ThoughtSpot

ThoughtSpot Application Integration Guide

Release 6.3

November, 2020

Table of Contents

Introduction to Embedding

Summary: You can embed ThoughtSpot in an external application, website, or portal.

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 [See page 54]
- Embedded Charts and Pinboards [See page 45]
- Data REST API [See page 23]
- Runtime Filters [See page 62]
- Metadata API [See page 86]

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 Pinboards 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 (https://brand.thoughtspot.com/d/Vtg4Zg2mqTbE/brand-guidelines).

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

   - Logos & Favicon [See page 70]
   - Color Palettes [See page 76]
   - Background Color [See page 75]
   - Chart and Table Fonts [See page 71]
   - Footer Text [See page 79]

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

Summary: You may need to log into the Linux shell to administer your ThoughtSpot cluster.

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

Summary: Learn how to log into ThoughtSpot.

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` [See page 6] and `thoughtspot` [See page 6] user can SSH into the cluster. After authenticating, either user can use and and all of the following utilities:

- tscli [See page 0]; `thoughtspot` [See page 6] user cannot use commands that require `sudo` or `root` privileges
- tsload [See page 0]
- tql [See page 0]

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.
   ThoughtSpot provides a default password for both the `admin` and `thoughtspot` shell users. ThoughtSpot recommends that you change the passwords for both these users on first login. Follow your company's policies on password security.

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

81.0.416.53, 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

> **Summary:** You can use ThoughtSpot within your own Web application with the ThoughtSpot 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 [See page 0] 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

81.0.416.53, 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 [See page 0].

About SAML

Summary: Learn how to use SAML to enable SSO.

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 [See page 13], for instructions to configure SAML in ThoughtSpot.
- Configure SAML with CA SiteMinder [See page 17], for configuring SAML specifically with CA SiteMinder.

Configure SAML

Summary: You can configure Security Assertion Markup Language (SAML) using ThoughtSpot's command line interface, tscli.

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 [See page 13], or through the Admin Console [See page 14].

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.

Configure SAML using tscli

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.

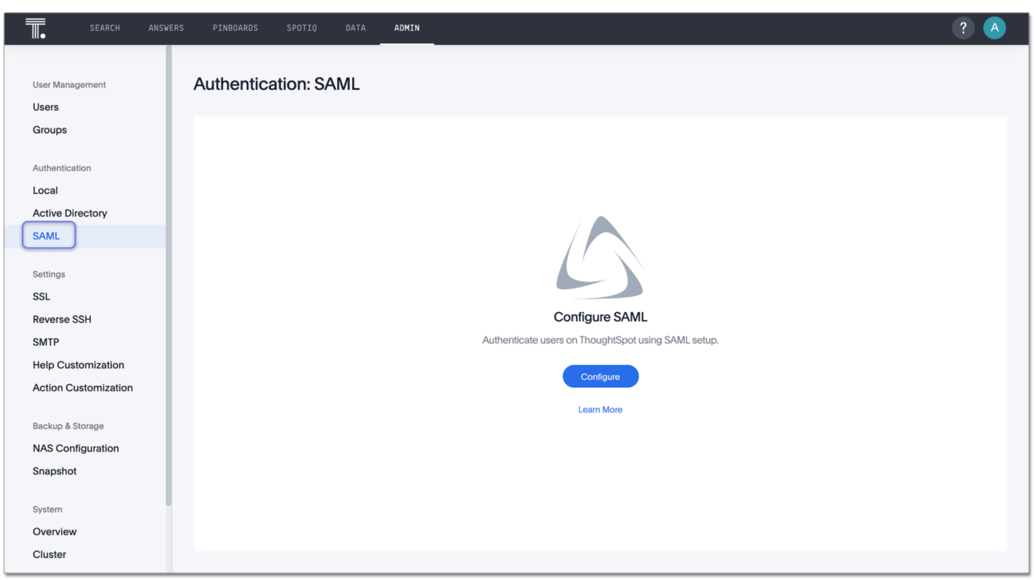
Configure SAML using the Admin Console

You can use ThoughtSpot's integration with SAML for user authentication. By default, local authentication [See page 0] is enabled. You can also configure a SAML integration with an external Identity Provider (IdP), allowing your ThoughtSpot users to log in using one of the supported Identity Providers: Okta, Ping Identity, CA SiteMinder, or ADFS.

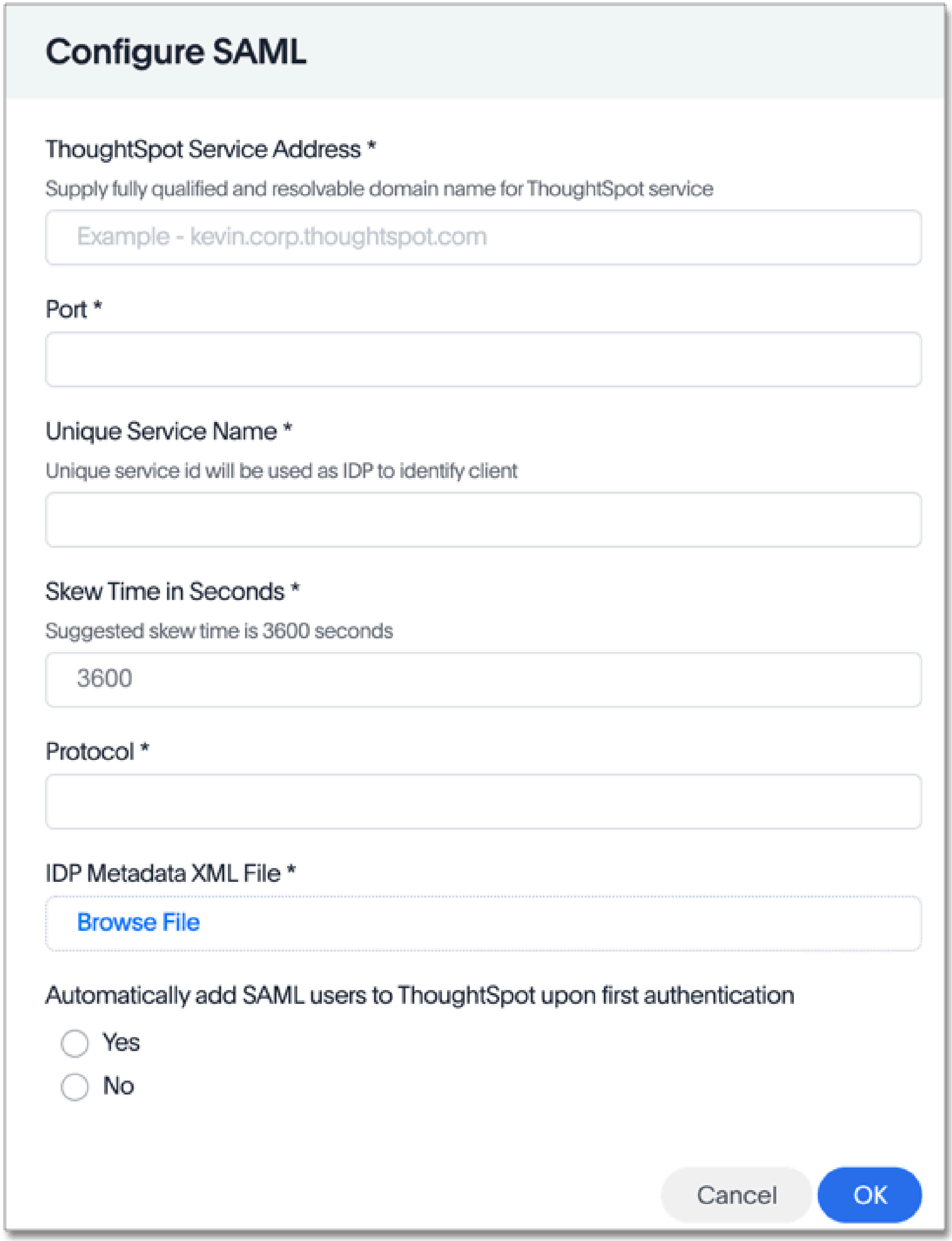
You can configure the SAML integration through the Admin Console.

> **Note:** If you configure authentication through SAML, you cannot also configure authentication through Active Directory.

Navigate to the Admin Console by clicking on the **Admin** tab from the top navigation bar. Select **SAML** from the side navigation bar that appears.



Click the **Configure** button in the middle of the screen.



Fill in the following parameters:

1. **ThoughtSpot Service Address**: A fully qualified and resolvable domain name for the ThoughtSpot service. For example, *thoughtspot.thoughtspot-customer.com*.
2. **Port**: Port of the server where your ThoughtSpot instance is running. For example, port `443`.
3. **Unique Service Name**: The unique key used by your Identity Provider to identify the client. For example, *urn:thoughtspot:callosum:saml*.

4. **Skew Time in Seconds**: The allowed skew time, after which the authentication response is rejected and sent back from the IDP. *86400* is a popular choice. The default is *3600*.
5. **Protocol**: The authentication mechanism for ThoughtSpot. For example, `http` or `https`.
6. **IDP Metadata XML File**: The absolute path to your Identity Provider's metadata file. This file is provided by your IDP. You need this file so that the configuration persists over upgrades. It is a best practice to set it up on persistent/HA storage (NAS volumes) or in the same absolute path on all nodes in the cluster. For example, *idp-meta.xml*.
7. **Automatically add SAML users to ThoughtSpot upon first authentication**: Choose whether or not to add SAML users to ThoughtSpot when they first authenticate. If you choose 'yes', then new users will be automatically created in ThoughtSpot upon first successful SSO login. If you choose 'no', then SAML users will not be added in ThoughtSpot upon first successful SSO login. Instead, you must add users manually [See page 0].

After you fill in all parameters, click **OK**.

> **Note:** ThoughtSpot adds external users, or users that authenticate through SAML or Active Directory, to the **all** group by default. This group has no privileges. You must manually assign users to ThoughtSpot groups to give them privileges, such as **can upload user data**, or **can manage data**.

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 [See page 0]. 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

Summary: Learn how to configure ADFS to work with ThoughtSpot.

You can configure Active Directory Federated Services (ADFS) to work with ThoughtSpot.

Prerequisites

Before you configure ADFS, complete these prerequisites.

1. Configure SAML in ThoughtSpot [See page 0].
2. Install ADFS 2.0 (https://www.microsoft.com/en-us/download/details.aspx?id=10909).

   ThoughtSpot supports ADFS 3.0, but recommends ADFS 2.0.
3. Make sure you can run ADFS 2.0 Federation Server Configuration Wizard from the ADFS 2.0 Management Console.
4. Make sure that the 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` in the command line on both machines, supplying the DNS of the other server. The syntax is `nslookup [HOST] [SERVER]`. Replace HOST and SERVER with your specific information.

   ```
   $ nslookup [HOST] [SERVER]
   ```

ADFS 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).

Step 1: Initialize IdP metadata

1. **Download the IdP metadata** Download the ADFS 2.0 IdP metadata from the ADFS server. You can reference this file by its URL:

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

2. **SSH into your cluster** Log into the Linux shell using SSH.

```
$ ssh admin@<cluster-IP>
```

3. **Change directories to the SAML directory**

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

4. **Update the metadata** 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. **Restart Tomcat** Contact ThoughtSpot Support [See page 0] for help restarting ThoughtSpot's Tomcat instance.

Step 2: Initialize the Service Provider metadata

1: Import metadata

1. Open the ADFS 2.0 Management Console.
2. Select **Relying Party Trusts > 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**.
   - There may be a warning that some of the content of the metadata is not supported. You can safely ignore this warning.

2: Edit claim rules

1. In the **Ready to Add Trust** section, make sure that the tab endpoints contain multiple endpoint values.
   - If not, verify that your metadata was generated with the HTTPS protocol URLs.
2. Ensure that the **Open the Edit Claim Rules dialog** checkbox is checked.
3. Click **Next**.
4. Select **Add Rule**.

5. Choose **Send LDAP Attributes as Claims**.
6. Click **Next**.
7. For **NameID** enter the name of your claim rule.
8. For **Attribute store**, choose “Active Directory”.
9. For **LDAP Attribute**, choose “SAM-Account-Name”.
10. For **Outgoing claim type**, choose “Name ID”.
    - If you are using ADFS 3.0, you might need to configure the Name ID as a Pass Through claim.
11. Finish the wizard.
12. Confirm that all information is correct in the claim rules window.

3. Specify your level of security

1. Open the provider by double-clicking it.
2. Select the **Advanced** tab.
3. Under **Secure hash algorithm**, choose “SHA-256”.
    - If you have trouble with SHA-256, try SHA-1 instead.

You can now use ADFS.

Step 3: Test your ADFS integration

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

About the Data REST API

Summary: The REST API allows you 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 [See page 9].

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 [See page 50] 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 [See page 35] 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 [See page 81].

Related information

- API Reference guide [See page 81].
- Direct Search-to-Embed API [See page 35].

Calling the Data REST API

> **Summary:** Learn how to call 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/pinboardda
ta?id=7752fa9e-db22-415e-bf34-e082c4bc41c3
```

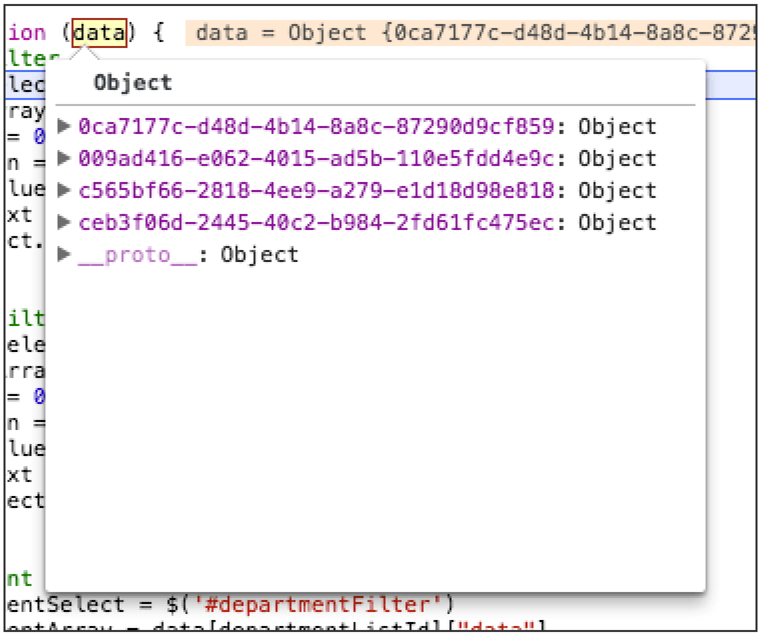
To retrieve data from a specific visualization within a pinboard, you would append the ID number of the visualization using the vizid parameter:

```
https://<thoughtspot_server>/callosum/v1/tspublic/v1/pinboardda
ta?id=7752fa9e-db22-415e-bf34-e082c4bc41c3&vizid=%5B1e99d70f-c1
dc-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.



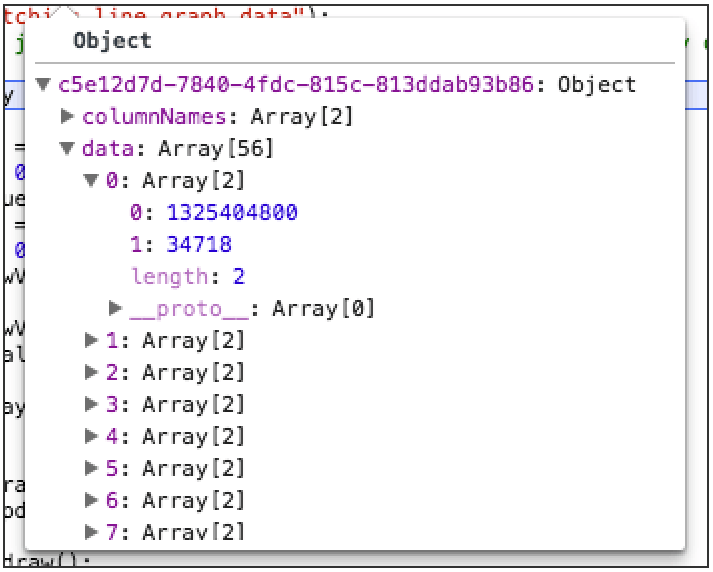
If you make a call to a specific viz on a pinboard, it will return just one object. The JSON object format for the data that is returned from ThoughtSpot is:

```
{
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 [See page 62].

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

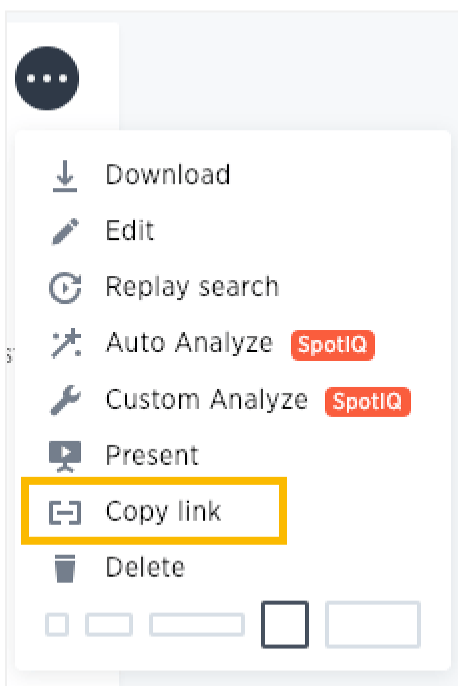
Summary: This procedure shows how to use the REST API to get data out of ThoughtSpot, so you can use it in a Web page, portal, or application.

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

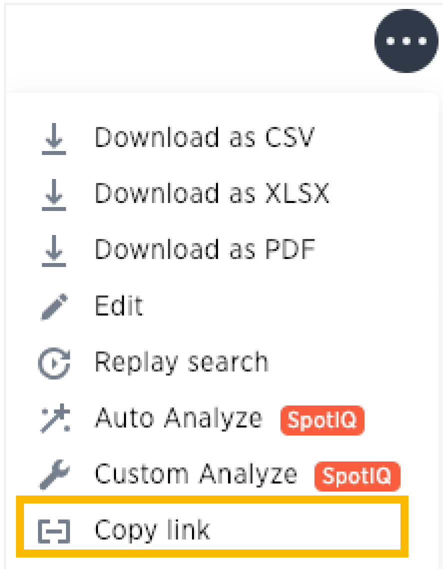
Before you can use the Data REST API, you need to enable the JavaScript API (JS API) [See page 50] and authenticate to ThoughtSpot.

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**.



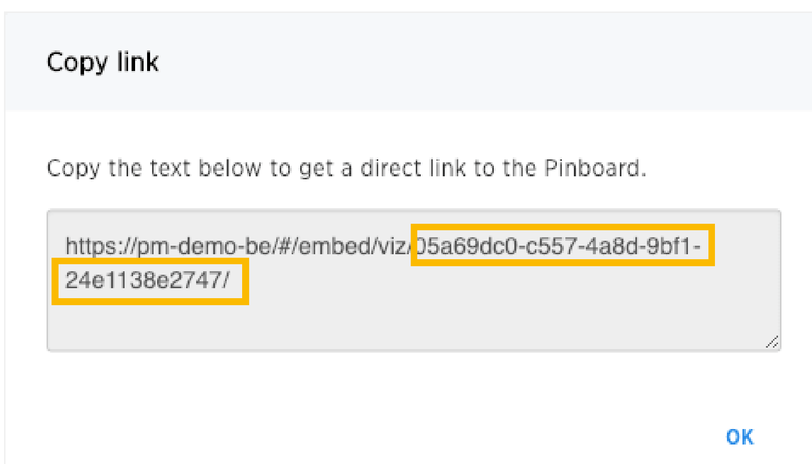
- A visualization, click the **Copy Link** icon in the upper right corner of the table or chart.



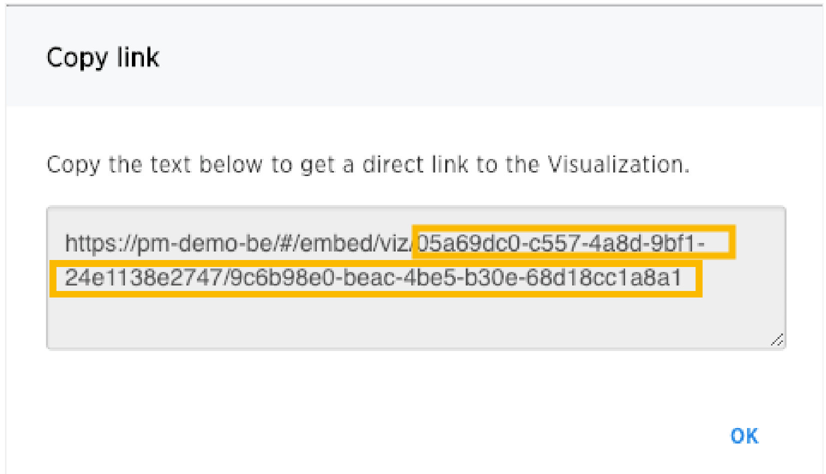
4. Copy the ID number from the link shown. Paste it somewhere so that you can use it later to construct the URL to use when calling the REST API.

   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 [See page 62].

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) [See page 50] on the receiving page of the target application.
2. Authenticate to ThoughtSpot [See page 9] on the receiving page of the target application.
3. Embed the ThoughtSpot application [See page 54] 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> sends 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.

You can use the Data Push API and a custom menu item to initiate an action in another application based on the result of a search in ThoughtSpot.

These custom menu items are visible when you select the more options menu on an Answer, or during a search, if you use the search terms that the external application expects to receive.

About the Data Push API

The Data Push API allows you to open a web page in the context of the ThoughtSpot application. This third party web page then has access to 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.

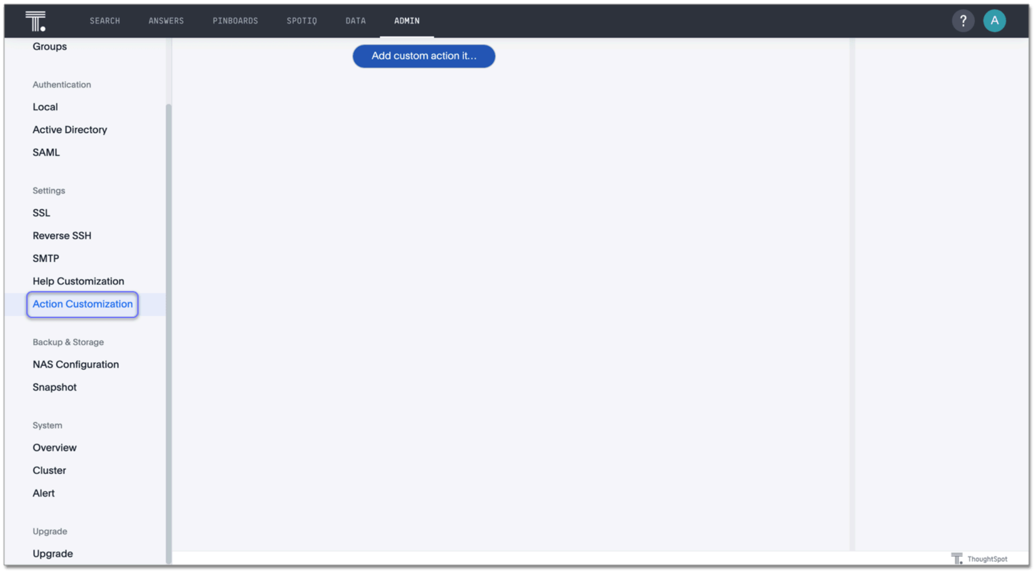
For example, you may search in ThoughtSpot for customers who are coming due for renewal of their contract in the next month. You could then trigger an action that brings up a web page from an external billing system. The billing system could be set up to read the data (list of names, emails, products, 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 turn on the Data Push API functionality, contact ThoughtSpot Support [See page 0].

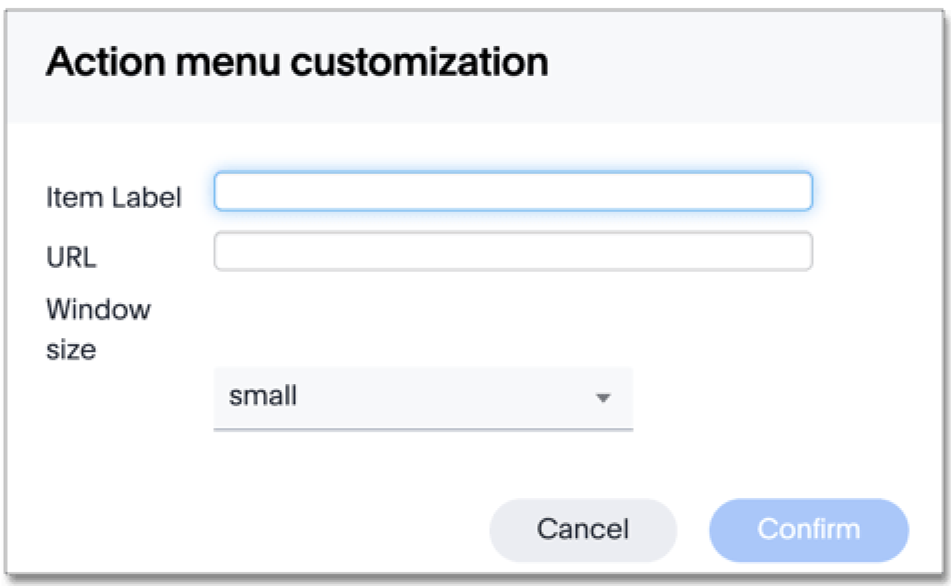
The data is available to the external application in JSON (JavaScript Object Notation) format. You must parse the JSON to get the data values you need using JavaScript in the receiving application.

Create a custom menu item from the Admin Console

To add custom items to the more options menu for Answers, navigate to the Admin Console by clicking on the **Admin** tab from the top navigation bar. Select **Action Customization** from the side navigation bar that appears.



Select **Add custom action item**.



Specify a label for the item, such as *billing renewal email*, and add the URL of the external application. Choose the size of the window you would like to open up in ThoughtSpot when a user clicks on this action, and click **Confirm**.

You can now see your action on ThoughtSpot searches and Answers, if you use the search terms that the external application expects to receive. Your action has a **custom** tag to differentiate it from ThoughtSpot actions.

When a user clicks on your action, the application you specified appears, and completes the actions specified by the Data Push API.

> **Note:** In order for your action to work correctly, you ***must*** use the search terms that the external application expects to receive in your Answer or unsaved ThoughtSpot search.

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 use all or part of ThoughtSpot inside another client application.

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:

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 embedding

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. Consider the possible link types, and how they open:

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>` element

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 be *already logged into ThoughtSpot*, before interacting with the client application. This approach is for testing the client. Do not use this in a production environment.

SAML

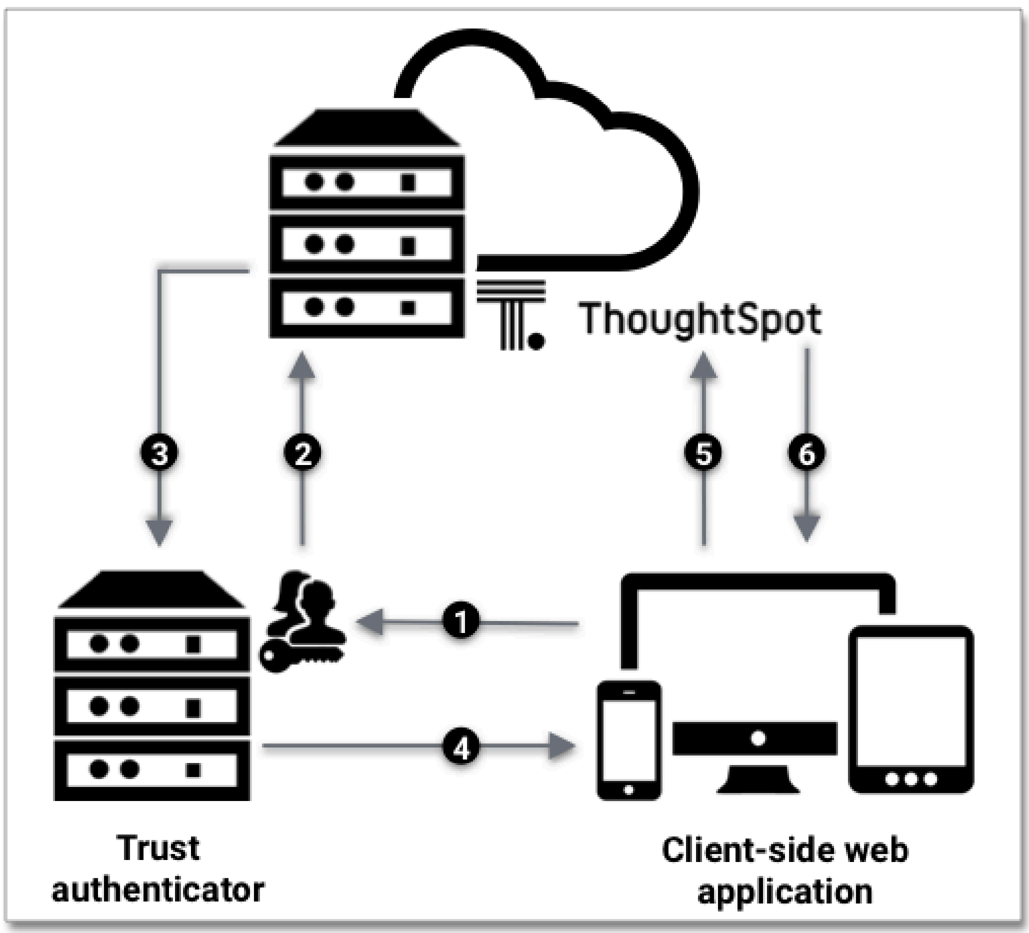
Before you can embed all or part of ThoughtSpot, you must authenticate to ThoughtSpot using SAML with the public REST API call. After authentication, a URL is provided to call the desired visualization, and populate it into an `<iframe>` element.

You must configure SAML [See page 0] 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 who is already logged into client-application interacts with a ThoughtSpot embed point, which initiates the following processes:

1. The client-side application requests a user token from the trusted authenticator.
2. The trusted authenticator requests a 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 application (such as ThoughtSpot) can present security vulnerabilities. Cross-origin or cross-domain verification prevents such security risks.

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 ThoughtSpot Support. 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 [See page 0].

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

> **Summary:** Learn how to embed a visualization or Pinboard in your own Web page.

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

To build this sample, you must have access to a text editor and a ThoughtSpot instance with a visualization. Experience working with Javascript also helps.

Get the link

You can get the link for an entire Pinboard, or single visualization

This procedure assumes the Pinboard with the visualization you want to embed already exists.

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:

   - **Pinboard embedding**: click the **More** menu at the level of the Pinboard, ,

     and select **[-] Copy embed link**.

     The format for the link is:

     ```
     <protocol>:<host>:<port>/#/embed/viz/<pinboardID>
     ```

- **Vizualization embedding**: click the **More** menu on the visual you plan to embed, , and select **[-] Copy embed link**.

  The format for the link is:

  ```
   <protocol>:<host>:<port>/#/embed/viz/<pinboardI
  D>/<vizualizationId>
  ```

Edit the test.html file

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 [See page 0] file.

1. Create an empty directory called `test`.
2. Save the `test.html` file to the `test` directory.
3. Download [See page 0] 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 must 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 in to 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`

   See if your Pinboard or visualization appears in the browser window.

6. Check the browser console.

   On success, the console reports a message similar to the following:

```
 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, and look for errors in the page loading, usually on the **Console** tab.

You may see an error similar to the following:

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

This occurs when the cross domain (CORS) setting is incorrect on your ThoughtSpot cluster. Contact ThoughtSpot Support (https://community.thoughtspot.com/customers/s/contactsupport) for more help.

Authentication flow with embed

> **Summary:** You can enable Single Sign On (SSO) with your embedded version of ThoughtSpot.

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 [See page 0] 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 [See page 45] or use the REST API to get data [See page 32] 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.

Advantages of full embedding

Fully embedding ThoughtSpot content gives your users the ability to perform many essential tasks:

- create Answers and Pinboards
- share objects with other users
- upload data, and refresh uploaded data
- relate uploaded data to existing worksheets

This is useful for supplying the full search experience into an `iframe` with different navigation views and toggle options.

Limitations of full embedding

There are some limitations; users in a fully embedded applications cannot perform the following tasks:

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

Before committing to fully embed the application, review the article Understand embedding [See page 41].

Example of full application embedding

The following sample demonstrates how you can fully embed an 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" allow="fullscreen" src="/?embedAp
p=true#/" height="80%" width="80%"></iframe>
   </body>
</html>
```

The function `updateIframeUrl(id)` contains the logic to change the src URL of the `<iframe>` when your users click navigation buttons.

Showing applications in a full screen

To have an embedded ThoughtSpot application show in a full screen, the `<iframe>` element must have the `allow="fullscreen"` parameter or its analogues.

Browser support for full screen embedding

Note the following support across browsers:

Supported desktop browsers

Chrome, Edge, Firefox, and Opera

Supported mobile browsers

Android webview, Chrome for Android, Firefox for Android, Opera for Android, and Samsung Internet

NOT supported desktop browsers

Internet Explorer, Safari

NOT supported mobile browser

Safari on iOS

Framework support for full screen embedding

Additionally, consider the following framework factors:

allowfullscreen

This attribute is the legacy precursor of `allow="fullscreen"`, and may still work with some browsers.
Set to `true` if the `<iframe>` can activate fullscreen mode by calling the `requestFullscreen()` method.
We strongly recommend that you update your embedding scripts to use the new approach inside the `<iframe>` tag.

Spring MVC

This framework supports the `allowfullscreen="true"` parameter inside the `iframe` tag.

React

This framework is case sensitive, and uses the attribute `allowFullScreen` inside the `iframe` tag.

Hide the ThoughtSpot navigation bar

To hide the primary navigation, configure these:

- Ensure the app is in an `<iframe/>` .
- Set the `embedApp` flag to `true` to specify that the application is embedded.
- Set the `primaryNavHidden` flag to `true` (the default) to specify that navigation visibility is off.

If either flag is `false`, primary navigation appears.

Error messages and full embed

In ThoughtSpot, you can disable error messages within the ThoughtSpot `iframe`. We provide APIs so you can access error messages and display them in your application's UI appropriately.

This approach suppressing error messages inside the `<iframe>`, and uses the `window.postMessage` function to passes them through to the parent application, which acts as the listener.

You can view these 'hidden' messages in the console logs. Contact ThoughtSpot Support 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>`.

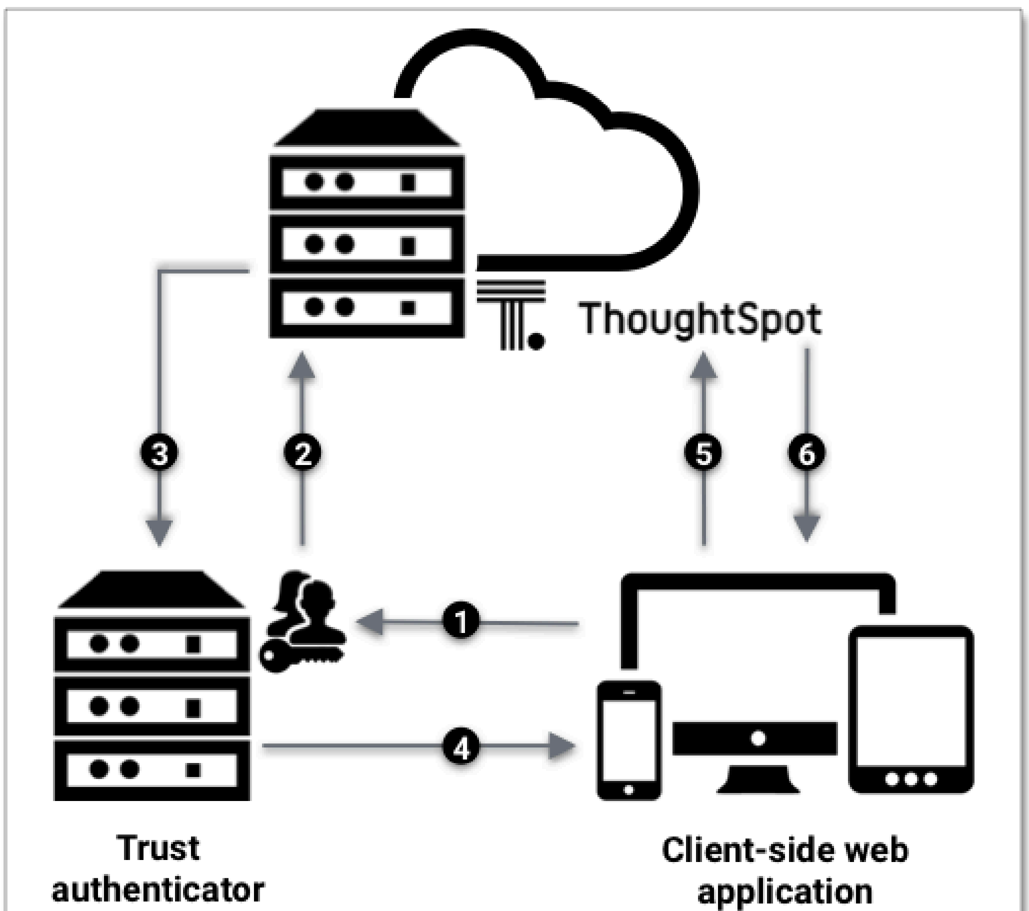
Configure trusted authentication

Summary: Learn how to configure trusted authentication.

If your organization has a trusted authentication server, you can use it 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 that user's behalf. This ensures that the user authentication persists across all subsequent user sessions.

How to authenticate users

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



A user already logged into the 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 must already be 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 and the user token to ThoughtSpot.

6. ThoughtSpot validates the token, and returns the information that the authenticated user requested.

Enable trusted authentication and get a token

Generated tokens do not expire.

1. Log in to 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. Your installation can have only one authentication token at a time. Every time you call to enable overwrite, the existing token and return a new one.

Disable a token

To disable a token and not overwrite it, enter the following command:

```
tscli tokenauthentication disable
```

Trusted authentication call

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

   This is a REST request for an embedded ThoughtSpot object, page, or 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 that 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/session/auth/token
   ```

   This post request takes the following parameters:

   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 a token to the original requestor.

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

   For example, the customer requests a specific object:

```
 GET https://<instance>/callosum/v1/session/login/toke
n?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 re-authenticate requests for each new object.

About Runtime Filters

Summary: Use runtime filters to filter an embedded answer or pinboard.

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 [See page 41] or the REST API [See page 23]. 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`
- `BIGINT`
- `INT`
- `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

Summary: Learn how to 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 [See page 63].

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) [See page 50] and authenticate to ThoughtSpot.
2. Use the Data API [See page 32] or Visualization Embedding [See page 45] 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 [See page 67] 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=<value\>**
    #/pinboard/<pinboard_id>
    ```

    - For Embedding a visualization:

    ```
    http://<thoughtspot_server>:<port>/
    ?**col1=<column_name\>&op1=<operator\>&val1=<value\>**
    #/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

Summary: Runtime filters have several operators you can use to filter your embedded pinboards and visualizations.

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 in the Admin Console 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 not enabled by default. To enable style customization, contact ThoughtSpot Support [See page 0]. The ThoughtSpot logo in the middle of the page is automatically removed when Style Customization is enabled.

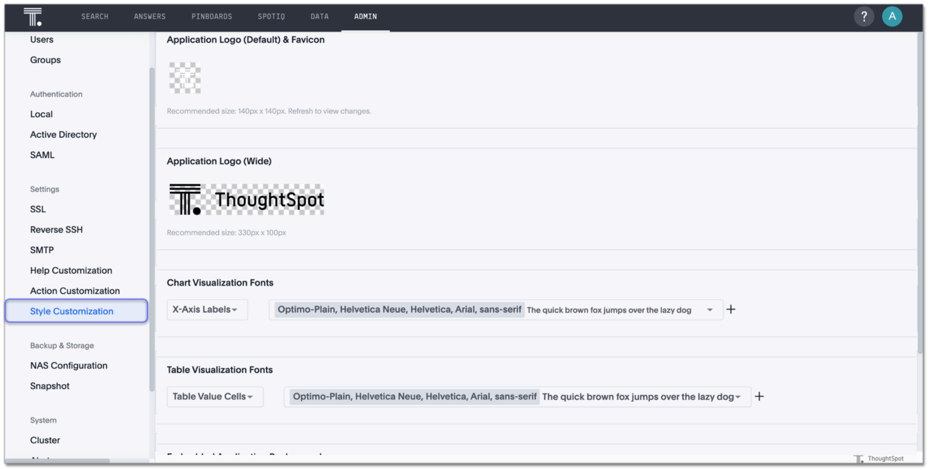
Change style customization

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. Navigate to the Admin Console by clicking on the **Admin** tab from the top navigation bar. Select **Style Customization** from the side navigation bar that appears.



In the **Style Customization** page, you can perform the following actions:

- Upload application logos [See page 70]
- Set chart and table visualization fonts [See page 71]
- Choose a background color [See page 75]
- Select chart color palettes [See page 76]
- Change the footer text [See page 79]

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. Navigate to the Admin Console by clicking on the **Admin** tab from the top navigation bar.

   Select **Style Customization** from the side navigation bar that appears.

   If the Style Customization option is not available, contact ThoughtSpot Support [See page 0] to enable it.

2. 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.

3. Click the wide icon under **Application Logo (Wide)** to browse for and select your own wide logo.

   The recommended size is 330px by 100px. The accepted file formats are jpg, jpeg, and png. This logo appears on the login screen. You may need to test a few versions to make sure it appears correctly.

4. 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

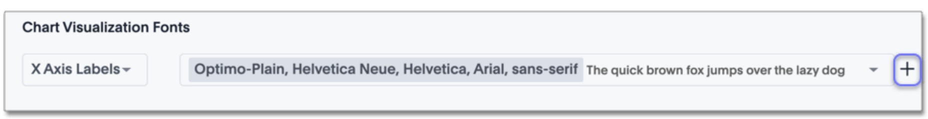
Summary: Learn how to add and edit fonts to customize ThoughtSpot's appearance.

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:

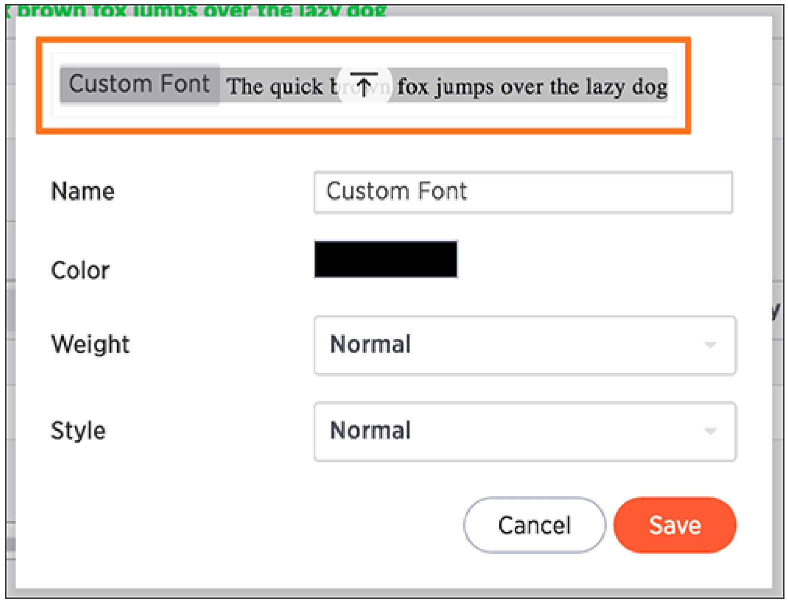
1. Navigate to the Admin Console by clicking on the **Admin** tab from the top navigation bar.
   Select **Style Customization** from the side navigation bar that appears.
   If the Style Customization option is not available, contact ThoughtSpot Support [See page 0]
   to enable it.

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



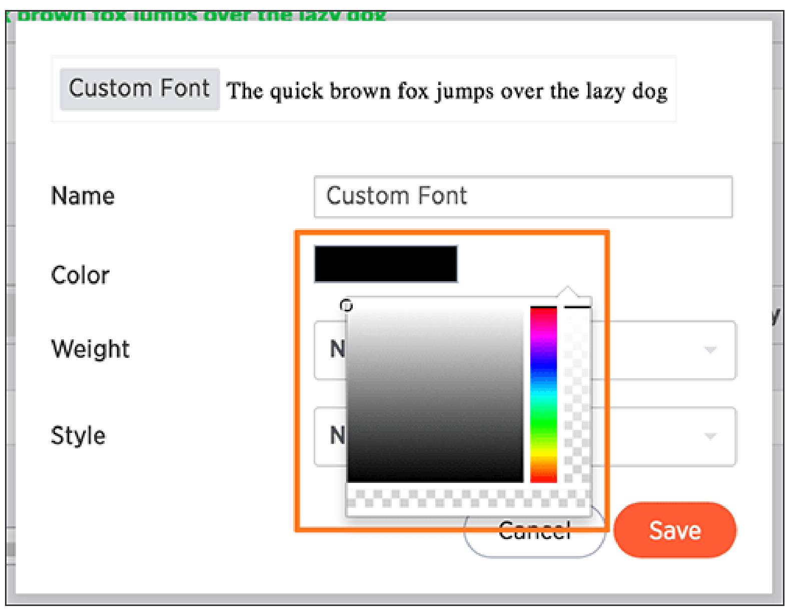
3. In the add new font menu, select the details for the font:

   a. Upload your custom font.

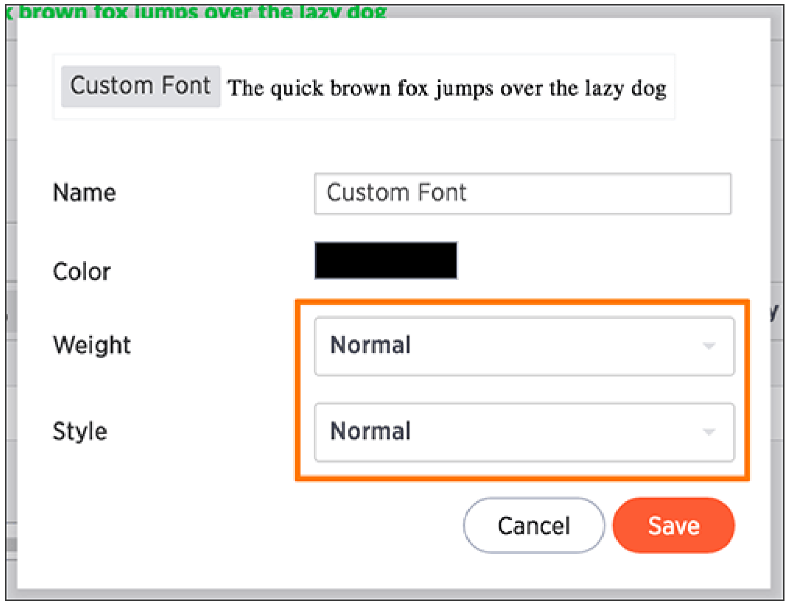


Only WOFF font types are supported.

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.

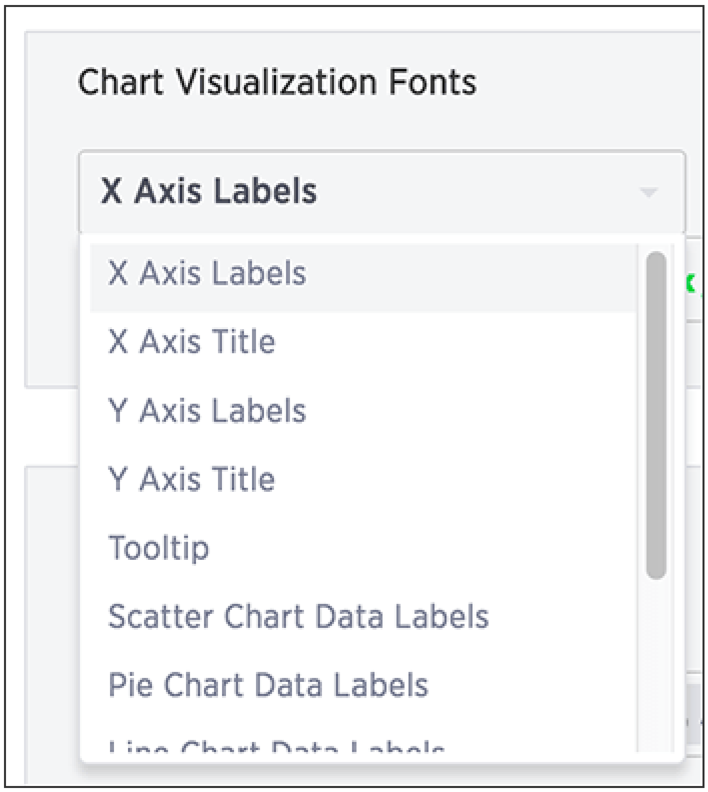
d. Click **Save**.

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

5. Make any changes to the details of the font in the edit menu and click **Save**.

6. Click the custom font drop down to choose your custom font.

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



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

9. 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

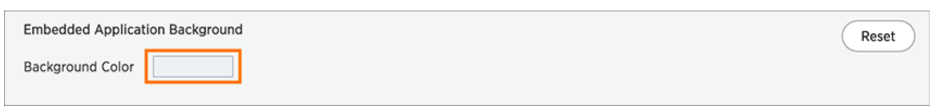
Summary: You can customize ThoughtSpot's background color to match your company's theme.

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.

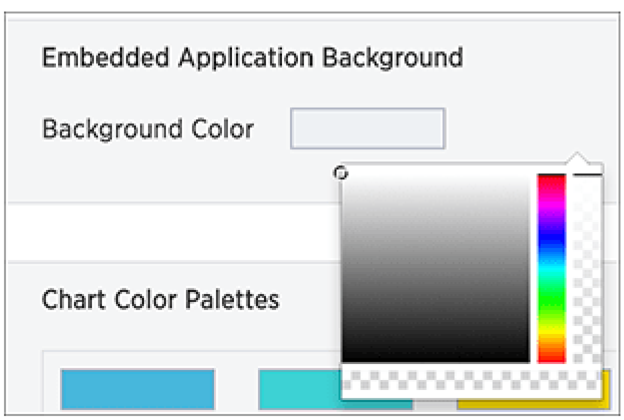
This feature is only applicable when embedding ThoughtSpot in an external web portal or application. To choose a background color:

1. Navigate to the Admin Console by clicking on the **Admin** tab from the top navigation bar.
   Select **Style Customization** from the side navigation bar that appears.
   If the Style Customization option is not available, contact ThoughtSpot Support [See page 0] to enable it.

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



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



4. 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

Summary: 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.

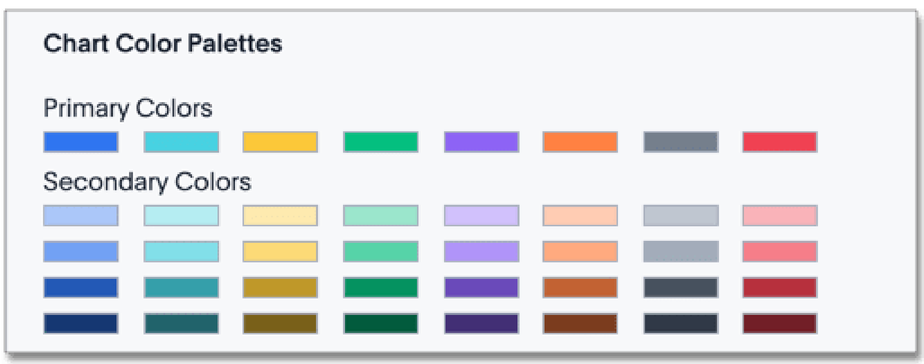
To select the chart color palettes:

1. Navigate to the Admin Console by clicking on the **Admin** tab from the top navigation bar.

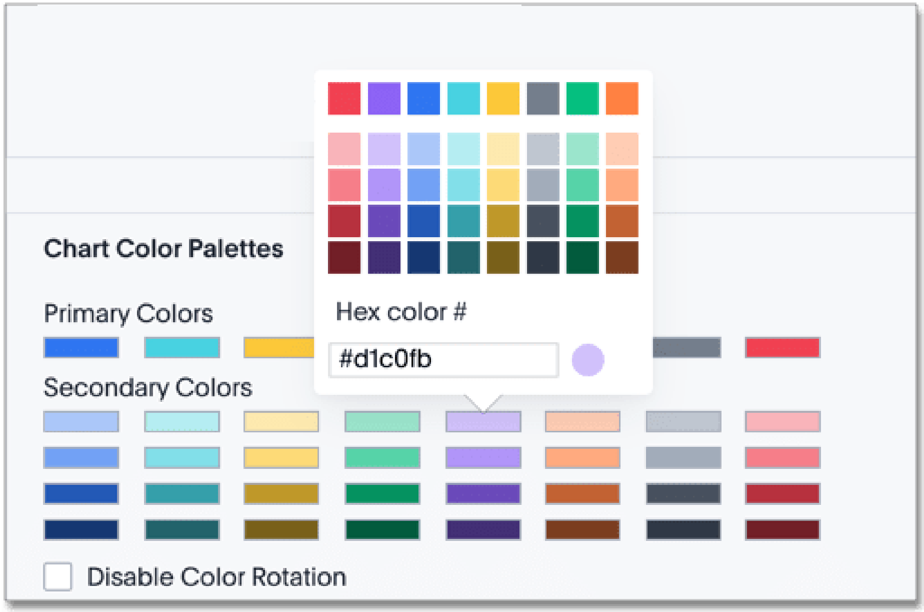
   Select **Style Customization** from the side navigation bar that appears.

   If the Style Customization option is not available, contact ThoughtSpot Support [See page 0] to enable it.

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

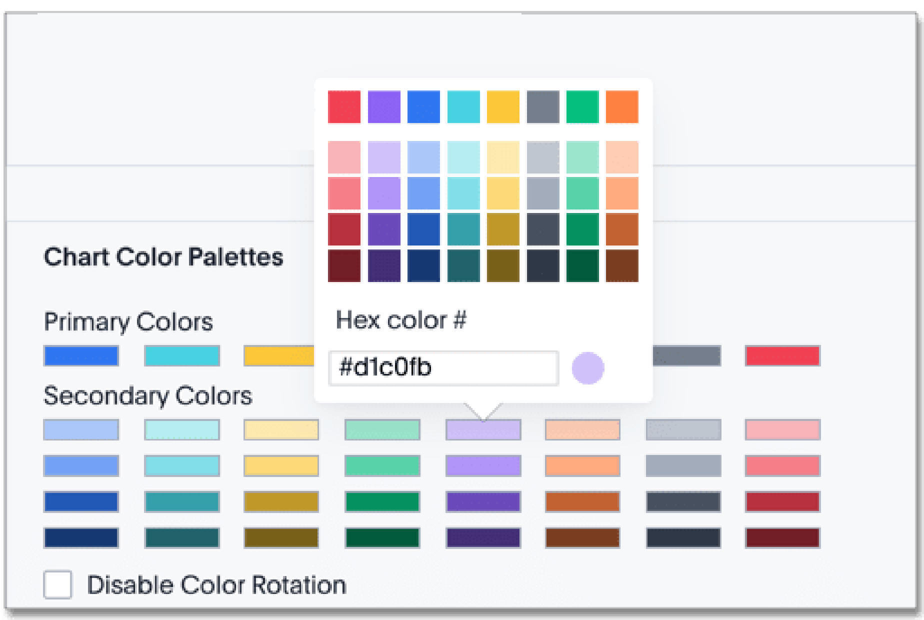


3. Click the color you would like to change in the **primary** color palette, and use the color menu to choose your new color. You can also add a HEX color number directly.



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.

4. Click the color you would like to change in the **secondary** color palette, and use the color menu to choose your new color. You can also add a HEX color number directly.



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.

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

Disable color rotation

You can disable color rotation for single-color charts. If you disable color rotation, ThoughtSpot generates single-color charts in the order of your color palette, left to right. If you leave color rotation enabled, ThoughtSpot generates the color of single-color charts randomly.

Change the footer text

Summary: You can customize ThoughtSpot's footer test to add a company-specific message.

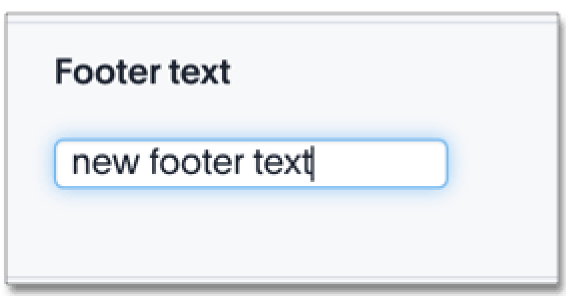
The ThoughtSpot footer appears by default in the ThoughtSpot application. It also appears with an embed application that encompasses an individual pinboard or a full application. In embed applications that have a single visualization, you can ask your ThoughtSpot support engineer to disable the footer.

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. Navigate to the Admin Console by clicking on the **Admin** tab from the top navigation bar. Select **Style Customization** from the side navigation bar that appears.
   If the Style Customization option is not available, contact ThoughtSpot Support [See page 0] to enable it.
2. Click the text box under **Footer text**.
3. Enter your new text message.



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

4. 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

> **Summary:** ThoughtSpot has several public APIs.

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 [See page 83]	`POST` /tspublic/v1/pinboarddata	Get the pinboard data from the ThoughtSpot system
Metadata [See page 86]	`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 [See page 93]	`POST` /tspublic/v1/session/login	Authenticate and login a user
	`POST` /tspublic/v1/session/logout	Logout a user out of an existing session
User [See page 96]	`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 [See page 105]	`POST` /tspublic/v1/group/addprivilege	Add a privilege to a group
	`POST` /tspublic/v1/group/removeprivilege	Remove a privilege from a group
Materialization [See page 108]	`POST` /tspublic/v1/materialization/refreshview/{id}	Re-execute the query and load data into the materialized view
Search Data [See page 110]	`POST` /tspublic/v1/searchdata	Search data from a specific data source in the ThoughtSpot system.

See About the REST API [See page 23] for information on how to call and use the REST APIs.

Pinboard Data API

> **Summary:** You can retrieve the data in a pinboard or other ThoughtSpot visualization.

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

Summary: The Metadata APIs allow you to fetch metadata for ThoughtSpot objects.

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/listvizhead
ers?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/listobjectheaders?type=PINBOARD_ANSWER_BOOK&subtypes=WORKSHEET&category=ALL&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

> **Summary:** The Session APIs enable you to manage the sessions of 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-url
encoded' --header 'Accept: application/json' --header 'X-Reques
ted-By: ThoughtSpot' -d 'username=test&password=fhfh2323bbn&rem
emberme=false' 'https://<instance>/callosum/v1/tspublic/v1/sess
ion/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

> **Summary:** The User APIs enable you to manage user- and group-related operations in the ThoughtSpot system.

You can use the User APIs to manage your users and groups in ThoughtSpot. 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/owners
hip?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. 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-url
encoded' --header 'Accept: application/json' -d 'applyChanges=f
alse' 'https://<instance>/callosum/v1/tspublic/v1/user/sync'
```

Request URL

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

Response Example

This example covers user objects (with emails), group objects, and their relationships.

- `created` and `modified` dates may be left blank for new users.
- `principalTypeEnum` value specifies if the principal is a user or a group.

Here, `test1` user belongs to two groups - `Customer Success` and `Marketing`. `test2` belongs to the group `Administrator`. `All` is a default group to which every user belongs; you may omit it from the input.

Set `visibility` to `NON_SHARABLE` if you do not want users to be able to share objects [See page 0] with users in this group.

```
[
  { "name": "Customer Success",
    "displayName": "Customer Success",
    "description": "CS",
    "created": 1568926267025,
    "modified": 1568926982242,
    "principalTypeEnum": "LOCAL_GROUP",
    "groupNames": [],
    "visibility": "DEFAULT" },

  { "name": "All",
    "displayName": "All Group",
    "created": 1354006445722,
    "modified": 1354006445722,
    "principalTypeEnum": "LOCAL_GROUP",
    "groupNames": [],
    "visibility": "DEFAULT" },

  { "name": "Marketing",
    "displayName": "Marketing",
    "description": "Marketing Group",
    "created": 1587573582931,
    "modified": 1587573583003,
    "principalTypeEnum": "LOCAL_GROUP",
    "groupNames": [],
    "visibility": "DEFAULT" },

  { "name": "test1",
    "displayName": "test one",
    "description": "",
    "created": 1587573554475,
    "modified": 1587573589986,
    "mail": "test1@test.com",
    "principalTypeEnum": "LOCAL_USER",
    "groupNames": [ "All", "Customer Success", "Marketing" ],
    "visibility": "DEFAULT" },

  { "name": "test2",
    "displayName": "test two",
    "created": 1587573621279,
    "modified": 1587573621674,
    "mail": "test2@test.com",
    "principalTypeEnum": "LOCAL_USER",
    "groupNames": [ "Administrator", "All" ],
    "visibility": "DEFAULT" }
```

```
]
```

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`

Property	Description
`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://<insta
nce>/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

> **Summary:** 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 the `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 `DATADOWN-LOADING` 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-url
encoded' --header 'Accept: application/json' --header 'X-Reques
ted-By: ThoughtSpot' -d 'privilege=DATADOWNLOADING&groupNames=A
LL_GROUP' 'https://<instance>/callosum/v1/tspublic/v1/group/add
privilege'
```

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 `DATADOWN-LOADING` 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-url
encoded' --header 'Accept: application/json' --header 'X-Reques
ted-By: ThoughtSpot' -d 'privilege=USERDATAUPLOADING&groupName
s=ALL_GROUP' 'https://<instance>/callosum/v1/tspublic/v1/group/
removeprivilege'
```

Request URL

```
https://<instance>/callosum/v1/tspublic/v1/group/removeprivileg
e
```

Response Example

```
Not applicable
204 - Success
```

Materialization API

> **Summary:** The materialization API allows you to synchronize a view's data with the latest data.

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 that occurred.

> **Note:** To refresh a materialized view, you must have the **Can administer ThoughtSpot** privilege [See page 0].

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: app
lication/json' --header 'X-Requested-By: ThoughtSpot' 'http
s://<instance>/callosum/v1/tspublic/v1/materialization/refreshv
iew/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
```

Search Data API

Summary: To use the data retrieved from a search query programmatically, you can first query this data using the ThoughtSpot Search Data API.

It is often difficult to programmatically use the result set of a query that runs in the ThoughtSpot UI search bar. To use the data that we retrieve from a query programmatically, you can use ThoughtSpot Search Data API.

When issuing a query through the ThoughtSpot UI, users make selections to disambiguate a query. Because selection is not possible with an API approach, we modified the API query language to include query disambiguation. See Components of a search query [See page 111].

Resource URL

`post` /tspublic/v1/searchdata

Request Parameters

Query Parameter	Data Type	Description
`query_string`	string	The data search query. See Components of a search query [See page 111].
`data_source_guid`	string	The GUID of the data source, either a worksheet, a view, or a table. Example: `["4fdf9d2c-6f34-4e3b-9fa6-bd0ca69676e1"]`
`batchsize`	integer	The batch size for loading search objects. The system default is `-1` .

Query Parameter	Data Type	Description
`pagenumber`	integer	Alternate way to specify `1` -based offset: `indexingOffset = (pageNumber - 1) * batchSize` The system default is `-1` .
`offset`	integer	Specify a `1` -based offset. The system default is `-1` .
`formattype`	string	The format of the data. Valid values are `COMPACT` (default) or `FULL` JSON.

Components of a search query

In ThoughtSpot Query Language, we classify components of a query into various types of tokens: *Column [See page 0]*, *Operator [See page 0]*, *Value [See page 0]*, *Date Bucket [See page 0]*, *Keyword [See page 0]*, and *Calendar [See page 0]*:

Column

Columns must be enclosed in square brackets, `[ ]` .

Example In the query `revenue by ship mode` , both `revenue` and `ship mode` are columns. A valid query for the API is:

```
[revenue] by [ship mode]
```

Operator

ThoughtSpot supports various operators such as `=` , `!=` , `>` , `>=` , `<=` , `<` , `contains` , `not contains` , and so on. Use these operators in the API query in the same manner as in the UI.

Example Specify `revenue` over 1000, and limit `ship mode` to 'air':

```
[revenue] > 1000 [ship mode] = ‘air’
```

Value

String (text) and date values must be enclosed within quotes, `' '`. Do not use quotes for numeric values, except for dates.

When using multiple values, separate them by a comma, `,`.

Example When a ThoughtSpot UI query is `revenue top 2 ship mode`, the equivalent API query is:

```
[revenue] top 2 [ship mode]
```

Example When a ThoughtSpot UI query is `revenue ship mode = air`, the equivalent API query is:

```
[revenue] [ship mode] = ‘air’
```

Date Bucket

In the ThoughtSpot UI, when there are several date columns, users can bind date bucket tokens to a specific column. When using the API, this binding between the date column and the date bucket must be made explicit. The column with which the date bucket is bound, and the date bucket token, must be separated by a period, `.`.

Single word date buckets can be expressed *as is*. Multi-word date buckets must be enclosed within quotes.

Example When a ThoughtSpot UI query is `revenue commit date monthly`, and if `monthly` is bound to `commit date`, the equivalent API query is:

```
[revenue] [commit date].monthly
```

Example When a ThoughtSpot UI query is `revenue day of week = 5`, and if `day of week` is bound to `commit date`, the equivalent API query is:

```
[revenue] [commit date].'day of week' = 5
```

Keyword

Use keywords in the API query in the same manner as in the UI.

Example When a ThoughtSpot UI query uses keywords `growth of` and `sort by`, the equivalent API query is:

```
growth of [revenue] by [commit date]
```

Calendar

You can specify a custom calendar in the query. Use the `calendar.`*`calendar_name`* format explicitly.

When the calendar name contains multiple words, these words must be enclosed in single quotes.

Example When a ThoughtSpot UI query is `revenue by commit date fiscal`, where the name of the calendar is `fiscal`, the equivalent API query is:

```
[revenue] by [commit date] calendar.fiscal
```

Example When a ThoughtSpot UI query is `revenue by commit date my calendar`, where the name of the calendar is `my calendar`, the equivalent API query is:

```
[revenue] by [commit date] calendar.'my calendar'
```

Request example

cURL - COMPACT

```
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/searchdata?que
ry_string=%5Bsales%5D%20%5Bstore%20region%5D&data_source_guid=0
6517bd1-84c0-4bc6-bd09-f57af52e8316&batchsize=-1&pagenumbe
r=-1&offset=-1&formattype=COMPACT'
```

Request URL - COMPACT

```
https://<instance>/callosum/v1/tspublic/v1/searchdata?query_str
ing=%5Bsales%5D%20%5Bstore%20region%5D&data_source_guid=06517bd
1-84c0-4bc6-bd09-f57af52e8316&batchsize=-1&pagenumber=-1&offse
t=-1&formattype=COMPACT
```

cURL - FULL

```
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/searchdata?que
ry_string=%5Bsales%5D%20%5Bstore%20region%5D&data_source_guid=0
6517bd1-84c0-4bc6-bd09-f57af52e8316&batchsize=-1&pagenumbe
r=-1&offset=-1&formattype=FULL’
```

Request URL - FULL

```
https://<instance>/callosum/v1/tspublic/v1/searchdata?query_str
ing=%5Bsales%5D%20%5Bstore%20region%5D&data_source_guid=06517bd
1-84c0-4bc6-bd09-f57af52e8316&batchsize=-1&pagenumber=-1&offse
t=-1&formattype=FULL
```

Response example

```
{
  "columnNames": [
    "Store Region",
    "Total Sales"
  ],
  "data": [
    [
      "east",
      18934491.05134509
    ],
    [
      "midwest",
      29157090.327609923
    ],
    [
      "south",
      25484693.074720126
    ],
    [
      "southwest",
      34241076.52103955
    ],
    [
      "west",
      30848491.458509445
    ]
  ],
  "samplingRatio": 1,
  "totalRowCount": 5,
  "rowCount": 5,
  "pageSize": 100000,
  "offset": 0
}
```

Using the Search Data API

To test the search query API, follow these steps:

Determine the GUID of the data source

1. In a Web Browser tab, navigate to the following address, and log in as admin user:

   ```
   https://<instance>/
   ```

2. In the top navigation, click **Data**.

   Alternatively, navigate to the following address:

   ```
   https://<instance>/#/data/tables/
   ```

3. Select a data source that you plan to query: a worksheet, a table, or a view.

4. In the address bar of the Web Browse, note the GUID of the selected data source; it is the last string of the address. For example, in the following address string, the GUID is `9d93a6b8-ca3a-4146-a1a1-e908b71b963f`:

   ```
    https://<instance>/#/data/tables/9d93a6b8-ca3a-4146-a1a
   1-e908b71b963f
   ```

5. Copy and save the GUID.

Run the Search Data API

1. In another browser, navigate to the following address:

```
  https://<instance>/external/swagger/#!/tspublic%2Fv1/searchDa
ta
```

1. Click on `POST /tspublic/v1/searchdataTS`.

The parameter interface appears.

2. In the **Parameters** interface, enter the values for the following:

    - `query_string` is the actual search query. See Components of a search query [See page 111].
    - `data_source_guid` is the identifier you obtained earlier, in Determine the GUID of the data sources [See page 116].

    You can leave other parameters at their default value.

3. Click **Try it out!**, and note the results.

    You may wish to check that the same query, when you run it in the ThoughtSpot UI search bar (with slightly different syntax), returns the same data.

Limitations of Search Query API

- To avoid join path ambiguities, a query can use only a **single data source**.
- Search execution of query strings is **case insensitive**.
- All Column names in the data source must have **unique names** that also pass the “case insensitivity” test.

  For example, Columns `[Revenue]` and `[revenue]` are not unique.
- Column names cannot contain square brackets, `[` or `]`.
- Values must be enclosed in quotes, `‘’`, but they cannot contain quotes.
- The API does not support **in-query formula** definitions. To use a formula, first create it on the worksheet or a table using the ThoughtSpot UI, and then use the named formula inside the API query.
- Users must be **authenticated** and have **read access** to the data source.

- Your browser locale must be `en-US`. Swagger does not accept other variations of English, such as British English, or other languages. Your search keywords must also be in American English. Your column names and other data values do **not** need to be in American English. You can change your preferred locale to `en-US` in your browser settings.