Optionally, move the file to a permanent location on your machine.

   When you expand the downloaded file it will create a directory in the location.

7. Unzip the zip file:

   ```
   gunzip ThoughtSpot_linux_odbc_<version>.tar.gz
   ```

8. Extract the contents of the `tar` file.

   ```
   tar –xvf ThoughtSpot_linux_odbc_<version>.tar
   ```

   This extracts a subdirectory called `linux` into the current directory.

9. Take a moment to examine the contents of the new directory.

   The structure contains a Simba client library, supporting libraries and setup files for two different architectures. It also continues error messages for multiple languages.

```
├── Bin
│   ├── Linux_x86
│   │   └── libSimbaClient.so
│   └── Linux_x8664
│       └── libSimbaClient.so
├── ErrorMessages
│   ├── de-DE
│   │   ├── ...
│   ├── en-US
│   │   ├── CSCommonMessages.xml
│   │   ├── ClientMessages.xml
│   │   ├── JNIDSIMessages.xml
│   │   ├── ODBCMessages.xml
│   │   ├── SQLEngineMessages.xml
│   │   └── ServerMessages.xml
│   ├── es-ES
│   ├── fr-FR
│   ├── it-IT
│   └── ja-JP
├── Lib
│   ├── Linux_x86
│   │   ├── libcrypto.so
│   │   ├── libcrypto.so.1.0.0
│   │   ├── ...
│   └── Linux_x8664
│       ├── ...
└── Setup
    ├── .simba.javaquickstart.ini
    ├── .simba.javaultralight.ini
    ├── .simba.quickstart.ini
    ├── .simba.ultralight.ini
    ├── .simbaclient.ini
    ├── .simbaserver.javaquickstart.ini
    ├── .simbaserver.javaultralight.ini
    ├── .simbaserver.quickstart.ini
    ├── .simbaserver.ultralight.ini
    ├── odbc.ini
    └── odbcinst.ini

14 directories, 92 files
```

The `linux/Setup` directory contains the key ODBC configuration files and sample Simba client configurations you can use later in this procedure.

10. You must know your workstation architecture to continue, confirm your workstation's architecture.

    You can use the `arch` or the `uname` command or both.

    ```
     $ arch
       x86_64
     $ uname -a
       Linux nebula-docs-production-4vfnv 4.4.108-1.el7.elre
    po.x86_64 #1 SMP Mon Dec 25 09:55:39 EST 2017 x86_64 x8
    6_64 x86_64 GNU/Linux
    ```

    In previous examples, the workstation is a 64 bit workstation. Your workstation may be 32-bit. You can use this architecture information in the procedures that follow.

(Optional) Install unixODBC tools for testing

The procedures on this page rely on the unixODBC tools to test your configuration and connection. If you are experienced with ODBC and want to skip this, you can. Simply substitute your preferred mechanism in the subsequent procedures where references are made to the unixODBC tools.

> ❗ **Warning:** Your ThoughtSpot installation contains a version of the unixODBC tools. These tools are incompatible with CentOS. Do not use these tools if you are performing this procedure on your ThoughtSpot server.

1. Search for the unixODBC tools on your system.

   The `yum` package manager searches for software already installed or available on your system or from the configured repositories. Depending on your workstation configuration, you may need to use the `sudo` command with your workstation.

```
$ yum search unixODBC
...
 * updates: repos-lax.psychz.net
================ N/S matched: unixODB
C============================
opensips-unixodbc.x86_64 : OpenSIPS unixODBC Storage sup
port
unixODBC-devel.i686 : Development files for programs whi
ch will use the unixODBC library
unixODBC-devel.x86_64 : Development files for programs w
hich will use the unixODBC library
erlang-odbc.x86_64 : A library for unixODBC support in E
rlang
freeradius-unixODBC.x86_64 : Unix ODBC support for freer
adius
unixODBC.i686 : A complete ODBC driver manager for Linux
unixODBC.x86_64 : A complete ODBC driver manager for Lin
ux
```

Make note of the correct package to install for your architecture.

2. Install the appropriate package for your architecture.

   In this case the command installs the tools for a 64-bit architecture. A 32-bit package needs the `unixODBC.i686` package.

```
 [admin@nebula-docs-odbc-test-cxmrn ~]$ yum install unix
ODBC.x86_64
 Loaded plugins: fastestmirror, ovl
 Loading mirror speeds from cached hostfile
  * base: mirror.linuxfix.com
  * elrepo: repos.lax-noc.com
  * epel: mirror.hmc.edu
  * extras: centos-distro.cavecreek.net
  * rpmforge: mirror.lstn.net
  * updates: repos-lax.psychz.net
 Resolving Dependencies
 --> Running transaction check
 ---> Package unixODBC.x86_64 0:2.3.1-11.el7 will be ins
talled
 ...
 Complete!
```

3. Verify the files were installed.

```
$ ls /usr/bin/isql
/usr/bin/isql
$ ls /usr/bin/odbcinst
/usr/bin/odbcinst
```

Set up your environment

In this section, you set parameters in your workstation to support your ODBC connection.

1. Copy the library for your architecture from the `Lib` directory on your Linux machine.

Library	Architecture
`/linux/Lib/Linux_x86`	32-bit
`/linux/Lib/Linux_x8664`	64-bit

2. Add the location's path to the `LD_LIBRARY_PATH` environment variable.

   For example if your architecture is 64-bit and you keep the library in your `home` directory:

```
export LD_LIBRARY_PATH=~/linux/Lib/Linux_x8664/
```

3. Use the `echo` command to verify the path was added correctly.

```
echo $LD_LIBRARY_PATH
```

4. Copy the `odbc.ini` file to the `/etc` directory.

```
$ cp ~/linux/Setup/odbc.ini  /etc
```

If you have trouble making the copy, use the `sudo` command to make the move.

5. Copy the `odbcinst.ini` file to the `/etc` directory.

```
$ cp ~/linux/Setup/odbcinst.ini  /etc
```

6. Copy the hidden `.simba.quickstart.ini` file to the `/etc` directory, renaming it in the process to `simbaclient.ini`.

```
$ cp ~/linux/Setup/.simba.quickstart.ini  /etc/simbacli
ent.ini
```

7. Update your environment with the `ODBCSYSINI` and `ODBCINI` variables.

```
$ export ODBCSYSINI=/etc/
$ export ODBCINI=/etc/odbc.ini
```

8. Use the `/usr/bin/odbcinst` command to confirm your settings:

```
$ /usr/bin/odbcinst -j
unixODBC 2.3.1
DRIVERS............: /etc/odbcinst.ini
SYSTEM DATA SOURCES: /etc/odbc.ini
FILE DATA SOURCES..: /etc/ODBCDataSources
USER DATA SOURCES..: /etc/odbc.ini
SQLULEN Size.......: 8
SQLLEN Size........: 8
SQLSETPOSIROW Size.: 8
```

Edit the /etc/simbaclient.ini file

When you are ready, follow this procedure to configure the driver.

1. Edit the `/etc/simbaclient.ini` file with your favorite editor. 2.

2. Change the `ErrorMessagesPath` property to point to the location where you unzipped the client.

```
[Driver]
ErrorMessagesPath=<path_to_error_messages_directory>
```

3. Comment out the `# Generic ODBCInstLib` value.
4. Uncomment the `ODBCInstLib` property.

   When you are done, your file looks like the following:

```
# Generic ODBCInstLib
#   iODBC
#ODBCInstLib=libiodbcinst.so

#   SimbaDM / unixODBC
ODBCInstLib=libodbcinst.so
```

5. Save and close the `/etc/simbaclient.ini` file.

Edit the odbcinst.ini file

The `odbcinst.ini` file is a registry and configuration file for ODBC drivers. Depending on your workstation architecture, you configure the 32-bit or 64-bit driver.

1. Open the file `/etc/odbcinst.ini` in your favorite editor.
2. Comment out the driver that you don't need.

   For example, if you are using 64-bit, comment out 32-bit.

3. Edit the `Driver` line so that it contains the path to the file `libSimbaClient.so`

   Use the path where you copied the library files. For example, for the 64-bit ODBC driver:

```
 [ThoughtSpot(x64)]
 APILevel            = 1
 ConnectFunctions    = YYY
 Description         = ThoughtSpot 64bit ODBC driver
 Driver              = /home/admin/linux/Bin/Linux_x866
4/libSimbaClient.so
 DriverODBCVer       = 03.52
 SQLLevel            = 1
```

4. Make sure the remaining driver is named `ThoughtSpot` without any special characters.

   When you are done, you should see something similar to the following:

```
 #[ThoughtSpot]
 #APILevel            = 1
 #ConnectFunctions    = YYY
 #Description         = ThoughtSpot 32bit ODBC driver
 #Driver              = /usr/local/scaligent/toolchain/l
ocal/simba/odbc/linux/Bin/Linux_x86/libSimbaClient.so
 #DriverODBCVer       = 03.80
 #SQLLevel            = 1

 [ThoughtSpot]
 APILevel            = 1
 ConnectFunctions    = YYY
 Description         = ThoughtSpot 64bit ODBC driver
 Driver              = /home/admin/linux/Bin/Linux_x866
4/libSimbaClient.so
 DriverODBCVer       = 03.80
 SQLLevel            = 1
```

5. Save and close the `/etc/odbcinst.ini` file.

Edit the odbc.ini file

The `odbc.ini` file is a registry and configuration file for ODBC DSNs (Data Source Names). This file relies on the drivers registered in the `/etc/odbcinst.ini` file. Depending on your workstation architecture, you configure the 32-bit or 64-bit driver.

1. Open the file `/etc/odbc.ini` in the editor of your choice.

2. Comment out the configuration that you don't need.

   For example, if you are using 64-bit, comment out 32-bit.

3. Locate the `Description` section for the type of Linux you are using (32-bit or 64-bit).
4. Locate the line that begins with `ServerList`.
5. Replace `127.0.0.1` with a comma separated list of the IP addresses of each node on the ThoughtSpot instance.

   The syntax for the `ServerList` is:

   ```
   ServerList = <node1_IP> 12345, <node2_IP> 12345 [, <node3_IP> 12345, ...]
   ```

   If you need to obtain the IP addresses of the ThoughtSpot cluster nodes, run the command `tscli node ls` from a Linux shell on a ThoughtSpot appliance.

6. Do not edit the port number, leave it as `12345`.

   When you are done, your entry will look similar to the following (this example is for the 64-bit ODBC driver):

   ```
   [ThoughtSpot]
   Description = ThoughtSpot 64-bit ODBC Driver
   Driver = ThoughtSpot
   ServerList =  172.18.231.17 12345
   Locale = en-US
   ErrorMessagesPath = /home/admin/linux/ErrorMessages
   UseSsl = 0
   #SSLCertFile = # Set the SSL certificate file path. The certificate file can be obtained by extracting the SDK tarball
   #LogLevel = 0 # Set log level to enable debug logging
   #LogPath = # Set the debug log files path
   DATABASE = # Set the default database to connect to
   SCHEMA = # Set the default schema to connect to
   ```

7. Save and close the `odbc.ini` file.

Test your ODBC connection

At this point, you can test your ODBC connection to ThoughtSpot. It is important to recall that the username/password you use belongs to a ThoughtSpot application user. Typically, this user is a user with data management or administrative privileges on the application.

1. Before trying the ODBC connection, make sure you can use this username/password to login into the ThoughtSpot application.
2. Confirm the user's privileges by going to the **Data** tab.
3. Go back to your workstation's terminal shell.
4. Use the `/usr/bin/isql` and confirm you can connect.

   Specify the `ThoughtSpot` DSN:

```
/usr/bin/isql –v ThoughtSpot tsadmin adminpwd
+---------------------------------------+
| Connected!                            |
|                                       |
| sql–statement                         |
| help [tablename]                      |
| quit                                  |
|                                       |
+---------------------------------------+
SQL>
```

Now, you are ready to begin using the connection you've configured.

Best Practices for Using ODBC

Summary: To successfully use ODBC, following these best practices is recommended.

When developing tools that use the ODBC driver, use these best practices:

- When setting up ODBC for the first time, begin by using the ThoughtSpot `tsload` for the initial data loads. This allows you to do more in-depth troubleshooting on any initial loading issues. After initial loads work properly, switch to ODBC to perform incremental loads.

- You should create the parameterized SQL statement outside of ODBC. Using this method, the SQL statement can be sent to ThoughtSpot in batches by the ODBC driver, so you only have to update the memory itself. ETL tools have this implemented already (end users shouldn't have to actually write the `INSERT` statement). But as a developer, you may be writing code that leverages the ODBC driver, so this tip can help you write your SQL for the best performance with the driver.

- Data can be loaded into a table through multiple parallel connections. You can achieve this by splitting the input data into multiple parts. Then, load those individual parts through multiple parallel connections. You can use parallel loading even while loading to a single table or multiple tables at the same time.

- When doing an incremental data load, note that the same `UPSERT` behavior that occurs in TQL also occurs. This means that if you import a row whose primary key matches an existing row, the existing row will be updated with the new values.

Related information

- Enable ODBC logs
- Introduction to loading and managing data
- Loading and constraints

JDBC Driver Overview

Summary: Use JDBC to interact with databases in a standard manner.

Java Database Connectivity (JDBC) is a Java standard API that allows applications to interact with databases in a standard manner. ThoughtSpot has JDBC support through a JDBC driver that we provide.

Connector type

There are different types of JDBC connectors. Driver types categorize the technology used to connect to the database. The ThoughtSpot JDBC drive is a type 4 connector. It uses Java to implement a networking protocol for communicating with ThoughtSpot.

This driver is Java driver. There is no client installation or configuration.

When to use JDBC

JDBC can be used whenever you want to connect to ThoughtSpot to insert data programmatically from a Java program or application. You should begin by using the ThoughtSpot Loader for initial data loads and then use JDBC for incremental loads. This is because the ThoughtSpot Loader is generally faster than JDBC. Information on using the ThoughtSpot Loader is available in the ThoughtSpot Administrator Guide.

Version Compatibility

To ensure compatibility, always use the JDBC driver with the same version number as the ThoughtSpot instance to which you are connecting.

Performance Considerations

These are some general recommendations for maximizing the performance of JDBC:

- Insert in batches rather than doing single inserts at a time using the

`PreparedStatement::addBatch()` and `PreparedStatement::executeBatch` commands.

- If you need to upload a lot of data, consider running multiple connections with batch inserts in parallel.

Note: The ETL tool must add a data transformation step if the source column data type does not exactly match the target's, ThoughtSpot's, column data type. The driver does not do any implicit conversions.

Use the JDBC Driver

Summary: How to configure the JDBC driver.

ThoughtSpot's ODBC connection relies on the SimbaEngine X SDK to connect through ODBC or JDBC to ThoughtSpot's remote data stores. The instructions on this page explain how to configure the JDBC driver.

The ThoughtSpot JDBC driver is supplied by a `.jar` file you install on a workstation. This workstation is the same machine where you plan to run your ETL activities.

JDBC configuration parameters

Information	Description
Driver name	`com.simba.client.core.jdbc4.SCJDBC4Driver`
Server IP address	The ThoughtSpot appliance URL or IP address.
Simba port	The simba port, which is `12345` by default.
Database name	This is not the machine login username. The ThoughtSpot Database name to connect to.
username	The name of a ThoughtSpot user with administrator permissions.
password	The password of a ThoughtSpot application user. This is not the machine or SSH userpassword.

For more JDBC configuration options, see also:

- JDBC properties reference in this ThoughtSpot documentation
- SimbaClient for JDBC Configuration Properties reference

Check the ThoughtSpot IP and the simba_server status

Before you begin, you need to know the IP address or DNS name of the server you intend to connect your server to.

1. SSH as `admin` or the `thoughtspot` user to your ThoughtSpot node.
2. Verify the node IP(s).

```
$ tscli node ls
172.18.231.17
172.18.231.18
```

3. Make a note of each IP; there may be more than one.
4. Configure the ThoughtSpot firewall to allow connections from your ETL client, by running the following command on any ThoughtSpot node: `tscli firewall open-ports --ports 12345`
5. Exit or close the shell.

Install the driver

The JDBC driver is a `.jar` packaged application. To use the package, you download it, install it

1. Log in to the local machine where you want to install the JDBC driver.
2. Click **Here** to download the JDBC driver.
3. Click **JDBC Driver** to download the file `thoughtspot_jdbc<version>.jar`.
4. Move the driver to the desired directory on your local machine.
5. Add the downloaded JDBC driver to your Java class path on the local machine.

Write your application

Using JDBC with ThoughtSpot is the same as using any other JDBC driver with any other database. You must provide the connection information, create a connection, execute statements, and close the connection.

Specify each of the nodes in the cluster in the connection string, as shown. This enables high availability for JDBC connections. To find out the nodes in the cluster, you can run the command `tscli node ls` from the Linux shell on the ThoughtSpot instance.

The format for the connection is:

```
jdbc:simba://<node1>:12345,<node2>:12345,<node3>:12345[,…];
           LoginTimeout=<seconds>;DATABASE=<db>;SCHEMA=<schema>
```

For example:

```
jdbc:simba://192.168.2.248:12345,192.168.2.249:12345,192.16
8.2.247:12345;
           LoginTimeout=5;DATABASE=test;SCHEMA=falcon_default_s
chema
```

As shown, the `DATABASE` and `SCHEMA` parameters need to be in all caps. For the `simba` JDBC driver to work with Spark, the `DATABASE` and `SCHEMA` must be specified in the URL. They cannot be specified as a name/value pair as a map or property. For example:

```
val tssqldf1 = sparkSession.read.format("jdbc").options(Map("ur
l" ->
"jdbc:simba://10.84.78.181:12345;DATABASE=movieratings;SCHEMA=f
alcon_default_schema", "driver" ->
"com.simba.client.core.jdbc4.SCJDBC4Driver", "dbtable" -> "Movi
es", "user" ->
"tsadmin", "password" -> "admin")).load()
```

This `InsertData.java` example shows how to use ThoughtSpot with JDBC. This is an example of a reference JDBC application:

```
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class InsertData {

  // JDBC class to use.
  private static final String DB_DRIVER = "com.simba.client.cor
e.jdbc4.SCJDBC4Driver";
  // jdbc_example should be an existing database.

  private static final String DB_CONNECTION = "jdbc:simba://19
2.168.2.129:12345;
     192.168.2.249:12345,192.168.2.247:12345;
     LoginTimeout=5;DATABASE=jdbc_example;SCHEMA=falcon_defaul
t_schema";

  private static final String TABLE_NAME = "jdbc_example";
  private static final String DB_USER = "<username>";
  private static final String DB_PASSWORD = "<password>";

  // Assuming everything in local directory use:
  //   javac InsertData.java
  //   java -cp .:thoughtspot_jdbc4.jar InsertData
  public static void main(String[] argv) {

    try {
      insertRecordsIntoTable();
    }
    catch (SQLException e) {
      System.out.println(e.getMessage());
    }
  }

  /**
   * Insert some records using batch updates.
   *  Assumes a table exists:  CREATE TABLE "jdbc_example" ( "t
ext" varchar(10) );
   */
  private static void insertRecordsIntoTable() throws SQLExcept
ion {

    System.out.println("Inserting records.");
    Connection dbConnection = getDBConnection();
```

```
    PreparedStatement preparedStatement = null;
    String insertTableSQL = "INSERT INTO falcon_default_schem
a.jdbc_example (text) VALUES (?)";

    try {
      preparedStatement = dbConnection.prepareStatement(insertT
ableSQL);

      // Create multiple statements and add to a batch update.
      for (int cnt = 1; cnt <= 10; cnt++) {
        preparedStatement.setString(1, "some string " + cnt);
        preparedStatement.addBatch();
        System.out.println("Record " + cnt + " was added to th
e batch!");
      }
      preparedStatement.executeBatch();  // For large numbers o
f records, recommend doing sets of executeBatch commands.
      System.out.println("Records committed");

    }
    catch (SQLException sqle) {
      sqle.printStackTrace();
    }
    finally {

      if (preparedStatement != null) {
        preparedStatement.close();
      }
      if (dbConnection != null) {
        dbConnection.close();
      }
    }
  }

  /** Create a connection to the database. */
  private static Connection getDBConnection() {
    Connection dbConnection = null;
    try {
      Class.forName(DB_DRIVER);
    }
    catch (ClassNotFoundException e) {
      System.out.println(e.getMessage());
    }
    try {
      dbConnection = DriverManager.getConnection(DB_CONNECTIO
```

```
N, DB_USER,DB_PASSWORD);
      return dbConnection;
    }
    catch (SQLException sqle) {
      System.out.println(sqle.getMessage());
    }

    return dbConnection;
  }

}
```

Related Information

- Enable JDBC logs
- Connection configuration
- Supported SQL commands

Set up the JDBC driver for Pentaho

Summary: JDBC to connect to the ThoughtSpot Simba server from Pentaho.

You can use the Pentaho Data Integration (PDI) to create a JDBC connection. The Pentaho Data Integration (PDI) suite is a comprehensive data integration and business analytics platform. You can use it to create a JDBC connection to ThoughtSpot.

PDI consists of a core data integration (ETL) engine and GUI applications that allow you to define data integration jobs and transformations. Through Pentaho, we primarily use the JDBC driver to set up a connection. The process is not as complicated as with SSIS, and is much more lenient.

Community and enterprise editions of PDI are available. Using the community edition is sufficient, though you may use the enterprise edition, which is subscription based, and therefore contains extra features and provides technical support.

Use JDBC to connect to the ThoughtSpot Simba server from Pentaho. The connection will be made between a new ThoughtSpot Table Input and Output objects.

Check the ThoughtSpot IP and the simba_server status

Before you begin, you need to know the IP address or DNS name of the server you intend to connect your server to.

1. SSH as `admin` or the `thoughtspot` user to your ThoughtSpot node.
2. Verify the node IP(s).

```
$ tscli node ls
172.18.231.17
172.18.231.18
```

3. Make a note of each IP; there may be more than one.
4. Configure the ThoughtSpot firewall to allow connections from your ETL client, by running the following command on any ThoughtSpot node: `tscli firewall open-ports --ports`

`12345`

5. Exit or close the shell.

Install the Simba drivers in the Pentaho directories

Before starting the Pentaho Data Integration (PDI) client and creating the connection, ensure that the Simba JDBC client libraries are present in the Pentaho client/server machines. This will ensure that the drivers picked up at runtime.

1. Log in to the local machine where you have alredy installed the Pentaho Data Integration (PDI) client.
2. Click **Here** to download the JDBC driver.
3. Click **JDBC Driver** to download the file `thoughtspot_jdbc<version>.jar`.
4. Copy the `thoughtspot_jdbc<version>.jar` file to the following directories:
    - `<Pentaho_install_dir>/server/data-integration-server/tomcat/webapps/pentaho-di/WED-INF/lib/`
    - `<Pentaho_install_dir>/design-tools/data-integration/lib/`
    - `<Pentaho_install_dir>/server/data-integration-server/tomcat/lib/`
    - `<Pentaho_install_dir>/design-tools/data-integration/plugins/spoon/agile-bi/lib/`

Set up the driver

This section explains how to set up the JDBC driver using Pentaho. These instructions use Spoon, the graphical transformation and job designer associated with the PDI suite. It is also known as the Kettle project.

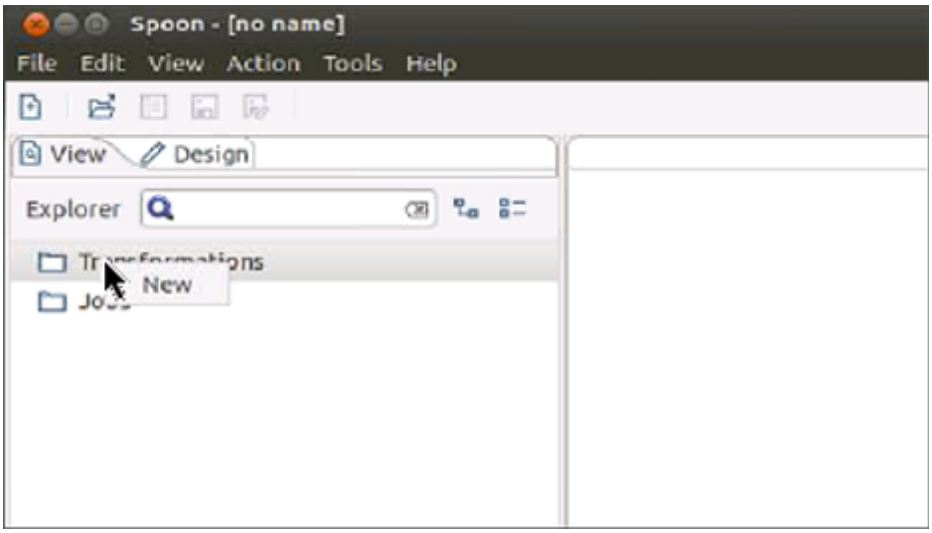
Create a transformation

Do the following on your ETL workstation with the Pentaho client:

1. Open the PDI client.

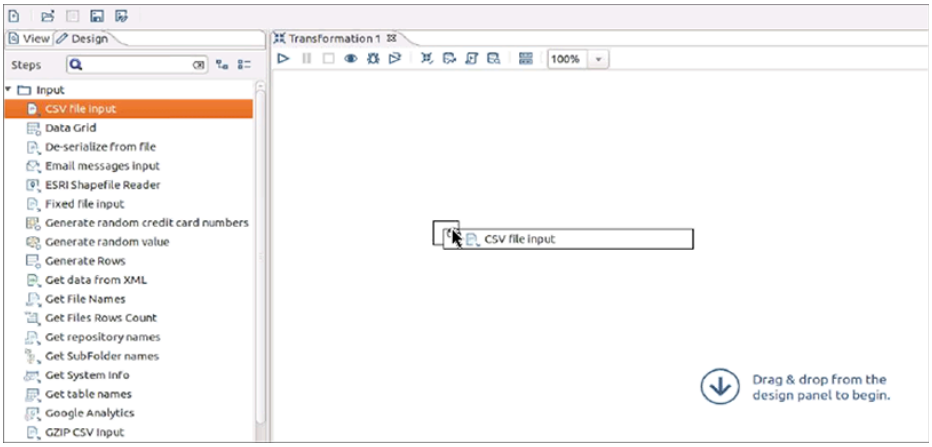
```
./spoon.sh &>/dev/null &
```

2. Right click **View > Transformations** tab.
3. Click **New** to create a new transformation.

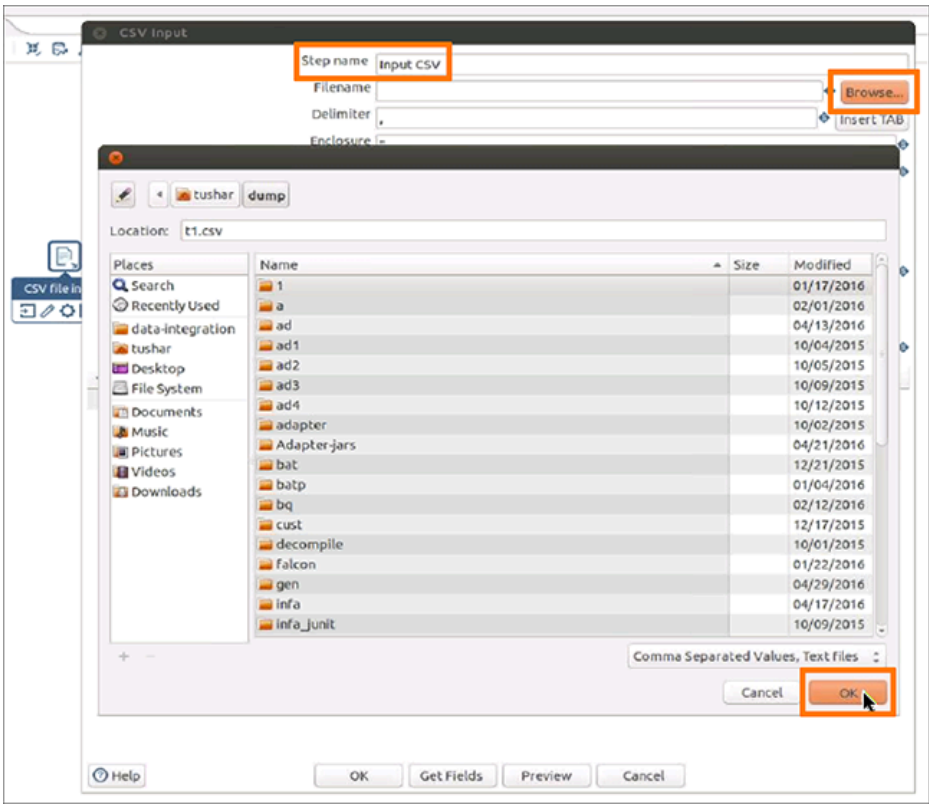


4. Click **Input** under the **Design** tab to expand it.
5. Drag and drop **CSV File Input** to the **Transformation** window.

   This opens a new CSV file.



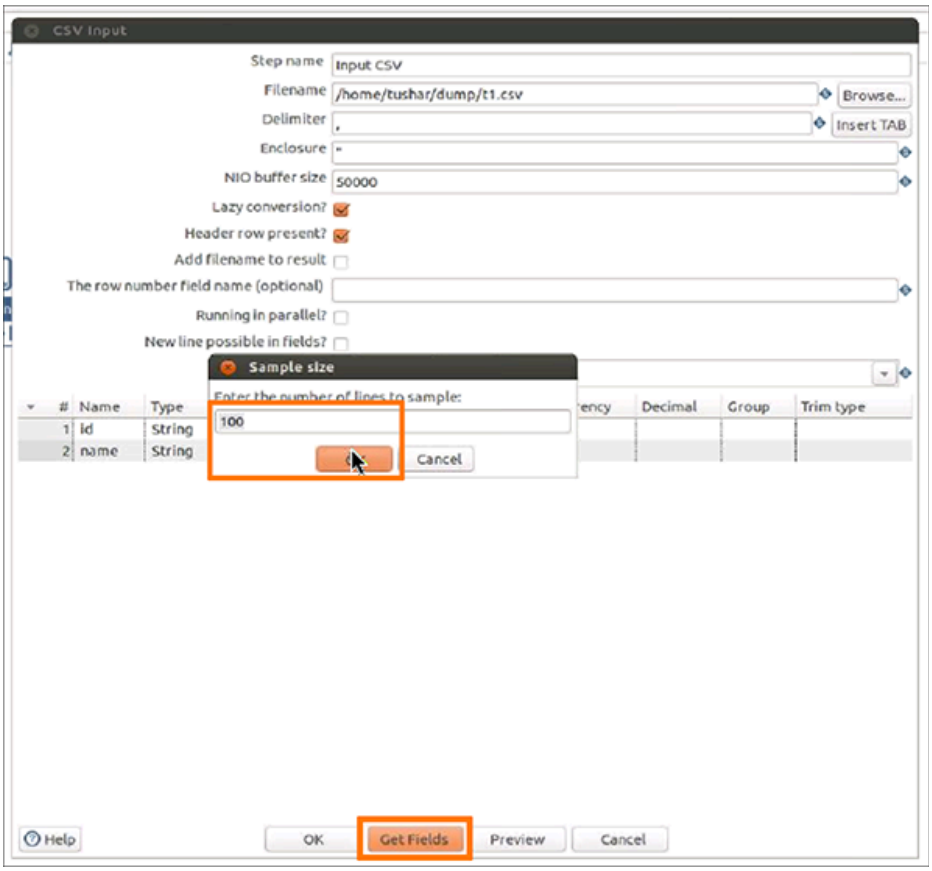
6. Double-click the **CSV File Input** icon to open the **CSV Input** dialog .
7. Name the **Step**.
8. Click **Browse** next to the **Filename** field and provide the file you want to read from.
9. Click **OK**.



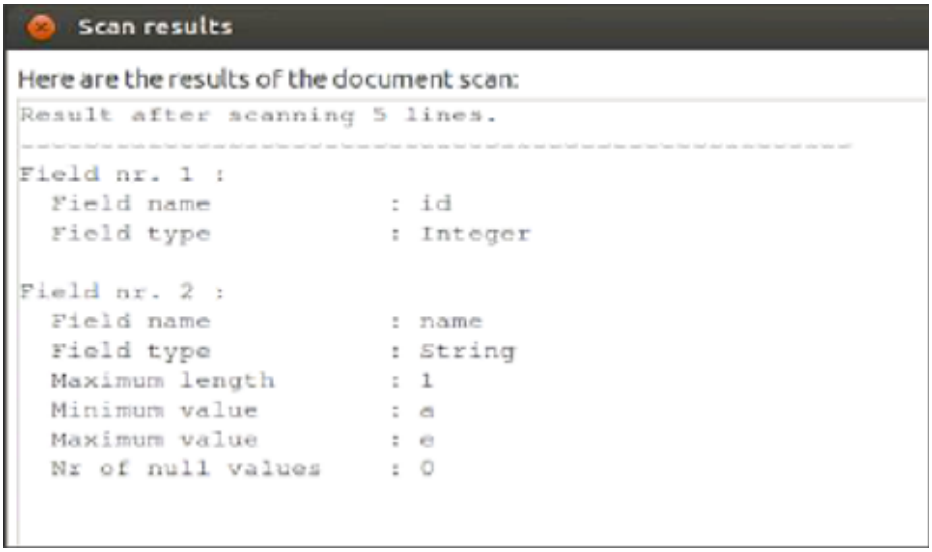
10. In the CSV Input dialog, click **Get Fields**.
11. Enter the number of lines you would like to sample in the Sample size dialog.

The default setting is 100.

1. Click **OK** when you are ready.



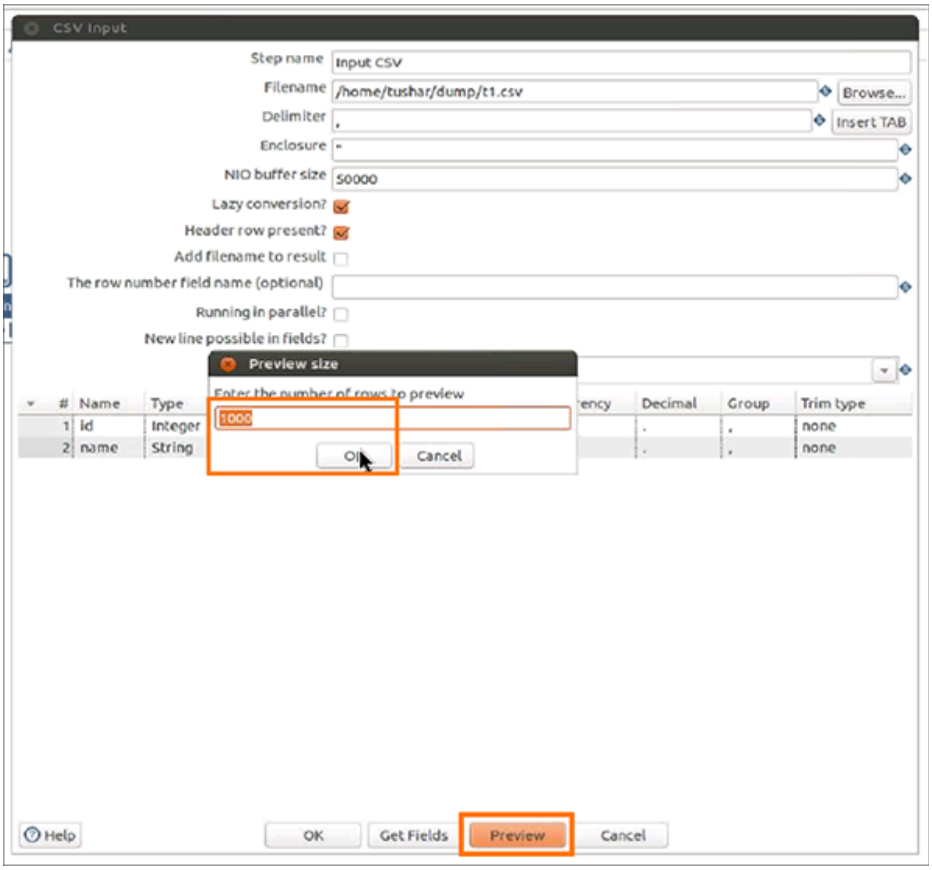
The tool reads the file and suggests the field name and type.



2. Click **Preview** to preview the data.

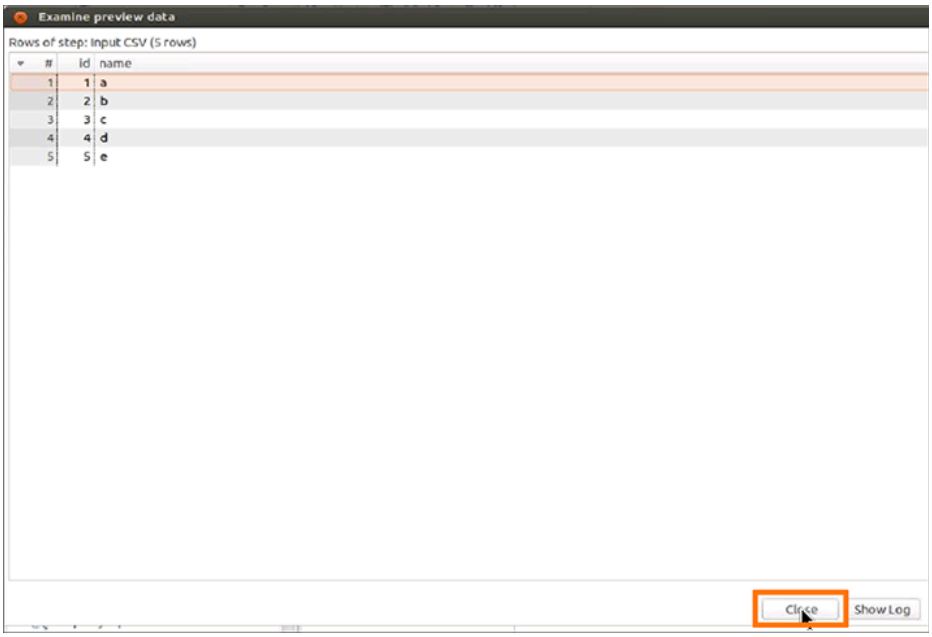
3. Enter the number of rows to preview in the **Preview size** dialog.

The default setting is 1000. Click **OK** to start the transformation in preview.



4. Examine the preview data, then click **Close**.

You may want to verify that you are able to read the data using the SQL query from ThoughtSpot.

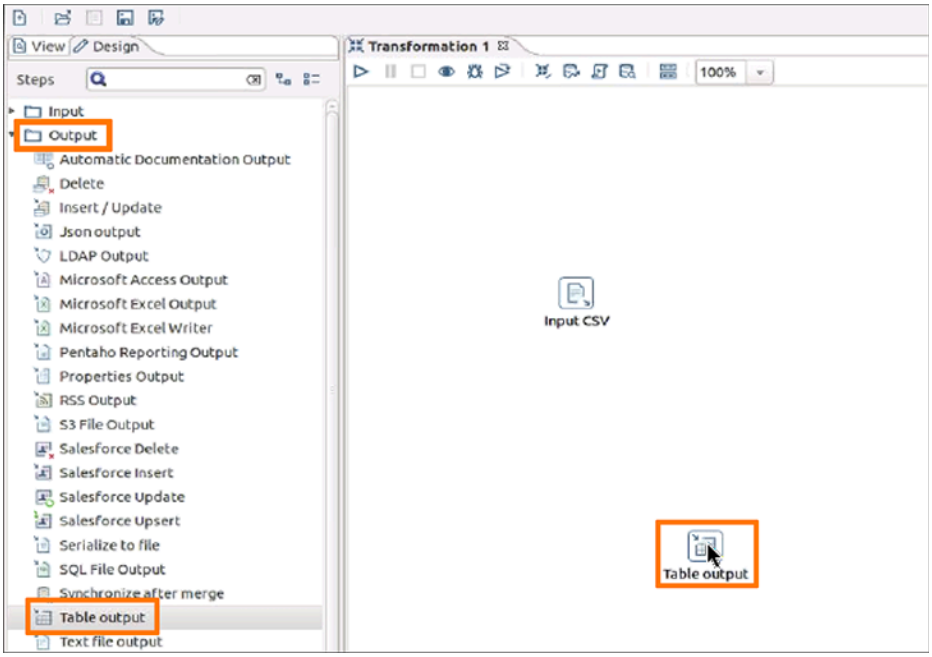


5. Click **OK** in the CSV Input dialog to confirm your CSV input settings.

Define the Output

1. Click **Design > Output**.

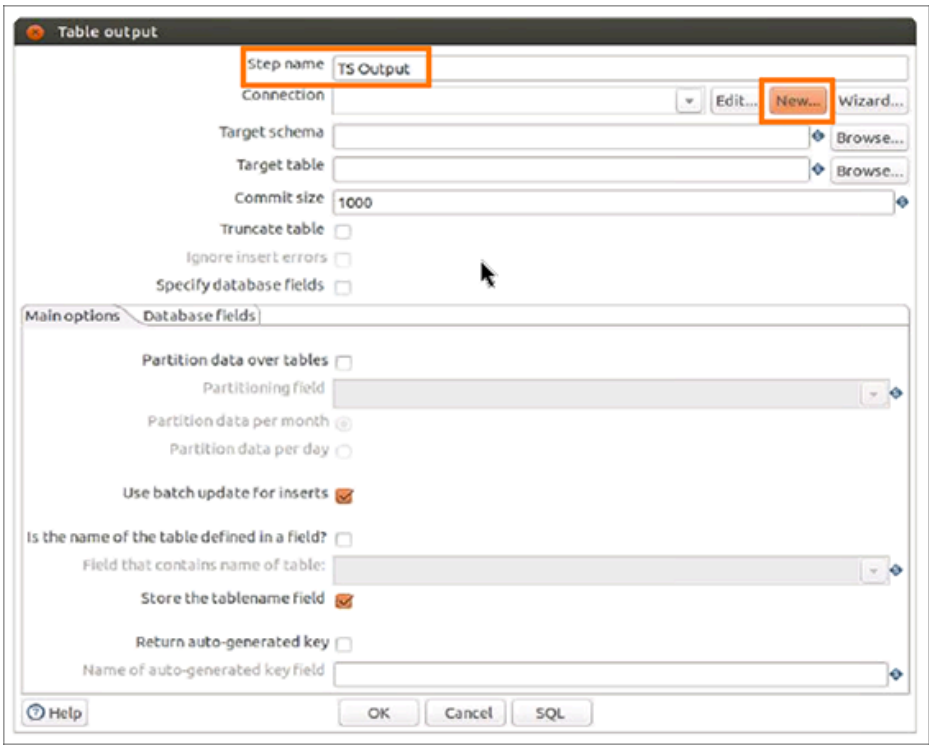
2. Drag and drop **Table output** to the **Transformation** window.



3. Double click the **Table output** icon to open the Table output dialog.

4. Enter a **Step name**.

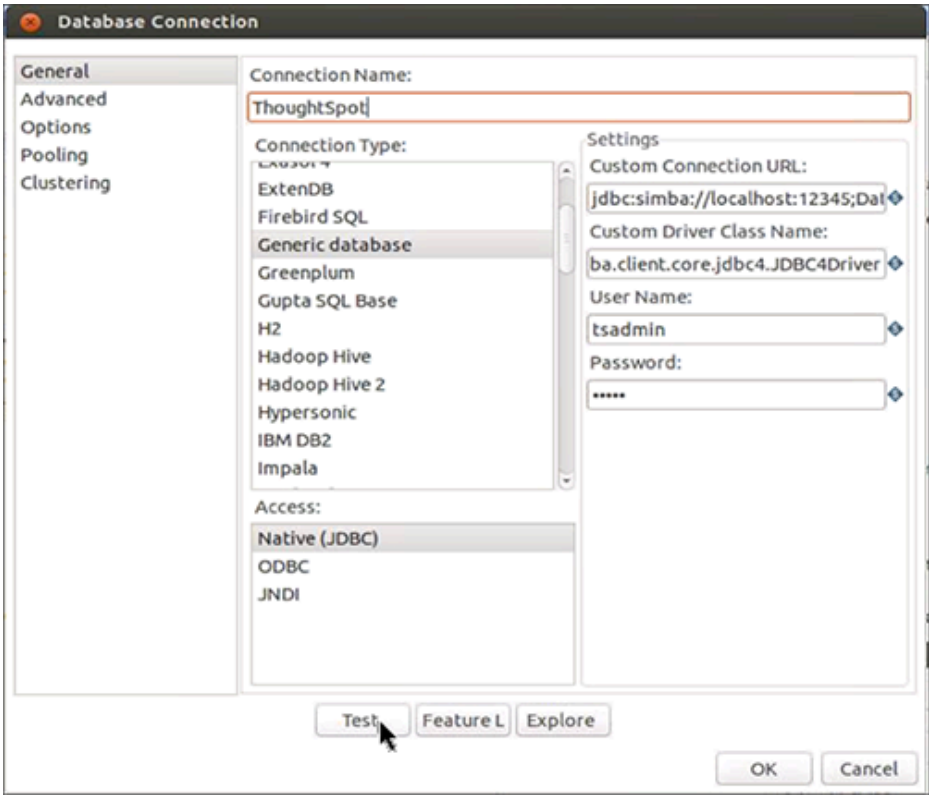
5. Click **New** to create a new connection.



6. Enter or select the following information in the Database Connection dialog:

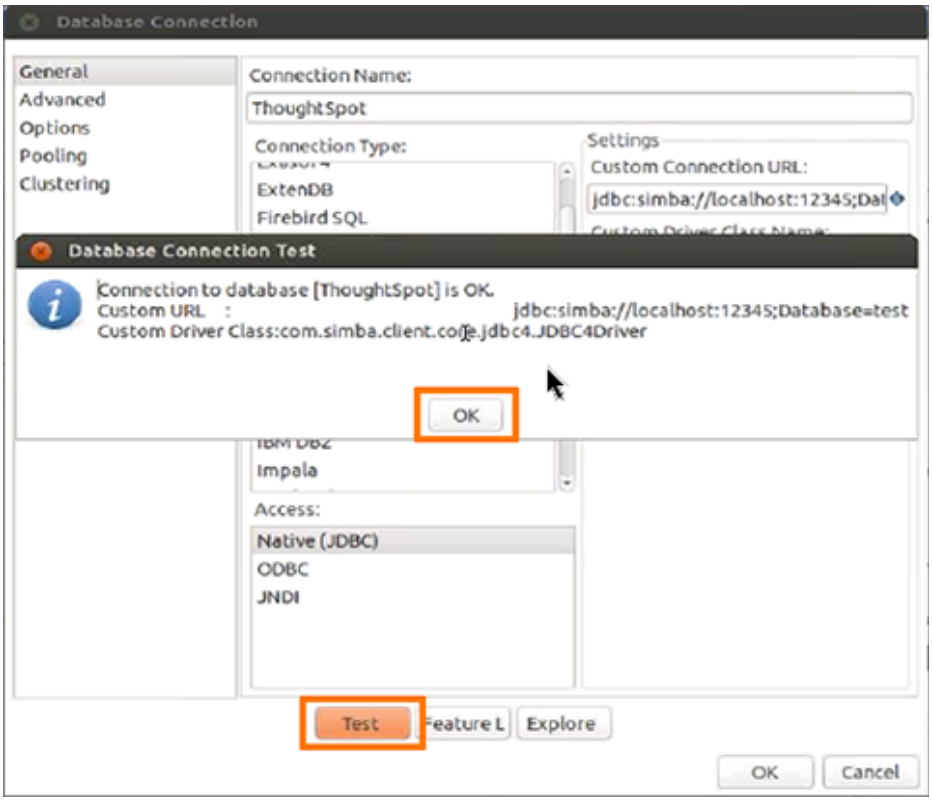
Field	Description
Connection Name	Any string.
Connection Type	Generic database
Access	Native (JDBC)
Custom Connection URL	`jdbc:simba://SERVER_IP:12345;Database=DATABASE_or_SCHEMA_NAME </code>` The IP is a node in your ThoughtSpot cluster. The name or schema of the database you want to connect to. Use TQL to create a database name if needed. Ensure that there are no leading or trailing spaces.

Custom Driver Class Name	`com.simba.client.core.jdbc4.JDBC4Driver` Ensure that there are no leading or trailing spaces.
User Name	A ThoughtSpot username. If you leave this empty, you are prompted for it at connection time. This user should have **Data Management** privileges on ThoughtSpot.
Password	The password for the **User Name**. If you leave this empty, you are prompted for it at connection time.



7. Click **Test** to test your database connection.

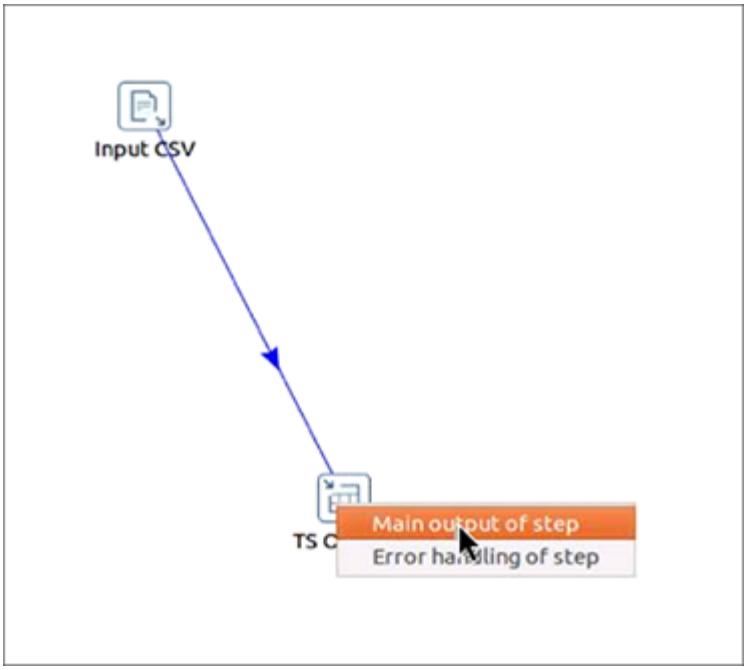
8. If you are able to make a successful connection to the ThoughtSpot Simba Server, click **OK**.



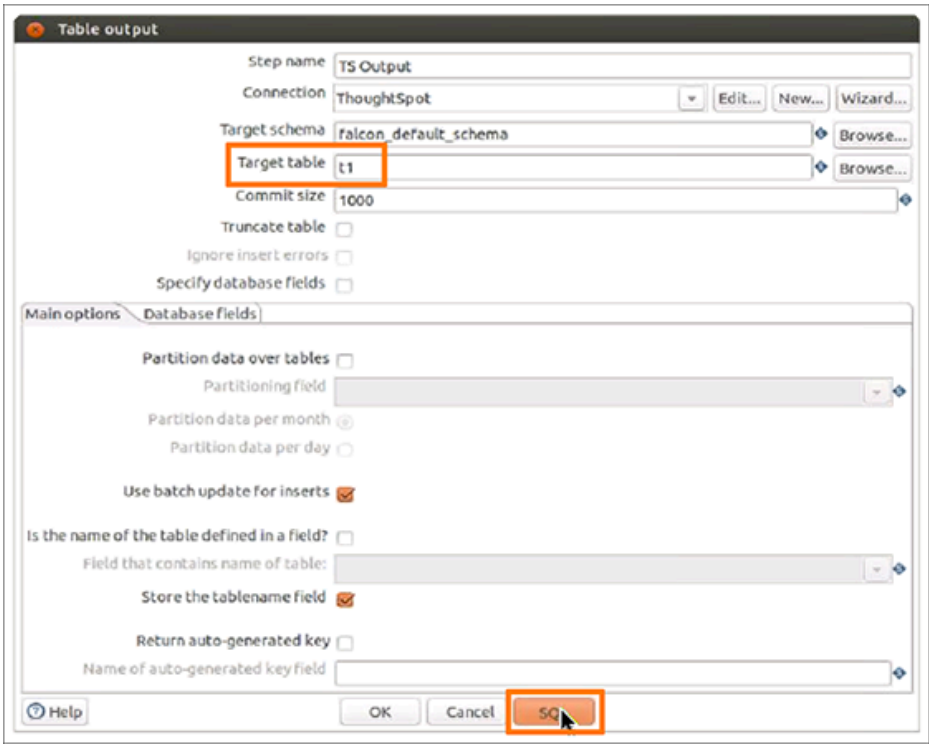
9. Click **OK** in the Database Connection dialog to create the new connection.

Import data

1. In the **Table output** dialog, select the connection you just created.
2. Click **Browse** next to the **Target schema** field and select your **Target schema**.
3. Click **OK** when you are done.
4. Connect the **Input CSV** icon to the **Table output** icon by clicking and dragging an arrow.
5. When prompted, choose **Main output of step**.

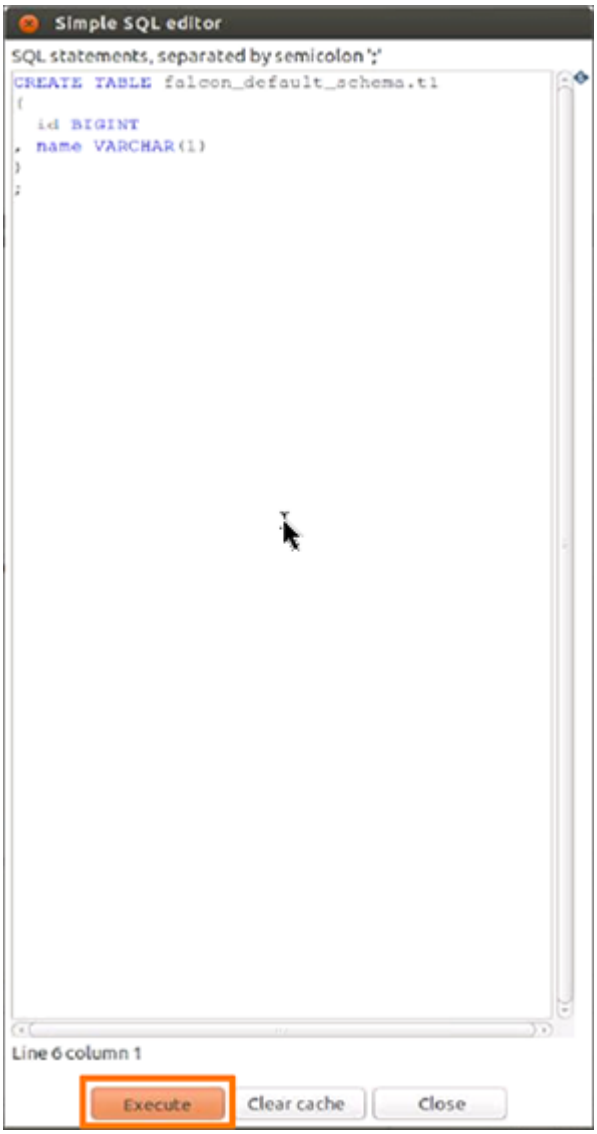


6. Double click the **Table output** icon to reopen the **Table output** dialog.
7. Enter a **Target table name**.
8. Click **SQL**.



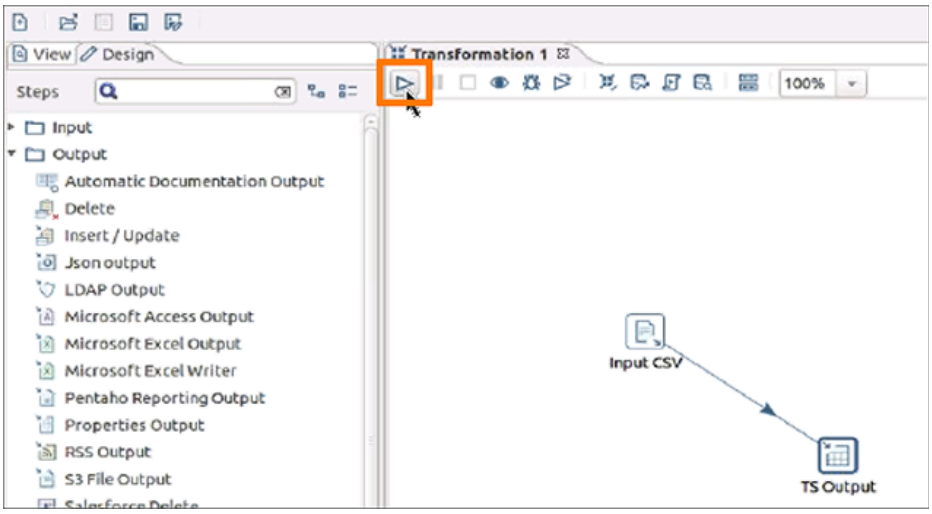
9. In the **Simple SQL editor** dialog, click **Execute**.

   The system processes and then displays the results of the SQL statements.

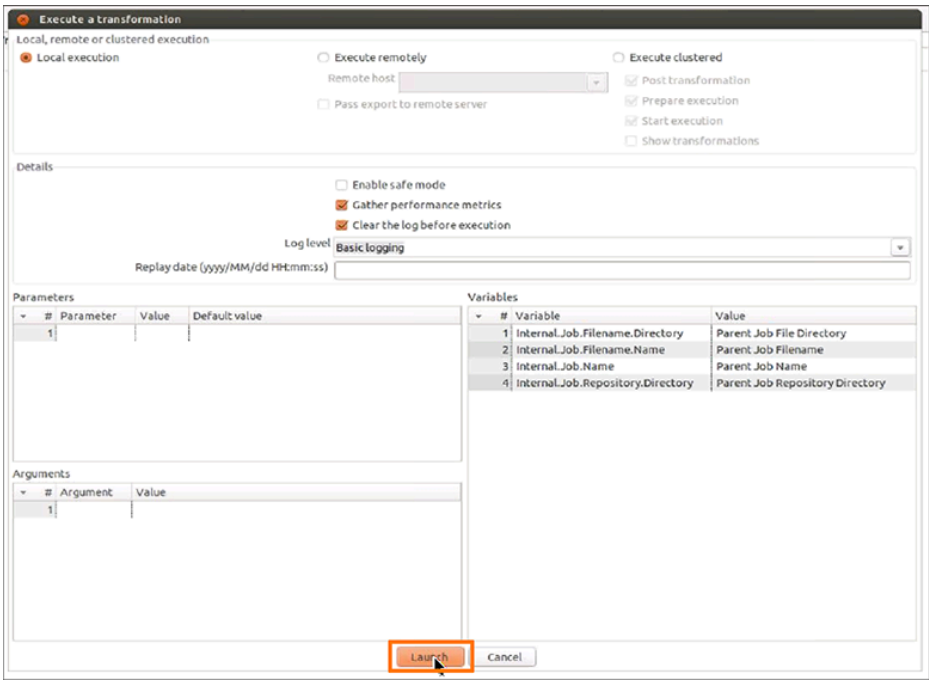


10. Close all open dialogs.

11. Click the **Play** button at the top of the **Transformation** window to execute the transformation.

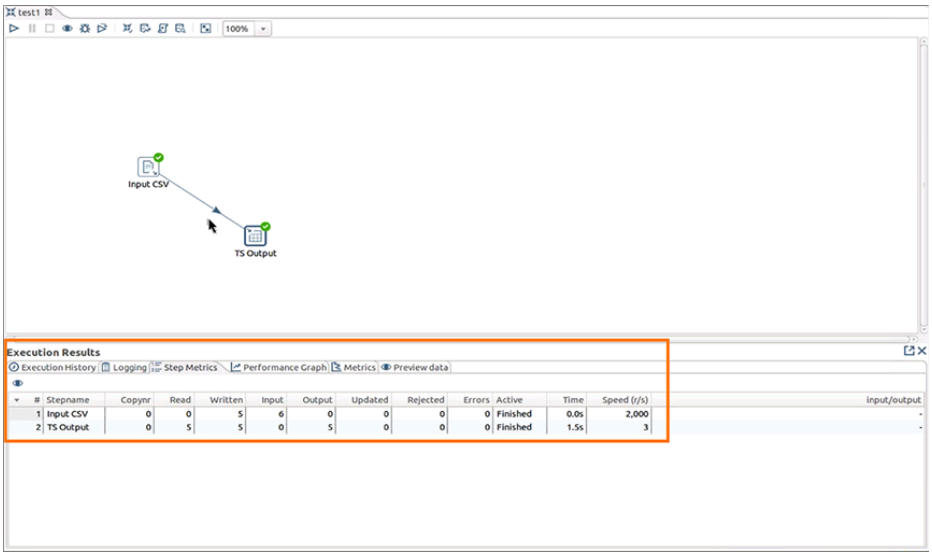


12. Click **Launch** in the **Execute a transformation** dialog.



The system prompts you to save it if you have not already.

13. View the **Execution Results**.


test1
100%
Input CSV
TS Output
Execution Results
Execution History
Logging
Step Metrics
Performance Graph
Metrics
Preview data
Stepname Copynr Read Written Input Output Updated Rejected Errors Active Time Speed (r/s) input/output
1 Input CSV 0 0 5 6 0 0 0 0 Finished 0.0s 2,000 -
2 TS Output 0 5 5 0 5 0 0 0 Finished 1.5s 3 -

Troubleshooting Data Integrations

Summary: Learn how to fix connection issues.

This section can help if you're having trouble creating a connection or need to find out more information about what is going on with ODBC or JDBC.

The information contained here is very basic, and mostly about how to enable logs on the client side. If you need more detailed troubleshooting information or help, please contact ThoughtSpot Support.

- **Enable ODBC Logs**

  If you need more information in order to troubleshoot ODBC connections, you can enable logging for ODBC.
- **Enable JDBC Logs**

  To enable logging for JDBC, add the logging parameters to the connect string. Logs are stored on ThoughtSpot.
- **Schema not found error with ODBC**

  When connecting with ODBC, you need to specify both the database and schema to connect to. If no schema is supplied, you will get an error indicating that the schema could not be found.
- **How to improve throughput of the load**

  The transaction/commit size value can improve the throughput of the load when setting up the ODBC Driver.
- **ODBC tracing on Windows**

  Using logs to aid in troubleshooting.

Enable ODBC Logs

Summary: Learn how to troubleshoot ODBC connections.

If you need more information in order to troubleshoot ODBC connections, you can enable logging for ODBC on the workstation you use for connecting to ThoughtSpot. There are two points where you can enable logging:

- the workstation where you run your ETL activities
- the server where the Simba service is running

On both workstation and servers, the verbosity of the log is controlled by the `LogLevel` property. This property can be one of the following:

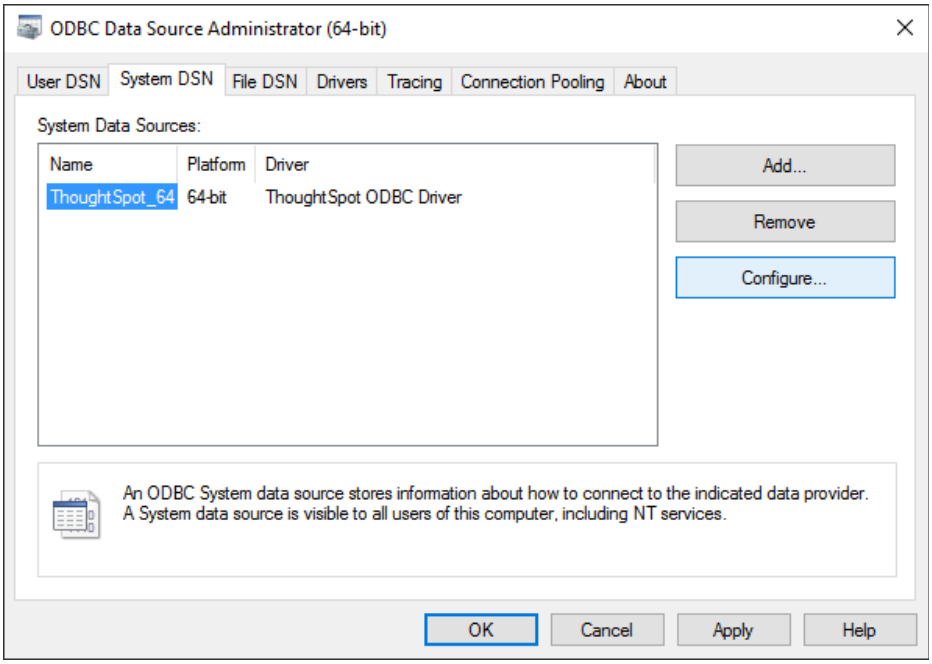
- `0` or `LOG_OFF`: no logging occurs
- `1` or `LOG_FATAL`: only log fatal errors
- `2` or `LOG_ERROR`: log all errors
- `3` or `LOG_WARNING`: log all errors and warnings
- `4` or `LOG_INFO`: log all errors, warnings, and informational messages
- `5` or `LOG_DEBUG`: log method entry and exit points and parameter values for debugging
- `6` or `LOG_TRACE`: log all method entry points

Larger values include the information from lessor values. For example, if you set `3` or `LOG_WARNING`, you log all warnings *and* all errors.

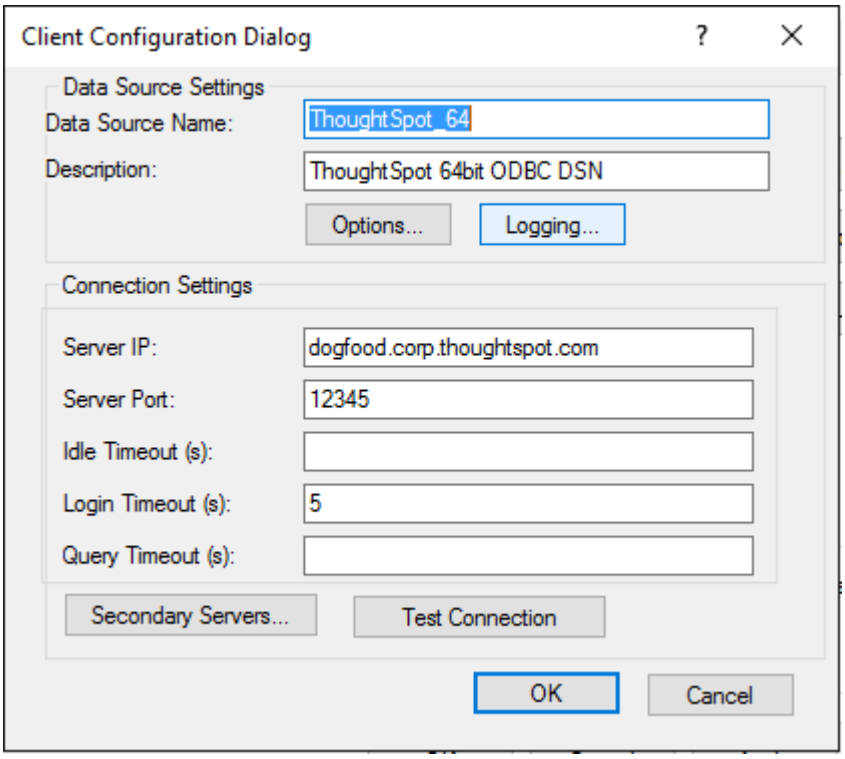
Enable ODBC logs on a Windows workstation

To enable ODBC logs on Windows:

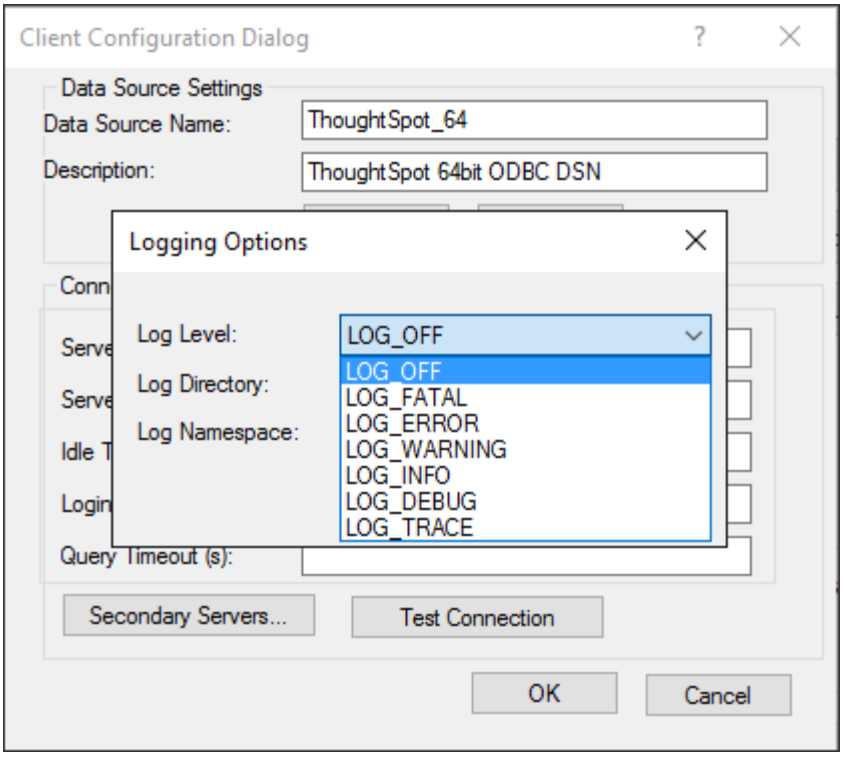
1. Open the **ODBC Data Source Administrator** and select the **System DSN** tab.
2. Select your ThoughtSpot data source and click **Configure**.



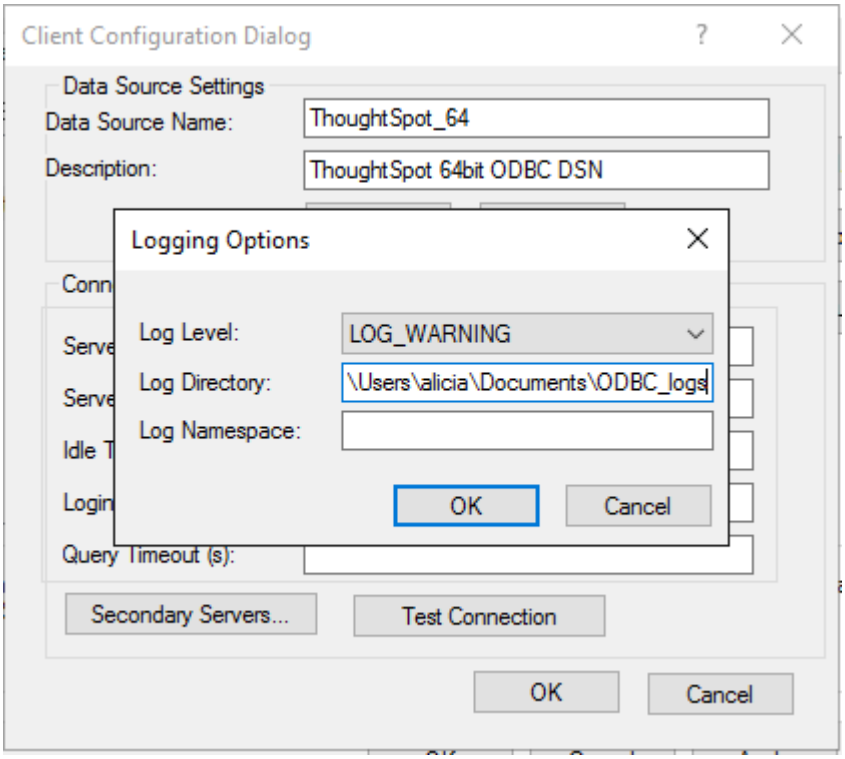
3. In the Client Configuration Dialog, click **Logging**.



4. Choose a **Log Level**, depending on what level of verbosity you want to show in the logs.



5. For **Log Directory:**, type in the fully qualified path where you want the logs to be saved.



6. Click **OK** to save your settings, and **OK** again, to dismiss the ODBC Data Source Administrator.
7. Run the ODBC load.
8. Locate the log file that was generated, and send it to ThoughtSpot Support with a description of the problem.

Enable ODBC logs on a Linux workstation

To enable logging on Linux, follow these instructions:

1. Navigate to the directory where you installed ODBC.
2. Open the `odbc.ini` file in a text editor.

   This file is the registry and configuration file for ODBC.
3. Locate the `LogLevel` and `LogPath` properties.
4. Uncomment the properties.
5. Enter a value for the `LogLevel` the.

   Acceptable values are from 1 to 6 with 6 being the most verbose.
6. Enter the fully qualified path for the `LogPath` values.

   The log will be written here. Your file will look similar to the following: Example for Linux 64-bit:

```
[ThoughtSpot]
Description = ThoughtSpot 64-bit ODBC Driver
Driver = ThoughtSpot
ServerList =  172.18.231.17 12345
Locale = en-US
ErrorMessagesPath = /home/admin/linux/ErrorMessages
UseSsl = 0
#SSLCertFile = # Set the SSL certificate file path. The
certificate file can be obtained by extracting the SDK t
arball
LogLevel = 3 # Set log level to enable debug logging
LogPath = /home/admin/odbc-logs # Set the debug log file
s path
DATABASE = # Set the default database to connect to
SCHEMA = # Set the default schema to connect to
```

7. Save and close the file.
8. To test the configuration, run the ODBC load and review the log files.

Control logs from the Simba server

You may want to collect logs from the Simba service. Do the following to procedure on every ThoughtSpot node running the Simba service.

1. SSH into the ThoughtSpot node.
2. Edit the `/etc/thoughtspot/linux.ini` file.

```
 ...
 [Driver]

 ## Note that this default DriverManagerEncoding of UT
F-32 is for iODBC. unixODBC uses UTF-16 by default.
 ## If unixODBC was compiled with -DSQL_WCHART_CONVERT,
then UTF-32 is the correct value.
 ## Execute 'odbc_config --cflags' to determine if you n
eed UTF-32 or UTF-16 on unixODBC
 DriverManagerEncoding=UTF-32
 DriverLocale=en-US
 ErrorMessagesPath=/usr/home/linux/ErrorMessages/
 LogLevel=0
 LogNamespace=
 LogPath=

 ....
```

3. Uncomment the `LogLevel` setting.

   The `LogLevel` is the level of logging to capture (0-6).

4. Set `LogPath` to a directory to save the logs.

   The `LogPath` is the fully qualified path where ThoughtSpot should write the logs.

5. Work with ThoughtSpot Support to restart the Simba serivce.

   The node IP may change because of the restart. If this happens, repeat the entire procedure.

Enable JDBC Logs

Summary: Configure logging parameter strings.

To enable logging for JDBC, add the logging parameters to the connect string. Logs are stored on ThoughtSpot. Before enabling JDBC logging, you need:

- The level of logging you want to capture.
- The path on the ThoughtSpot server where the logs will be written. Make sure the directory has the correct permissions so that the “admin” Linux user can write logs to it.

To enable JDBC logging:

1. When forming the connect string for JDBC, add these two parameter, separated by “&”:

   For example:

   ```
    jdbc:simba://192.168.2.248:12345;SERVERS=192.168.2.24
   9:12345,
    192.168.2.247:12345;Database=test;Schema=falcon_defaul
   t_schema;**LogLevel=3;LogPath=/usr/local/scaligent/log
   s**
   ```

   The `LogLevel` is the level of logging to capture (0-6). The `LogPath` is the fully qualified path where logs will be written on ThoughtSpot.

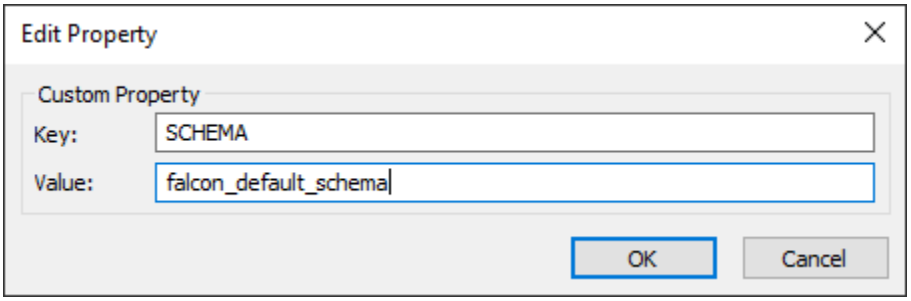
2. Run the JDBC code that uses the connection you modified.
3. Check the `LogPath` directory for logs generated by JDBC.

Schema not found error with ODBC

Summary: Correct schema not found errors.

When connecting with ODBC, you need to specify both the `DATABASE` and `SCHEMA` parameters. This is true even if you do not use schema names in ThoughtSpot. If you don't supply a `SCHEMA`, you get an error indicating that the schema could not be found.

The default schema name in ThoughtSpot is `falcon_default_schema`. To set the `SCHEMA` on Windows, adding a custom property with the key `SCHEMA` and the value `falcon_default_schema`.



On Linux, you can edit the properties in the `odbc.ini` file for the driver you are using:

```
[ThoughtSpot]
Description = ThoughtSpot 64-bit ODBC Driver
Driver = ThoughtSpot
ServerList =  172.18.231.17 12345
Locale = en-US
ErrorMessagesPath = /home/admin/linux/ErrorMessages
UseSsl = 0
#SSLCertFile = # Set the SSL certificate file path. The certifi
cate file can be obtained by extracting the SDK tarball
#LogLevel = 0 # Set log level to enable debug logging
#LogPath = # Set the debug log files path
DATABASE = # Set the default database to connect to
SCHEMA = # Set the default schema to connect to
```

Related information

- Configuring ODBC on Windows
- Configuring ODBC on LINUX
- ODBC and JDBC configuration properties

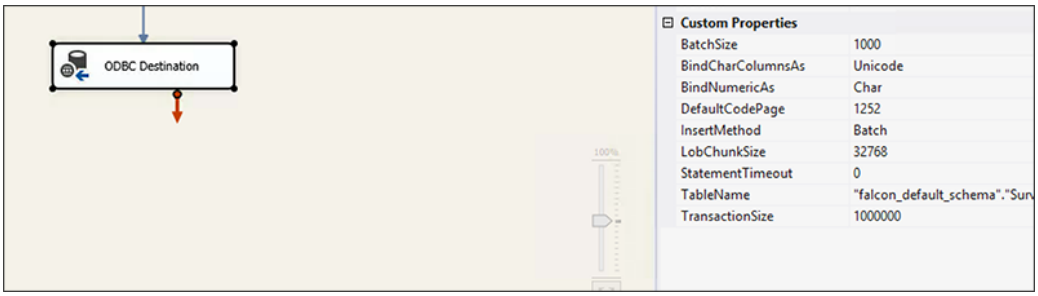
How to improve throughput

> **Summary:** Adjusting the transaction size may correct poor performance and low throughput.

The transaction/commit size value can improve the throughput of the load when setting up the ODBC Driver.

Adjusting the transaction size may correct poor performance and low throughput issues. The transaction size should be set to match the total number of rows that are expected to be loaded in the load cycle. However, increasing this value even higher should help improve throughput of the load.

> ❗ **Warning:** A high transaction size may slow down the ThoughtSpot system.



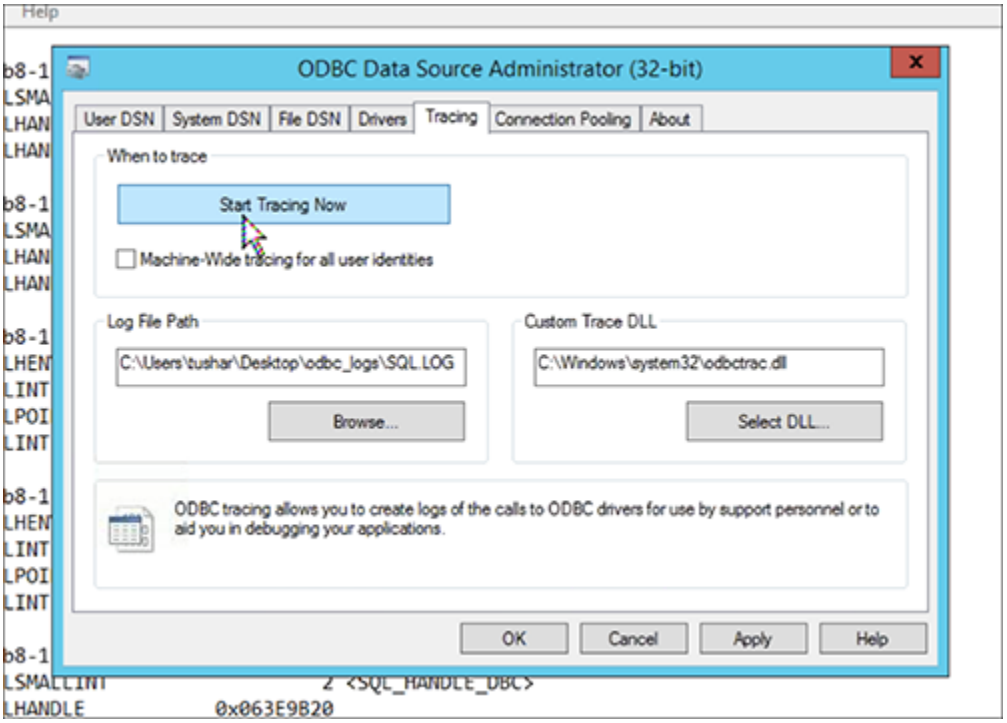
This is where the transaction size field exists for SSIS. Clicking on the ODBC destination reveals the properties on the right hand side, where the **Transaction Size** can be found.

See Set up the ODBC Driver for SSIS for more details on setting the transaction size.

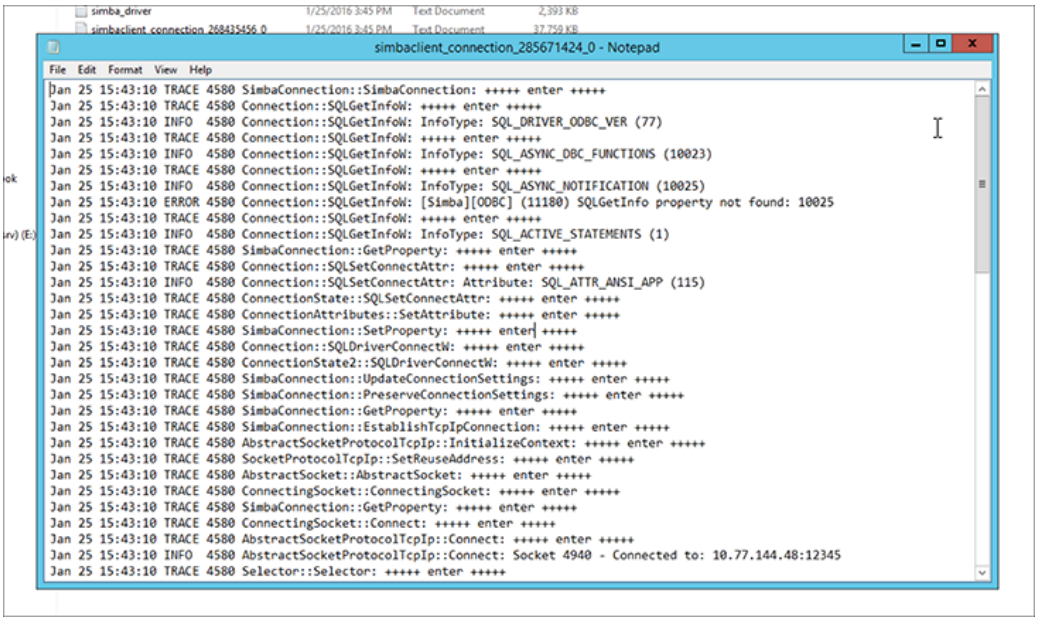
ODBC tracing on Windows

Summary: Using logs to aid in troubleshooting.

Windows shows ODBC specific tracing in the ODBC Data Source Administrator Tracing tab. You can start tracing there by clicking **Start Tracing Now**. This logs every ODBC call from this system, and prints the input and output for the call.



Although this is lower level information, it can still be helpful in troubleshooting. When you are not sure if it is our driver or the tool causing an issue, doing this trace will help narrow the inquiry.



If you start or stop tracing, make sure you do not have the SSIS client open. Close it, change the trace, and reopen.

Supported SQL commands

> **Summary:** The ThoughtSpot connection drivers support a limited set of SQL commands.

The ODBC and JDBC drivers support a limited set of SQL commands. When developing software that uses a ThoughtSpot ODBC driver, use this reference of supported commands. This reference is intended for developers using other tools (ETL, etc.) to connect to ThoughtSpot through the ODBC or JDBC driver.

> **Note:** ThoughtSpot displays `VARCHAR` fields using lower case, regardless of what the original casing of your loaded data is.

ODBC

These SQL commands are supported for ODBC:

- `CREATE TABLE`

  Creates a table with the specified column definitions and constraints. The table is replicated on each node.

  ```
  CREATE TABLE country_dim (id_number int, country varch
  ar, CONSTRAINT PRIMARY KEY (id_number));
  ```

- `INSERT`

  Creates placeholders in the table to receive the data.

  ```
  INSERT INTO TABLE country_dim (?, ?);
  ```

- `DELETE FROM <table>`

Deletes `ALL` rows from the specified table. Use the `WHERE` clause to specify only certain rows to be deleted. Example: You could remove all data for sales before a certain date to free up space in ThoughtSpot.

```
  DELETE FROM country_dim;
```

- `SELECT <cols_or_expression> FROM <table_list> [WHERE <predicates>] [GROUP BY <expressions>] [ORDER BY <expressions>]`

  Fetches the specified set of table data.

```
  SELECT id_number, country FROM country_dim WHERE id_nu
mber > 200;
```

JDBC

`TRUNCATE` is not supported. Instead, use `DELETE FROM TABLE` which is functionally equivalent to “truncate table” in terms of table compression and so forth.

Connection configuration

Summary: Lists the properties you can set for ODBC or JDBC connections

This section lists the properties you can set for ODBC or JDBC connections.

Setting Properties for ODBC

Not all the parameters Simba accepts are supported by the ThoughtSpot ODBC clients, and ThoughtSpot has added some properties, which are listed separately here. All configuration properties use the type String (text).

You can set these properties on Windows by using the ODBC Administrator client. For Linux, the properties are located in three files, depending on the property type:

Property Type	Location
DSN	`odbc.ini` file
Driver	`odbsinst.ini` file
SimbaSetting Reader	`simbaclient.ini` file

Setting Properties for JDBC

For JDBC, these properties are passed as key value pairs in the connect string. For more information, see Use the JDBC Driver.

Properties Reference

The following tables summarize the configuration properties.

Property	Type	Description
DATABASE	DSN or Driver	The default database to connect to.
SCHEMA	DSN or Driver	The default schema to connect to.
Description	DSN	A brief, human-readable description of the DSN. This describes the DSN to users who are deciding which DSN to use.
Driver	DSN or Driver	In the driver configuration location, Driver should contain the path to the driver binary. In the DSN configuration location, Driver could contain the path to the driver binary, or it could contain the driver entry in the registry.
IdleTimeout	DSN	The time to wait for a response from the server, in seconds. This property is optional, but SimbaClient will wait indefinitely for SimbaServer to respond to a request made to the server unless you specify a timeout period. IdleTimeout specifies how many seconds that SimbaClient will wait before aborting the attempt and returning to the application with an error. This timeout corresponds to ODBC's CONNECTION_TIMEOUT property and is only used when more specific timeouts, such as QUERY_TIMEOUT or LOGIN_TIMEOUT aren't applicable.
Locale	DSN	The connection locale. If this value is set, it overrides the driver-wide locale. For example, the driver-wide locale could be en-US. If the client would prefer fr-CA, it can set the connection locale to fr-CA. Values are composed of a 2-letter language code (in lower case), and an optional 2-letter country code (in upper case). If the country code is specified, it must be separated from the language code by a hyphen (-).
LoginTimeout	DSN	The timeout, in seconds, to wait for a response from the server when attempting to log in. A value of 0 means no timeout. The default value is 60.
QueryTimeout	DSN	The timeout, in seconds, to wait for a response from the server during Prepare, Execute, and ExecuteDirect. A value of 0 means no timeout. The default value is 60.
ServerList	DSN	A comma separated list of all servers (IP address and port number) to connect to. SimbaClient must be able to find SimbaServer on the network. This property enables server discovery. SimbaClient will try to make a network connection to the servers in the order specified until a connection is made.
LogLevel	SimbaSettingReader	Controls the granularity of the messages and events that are logged. With this keyword, you can control the amount of log output by controlling the kinds of events that are logged. Possible values (case sensitive): • 0 or LOG_OFF : no logging occurs • 1 or LOG_FATAL : only log fatal errors • 2 or LOG_ERROR : log all errors • 3 or LOG_WARNING : log all errors and warnings • 4 or LOG_INFO : log all errors, warnings, and informational messages • 5 or LOG_DEBUG : log method entry and exit points and parameter values for debugging • 6 or LOG_TRACE : log all method entry points

Property	Type	Description
`LogPath`	SimbaSetting Reader	Specifies the directory where the log files are created. For example: `LogPath=C:\Simba Technologies\Temp` If this value is not set, the log files are written to the current working directory of the SimbaClient.
`LogFileSize`	SimbaSetting Reader	The size of each log file, in bytes. The default values is 20971520 bytes. When the maximum size of the file is reached, a new file is created.
`LogFileCount`	SimbaSetting Reader	The number of log files to create. When the maximum number of log files has been created, the oldest file will be deleted and a new one created. The default value is 50.
`username`	UID	Part of a user username/password combination. This combination should correspond to a ThoughtSpot application user with permissions appropriate to your ETL requirements. Typically, this user is a user with data management or administrative privileges on the application.
`password`	Password	Part of a user username/password combination. This combination should correspond to a ThoughtSpot application user with permissions appropriate to your ETL requirements. Typically, this user is a user with data management or administrative privileges on the application.