Application Integration Guide

Version 3.3

May 2016

Contents

Chapter 1: Introduction

Topics:

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 the ThoughtSpot Analytic Search Appliance 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	Access remotely via SSH from the command prompt on a client machine.	Contact ThoughtSpot to obtain the default password.
Application User	tsadmin	Access through a Web browser.	Contact ThoughtSpot to obtain the default password.

Log In to ThoughtSpot From a Browser

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

Before accessing ThoughtSpot, you need:

- The Web address (IP address or server name) of any node in the appliance.
- 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	• Windows 7 or greater • Linux • MacOS
Mozilla Firefox	14 and above	• Windows 7 or greater • Linux • MacOS
Internet Explorer	10 and 11	• Windows 7 or greater

To log in to the appliance from a browser:

1. Open the browser and type in the Web address of any node in the ThoughtSpot Analytic Search Appliance:

   `http://<hostname_or_IP>`

2. If you see an untrusted connection warning, see Browser Untrusted Connection Error.
3. Enter your username and password and click **Enter Now**.

Log In to the Linux Shell Using SSH

To perform basic administration on the appliance, 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 the ThoughtSpot Analytic Search Appliance. 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 appliance:

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

3. Enter the password for the admin user.

Chapter 2: About SAML

Topics:

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 Analytic Search Appliance.

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 Analytic Search Appliance 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.

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

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:

   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:

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 JavaScript 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	10 or later
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 10, 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 re-directed 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. Log In to ThoughtSpot From a Browser.
2. Click on the **?** icon in the left menu to reach the Help Center.
3. Navigate to the **Downloads** page in the Help Center, and download the ThoughtSpot JS API JavaScript file.

4. 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">
```

5. 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>
```

6. 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 tripple backslash (\) as shown in this example:

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

Note that 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.

7. 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.
8. 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 nginix 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:

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 JSON. 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.

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.

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=1e99d70f-c1dc-4a52-9980-
cfd4d14ba6d6
```

Object Format for Returned Data

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 : {
          .
          }
}
```

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("embeded-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();
        }
```

Use the REST API to Get Data

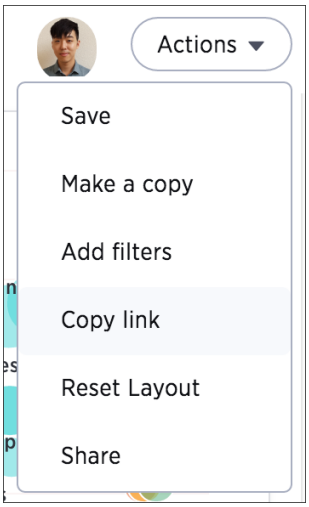
This procedures shows how to use the REST API to get data out of ThoughtSpot, so you can use it in a Web page, portal, or application. The data will be returned as JSON (JavaScript Object Notation).

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**.

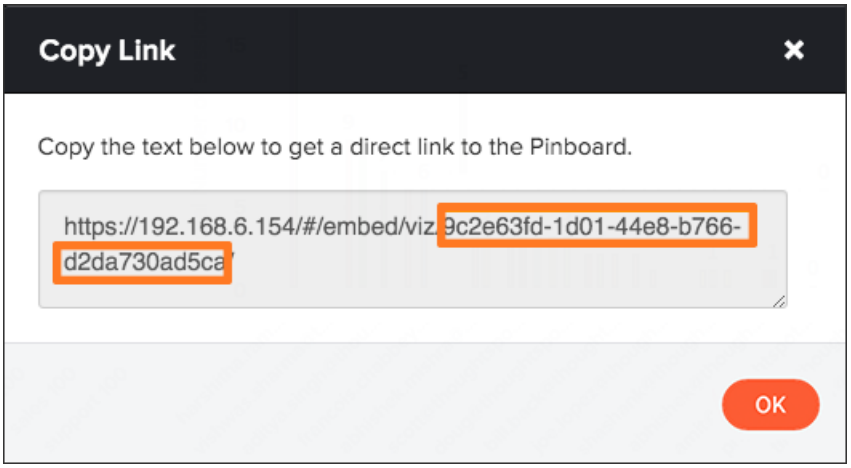


   - A visualization, click the **Copy Link** icon in the upper right corner of the table or chart.

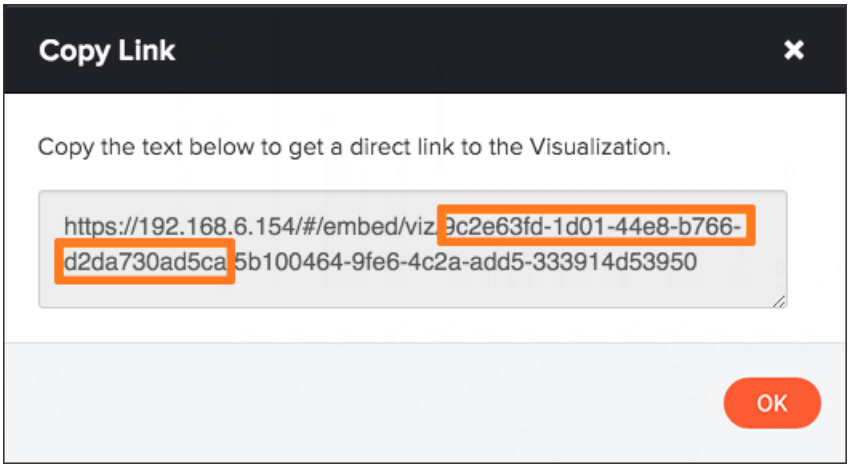
4. Copy the ID number from the link shown. Paste it somewhere so that you can use it later to construct the URL to use when calling the REST API.

   If the object is:

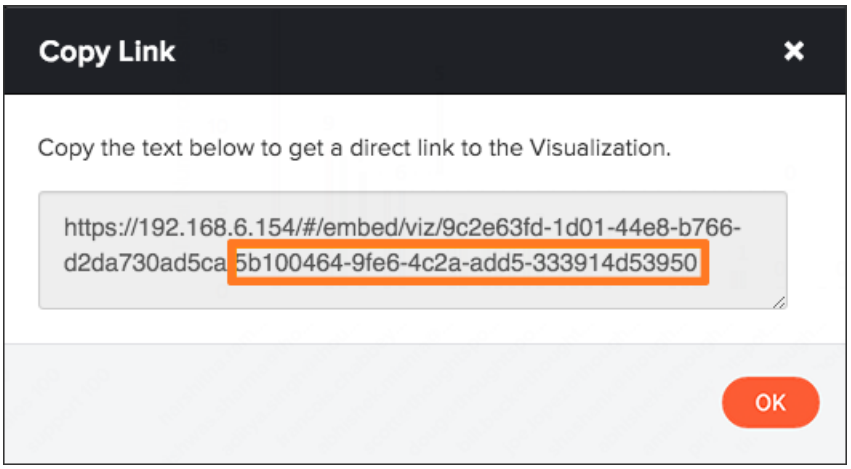
   - A pinboard, copy the identifier that appears after "viz/". Omit the trailing "/".



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



Then skip the "/" and copy what follows it. 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=<visualization_id>
   ```

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 via the HTTP POST method.
8. 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:

Embedding allows you to display a pinboard from ThoughtSpot on your own Web page, Web portal, or application. 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. Only the visualization is displayed, without the ThoughtSpot navigation or controls.

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

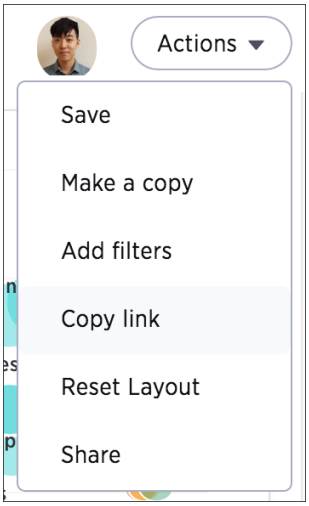
Embedding allows you to include a visualization (table or chart) or pinboard from ThoughtSpot in your own static Web page, Web portal, or application.

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**.



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

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

Chapter 6: About Runtime Filters

Topics:

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 pinboards 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

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

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>/
?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=<value>
```

   - For the REST API with a visualization:

```
http://<thoughtspot_server>:<port>
/callosum/v1/tspublic/v1/pinboarddata
?id=<pinboard_id>&vizid=<visualization_id>
&col1=<column_name>&op1=<operator>&val1=<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:

```
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>
```

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.

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
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

Operator	Description	Number of Values
BW_INC	between inclusive	2
BW	between non-inclusive	2
IN	is included in this list of values	multiple