



Application Integration Guide

Version 6.0 December 02, 2019

Copyright for ThoughtSpot publications. © 2019 ThoughtSpot, Inc. All rights reserved.

ThoughtSpot, Inc. 910 Hermosa Ct
Sunnyvale, CA 94085

All rights reserved. This product is protected by U.S. and international copyright and intellectual property laws. ThoughtSpot is a trademark of ThoughtSpot, Inc. in the United States and/or other jurisdictions. All other marks and names mentioned herein may be trademarks of their respective companies.

Table of Contents

| | |
|---|-----------|
| Overview..... | 3 |
| Introduction to Embedding..... | 5 |
| Log in to the Linux shell using SSH | 7 |
| Log in credentials | 8 |
| Use the JavaScript API | 11 |
| SAML | |
| About SAML | 13 |
| Configure SAML | 14 |
| Configure CA SiteMinder | 15 |
| Active Directory | |
| Configure Active Directory Federated Services..... | 18 |
| Initialize the Identity Provider Metadata | 19 |
| Initialize the Service Provider Metadata | 20 |
| Test the ADFS Integration | 21 |
| REST API | |
| About the REST API | 22 |
| Calling the REST API..... | 24 |
| REST API pagination | 28 |
| Use the REST API to get data | 31 |
| Use the Embedded Search API..... | 34 |
| Use the Data Push API..... | 36 |
| Embed ThoughtSpot | |
| Understand embedding | 41 |
| Embed pinboard or visualization | 46 |
| Authentication flow with embed | 52 |
| Full application embedding | 56 |
| Configure trusted authentication..... | 59 |
| Runtime Filters | |
| About Runtime Filters | 63 |
| Apply a Runtime Filter..... | 66 |
| Runtime Filter Operators | 68 |

Style Customization

| | |
|---|----|
| Customize the application style..... | 69 |
| Upload application logos | 71 |
| Set chart and table visualization fonts | 72 |
| Choose a background color | 76 |
| Select chart color palettes | 77 |
| Change the footer text | 79 |

API Reference

| | |
|---------------------------|-----|
| Introduction..... | 81 |
| Pinboard Data API | 83 |
| Metadata API | 86 |
| Session API | 93 |
| User API..... | 96 |
| Group API..... | 104 |
| Materialization API | 107 |

ThoughtSpot mobile overview

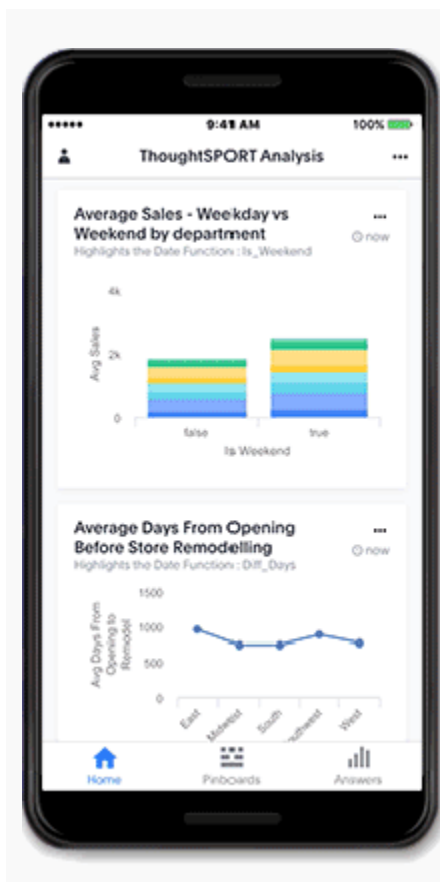
ThoughtSpot mobile app allows you to discover insights in seconds from billions of rows of data, right from the palm of your hand. You can access your ThoughtSpot cluster to search answers and pinboards. You can also create pinboards using existing answers and use pinboard filters.

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

Requirements

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

Features



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

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

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

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

Favorites
Tag your favorite Pinboards and Answers for quick access.

Getting Started

For administrators:

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

For users:

- To install and set up the app, see [Install and set up mobile app](#).

Introduction to Embedding

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

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

ThoughtSpot Extended Enterprise Edition includes these capabilities:

- [Full Application Embedding](#)
- [Embedded Charts and Pinboards](#)
- [Data REST API](#)
- [Runtime Filters](#)
- [Metadata API](#)

Rights and Obligations

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

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

concerning data privacy, such as GDPR and HIPAA.

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

If you are using either the **Visualization Embedding** or **Pinboard Embedding** in your application, it is okay to disable the NPS survey.

8. Deployments of ThoughtSpot Extended Enterprise must conform to [logo restrictions](#).

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

- [Logos & Favicon](#)
- [Color Palettes](#)
- [Background Color](#)
- [Chart and Table Fonts](#)
- [Footer Text](#)

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

Section Contents

Here are the main topics for Embedding with Extended Enterprise:

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

Log in to the Linux shell using SSH

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

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

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

```
ssh admin<hostname_or_IP>
```

3. Enter the password for the admin user.

Log in credentials

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

Administrative access

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

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

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

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

SSH to the appliance

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

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

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

```
ssh admin<hostname_or_IP>
```

3. Enter the password for the admin user.

Log in to the ThoughtSpot application

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

Before accessing ThoughtSpot, you need the following:

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

ThoughtSpot supports the following Web browsers:

Firefox

68.x, 69.x, and later

Chrome

76.x, 77.x, and later

Internet Explorer

11.x, and later

Edge

44.x, and later

Safari

13.x, and later

☑ **Tip:** We support, but do not recommend, the use of the Internet Explorer.
Depending on your environment, you can experience performance or UI issues.

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

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

`http://<hostname_or_IP>`

2. Enter your username and password, and click **Sign in**.

Using the JavaScript API

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

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

You can download the [ThoughtSpot JavaScript library](#) from our secure storage server.

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

Browser Support

The JS API works in the following browsers:

Firefox

68.x, 69.x, and later

Chrome

76.x, 77.x, and later

Internet Explorer

11.x, and later

Edge

44.x, and later

Safari

13.x, and later

Internet Explorer 10

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

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

- **Add a Custom Response Header** We recommend this approach because it is more robust, offers more control, and has a lower risk of introducing a bug to your code. In general, you must set the response header to match the server and the technology.
 - set the header name to "X-UA-Compatible"
 - set the value to "IE=Edge"
- **Add a Meta Tag** Add this meta tag as the *first* tag in the header section of the page:

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

Cross-Origin HTTP Requests (CORS)

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

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

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

To enable CORS between your client applications and your ThoughtSpot instance, you must work with [ThoughtSpot Support](#).

About SAML

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

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

- [Configure SAML](#), for instructions to configure SAML in ThoughtSpot.
- [Configure SAML with CA SiteMinder](#), for configuring SAML specifically with CA SiteMinder.

Configure SAML

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

Before configuring SAML, you need this information:

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

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

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

```
tscli saml configure
```

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

Configure CA SiteMinder

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

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

1. Configure the Local Identity Provider Entity as follows:

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

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

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

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

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

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

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

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

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

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

Configure Active Directory Federated Services

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

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

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

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

1. [Initialize IdP metadata.](#)
2. [Initialize the Service Provider metadata.](#)
3. [Test your ADFS integration.](#)

Initialize the Identity Provider Metadata

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

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

To initialize the IdP metadata on AD FS:

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

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

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

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

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

Initialize the Service Provider Metadata

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

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

To initialize the Service Provider metadata on AD FS:

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

Test the ADFS Integration

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

About the Data REST API

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

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

Authentication

Before you can use the Data REST API, you must authenticate to ThoughtSpot using SAML with the [JavaScript API](#).

Cross Domain Verification

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

Data REST API capabilities

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

You can use the REST API to do things like:

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

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

Direct Search-to-Embed API

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

Public API reference

You can find more information on our public APIs in the [Reference guide](#).

Related information

- [API Reference guide](#).
- [Direct Search-to-Embed API](#).

Calling the Data REST API

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

Specify the pinboard or visualization example

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

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

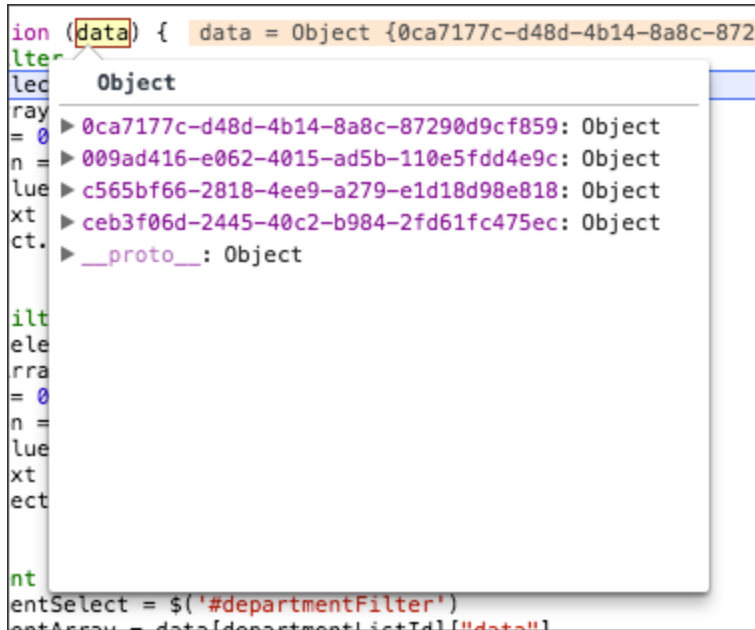
To retrieve data from a specific visualization within a pinboard, you would append the ID number of the visualization using the vizid parameter:

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

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

Object Format for Returned Data

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



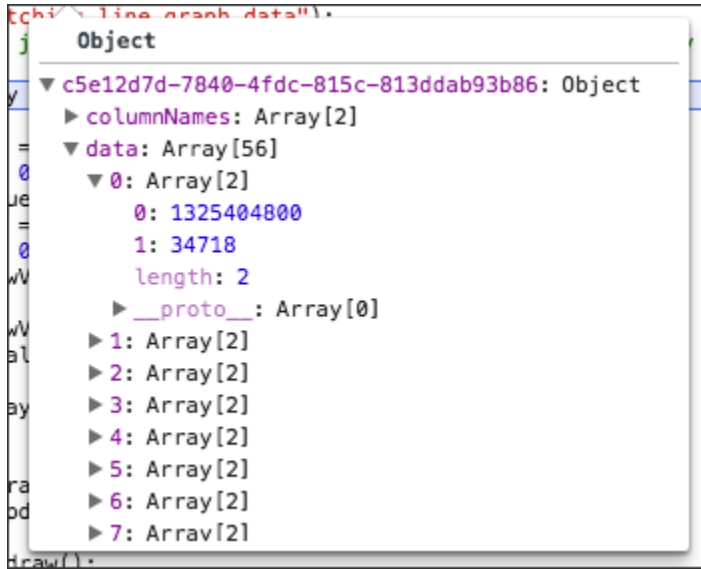
If you make a call to a specific viz on a pinboard, it will return just one object. The JSON object format for the data that is returned from ThoughtSpot is:

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

Each object contains four components:

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

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



The REST API supports filtering the data returned through parameters that you pass within the URL.

These are called [Runtime Filters](#).

Example

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

```

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

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

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

```

Data REST API pagination

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

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

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

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

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

`Offset` determines the starting point.

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

`PageNumber` is an alternate way to determine the offset. You must make a call with `pageNumber = 1` first. Then you can access any page. Calling with `pageNumber != 1` as the initial call will fail.

`pageNumber = 0` is not a valid value.

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

`FormatType` is the JSON format type.

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

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

Example

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

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


Use the Data REST API to get data

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

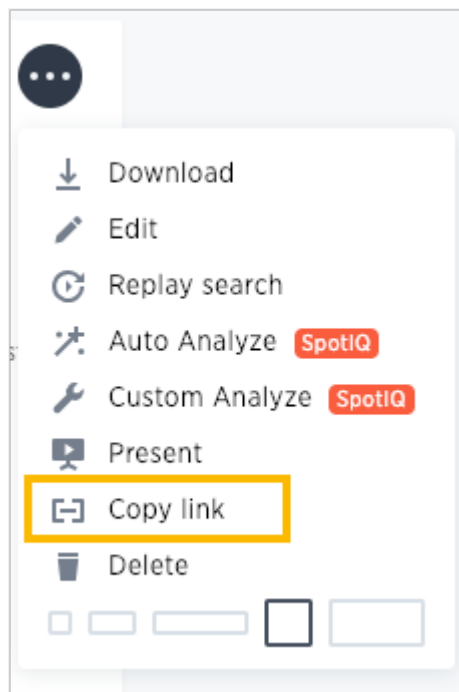
Data retrieved using the Data REST API is returned as JSON (JavaScript Object Notation).

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

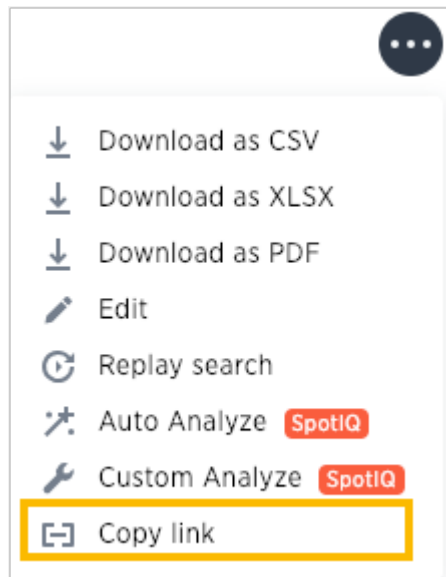
Use this procedure to construct the URL you will use to call the Data REST API:

1. Log in to ThoughtSpot from a browser.
2. Navigate to the pinboard from which you want to get data. If it doesn't exist yet, create it now.
3. Find the ID number of the object you want to get the data from. If the object is:

- A pinboard, click ellipses icon  the and select **Copy Link**.



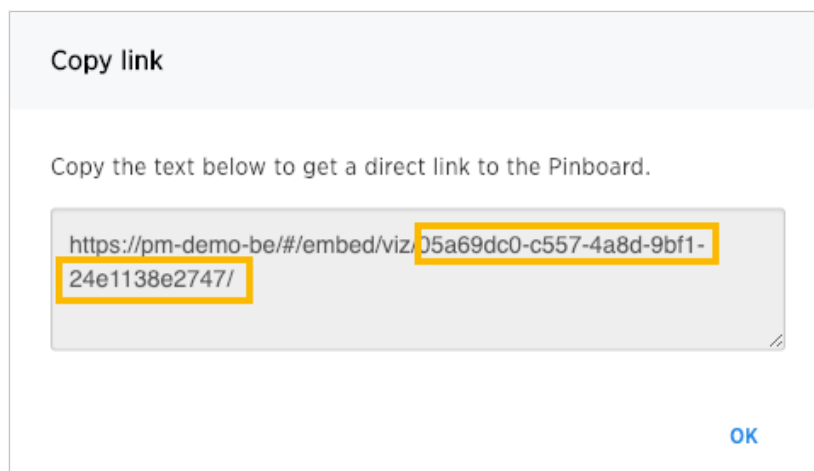
- A visualization, click the **Copy Link** icon in the upper right corner of the table or chart.



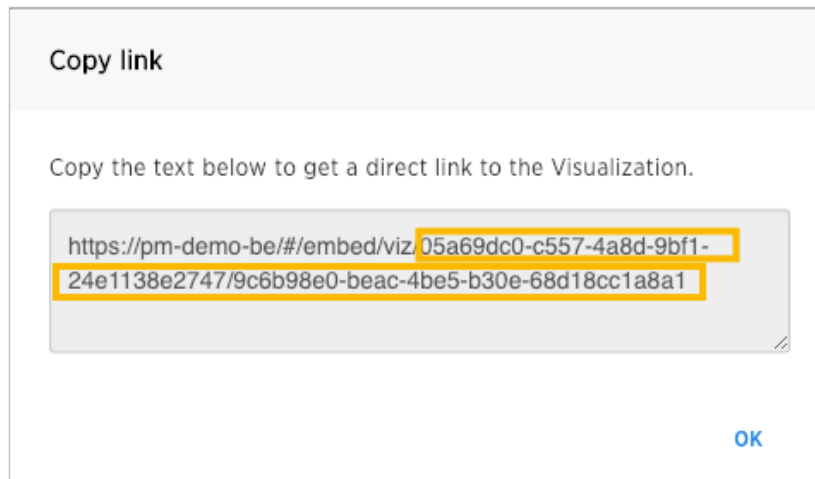
4. Copy the ID number from the link shown. Paste it somewhere so that you can use it later to construct the URL to use when calling the REST API.

If the object is:

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



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



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

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

For a visualization, the URL takes the form:

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

6. If you want to apply any filters to the data that will be returned, apply [Runtime Filters](#).
7. Now your URL is complete, and you can use it to access the data directly through the HTTP POST method. The Data REST API returns the data formatted as JSON.
8. Retrieve the data from the JSON and display it in your Web page, Web portal, or application.

Use the Embedded Search API to pull data from ThoughtSpot

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

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

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

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

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

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

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

Now when a user searches, the iFrame will send data to the subscription. The parent web page or application receives the data as JSON, and can do whatever you want with it.

5. You can set up your web page or application to display or otherwise act on the data it receives from the subscription.

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

Use the Data Push API

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

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

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

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

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

Create an Custom Action

To create a Custom Action, you must have the [Can administrator ThoughtSpot privilege](#).

Use this procedure to create an Custom Action in ThoughtSpot:

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

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

Action menu customization

Item Label

test action

URL

https://my_SAP_application.company.com/ord


Window size

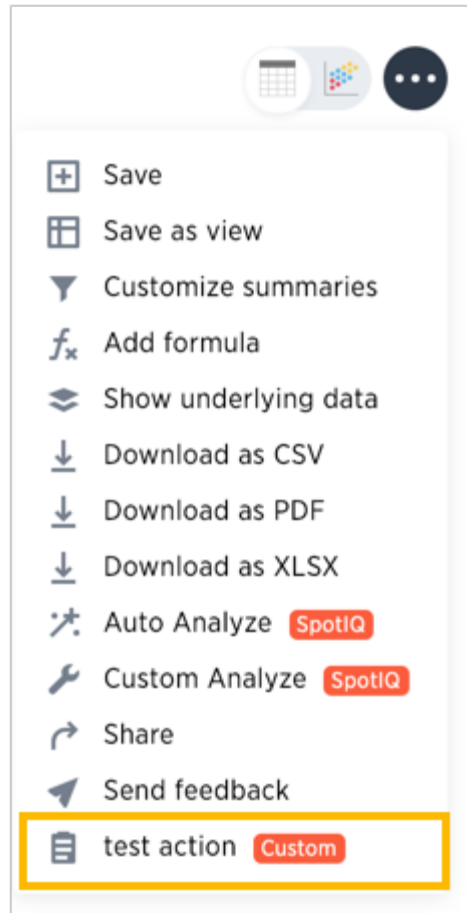
medium

▼

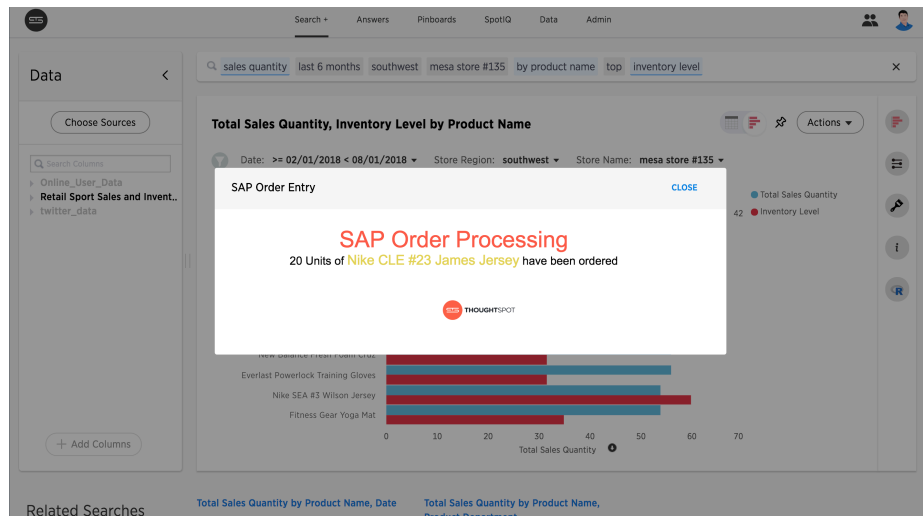
Cancel

CONFIRM

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



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



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

Sample application

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

```

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

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

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

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

```

Understand embedding

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

Decide what to embed and where

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

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

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

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

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

Configuration requirements for embedding

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

Optional settings for embedding

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

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

Changing this setting opens the links:

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

Choose an authentication methodology

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

No Authentication

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

SAML

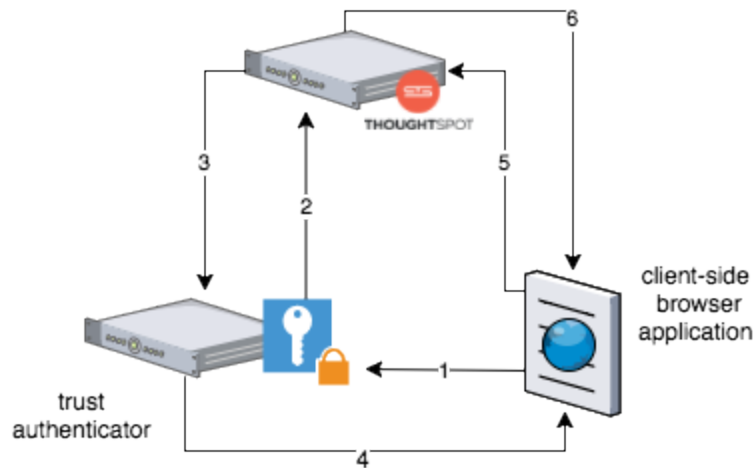
Before you can embed all or part of ThoughtSpot, you must authenticate to ThoughtSpot using SAML with the the public REST API call. After authentication, a URL is provided to call the desired visualization and populate it into an `iframe` .

You must [configure SAML](#) on your ThoughtSpot instance before using this method.

Trusted authentication service

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

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



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

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

Plan for Cross-Origin HTTP Requests (CORS)

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

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

Decide if you need to change the feedback email

ThoughtSpot has an automated feature that collects feedback from users and sends it to

`support@thoughtspot.com`. Depending on what and how you embed, user actions with your

embedded application can trigger feedback. You can continue to forward feedback in this manner or

direct the feedback to another email. To learn how to change the feedback email, see [Manage the feedback contact](#).

Remove the ThoughtSpot branded footer

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

Embed pinboard or visualization


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

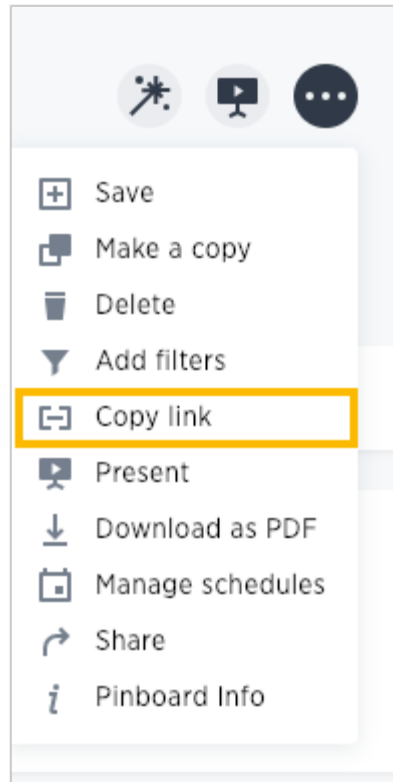
To build this sample, you need to have access to a text editor and a ThoughtSpot instance with a visualization. You should also have some experience working with Javascript.

Get the link for an entire pinboard or single visualization

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

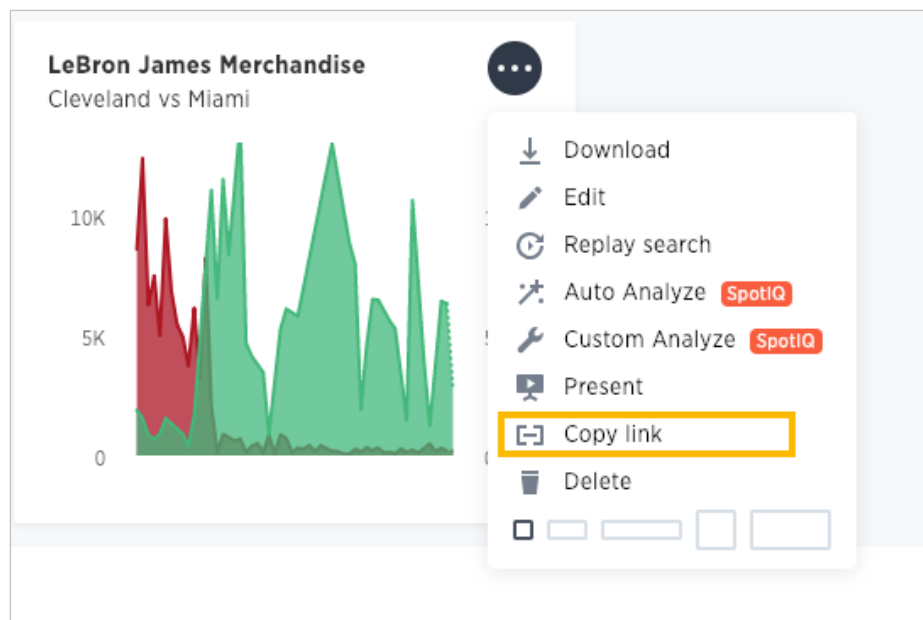
1. Log in to ThoughtSpot from a browser.
2. Navigate to a visualization on the **Pinboard** tab.
3. Open a pinboard.
4. Copy the URL for the entire pinboard and for a single visualization.

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



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

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



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

Edit the test.html

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

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

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

```
var protocol = "THOUGHTSPOT_PROTOCOL";  
var hostPort = "HOST_PORT";  
  
var pinboardId = "PINBOARD_ID";  
var vizualizationId = "VIZUALIZATON_ID";
```

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

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

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

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

```
var protocol = "http";  
var hostPort = "172.18.202.35:8088";  
  
var pinboardId = "061457a2-27bc-43a9-9754-0cd873691bf  
0";  
var visualizationId = "9985fccf-b28d-4262-b54b-29619a38  
348e";
```

The protocol (`http` or `https`) of your client and your ThoughtSpot instance should match.

You can use this identifier in the next part.

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

Enable CORS for your client domain

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

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

Test the example page

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

1. Log into ThoughtSpot.

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

2. Change to your `test` directory.

3. Start the `simplehttpserver` web server.

```
python -m SimpleHTTPServer 8000
```

4. Open your browser's **Developer** tools.
5. Navigate to the test page in your browser.

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

You should see something similar to the following:



6. Check the browser console.

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

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

Troubleshooting embeds

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

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

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

Authentication flow with embed

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

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

Authenticate when the window is initialized

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

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

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

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

Example of code flow

To authenticate with SSO.

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

```
<head>
  <script type="text/javascript" src="<protocol><your.thoughtspot.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 (isUserAuthenticatedToThoughtSpot) {
        // load an embedded ThoughtSpot
        // visualization or
        // make a ThoughtSpot data API call
      } else {
        // the current user into your system is not authenticated
        // into your ThoughtSpot instance, case in any other way suitable
        // to your application logic. Do NOT call setUpThoughtSpotAPI again
        // here as that could create an infinite cycle.
      }
    },
    function() {
      // the user got logged out from ThoughtSpot, possibly because
      // their session with ThoughtSpot expired, you can call setUpThoughtSpotAPI()
      // again to re-authenticate the user or handle this case in any other way
      // suitable to your application logic.
    },
    thoughtspotHost
  );
}
</script>

```

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

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

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

Full application embedding

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

Fully embedding ThoughtSpot content gives your users the ability to:

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

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

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

Before you try the technique, make sure you have read, [Understand embedding](#) in this section.

A single page with the full application embedded

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

```

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

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

    </script>
  </head>
  <body style="height: 100%; width: 100%">
    <button onclick="updateIframeUrl('homepage')">Homepage</button>
    <button onclick="updateIframeUrl('search')">Search</button>
    <button onclick="updateIframeUrl('answerList')">Answer list</button>
    <button onclick="updateIframeUrl('pinboardList')">Pinboard list</button>
    <button onclick="updateIframeUrl('data')">Data</button>
  </body>
</html>

```

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

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

Hide the ThoughtSpot navigation bar

To hide the primary navigation, configure this:

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

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

Error messages and full embed

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

Additional notes

Here are some additional notes about the full embed feature:

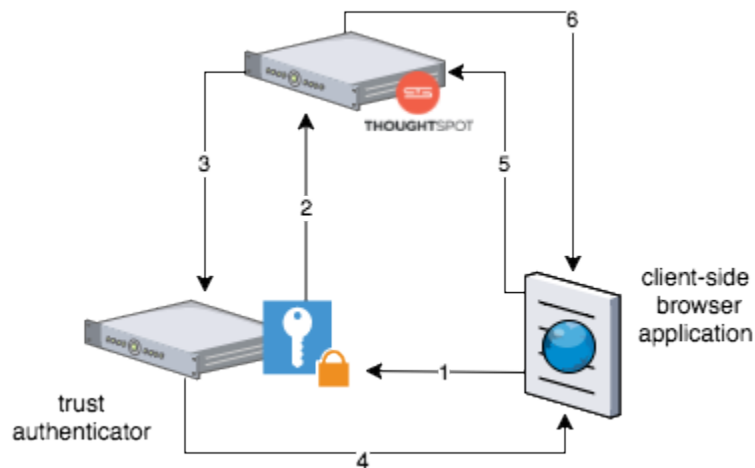
- Call `thoughtspot.<customerURL>.com/#/answer` and use that to access the search functionality.
- Call `thoughtspot.<customerURL>.com/#/pinboards` and use that to access saved pinboards.
- Use SAML for authentication against ThoughtSpot within the iFrame.

Configure trusted authentication

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

How users are authenticated

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



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

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

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

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

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

Enable trusted authentication and get a token

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

```
[admin@ourthoughtspot ~]$ tscli tokenauthentication enable
```

Token generated. Copy the GUID in the box.

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

Override added successfully

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

```
tscli tokenauthentication disable
```

Generated tokens do not expire.

Trusted authentication call

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

This is a REST request for an embedded ThoughtSpot object, page, or the entire application.

Your trusted authenticator server intercepts the request. Your server application must determine at minimum:

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

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

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

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

This post takes the following parameters:

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

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

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

4. The trusted authenticator server returns token to the original requestor.
5. Client completes the user's request providing the token along with the request.

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

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

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

About Runtime Filters

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

Capabilities of Runtime Filters

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

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

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

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

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

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

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

Supported Data Types

You can use runtime filters on these data types:

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

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

Example Uses

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

Limitations of runtime filters

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

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

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

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

Apply a Runtime Filter

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

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

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

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

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

- For Embedding a pinboard:

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

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

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

Customize the application style

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

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

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

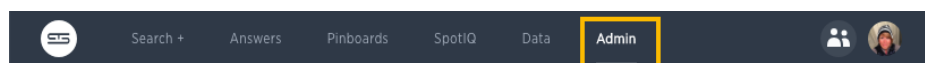
Style customization is enabled by default beginning in ThoughtSpot version 5.0. To disable style customization, contact ThoughtSpot Support. The ThoughtSpot logo in the middle of the page is automatically removed when Style Customization is enabled.

Change style customization

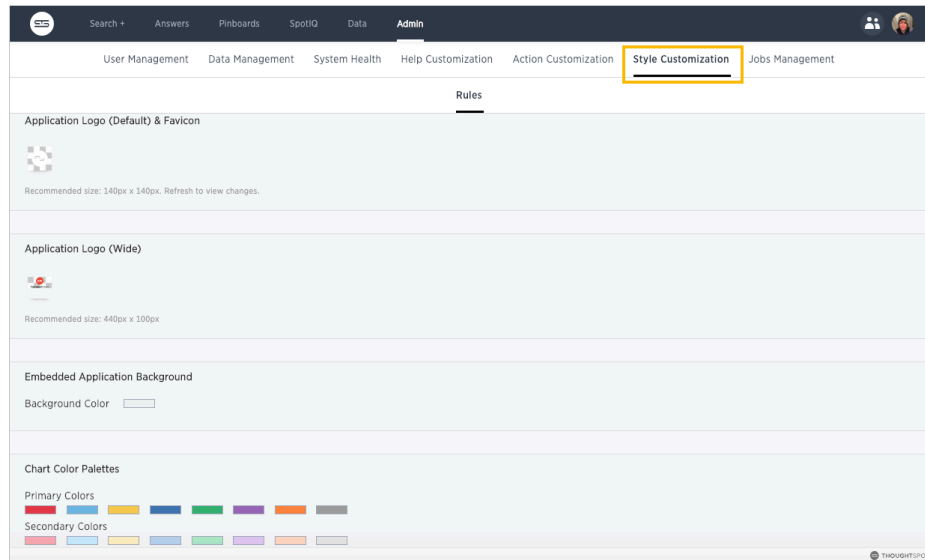
Make changes to the style of your ThoughtSpot interface in the **Style Customization** page. This option gives you defined, yet impactful capabilities for re-branding the interface, so having some understanding of typography and color schemes would be helpful.

To re-brand the interface:

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



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



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

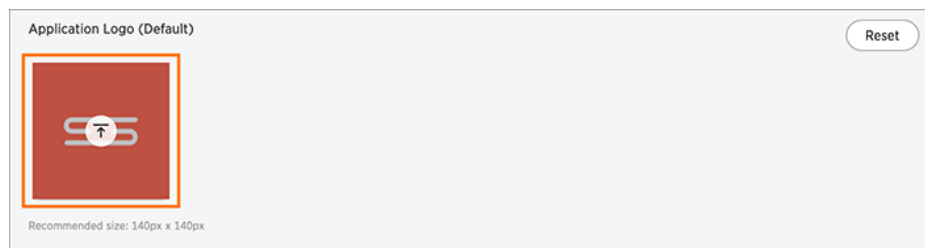
- [Upload application logos](#)
- [Set chart and table visualization fonts](#)
- [Choose a background color](#)
- [Select chart color palettes](#)
- [Change the footer text.](#)

Upload application logos

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

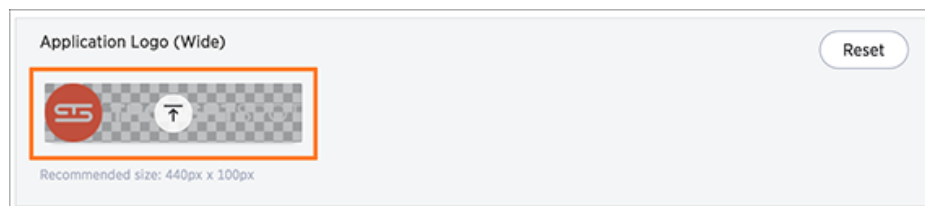
To upload your own default and wide application logos:

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



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

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



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

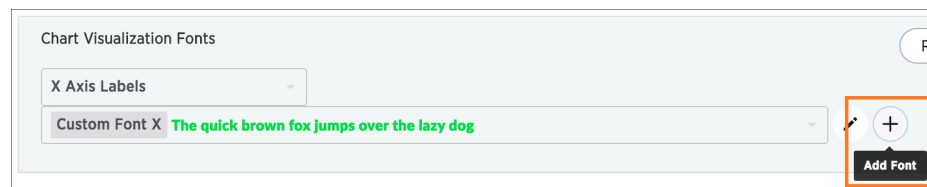
3. Click the **Reset** button on the upper right hand side of the sections if you would like to bring back the default logos.

Set chart and table visualization fonts

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

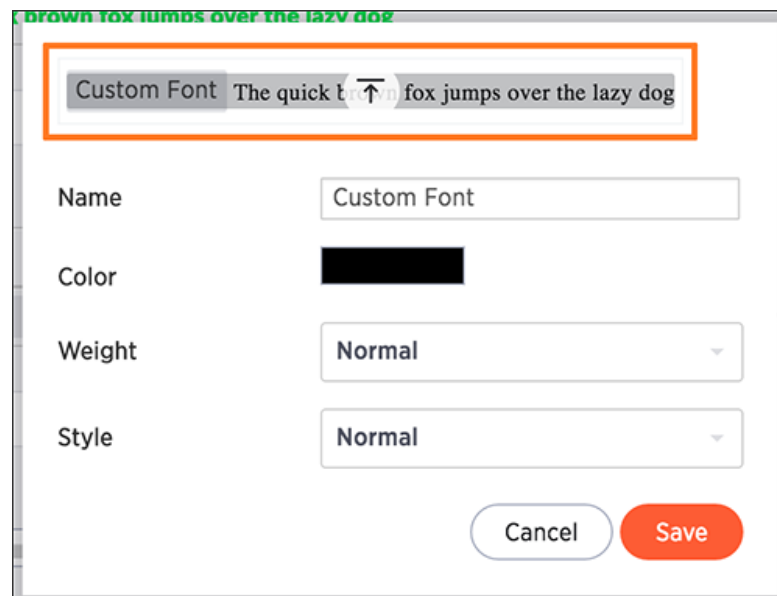
If you are confident in your knowledge of font visualizations, you can set your chart and table visualization fonts by following these steps:

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



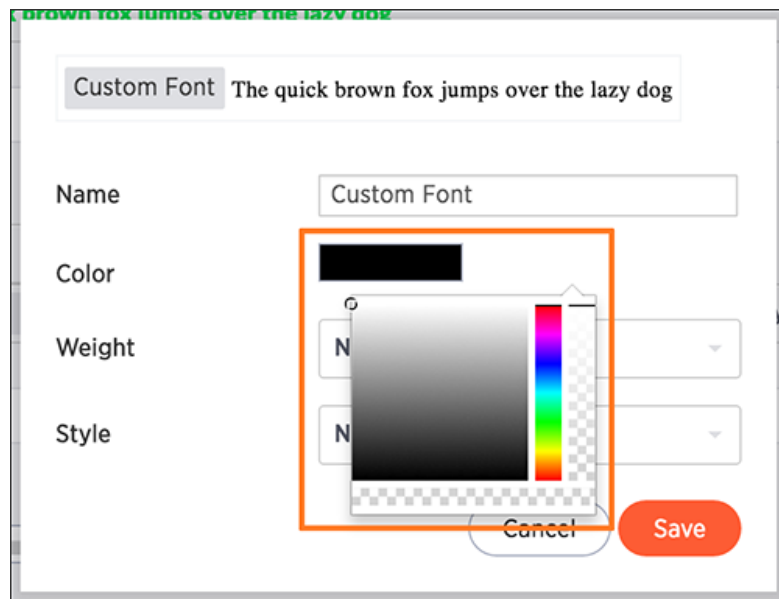
2. In the add new font menu, select the details for the font:

- a. Upload your custom font.

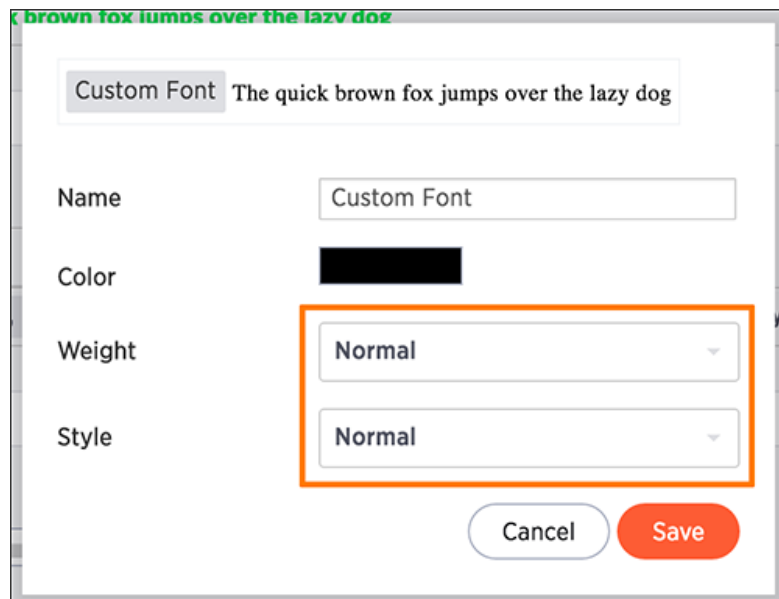


Only WOFF font types are supported.

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




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



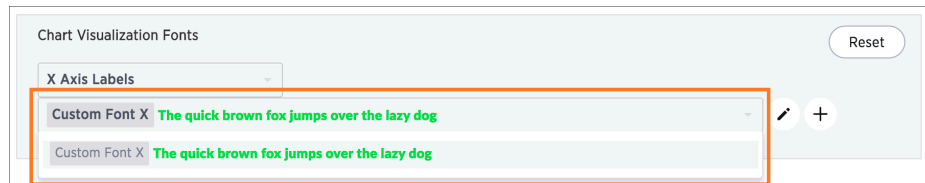
The font weight choices are normal, bold, and light. The style choices are normal, italic, and oblique.

d. Click **Save**.

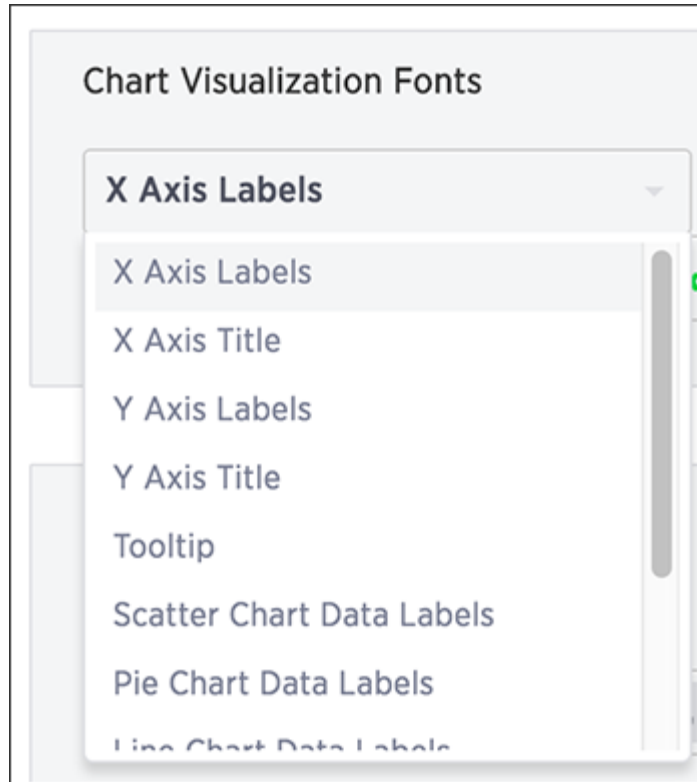
3. Click the **Edit Font** icon  to make changes to the font you just uploaded or to a pre-existing font.



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



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



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



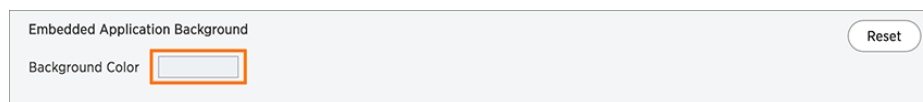
8. Click the **Reset** button on the upper right hand side of the sections if you would like to bring back the default fonts.

Choose a background color

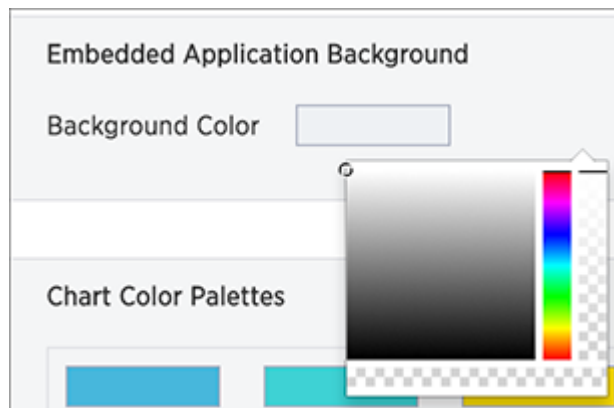
You can change the background color to match with your company's theme. The custom background color is in effect when using the API to embed visualizations and pinboards.

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

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



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



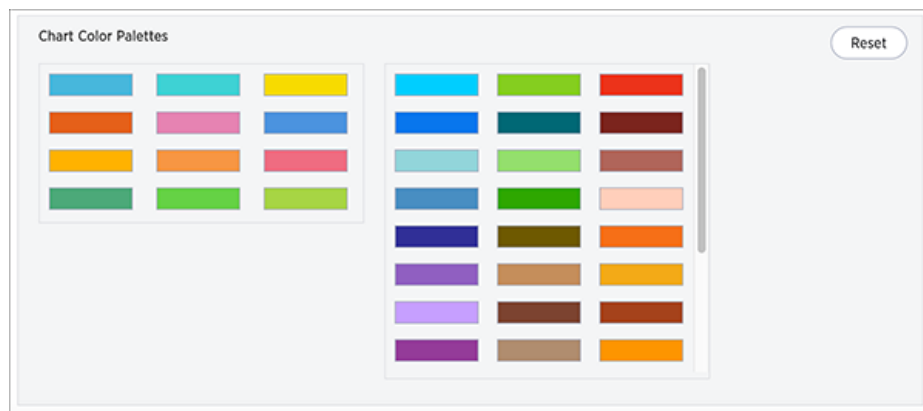
3. Click the **Reset** button on the upper right hand side of the section if you would like to bring back the default color.

Select chart color palettes

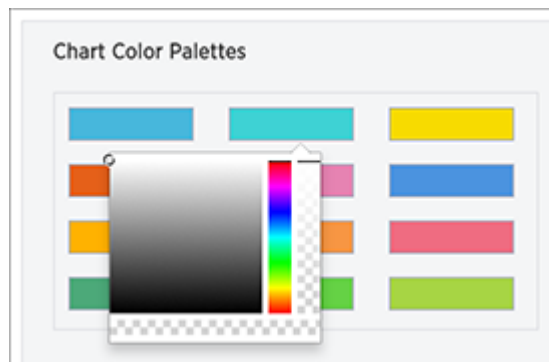
You can change the color palettes that are used to create your charts. Although it is suggested that you stick with the default settings, it is possible to create your own appealing color palettes if done correctly.

To select the chart color palettes:

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



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



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

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



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

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

Change the footer text

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

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

Add a message to the application footer

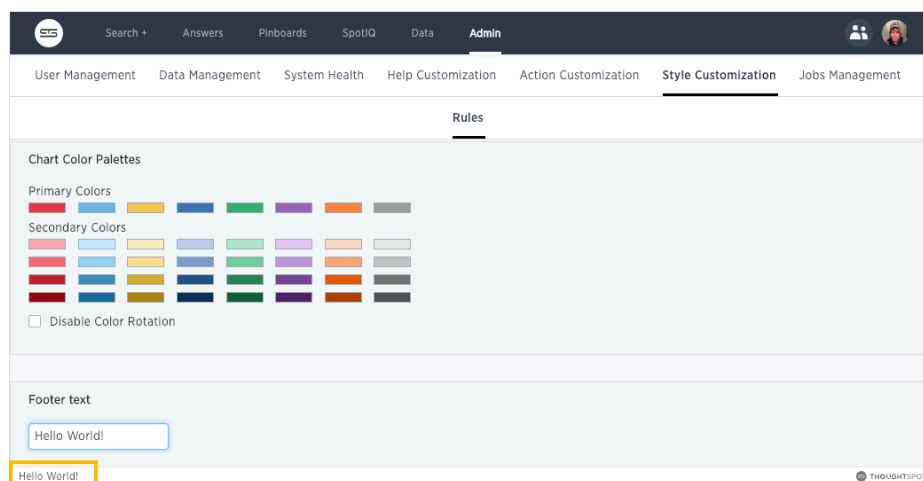
You can change the footer text to reflect your company's message. To change the footer text:

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



A screenshot of a configuration panel titled "Footer text". Inside the panel, there is a text input field containing the text "Hello World!".

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



A screenshot of the ThoughtSpot Admin console. The top navigation bar includes "Search +", "Answers", "Pinboards", "SpotIQ", "Data", and "Admin". Below the navigation bar, there are tabs for "User Management", "Data Management", "System Health", "Help Customization", "Action Customization", "Style Customization", and "Jobs Management". The "Style Customization" tab is selected. Under this tab, there is a "Rules" section. Below "Rules", there is a "Chart Color Palettes" section with "Primary Colors" and "Secondary Colors" palettes. At the bottom of the "Style Customization" section, there is a "Footer text" section with a text input field containing "Hello World!". A yellow box highlights the "Hello World!" text in the footer of the page.

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

back the default footer text.

Public API reference

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

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

See [About the REST API](#) for information on how to call and use the REST APIs.

Pinboard Data API

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

You may want to visualize the following:

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

Resource URL

POST /tspublic/v1/pinboarddata

Request Parameters

| Query Parameter | Data Type | Description |
|-----------------|-----------|---|
| id | string | The pinboard id in the system. |
| vizid | string | (Optional) The visualization id(s) on a pinboard. Use this parameter to fetch a specific visualization on a pinboard. The syntax is: ["4fdf9d2c-6f34-4e3b-9fa6-bd0ca69676e1", "....."] |
| batchsize | integer | The batch size for loading of pinboard objects. The system default is -1 . |
| pagenumber | integer | The system default is -1 . |
| offset | integer | The system default is -1 . Alternately, set the offset using the following code: 1-based indexing $\text{Offset} = (\text{pageNumber} - 1) * \text{batchSize}$ |
| 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' --header 'Accept: application/json' --header 'X-Requested-By: ThoughtSpot' 'https://<instance>/callosum/v1/tspublic/v1/pinboarddata?id=f4533461-cao5-4efa-a189-13815ab86770&batchsize=-1&pagenumber=-1&offset=-1&formattype=COMPACT'
```

Request URL

```
https://<instance>/callosum/v1/tspublic/v1/pinboarddata?id=f4533461-cao5-4efa-a189-13815ab86770&batchsize=-1&pagenumber=-1&offset=-1&formattype=COMPACT
```

Response Example

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

Metadata API

The Metadata APIs enable you to fetch metadata details for various objects in the ThoughtSpot system.

For example, you may want to see the visualization headers of a particular answer or a pinboard.

Get visualization headers

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

Resource URL

GET /tspublic/v1/metadata/listvizheaders

Request Parameters

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

Request Example

cURL

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


Request URL

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

Response Example

```
[
  {
    "id": "dd7f5467-99c3-4278-998b-6dd0c4346cd4",
    "name": "Headline Viz answer book guid max timestamp answer book guid != {null} sort by max timestamp descending today last 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 book 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 book guid != {null} sort by max timestamp descending today last 180 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: <ul style="list-style-type: none">QUESTION_ANSWER_BOOKPINBOARD_ANSWER_BOOKQUESTION_ANSWER_SHEETPINBOARD_ANSWER_SHEETLOGICAL_COLUMNLOGICAL_TABLELOGICAL_RELATIONSHIPTAGDATA_SOURCE |
| subtypes | string | Specifies the sub-types of metadata object. Valid values are: <ul style="list-style-type: none">ONE_TO_ONE_LOGICALWORKSHEETPRIVATE_WORKSHEETUSER_DEFINEDAGGR_WORKSHEET <p>Note: This parameter only applies to the LOGICAL_TABLE type.</p> |
| category | string | Specifies the metadata object category. Valid values are: <ul style="list-style-type: none">ALLMYFAVORITEREQUESTED |

| Query Parameter | Data Type | Description |
|-----------------|-----------|--|
| sort | string | Sort order of returned headers. Valid values are: <ul style="list-style-type: none"> • DEFAULT • NAME • DISPLAY_NAME • AUTHOR • CREATED • MODIFIED |
| sortascending | boolean | A flag to specify the sort order. A null value defines the default order. <ul style="list-style-type: none"> • Choose <code>true</code> to set ascending order • Choose <code>false</code> to set descending order |
| offset | integer | The batch offset to fetch the page headers. The system default is <code>-1</code> that implies first page. |
| batchsize | integer | The batch size of the object. A value of <code>-1</code> 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 <code>%</code> 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 TPCCH based visualization",
    "author": "59481331-ee53-42be-a548-bd87be6ddd4a",
    "created": 1450823023991,
    "modified": 1504281997165,
    "modifiedBy": "59481331-ee53-42be-a548-bd87be6ddd4a",
    "owner": "7752fa9e-db22-415e-bf34-e082c4bc41c3",
    "isAutoCreated": false,
    "isAutoDelete": false
  },
  {
    "id": "6715f768-8930-4180-9a3d-1efdbfaa8e7f",
    "name": "Headline Pinboard",
    "author": "59481331-ee53-42be-a548-bd87be6ddd4a",
    "created": 1519940021267,
    "modified": 1519945210514,
    "modifiedBy": "59481331-ee53-42be-a548-bd87be6ddd4a",
    "owner": "6715f768-8930-4180-9a3d-1efdbfaa8e7f",
    "isAutoCreated": false,
    "isAutoDelete": false
  },
  {
    "id": "601be8e5-140e-477c-8812-843795306438",
    "name": "Pinboard Filter - datatypes",
    "author": "59481331-ee53-42be-a548-bd87be6ddd4a",
    "created": 1519943239150,
    "modified": 1519944533160,
    "modifiedBy": "59481331-ee53-42be-a548-bd87be6ddd4a",
    "owner": "601be8e5-140e-477c-8812-843795306438",
    "isAutoCreated": false,
    "isAutoDelete": false
  }
]
```

Session API

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

Managing login

Use this API to authenticate and login a user.

Resource URL

POST /tspublic/v1/session/login

Request Parameters

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

Request Example

cURL

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

Request URL

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

Response Example

```
Not applicable  
204 – Successful login
```

Managing logout

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

Resource URL

```
POST /tspublic/v1/session/logout
```

Request Example

cURL

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' --header 'X-Requested-By: ThoughtSpot' 'https://<instance>/callosum/v1/tspublic/v1/session/logout'
```

Request URL

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


Response Example

Not applicable
204 – Successful logout

User API

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

Transfer ownership

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

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

Resource URL

POST /tspublic/v1/user/transfer/ownership

Request Parameters

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

Request Example

cURL

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

Request URL

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

Response Example

```
Not applicable  
204 – Successful login
```

Synchronize principals

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

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

Resource URL

POST /tspublic/v1/user/sync

Request Parameters

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

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

Request Example

cURL

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

Request URL

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

Response Example

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

Change password

Use this API to change the password of a user.

Resource URL

POST /tspublic/v1/user/updatepassword

Request Parameters

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

Request Example

cURL

```
curl -X POST --header 'Content-Type: application/x-www-form-urlencoded' --header 'Accept: application/json' --header 'X-Requested-By: ThoughtSpot' -d 'name=guest&tpassword=test&password=foo barfoobar' '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 | <p>Name of the principal.</p> <p>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”).</p> |
| 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 | <p>Type of the user created in the ThoughtSpot system.</p> <ul style="list-style-type: none">• LOCAL_USER (a user is validated through password saved in the ThoughtSpot database)• LOCAL_GROUP |
| password | <p>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:</p> <ul style="list-style-type: none">• if the user is of LOCAL_USER type,• when the user is created for the first time. <p>In subsequent update, the user password is not updated even if it changes in the source system.</p> |
| groupNames | Group names that a principal belongs to. Groups and users can belong to other groups. |

Resource URL

GET

</tspublic/v1/user/list>

Request Example

cURL

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

Request URL

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


Response Example

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

Group API

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

Add a privilege

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

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

Resource URL

POST `/tspublic/v1/group/addprivilege`

Request Parameters

| Form Parameter | Data Type | Description |
|----------------|-----------|---|
| privilege | string | Specifies a privilege type to add. Valid values are <code>DATADOWNLOADING</code> or <code>USERDATAUPLOADING</code> privilege. |
| groupNames | string | Specifies a group name to add the privilege to. Valid value is <code>ALL_GROUP</code> group. |

Request Example

cURL

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

Request URL

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

Response Example

```
Not applicable  
204 – Success
```

Remove a privilege

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

Resource URL

```
POST /tspublic/v1/group/removeprivilege
```

Request Parameters

| Form Parameter | Data Type | Description |
|----------------|-----------|--|
| privilege | string | Specifies a privilege type to delete. Valid values are <code>DATADOWNLOADING</code> or <code>USERDATAUPLOADING</code> privilege. |
| groupNames | string | Specifies a group name to delete the privilege from. Valid value is <code>ALL_GROUP</code> group. |

Request Example

cURL

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

Request URL

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

Response Example

```
Not applicable  
204 – Success
```

Materialization API

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

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

Note: To refresh a materialized view, you must have the [Can administer ThoughtSpot](#) privilege.

Resource URL

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

Request Parameters

| Path Parameter | Data Type | Description |
|-----------------|-----------|---------------------------|
| <code>id</code> | string | ID of the metadata object |

Request Example

cURL

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

Request URL

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

Response Example

```
Not applicable  
204 – Refresh submitted
```