



# Application Integration Guide

Version 4.2  
February 2017

---

# Contents

|                                                    |    |
|----------------------------------------------------|----|
| Chapter 1: Introduction.....                       | 4  |
| Login credentials for administration.....          | 5  |
| Log in to ThoughtSpot from a browser.....          | 5  |
| Log in to the Linux shell using SSH.....           | 6  |
| <br>                                               |    |
| Chapter 2: About SAML.....                         | 7  |
| Legacy configure SAML.....                         | 8  |
| Configure SAML.....                                | 10 |
| Configure CA SiteMinder.....                       | 11 |
| Configure Active Directory Federated Services..... | 14 |
| Initialize the Identity Provider Metadata.....     | 14 |
| Initialize the Service Provider Metadata.....      | 15 |
| Test the ADFS Integration.....                     | 16 |
| <br>                                               |    |
| Chapter 3: About the JavaScript API.....           | 17 |
| Enable the JavaScript API.....                     | 20 |
| <br>                                               |    |
| Chapter 4: About the REST API.....                 | 23 |
| Calling the REST API.....                          | 25 |
| REST API pagination.....                           | 28 |
| Use the REST API to get data.....                  | 30 |
| <br>                                               |    |
| Chapter 5: About Embedding.....                    | 34 |
| Embed a visualization.....                         | 35 |
| About full embed.....                              | 36 |
| <br>                                               |    |
| Chapter 6: About Runtime Filters.....              | 39 |

---

|                                           |    |
|-------------------------------------------|----|
| Apply a Runtime Filter.....               | 42 |
| Runtime Filter Operators.....             | 43 |
| <br>                                      |    |
| Chapter 7: About Style Customization..... | 45 |
| Perform Style Customization.....          | 46 |
| Upload application logos.....             | 47 |
| Choose a background color.....            | 48 |
| Select chart color palettes.....          | 49 |
| Change the footer text.....               | 51 |
| <br>                                      |    |
| Chapter 8: Reference guide.....           | 53 |
| Public API reference.....                 | 54 |

---

## Chapter 1: Introduction

---

Topics:

- [Login credentials for administration](#)
- [Log in to ThoughtSpot from a browser](#)
- [Log in to the Linux shell using SSH](#)

This guide explains how to integrate ThoughtSpot with other applications, including authentication, embedding, and APIs.

For information on how to integrate with other data sources for loading data, refer to the *Data Integration Guide*.

## Login credentials for administration

You will need administrative permissions to perform the actions discussed in this guide. You can access ThoughtSpot via SSH at the command prompt and from a Web browser.

There are two separate default administrator users, an operating system user that you type in at the Linux shell prompt, and an application user for access through a browser. Make sure you use the correct login and password for the method you are using to log in. Passwords are case sensitive.

Table 1: Default administrative user credentials

| Login Type       | User    | Access Method                                                                        | Password                                            |
|------------------|---------|--------------------------------------------------------------------------------------|-----------------------------------------------------|
| OS user          | admin   | <a href="#">Access remotely via SSH from the command prompt on a client machine.</a> | Contact ThoughtSpot to obtain the default password. |
| Application user | tsadmin | <a href="#">Access through a Web browser.</a>                                        | Contact ThoughtSpot to obtain the default password. |

## Log in to ThoughtSpot from a browser

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

Before accessing ThoughtSpot, you need:

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

Supported Web browsers include:

Table 2: Supported browsers

| Browser           | Version      | Operating System                                                                                     |
|-------------------|--------------|------------------------------------------------------------------------------------------------------|
| Google Chrome     | 20 and above | <ul style="list-style-type: none"> <li>Windows 7 or greater</li> <li>Linux</li> <li>MacOS</li> </ul> |
| Mozilla Firefox   | 14 and above | <ul style="list-style-type: none"> <li>Windows 7 or greater</li> <li>Linux</li> <li>MacOS</li> </ul> |
| Internet Explorer | 11           | <ul style="list-style-type: none"> <li>Windows 7 or greater</li> </ul>                               |

To log in to ThoughtSpot from a browser:

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

```
http://<hostname_or_IP>
```

2. Enter your username and password and click **Enter Now**.

## 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.

---

## Chapter 2: About SAML

---

Topics:

- [Legacy configure SAML](#)
- [Configure SAML](#)
- [Configure CA SiteMinder](#)
- [Configure Active Directory Federated Services](#)

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

## Legacy 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.

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

1. [Log in to the Linux shell using SSH.](#)
2. Change directories to the SAML directory:

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

3. Open the file `applicationContext-security.xml` in `vi` or another editor.
  - a) Find the section labeled “Entry point to initialize authentication, default values taken from properties file”.
  - b) Edit the value for the property `entityBaseURL` to supply the IP address of the server you want to use. Only supply the port (e.g. `:8080`) if the IP address for your server includes it, otherwise omit it. Be sure to use either `http:` or `https:`, depending on how your server is configured:

```
88 <property name="entityBaseURL" value="https://<your
server IP>:443/callosum/v1" />
```

- c) The next line contains the property `entityId`. The default value is “`urn:thoughtspot:callosum:saml`”. Change “`thoughtspot`” to the name of your cluster:

```
89 <property name="entityId" value="urn:<your
cluster>:callosum:saml"/>
```

- d) Find the section labeled “Provider of default SAML Context”. Edit the SAML context to change the IP address to your server’s IP. The default port is 80 for `http`, and 443 for `https`:

```
142 <property name="scheme" value="https"/>
143 <property name="serverName" value="<your server IP>"/>
```



```
144 <!-- Replace the value 8080 of serverPort with port of the load
balancer-->
145 <property name="serverPort" value="443"/>
```

e) Save the edited file.

#### 4. Change directories to the callosum directory:

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

#### 5. Open the file `callosumconfig_prod.json` in vi or another editor.

a) Set up autocreation of users by adding the following line above

"shiroConfig":

```
89 "shouldCacheLogicalModel": true,
90 "autoCreateUserFromLDAP":true,
91 "shiroConfig": {
```

b) Add the SAML realm as shown:

```
112 "adLdapRealm.domain": "ldap.thoughtspot.com",
113 "securityManager.realms": "$callosumRDBMSRealm, $callosumSamlRealm",
114 "authcStrategy":
    "org.apache.shiro.authc.pam.AtLeastOneSuccessfulStrategy",
```

c) Enable SAML as shown:

```
144 "enableNotificationOnShare": true,
145 "samlConfiguration": {
146 "enabled": true
147 }
```

#### 6. Restart Tomcat using these commands:

```
$ cd /usr/local/scaligent/release
$ tscli --adv service push tomcat /usr/local/scaligent/release/production/
orion/tomcat/tomcat_prod.config
```

7. After restarting Tomcat, open a Web browser and go to the ThoughtSpot login page. It should now show the Single Sign On option.

8. Retrieve the metadata by navigating to `https://<your server IP>/callosum/v1/saml/metadata`. The SP metadata file will download. Save it as `metadata.xml`. You will need this file when configuring your SAML service provider.

9. If you're using one of these SAML service providers, continue your configuration using these instructions:

- [Configure CA SiteMinder.](#)
- [Configure Active Directory Federated Services.](#)

Otherwise, refer to your SAML service provider for instructions how to import the metadata.

## 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 will need this information:

- IP of the server where your ThoughtSpot instance is running.
- Port of the server where your ThoughtSpot instance is running.
- Protocol, or the authentication mechanism for ThoughtSpot.
- Unique service name that is used as the unique key by IDP to identify the client.

It should be in the following format: `urn:thoughtspot:callosum:saml`

- Allowed skew time, which is the time after authentication response is rejected and sent back from the IDP. It is usually set to 86400.
- The absolute path to the idp-meta.xml file. This is needed so that the configuration persists over upgrades.
- 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, so set it to true if you do not know what it does.

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

1. [Log in to the Linux shell using SSH.](#)

- Execute the command to launch the interactive SAML configuration:

```
tscli saml configure
```

- Complete the configurator prompts with the information you gathered above.
- 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

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:

- Configure the Local Identity Provider Entity as follows:

Table 3: Configure Local Identity Provider Entity Settings

| 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 via a metadata import or manually. To configure the Remote SP entity manually, select **Create Entity**.

Create ThoughtSpot as a Remote Entity with following details:

Table 4: ThoughtSpot Remote Entity Settings

| 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:

Table 5: Configure Partnership settings

| Section              | Entry                      |
|----------------------|----------------------------|
| Add Partnership Name | Any (relevant name)        |
| Description          | Any (relevant description) |
| Local IDP ID         | Select Local IDP ID        |

| Section                           | Entry                                                |
|-----------------------------------|------------------------------------------------------|
| 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:

Table 6: Configure Assertion settings

| 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:

Table 7: Configure SSO and SLO settings

| 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

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. Restart Tomcat using these commands:

```
$ cd /usr/local/scaligent/release
$ tscli --adv service push tomcat /usr/local/scaligent/release/production/orion/tomcat/tomcat_prod.config
```

6. Next, [Initialize the Service Provider Metadata.](#)

## Initialize the Service Provider Metadata

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.



## Chapter 3: About the JavaScript API

Topics:

- [Enable the JavaScript API](#)

Use the ThoughtSpot JavaScript API to embed data or visualizations from ThoughtSpot in your own Web portal, application, or page.

### JavaScript API Capabilities

The ThoughtSpot JavaScript API (JS API) allows you to use your ThoughtSpot instance within your own Web application. The JS API has methods that allow you to:

- Authenticate to ThoughtSpot.
- Embed visualizations from ThoughtSpot in your Web page using iframes.
- Use the ThoughtSpot REST API to get data from ThoughtSpot and use it in your Web page.

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

### Browser Support

The JS API works in the following browsers:

Table 8: Web browsers supported by the JS API

| Browser           | Versions    |
|-------------------|-------------|
| Internet Explorer | 11          |
| Firefox           | 38 or later |

| Browser       | Versions    |
|---------------|-------------|
| Google Chrome | 47 or later |
| Safari        | 9 or later  |



**Note:** Microsoft introduced a compatibility mode in Internet Explorer 10, which displays your page using the version of Internet Explorer that is most compatible with the current page. Since we do not support any version below 11, this feature can sometimes break the code. There are two ways to force the emulation of Internet Explorer to the most up to date version:

- Add a Custom Response Header

This is the recommended approach since it is more robust, offers more control, and has a lower risk of introducing a bug to your code. The header name should be set to "X-UA-Compatible" and the value should be set to "IE=Edge". The response header should be based on the server it is set on and the technology being used.

- Add a Meta Tag

The following meta tag should be added to your header: `<meta http-equiv="X-UA-Compatible" content="IE=Edge" />`. This tag must be the first tag in the header section of the page.

### **Cross-Origin HTTP Requests**

Because you'll be making a call from your own Web page, portal, or application to ThoughtSpot, which has a different domain, you'll need to enable cross-origin HTTP requests. This controls what domains are allowed to use this code to authorize users and prevents other people from copying your code and running it on their site. For example, if your Web site is hosted on the domain example.com, you would need to set the following origin for your client ID: `http://example.com`. If you want to test your code locally, you'll also need to add the origin for your local server as well, for example: `http://localhost:8080`.

## Enable the JavaScript API

---

This procedure shows how to include the ThoughtSpot JavaScript API (JS API) in your web page, and then use it to authenticate to ThoughtSpot.

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.

If your ThoughtSpot system is configured for Single Sign On (SSO), the JS API call `window.thoughtspot.initialize` can cause the entire Web page to be redirected to your Identity Provider (IDP). This implies that you may not execute any of your application logic before `window.thoughtspot.initialize` has called your callback, because any possible redirection could interfere with your application logic. The recommended way of achieving this is to:

1. Place the JS API in the `<head>` section of the HTML on your Web page.
2. Ensure that the JS API script tag is the first script to be loaded in the page.
3. Ensure that you 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.

Note that `onAuthenticationExpiredCallback` is only available if you have at least one ThoughtSpot visualization `iframe` in your web page.

To enable the JS API:

1. Navigate to the Downloads page in the Help Center to download the ThoughtSpot JS API JavaScript file.
2. Include the ThoughtSpot JS API JavaScript file into your web page using an HTML include script like this in the `<head>` section:

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

3. From your application code, authenticate to ThoughtSpot using a call to the JavaScript method `window.thoughtspot.initialize` (`onInitializationCallback`, `onAuthenticationExpiredCallback`, `<Hostname_or_IP>`)

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. Set up CORS (Cross-Origin HTTP Request) to control what domains are allowed to use this code to authorize users:


a) [Log in to the Linux shell using SSH](#).

b) Issue the command to set the domains that will be allowed to access ThoughtSpot using the JS API using this syntax:

```
echo "https?://(localhost|.*:443)" | tscli --adv config set
--key "/config/nginx/corshosts"
```

When supplying an IP address, you have to escape the dots with a triple backslash (\) as shown in this example:

```
$ echo "https?://(localhost|10\\\.77\\\.20\\\.87:443)"
| tscli --adv config set --key "/config/nginx/corshosts"
```

 **Note:** By default this value is set to empty, to disallow any cross domain access. When this value is changed, the nginx service will be restarted automatically to reflect the change.

5. 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.
6. Test your Web page or application. If your the page no longer works, check the JavaScript console on your Web browser. If you see this error message, it means that the CORS cross domain setting on nginx was not completed correctly:

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

If this happens, go back to the step in this procedure where you set up CORS.

---

## Chapter 4: About the REST API

---

Topics:

- [Calling the REST API](#)
- [Use the REST API to get data](#)

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

### **Cross Domain Verification**

You'll need to enable cross domain verification when using the 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.

### **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'll need to 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.

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.

#### **Public API reference**

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



## Calling the REST API

---

To call the REST API, you'll 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'll 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
```

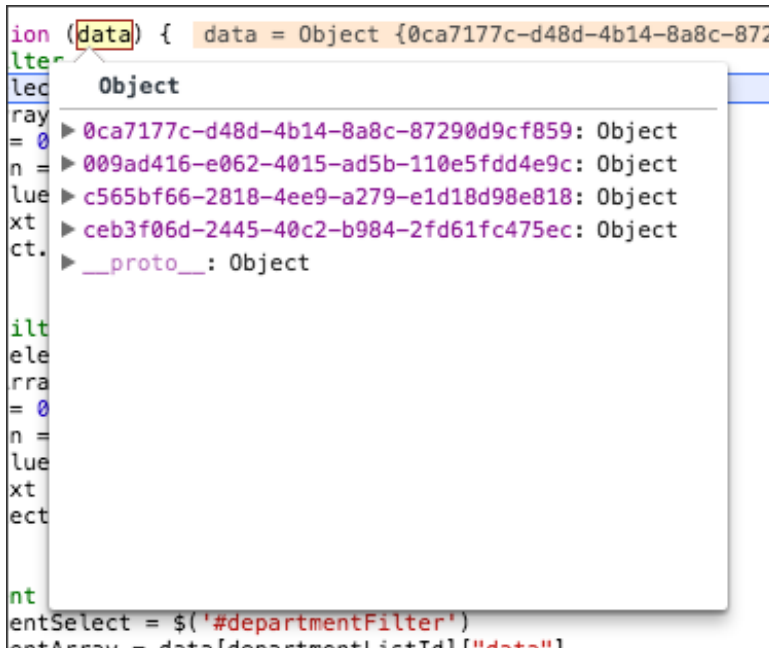
To retrieve data from a specific visualization within a pinboard, you would append the ID number of the visualization using the vizid parameter:

```
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:

**Figure 1: Parsed JSON data**

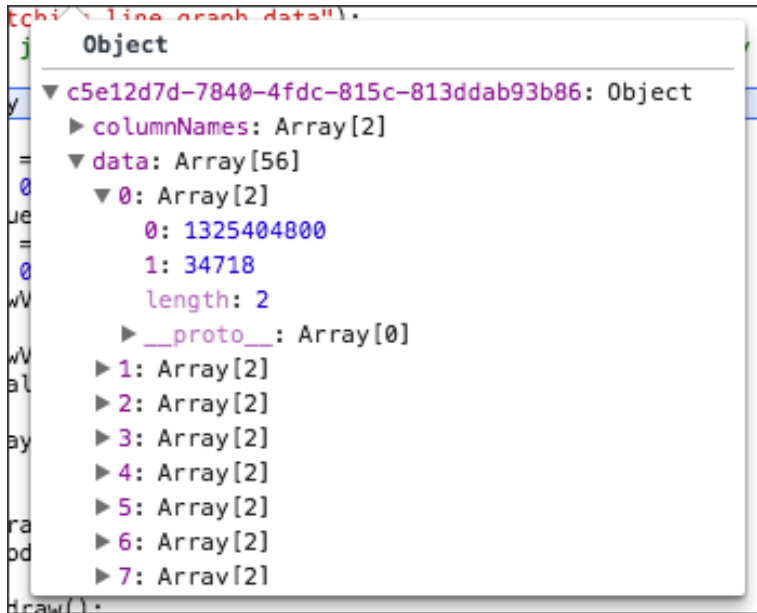
```

{
  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.



**Figure 2: columnNames and data arrays**

## Data Filters

The REST API supports filtering the data returned via 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/v1/" +
      "pinboarddata?id=" + pinboardId + "&" +
    
```

```

        "vizid=%5B" + vizId + "%5D";
    } else {
        var query = getQueryString(filters);
        myURL = "http://192.168.2.55:443/callosum/v1/tspublic/v1/" +
            "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("embedded-headline");
        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();
}

```

## REST API pagination

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 REST API.

To paginate results in your API response, you'll need to 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.

```

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

```

**!** **Important:** 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.

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"
    }

```

**📄** **Note:** 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 REST API to get data

---

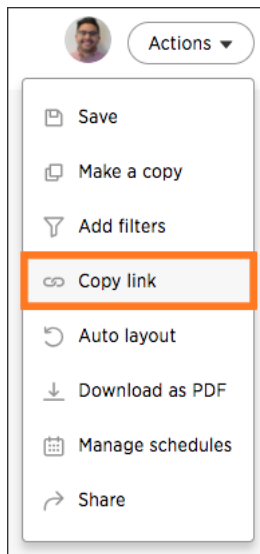
This procedure shows how to use the REST API to get data out of ThoughtSpot, so you can use it in a Web page, portal, or application. The data will be returned as JSON (JavaScript Object Notation).

Before you can use the REST API, you need to:

1. [Enable the JavaScript API \(JS API\)](#) and authenticate to ThoughtSpot.

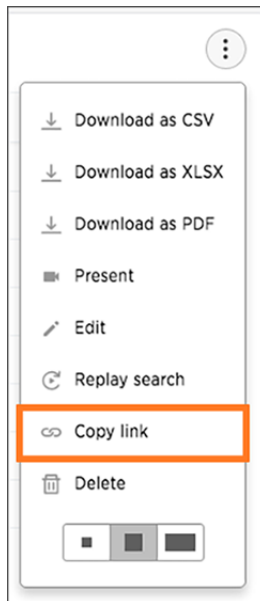
Use this procedure to construct the URL you will use to call the REST API:

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 **Actions** and select **Copy Link**.



**Figure 3: The Actions menu**

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

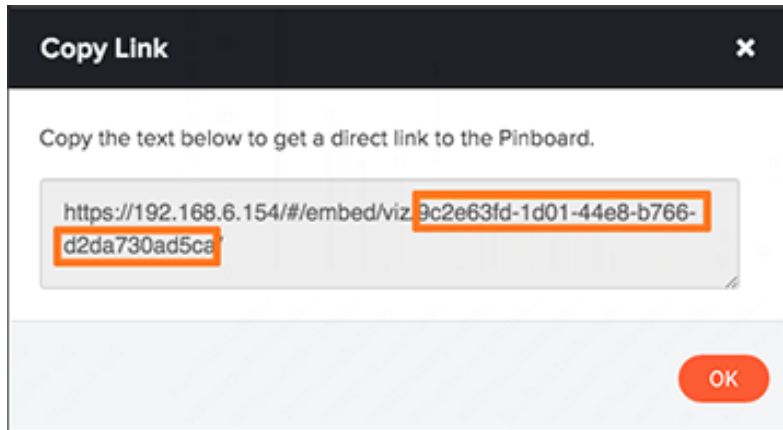


**Figure 4: Copy link**

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.

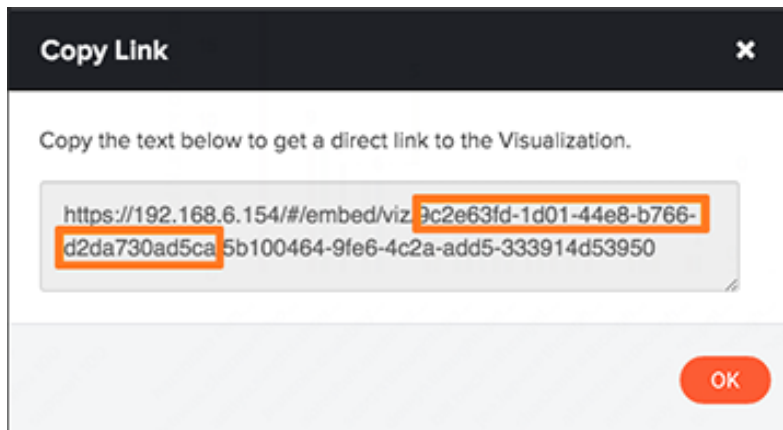
If the object is:

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



**Figure 5: The pinboard ID**

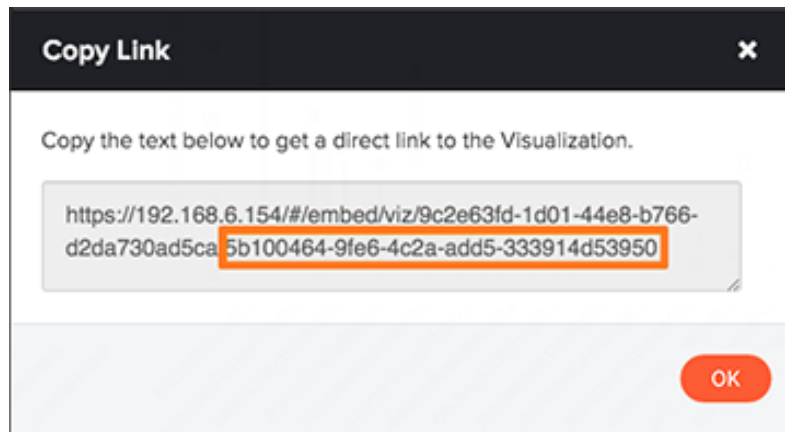
- A visualization (table or chart), copy the identifier that appears after "viz/". This is the pinboard ID.



**Figure 6: The pinboard ID**

Then skip the "/" and copy what follows it. This is the visualization ID.





**Figure 7: The visualization ID**

- 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
```

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

## Chapter 5: About Embedding

---

Topics:

- [Embed a visualization](#)
- [About full embed](#)

Embedding allows you to display a pinboard from ThoughtSpot on your own Web page, Web portal, or application. You can also enable full embedding, which allows you to create content in an embedded environment. When using Embedding, authentication is achieved through SAML.

After authentication, a URL is provided to call the desired visualization and populate it into an iframe.

When using this method, the visualization is rendered within an iframe on your Web page, portal, or application.

### **Authentication**

Before you can embed a visualization, you must authenticate to ThoughtSpot using SAML with the [JavaScript API](#).

### **Cross Domain Verification**

When using Embedding, you will use cross domain verification. This protects your data, so that another website cannot use the same URL to embed the visualization in its own Web pages. The procedure for [enabling the JavaScript API](#) authentication includes information on how to enable this.

## Embed a visualization

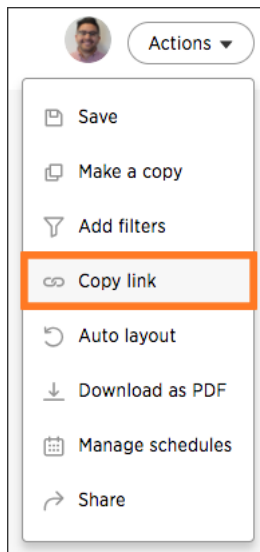
Embedding allows you to include a visualization (table or chart) or pinboard from ThoughtSpot in your own static Web page, Web portal, or application.

Before you can embed a visualization, you need to:

1. [Enable the JavaScript API \(JS API\)](#) and authenticate to ThoughtSpot.

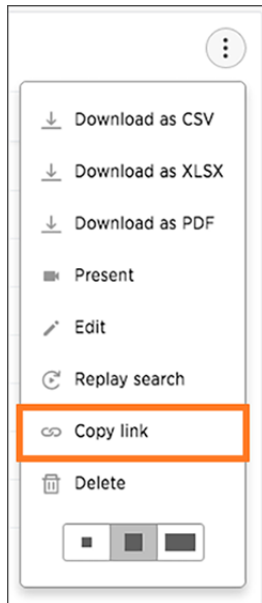
Use this procedure to construct the URL you will use to embed a visualization:

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 link for the object you want to get the data from. If the object is:
  - A pinboard, click **Actions** and select **Copy Link**.



**Figure 8: The Actions menu**

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



**Figure 9: Copy link**

4. Copy the link shown, and paste it into the iframe in your Web page, Web portal, or application.

## About full embed

---

Full embedding enhances our existing external sharing functionality. It allows you to create content in an embedded environment.

You can embed the full search experience into an iframe with different navigation views and toggle options. Before you can enable full embed, you need to [enable the JavaScript API \(JS API\)](#) and authenticate to ThoughtSpot.

Embedded content creation gives you the ability to:

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

You won't be able to:

- create worksheets, both regular and aggregated.
- modify profiles.
- view the Help Center.

### Demo code for sample full embed app (thoughtspot/blink/app/embed.html)

```

<!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"></script>
<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>

<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 you click on different navigation buttons.

The URL flag `embedApp=true` is used for the `iframe` source. The default value for it is set to `true`.

Here are some additional notes about the full embed feature:

- You can call `thoughtspot.<customerURL>.com/#/answer` and use that to access the search functionality.
- You can call `thoughtspot.<customerURL>.com/#/pinboards` and use that to access saved pinboards.
- 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.
- You can use SAML for authentication against ThoughtSpot within the `iFrame`.
- ThoughtSpot can hide the top navigation within the ThoughtSpot app. Contact ThoughtSpot Support to request this modification.
- ThoughtSpot can hide the expandable left panel that shows data sources and fields within the current data source. Contact ThoughtSpot Support to request this modification.

## Chapter 6: About Runtime Filters

Topics:

- [Apply a Runtime Filter](#)
- [Runtime Filter Operators](#)

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/?  
coll=Color&op1=EQ&vall=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&coll=Category  
&op1=EQ&vall=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**

Runtime Filters do not work directly on top of tables. You need to 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 (i.e. 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. 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, and paste it into a text editor.
2. 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>/
?coll=<column_name>&op1=<operator>&vall=<value>
#/pinboard/<pinboard_id>
```

- For Embedding a visualization:

```
http://<thoughtspot_server>:<port>/
?coll=<column_name>&op1=<operator>&vall=<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>
&coll=<column_name>&op1=<operator>&vall=<value>
```

- For the REST API with a visualization:

```
http://<thoughtspot_server>:<port>
/callosum/v1/tspublic/v1/pinboarddata
```

```
?id=<pinboard_id>&vizid=%5B<visualization_id>%5D
&coll=<column_name>&op1=<operator>&vall=<value>
```

3. If you want to add additional filters on a particular column, you can specify multiple values by separating them with "&" as in the example:

```
vall=foo&vall=bar
```

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

```
coll=<column_name>&op1=IN&vall=<value>&vall=<value>
```

4. Add additional filters by incrementing the number at the end of each parameter in the Runtime Filter for each filter you want to add, e.g. 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.

```
coll=region&op1=IN&vall=midwest&vall=south&vall=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.

Table 9: Runtime Filter Operators

| 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                |

| Operator   | Description                           | Number of Values |
|------------|---------------------------------------|------------------|
| 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         |

---

## Chapter 7: About Style Customization

---

Topics:

- [Perform Style Customization](#)

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

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 a premium feature, that can be enabled at additional cost. To enable style customization, contact ThoughtSpot Support.

## Perform 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.

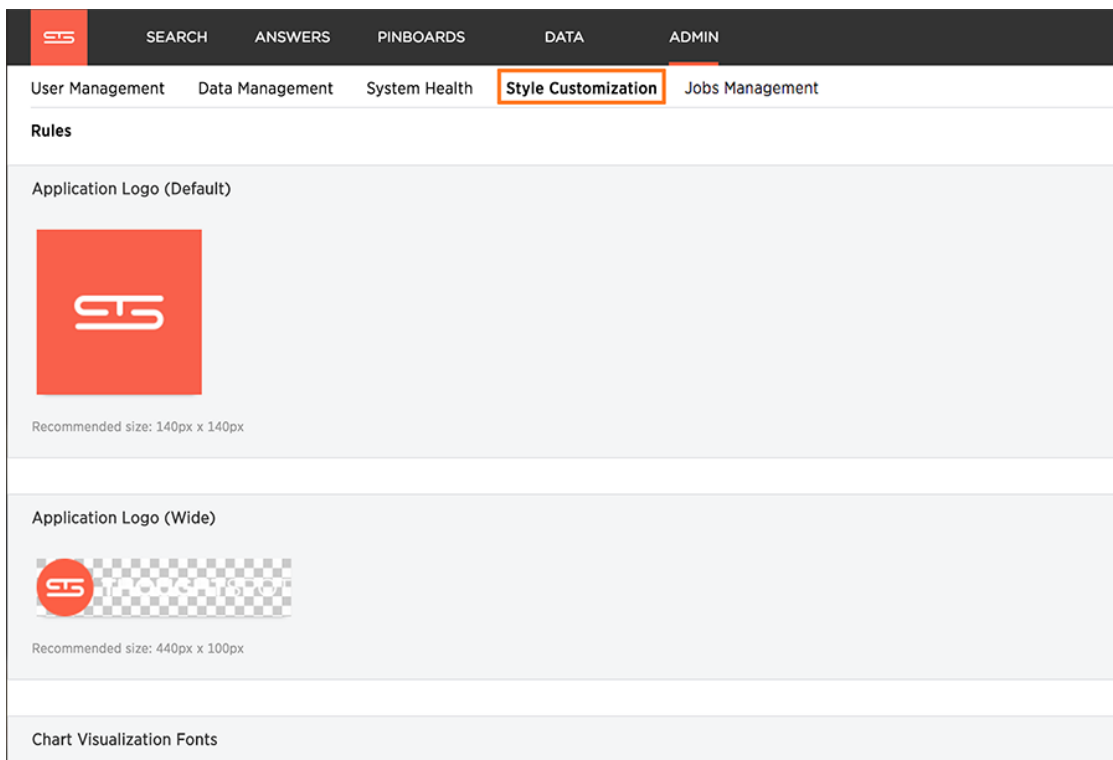
To re-brand the interface:

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



**Figure 10: The Admin icon**


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



**Figure 11: Style Customization menu**

Once in the menu page, you can:

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

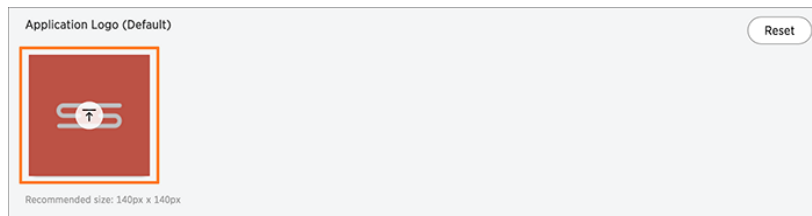
 **Note:** The ThoughtSpot logo in the middle of the page is automatically removed when Style Customization is enabled.

## Upload application logos

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

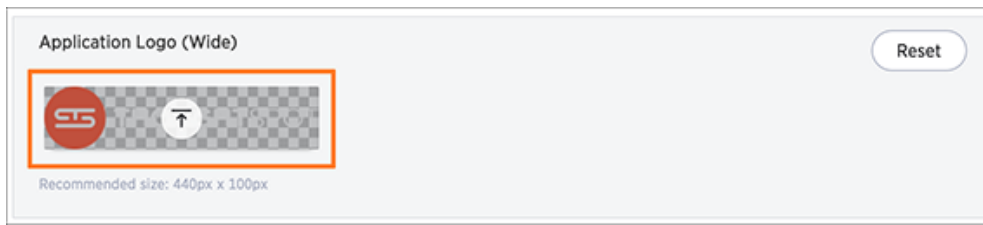


**Figure 12: Application Logo (Default)**

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



**Figure 13: Application Logo (Wide)**

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.

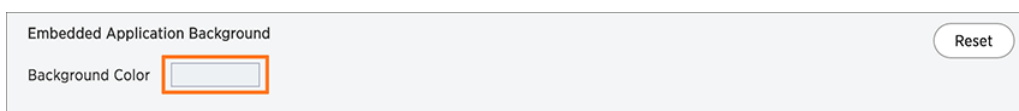
3. Click the **Reset** button on the upper right hand side of the sections if you would like to bring back the default logos.

### 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:

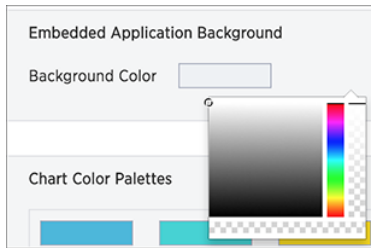
1. Click the background color box under **Application Background**.



**Figure 14: Application Background menu**

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





**Figure 15: Application Background Color**

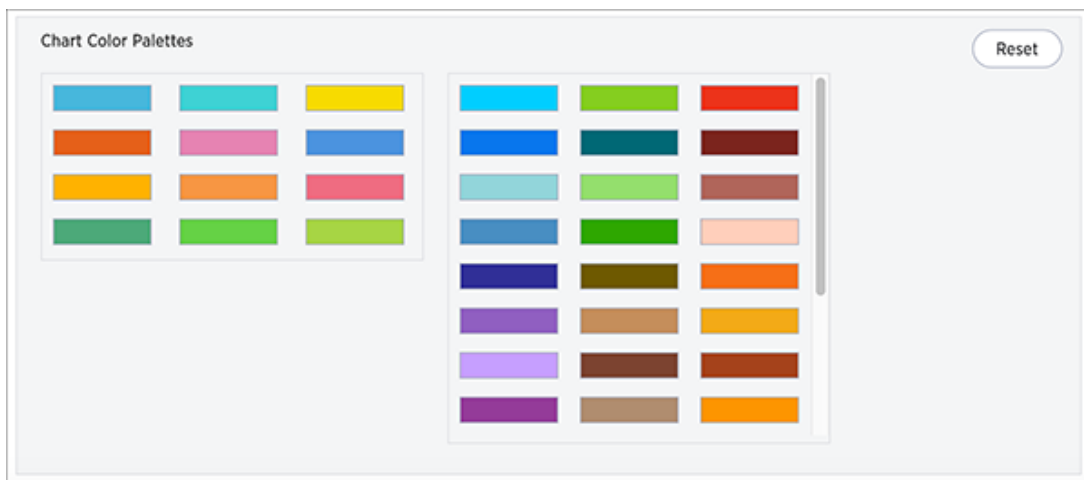
3. Click the **Reset** button on the upper right hand side of the section if you would like to bring back the default color.

### 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.

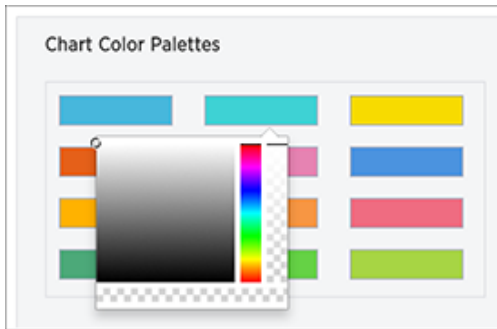
To select the chart color palettes:

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



**Figure 16: Chart Color Palettes section**

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



**Figure 17: Primary color palette**

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



**Figure 18: Secondary color palette**

The colors from the secondary color palette are used once 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

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

To change the footer text:

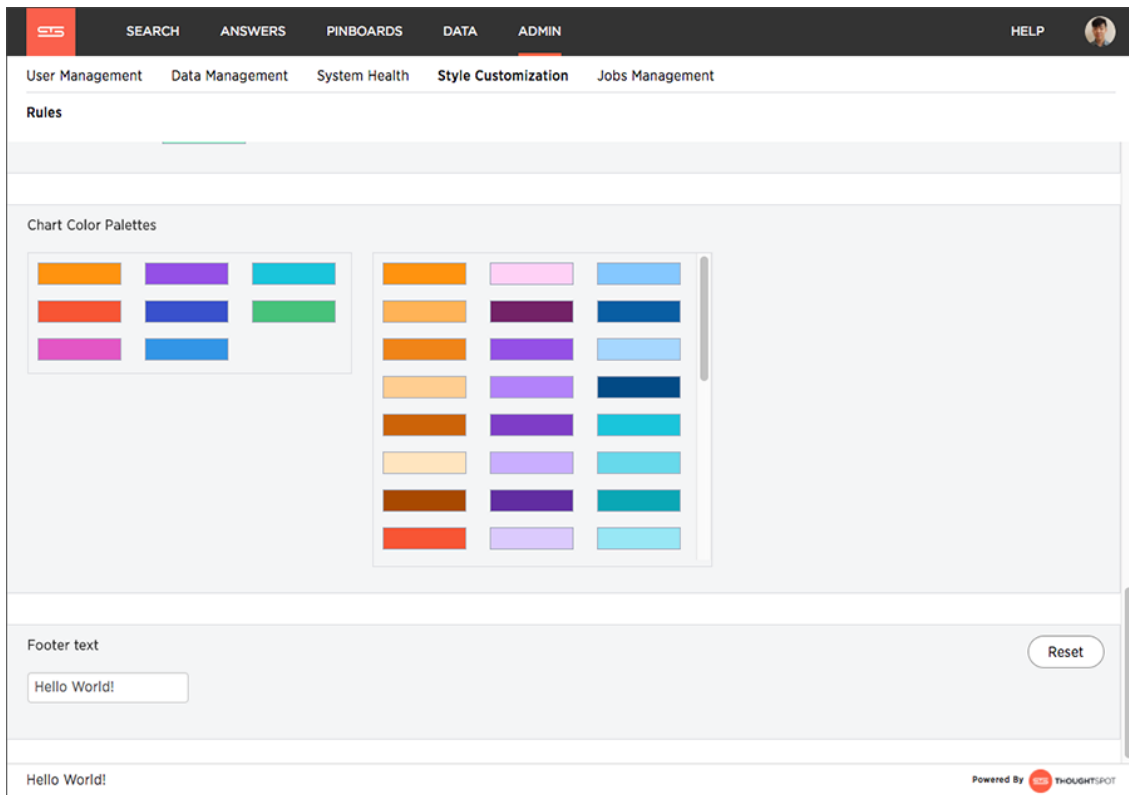
1. Click the text box under **Footer text**.
2. Enter your new text message.



The image shows a light gray rectangular panel. At the top left, the text "Footer text" is displayed. Below it, there is a white text input field with a thin gray border. Inside the input field, the text "Hello World!" is entered.

**Figure 19: Footer text menu**

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



**Figure 20: Footer text display**

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.

## Chapter 8: Reference guide

---

Topics:

- [Public API reference](#)

This Reference Guide contains ThoughtSpot's public APIs and endpoints.

Included in this guide are:

- [Public API reference](#) lists the available ThoughtSpot public APIs and their responses.

## Public API reference

This is a list of 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.

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

### Public name: pinboarddata

- Public namespace: Data
- Current URL path: /tspublic/v1/pinboarddata
- Implementation notes: Gets the pinboard data from the ThoughtSpot system. Returns one object if you make a call to a specific visualization on a pinboard.

Table 10: Input parameters

| Parameter  | Value   | Description                                                                                                                               | Parameter Type | Data Type |
|------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------|
| id         |         | GUID id of the pinboard                                                                                                                   | query          | string    |
| vizid      |         | Optional GUID ids of the visualizations                                                                                                   | query          | string    |
| batchsize  | -1      | Batch size                                                                                                                                | query          | integer   |
| pagenumber | -1      | PageNumber. Alternate way to set offset. This is 1-based indexing<br>$\text{indexingOffset} = (\text{pageNumber} - 1) * \text{batchSize}$ | query          | integer   |
| offset     | -1      | Offset                                                                                                                                    | query          | integer   |
| formattype | COMPACT | JSON format type. Values allowed are FULL and COMPACT.                                                                                    | query          | string    |

Table 11: Response messages

| HTTP Status Code | Reason                                     | Response Model | Headers |
|------------------|--------------------------------------------|----------------|---------|
| 200              | Gets the data of a pinboard/visualization. |                |         |
| 400              | Invalid pinboard id.                       |                |         |
| default          |                                            |                |         |

Response example:

#### Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' 'https://dogfood/callosum/v1/tspublic/v1/pinboarddata?batchsize=-1&pagenumber=-1&offset=-1&formattype=COMPACT'
```

#### Request URL

```
https://dogfood/callosum/v1/tspublic/v1/pinboarddata?batchsize=-1&pagenumber=-1&offset=-1&formattype=COMPACT
```

#### Response Body

```
no content
```

#### Response Code

```
400
```

#### Response Headers

```
{
  "x-callosum-incident-id": "2ff9d2e4-c928-4192-8250-8450de264ab7",
  "x-callosum-trace-id": "2e551b8d-d3f4-4cf1-af90-a49bb246ad92",
  "date": "Sun, 19 Feb 2017 03:39:41 GMT",
  "x-callosum-request-time-us": "11536",
  "server": "nginx",
  "pragma": "no-cache",
  "cache-control": "no-store, no-cache, must-revalidate, max-age=0, post-check=0, pre-check=0",
  "content-security-policy": "script-src 'self'",
  "connection": "keep-alive",
  "content-length": "0",
  "x-callosum-ip": "192.168.2.247",
  "content-type": null
}
```

### Public name: **pinboard/listvizheaders**

- Public namespace: Objects
- Current URL path: /tspublic/v1/metadata/listvizheaders
- Implementation notes: Gets 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.

Table 12: Input parameters

| Parameter | Value | Description            | Parameter Type | Data Type |
|-----------|-------|------------------------|----------------|-----------|
| id        |       | GUID of the reportbook | query          | string    |

Table 13: Response messages

| HTTP Status Code | Reason                          | Response Model | Headers |
|------------------|---------------------------------|----------------|---------|
| 200              | Gets the visualization headers. |                |         |
| 400              | Invalid pinboard GUID.          |                |         |
| default          |                                 |                |         |

Response example:

Curl

```
curl -X GET --header 'Accept: application/json' 'https://dogfood/callosum/v1/tspublic/v1/metadata/listvizheaders'
```

Request URL

```
https://dogfood/callosum/v1/tspublic/v1/metadata/listvizheaders
```

Response Body

```
[]
```

Response Code

```
200
```



## Response Headers

Header of each visualization header object obtained will have these first class citizens:

1. id (GUID)
2. name (String)
3. author (GUID)
4. created (Epoch)
5. modified (Epoch)
6. modifiedBy (GUID)
7. owner (GUID)
8. vizType (String : CHART, TABLE, etc.)
9. title

```
{
  "x-callosum-incident-id": "6f10c360-1dc8-4469-8d7d-41c7eab4d883",
  "x-callosum-trace-id": "70d91ca9-7278-4926-8161-bec75bfe3c5e",
  "date": "Sun, 19 Feb 2017 03:19:17 GMT",
  "content-encoding": "gzip",
  "x-callosum-request-time-us": "3437",
  "server": "nginx",
  "vary": "Accept-Encoding",
  "content-type": "application/json",
  "pragma": "no-cache",
  "cache-control": "no-store, no-cache, must-revalidate, max-age=0,
post-check=0, pre-check=0",
  "transfer-encoding": "chunked",
  "content-security-policy": "script-src 'self'",
  "connection": "keep-alive",
  "x-callosum-ip": "192.168.2.247",
  "x-ua-compatible": "IE=edge"
}
```

### Public name: principal/sync

- Public namespace: Configuration
- Current URL path: /tspublic/v1/user/sync
- Implementation notes: API to synchronize principal from external system with ThoughtSpot system. This API is for users and groups. It should help to keep

ThoughtSpot users and groups automatically synchronized with your external database.

Specifically, you will have to make a call to `/tspublic/v1/user/sync` containing all users and groups present in the external database. If the call succeeds, then it is guaranteed that the users and groups in ThoughtSpot match those specified in the list of objects passed to `/tspublic/v1/user/sync`. This means that:

- Objects (users or groups) present in ThoughtSpot, but not present in the list passed to a sync call will be deleted.
- Objects present in ThoughtSpot, and present in the list passed to a sync call 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 list will be created in ThoughtSpot.

The returned object represents the changes that were made in ThoughtSpot.

Table 14: Input parameters

| Parameter    | Value | Description                                                                                                                                                                                         | Parameter Type | Data Type |
|--------------|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------|
| principals   |       | formData. List of principal objects.                                                                                                                                                                | string         |           |
| applyChanges |       | formData. Flag to indicate whether to sync the users and groups to the system, and apply the difference evaluated. The API can be used to just validate the difference before applying the changes. | boolean        |           |

| Parameter       | Value | Description                                                                                                          | Parameter Type | Data Type |
|-----------------|-------|----------------------------------------------------------------------------------------------------------------------|----------------|-----------|
| defaultPassword |       | formData. If set then all of the created users will have a password that is the same as the defaultPassword applied. | string         |           |

Table 15: Response messages

| HTTP Status Code | Reason | Response Model | Headers |
|------------------|--------|----------------|---------|
| 200              |        |                |         |
| default          |        |                |         |

Response example:

Curl

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

Request URL

```
https://dogfood/callosum/v1/tspublic/v1/user/sync
```

Response Object Format

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

Response Code

```
415
```

Response Headers

```
{
```

```

"x-callosum-incident-id": "645499d1-d0cf-4b3b-bbdc-4296abb9a326",
"x-callosum-trace-id": "19f7ad7d-226a-4e88-a301-405f85125959",
"date": "Sun, 19 Feb 2017 03:55:52 GMT",
"x-callosum-request-time-us": "4545",
"server": "nginx",
"pragma": "no-cache",
"cache-control": "no-store, no-cache, must-revalidate, max-age=0,
post-check=0, pre-check=0",
"content-security-policy": "script-src 'self'",
"connection": "keep-alive",
"content-length": "0",
"x-callosum-ip": "192.168.2.247",
"content-type": null
}

```

### Public name: principal/list

- Public namespace: Configuration
- Current URL path: /tspublic/v1/user/list
- Implementation notes: API to get a list of all users, groups, and their inter-dependencies in the form of principal objects. This API is for users and groups.

Input Principal Object Format:

One principal object contains the following properties (\* denotes required properties):

- name\*: String to represent the name of the principal.
  - 📌 **Note:** This field, in conjunction with whether the object is a user or group, is used to identify a user/group. Consequently, this field is required to be unique (unique for users and groups separately. I.e. you can have user “x” and group “x”).
- displayName\*: String to represent the display name of the principal.
- description: String to describe the principal.
- mail: String to represent the email address of the user. This field should be populated only in case of user not group. It is ignored in the case of groups.
- principalTypeEnum\*: The value of this field should be one of the following:

- **LOCAL\_USER**: User created in the ThoughtSpot system and the validation of the user is done through password saved in the ThoughtSpot database.
- **LOCAL\_GROUP**: Groups created in the ThoughtSpot system.
- **password**: String to represent the password of the user. This field should be only populated in case of user not group. It is ignored in the case of groups. Also password is only required if the user is of LOCAL\_USER type. Password is only required when the user is created for the first time. In subsequent update of the user password is not updated even if it changes in the source system.
- **groupNames**: List of group names that a principal belongs to directly. Groups and users can belong to other groups.

Table 16: Response messages

| HTTP Status Code | Reason | Response Model | Headers |
|------------------|--------|----------------|---------|
| 200              |        |                |         |
| default          |        |                |         |

Response example:

Curl

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

Request URL

```
https://dogfood/callosum/v1/tspublic/v1/user/list
```

Response Body format

```
[
  {
    "name": "Group 1",
    "displayName": "Group Display Name 1",
    "description": "Group Description 1",
    "principalTypeEnum": "LOCAL_GROUP",
    "groupNames": []
  },
  {
```

```

    "name": "Test Name",
    "displayName": "Test DisplayName",
    "principalTypeEnum": "LOCAL_USER"
    "password": "password_123",
    "groupNames": ["Group_1"]
  }
]

```

### Response Body example

```

[
  {
    "name": "Sales Executives",
    "displayName": "Sales Executives",
    "description": "",
    "created": 1481827712854,
    "modified": 1481827713052,
    "principalTypeEnum": "LOCAL_GROUP",
    "groupNames": []
  },
  {
    "name": "Operations Demo",
    "displayName": "Operations Demo",
    "description": "",
    "created": 1436491036553,
    "modified": 1436498598655,
    "principalTypeEnum": "LOCAL_GROUP",
    "groupNames": []
  },
  {
    "name": "Sales Directors",
    "displayName": "Sales Directors",
    "description": "",
    "created": 1481827747555,
    "modified": 1485805361837,
    "principalTypeEnum": "LOCAL_GROUP",
    "groupNames": []
  },
  {
    "name": "Product",
    "displayName": "Product",
    "description": "",
    "created": 1409250574242,
    "modified": 1477525172084,
    "principalTypeEnum": "LOCAL_GROUP",
    "groupNames": []
  },
  {
    "name": "Sales Development",
    "displayName": "Sales Development",
    "description": "",
    "created": 1481831987186,
    "modified": 1481831987382,

```

```

    "principalTypeEnum": "LOCAL_GROUP",
    "groupNames": [
      "Sales"
    ]
  }
]

```

### Response Code

200

### Response Headers

```

{
  "x-callosum-incident-id": "1be6e07b-b7aa-4531-8597-8852760757f0",
  "x-callosum-trace-id": "e92c54ca-d5f1-44a6-ab8e-f6871bb0da8b",
  "date": "Sun, 19 Feb 2017 04:14:13 GMT",
  "content-encoding": "gzip",
  "x-callosum-request-time-us": "19720",
  "server": "nginx",
  "vary": "Accept-Encoding",
  "content-type": "application/json",
  "pragma": "no-cache",
  "cache-control": "no-store, no-cache, must-revalidate, max-age=0,
post-check=0, pre-check=0",
  "transfer-encoding": "chunked",
  "content-security-policy": "script-src 'self'",
  "connection": "keep-alive",
  "x-callosum-ip": "192.168.2.247",
  "x-ua-compatible": "IE=edge"
}

```