Administrator Guide

The Administrator Guide is your complete reference for handling administrative tasks on Tableau Server:

Before you install...

Make sure the computer on which you're installing Tableau Server meets the following requirements:

- **Supported operating systems**—You can install Tableau Server on Windows Server 2003 (SP1 or higher), Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, Windows Vista, Windows 7, or Windows 8. Although Tableau Server performs well on 32-bit operating systems, 64-bit operating systems are recommended. You may install Tableau Server on virtual or physical platforms.
- **Memory, cores, and disk space**—Tableau Server system requirements vary based on many factors. The following are recommendations based on the number of users on the server:

Deployment Type	*# Server Users*	*CPU*	*RAM*
Evaluation	1-2	2-core	4 GB
Small	<25	4-core	8 GB
Medium	<100	8-core	32 GB
Enterprise	>100	16-core	32 GB or more

- **Administrative account**—The account under which you install Tableau Server must have permission to install software and services.
- **Optional: Run As Account**—A Run As User account for the Tableau Server service to run under is useful if you're using NT Authentication with data sources or if you're planning on doing SQL Server impersonation. For more information, see Run As User and SQL Server Impersonation.
- **IIS and port 80**—Tableau Server's gateway listens on port 80, which is also used by Internet Information Services (IIS) by default. If you are installing Tableau Server on a machine that's also running IIS, you should modify the Tableau's gateway port number to avoid conflict with IIS. See TCP/IP Ports and Edit the Default Ports for details.

Configuration Information

When you install and configure Tableau Server you may be asked for the following information:

Option	Description	Your Information
Server Account	The server must have a user account that the service can	Username:

Option	Description	Your Information
	use. The default is the built-in Windows Network Service account. If you use a specific user account you'll need the domain name, user name, and password.	Password: Domain:
Active Directory	Instead of using Tableau's built-in user management system, you can authenticate through Active Directory. If so, you'll need the fully-qualified domain name.	Active Directory Domain:
Open port in Windows firewall	When selected Tableau Server will open the port used for http requests in the Windows Firewall software to allow other machines on your network to access the server.	__ - Yes __ - No

Ports

By default Tableau Server requires several TCP/IP ports to be available to the server. See the topic TCP/IP Ports for the full list, including which ports must be available for all installations vs. distributed installations or failover ready installations. The default ports can be changed if there is a conflict. See Edit the Default Ports to learn how.

Drivers

You may need to install additional database drivers. Download drivers from www.tableausoftware.com/support/drivers.

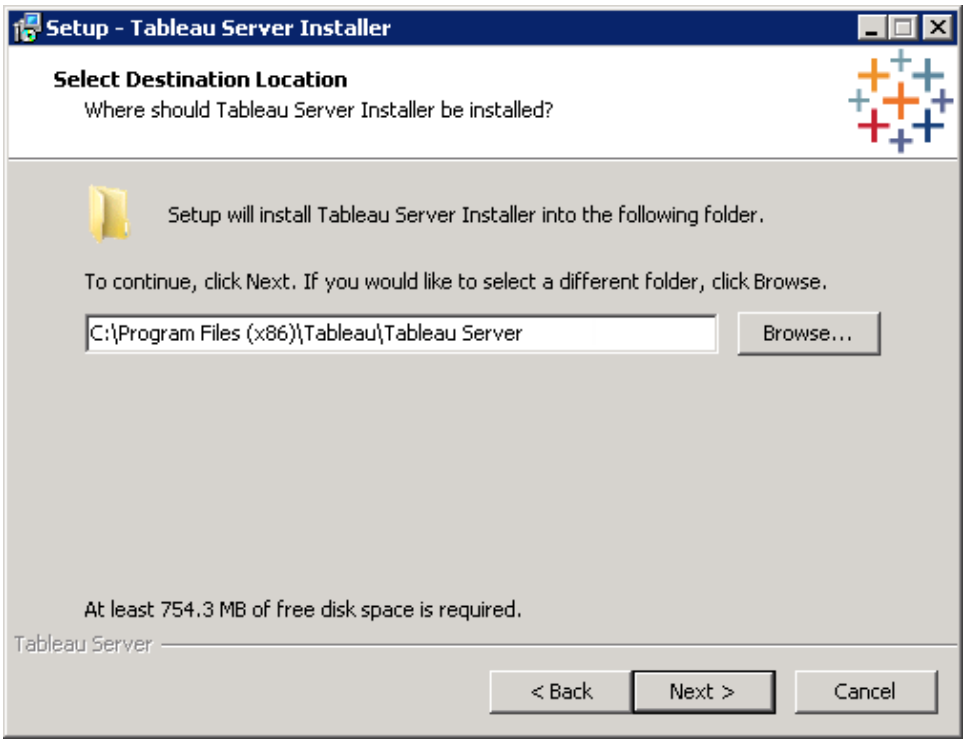
Install and Configure

Here are the main steps you need to take to install and configure Tableau Server:

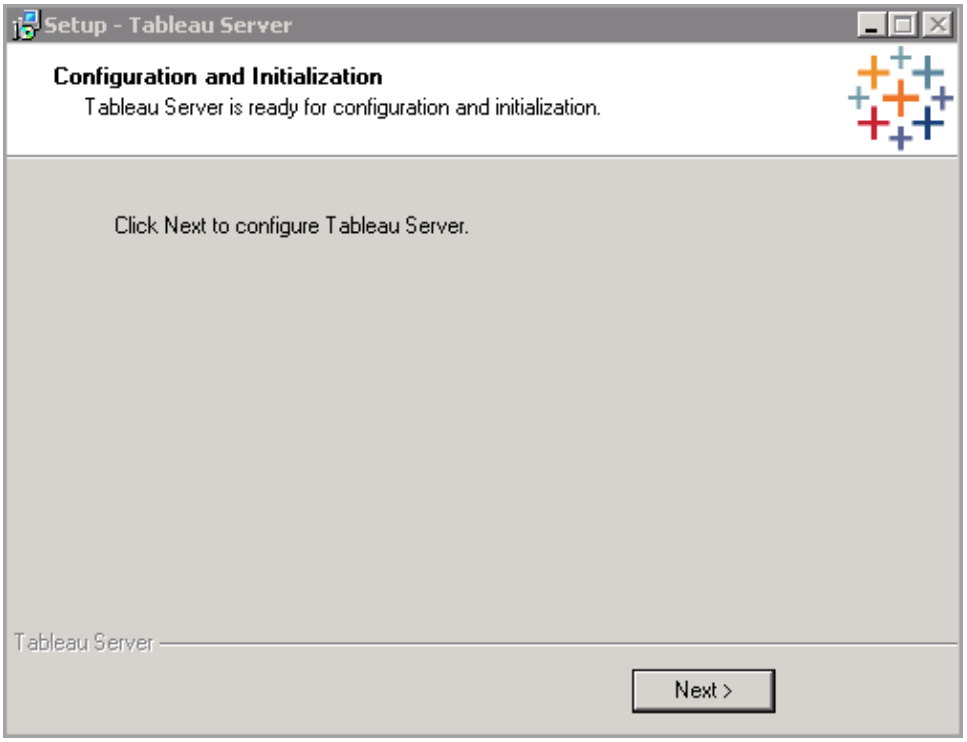
Run Server Setup

After you download the Tableau Server installation file, follow the instructions below to install the server.

1. Double-click the installation file.
2. Follow the on-screen instructions to complete Setup and install the application.



3. After the installation completes, click Next to open the Product Key Manager window.

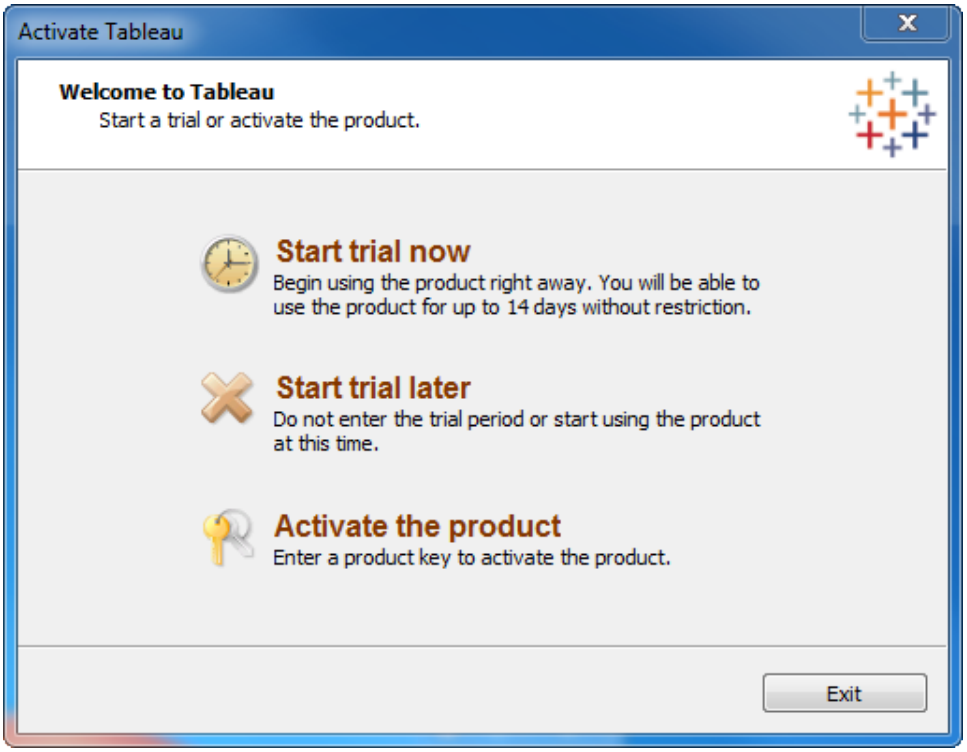


If you need to support characters that are not the Latin-1 set, install the Windows Language Packs via **Control Panel > Regional and Language Options**. The language packs will need to be installed on the primary server as well as any worker machines.

Activate Tableau

Tableau Server requires at least one product key that both activates the server and specifies the number of license levels you can assign to users. You can access your product keys from the Tableau Customer Account Center. After installing and configuring the server, the product key manager automatically opens so you can enter your product key and register the product. If you need to activate the product on a computer that is offline, see Activate Tableau Offline.

1. Select Activate and past in your product key:



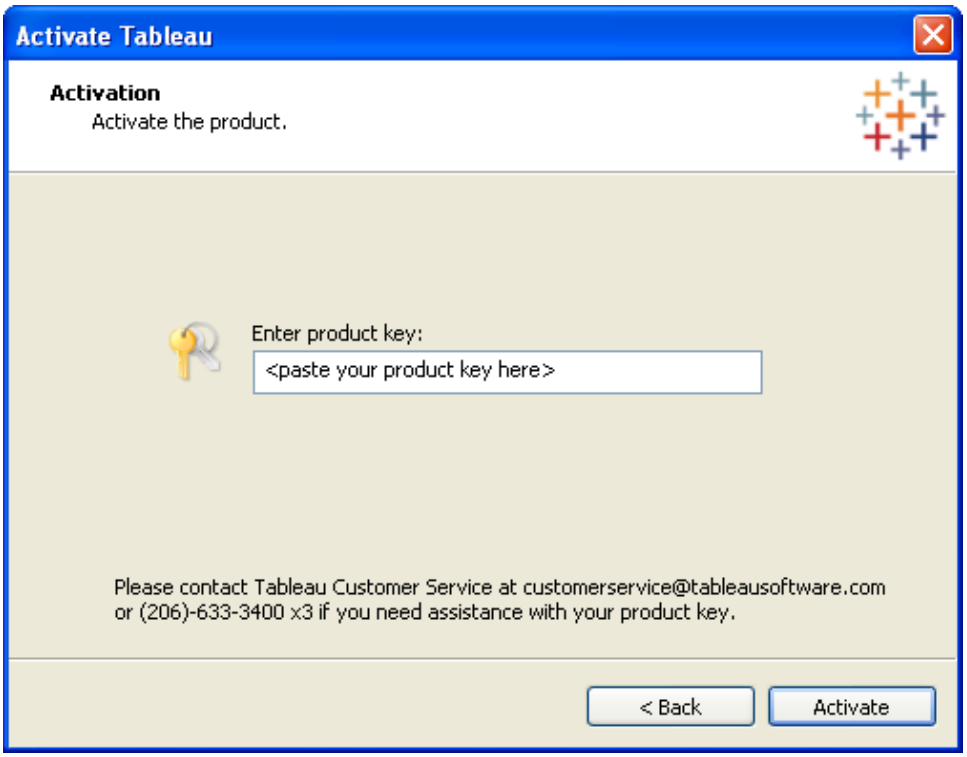
2. Refer to the download help page on the web site for step-by-step instructions.

Activate Tableau Offline

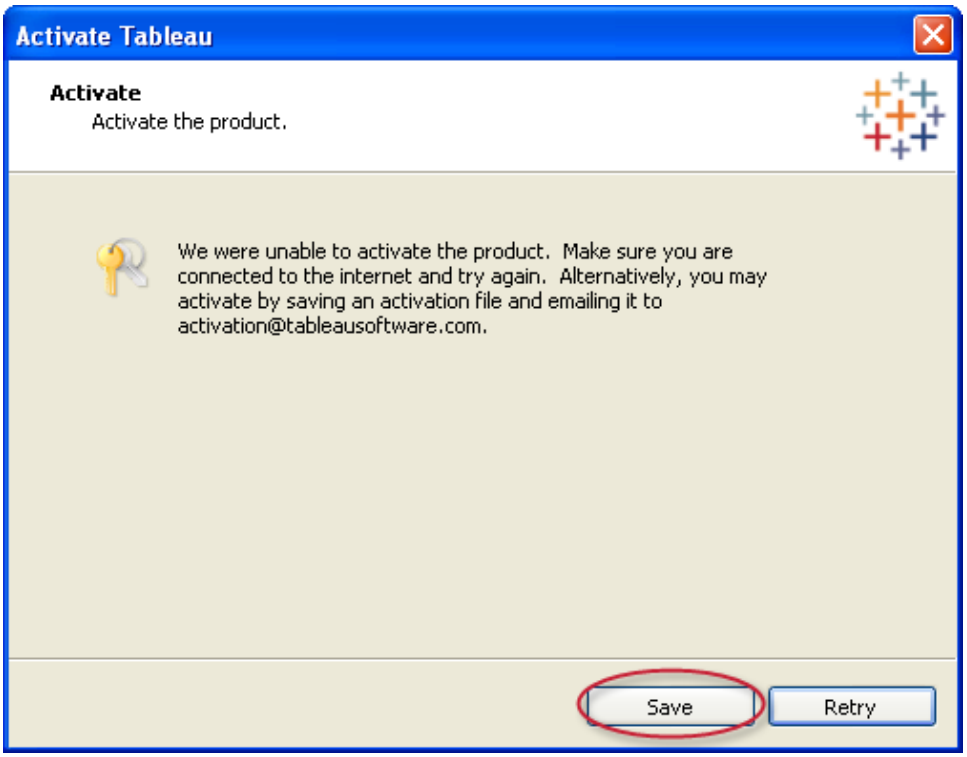
If you are working offline you can follow the steps below to complete offline activation.

1. When the product key manager opens click **Activate the product**.

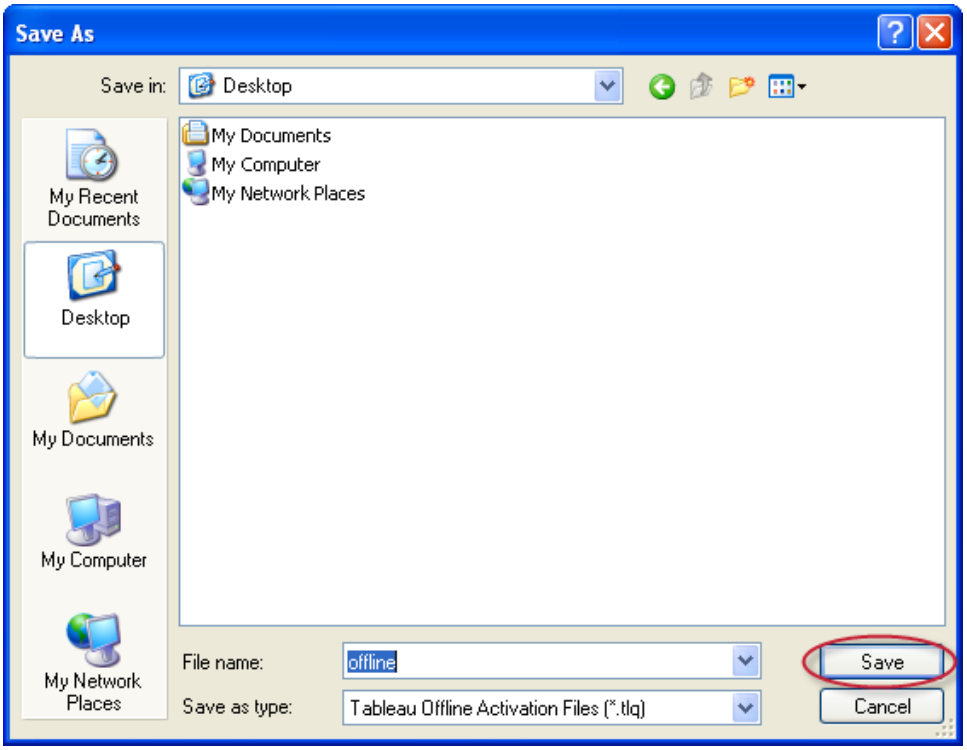
   Paste your server product key into the corresponding text box and click Activate. You can get your product key from the Customer Account Center on Tableau's web site.



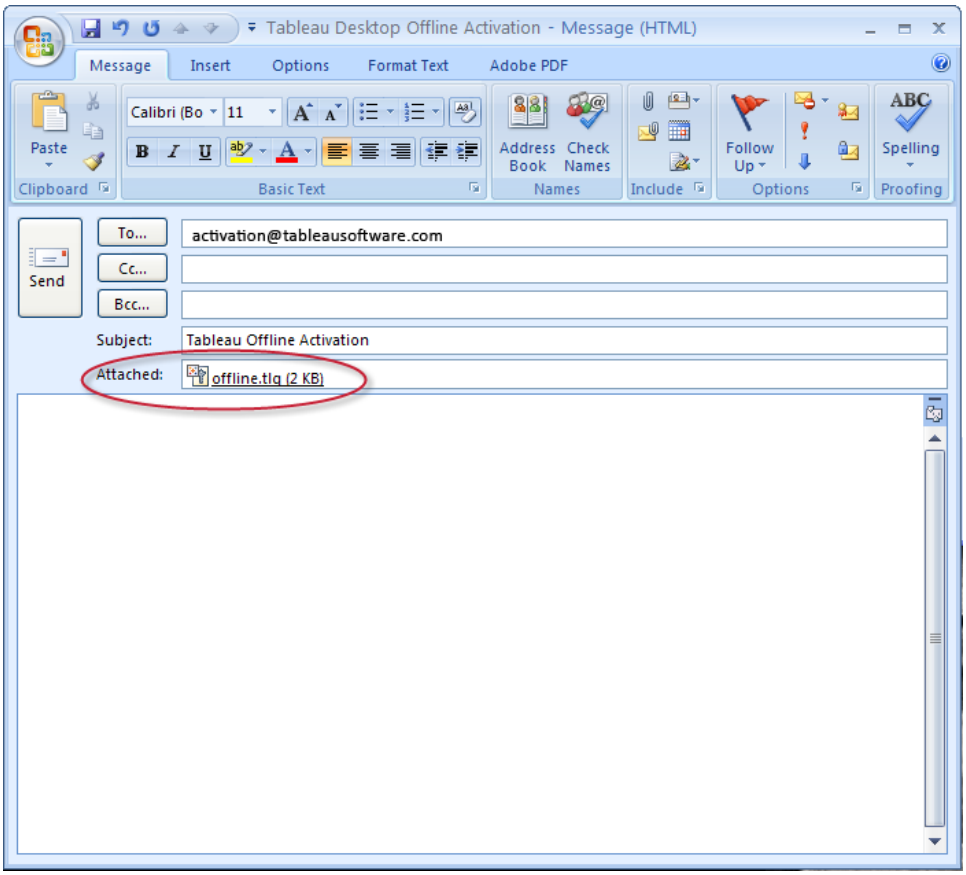
2. When you are offline, activation will fail and you are given the option to save a file that you can use for offline activation. Click **Save**.



3. Select a location for the file and click **Save**. The file is saved as *offline.tlq*.



4. Back in Tableau click **Exit** to close the Activation dialog box.
5. Move the file to a computer that is online and open an email editor. Create a new email to *activation@tableausoftware.com*. Attach the file to the email and click **Send**.



6. Tableau will email you a file called activation.tlf. Move this file to the computer where you are installing Tableau Server. If you have Tableau Desktop installed on the computer you can then **double-click** the new file to complete activation. If you do not have Tableau Desktop installed continue to steps 7 and 8.
7. On the computer where you are installing Tableau Server, open a command prompt as an administrator and run the following command:

   ```
   cd "C:\Program Files (x86)\Tableau\Tableau Server\8.0\bin"
   ```

8. Next, type `tabadmin activate --tlf <path>\activation.tlf`, where `<path>` is the location of the response file Tableau emailed to you. For example:

   ```
   tabadmin activate --tlf \Desktop\activation.tlf
   ```

   If you need additional assistance, contact Tableau at *activation@tableausoftware.com*.

Configure the Server

The Configure dialog displays during Setup. You can open it after Setup by selecting **All Programs > Tableau Server 8.0 > Configure Tableau Server** on the Windows Start menu. You need to stop the server before making any configuration changes. See Reconfigure the Server for steps.

There are two things to keep in mind about the settings you specify in the Configuration dialog box::

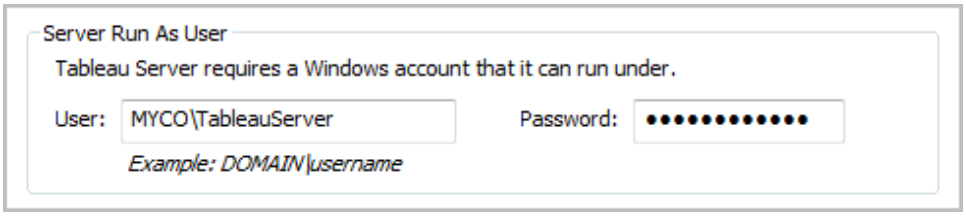
- **Settings are system-wide**: The settings you enter apply to the entire server. If the server is running multiple sites, these settings affect every site.
- **User Authentication is "permanent"**: All of the settings can be changed after Setup by stopping the server and reconfiguring. The exception is the **User Authentication** setting (General tab). It is "permanent" in the sense that changing from **Use Local Authentication** to **Use Active Directory** requires you to uninstall, then reinstall the server.

See the topics below for details on the different Configuration tabs:

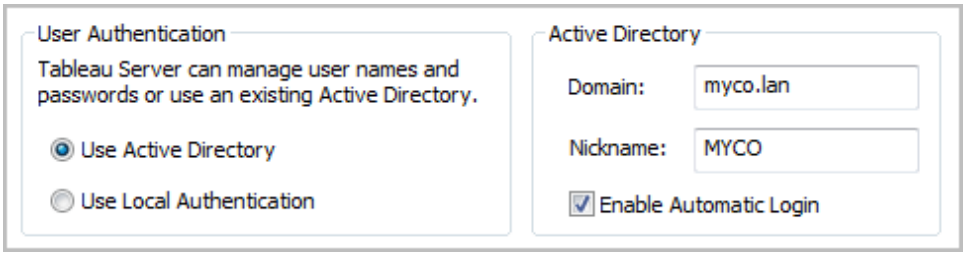
General

Use the steps below to configure options on the General tab:

1. By default, Tableau Server runs under the Network Service account. To use an account that will accommodate NT authentication with data sources, specify a user name and password. The user name should include the domain name. See Run As User to learn more about using a specific user account.



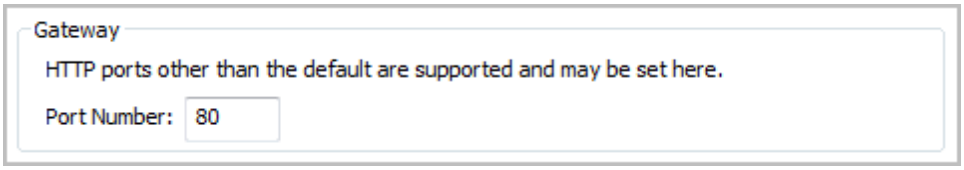
2. Select whether to use **Active Directory** to authenticate users on the server. Select **Use Local Authentication** to create users and assign passwords using Tableau Server's built-in user management system. You cannot switch between Active Directory and Local Authentication later.



3. **If you use Active Directory:**
    - You can optionally **Enable Automatic Login**, which uses Microsoft SSPI to automatically log in your users based on their Windows username and password. If you select this setting, you cannot also select Enable Guest later. Also, do not select **Enable Automatic Login** if you plan to configure Tableau Server for trusted authentication.
    - Be sure to type the fully qualified domain name (FQDN) and nickname.

To determine the FQDN: Select **Start > Run** then type `sysdm.cpl` in the Run textbox. In the System Properties dialog box, select the **Computer Name** tab. The FQDN is shown near the middle of the dialog box. The first time your users log in, they will need to use the fully qualified domain name (for example, `myco.lan\jsmith`). On subsequent logins, they can use the nickname (`myco\jsmith`).

4. The default port for web access to Tableau Server (via HTTP) is port 80. You may need to change the port number if you have another server running on port 80 or other networking needs. For example, you may have a hardware firewall or proxy in front of the Tableau Server host, which might make running a back-end system on port 80 undesirable.



5. Select whether to open a port in Windows Firewall. If you do not open this port, users on other machines may not be able to access the server.

6. Select whether to include sample data and users. The sample data can help you get familiar with Tableau Server, especially if you are installing a trial version of the product. Initially the sample user uses one interactor license. You can change this user to unlicensed in order to reclaim the license levels. See Licenses and User Rights to learn how. If you select to include the sample user, a single user is installed. The username and password are shown below:

Username	Password
Tableau Software	test

7. Optionally continue to the next page to configure Caching and Initial SQL options. If you do not want to configure these options click **OK**.

About Enable Guest & Enable Automatic Login

This topic provides some background on the **Enable Automatic Login** and **Enable Guest** settings and why they must not be used together.

Enable Automatic Login is an option you can select during Setup. It uses Active Directory and NTLM to authenticate Tableau Server users and automatically logs them into the server when they click a link for a view.

Enable Guest is a setting on the Maintenance page that can be selected if you have a core-based server license. It has the same result as **Enable Automatic Login**—users click a link

and they go directly to the view with no login—but unlike **Enable Automatic Login**, no authentication is performed. The Tableau Server Guest User account is used to access the server, but as long as **Enable Guest** is selected, anyone can use it. Administrators often limit the capabilities of the Guest User account. For example, they might edit the permissions of certain views so that Guest User is denied access.

Enable Automatic Login and **Enable Guest** are not supported or recommended as a combination. Selecting the first during Setup causes the second to become grayed out. In rare situations, however, both settings can be enabled. For example, if you don't select **Enable Automatic Login** during Setup, you can later select **Enable Guest**, then return to the Configuration utility and select **Enable Automatic Login**.

If the above happens, one symptom you may notice is that server users may have full access to a view, then after their sessions time out, they are denied access. This happens because the first automatic login and access level are based on the individual server user's Active Directory credentials, but the second (post-timeout) is based on Guest User's—and that account has a lower access level.

For more information on this topic, including workarounds for the above situation, see the Tableau Knowledge Base.

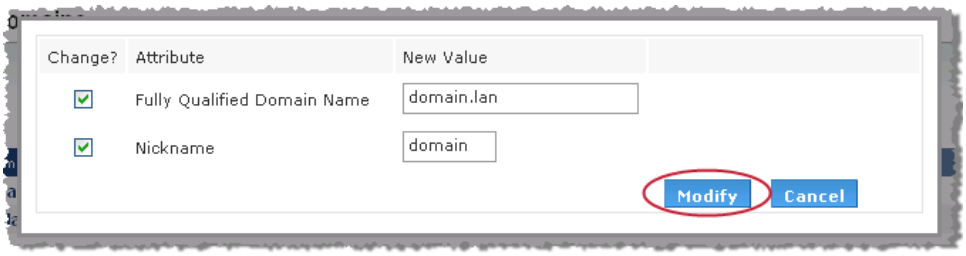
Domains

When you are using Active Directory authentication for the server you can view a list of the domains that are being used and edit their domain names and nicknames. You may need to do this, for example, to ensure that Tableau Server is using the correct nickname for SSPI authentication, or the correct domain name.

Modify Domain Names

To modify a domain name:

1. Select the Users link in the Administration area on the left side of the page.
2. Click the Domains link at the bottom of the list of users. The list of domains shows the number of users and groups that have been added to the server from each domain.
3. To display a list of users who are part of a domain, click the domain name.
4. To modify the domain name or nickname, click the **Edit** link, type a new, fully qualified domain name or a nickname, then click **Modify**.



You can modify the nickname for any domain the server is using. In general, you can modify the full domain name for any domain except the one that you used to

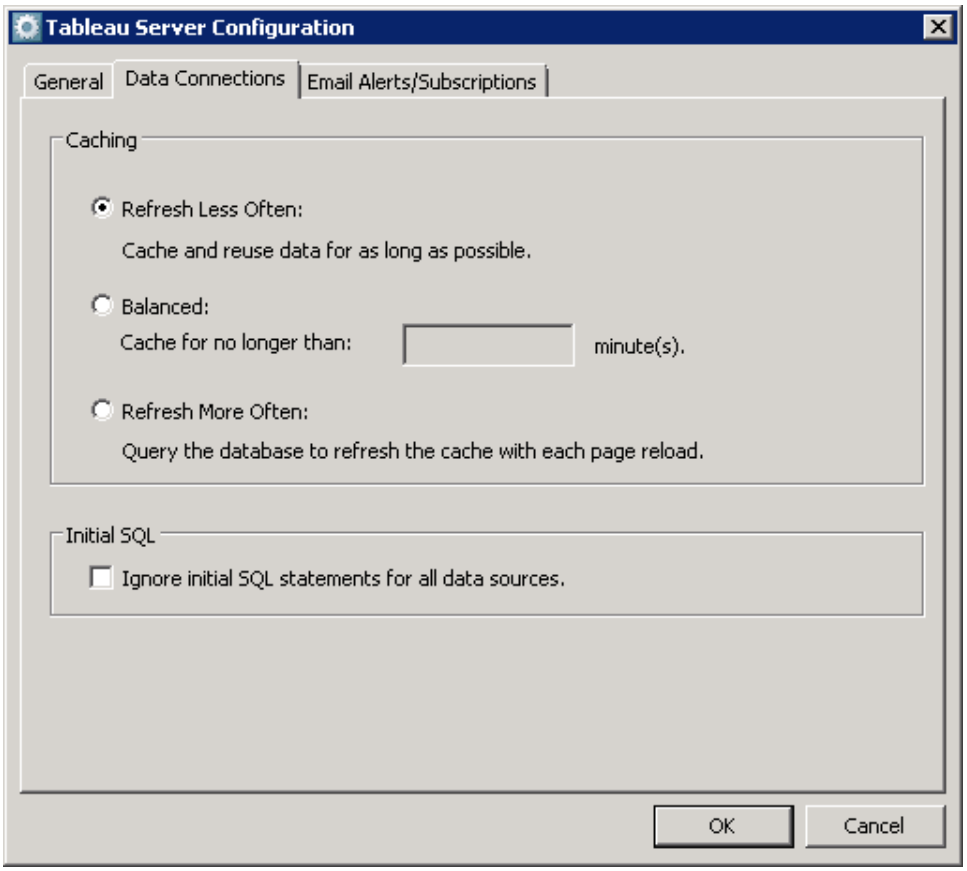
log in. However, if the user name that you are currently logged in with exists in both the current domain and the new domain you can modify the full name for the current domain.

Data Connections

Use the options on the Data Connections tab to configure caching and specify how you want to handle initial SQL statements from data sources.

Caching

Views published to Tableau Server are interactive and sometimes have a live connection to a database. As users interact with the views in a web browser, the data that is queried gets stored in a cache. Subsequent visits will pull the data from this cache if it is available. The Data Connections tab is where you configure aspects of caching that will apply to all data connections:



To configure caching, select from one of the following options: :

- **Refresh Less Often**—Data is cached and reused whenever it is available regardless of when it was added to the cache. This option minimizes the number of queries sent to the

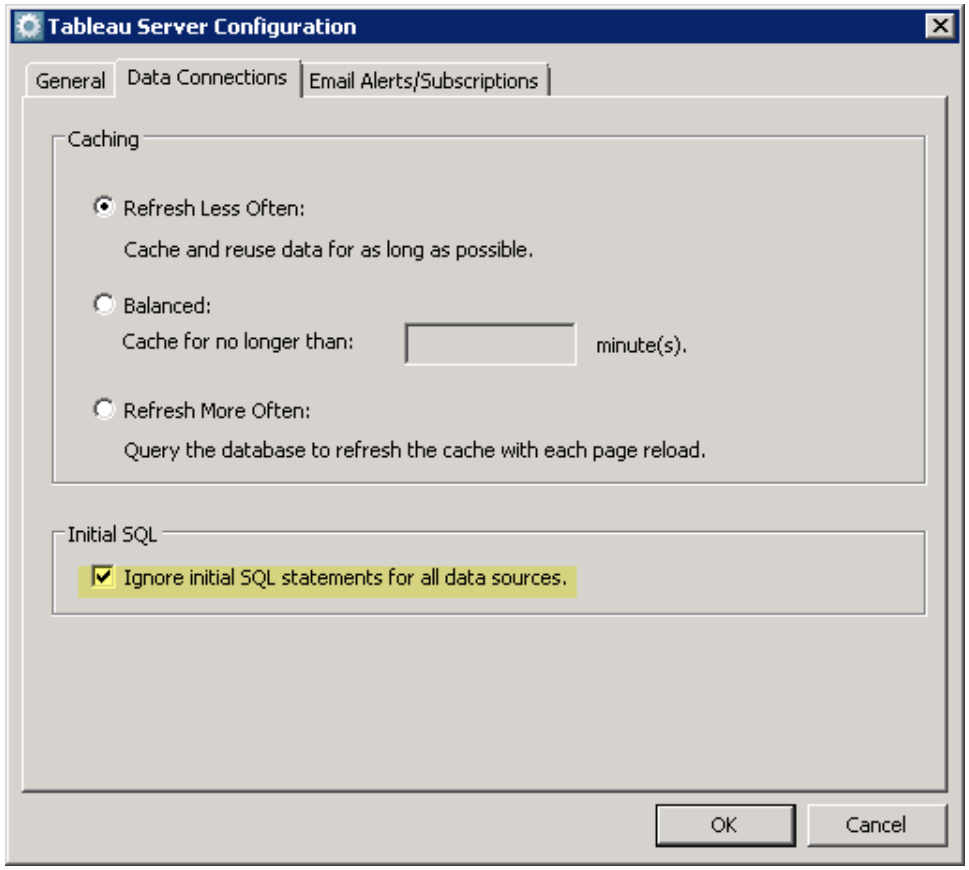
database. Select this option when data is not changing frequently. Refreshing less often may improve performance.

- **Balanced**—Data is removed from the cache after a specified number of minutes. If the data has been added to the cache within the specified time range the cached data will be used, otherwise new data will be queried from the database.
- **Refresh More Often**—The database is queried each time the page is loaded. The data is still cached and will be reused until the user reloads the page. This option will ensure users see the most up to date data; however, it may decrease performance.

> Regardless of how caching is configured, the user can click the **Refresh Data** button on the toolbar to force the server to send a query and retrieve new data.

Initial SQL

For views that connect to Teradata data sources, workbook creators can specify a SQL command that will run once, when the workbook is loaded in the browser. This is called an initial SQL statement. For performance or security reasons, some administrators may want to disable this functionality. The **Data Connections** tab is where you do this:



To disable initial SQL functionality, select the **Ignore initial SQL statements for all data sources** checkbox. Workbooks created with initial SQL statements will still open but the initial SQL commands will not be sent.

Email Alerts/Subscriptions

Tableau Server can send you an alert by email if there's a system failure and it can email subscriptions to Tableau Server users, which are snapshots of their favorite views. The **Email Alerts/Subscriptions** tab is where you specify the SMTP server that Tableau Server uses for sending email.

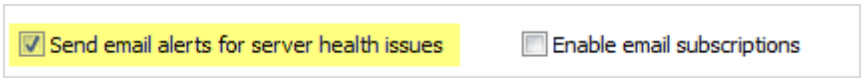
> For both alerts and subscriptions, encrypted SMTP connections are not supported.

Configure Email Alerts

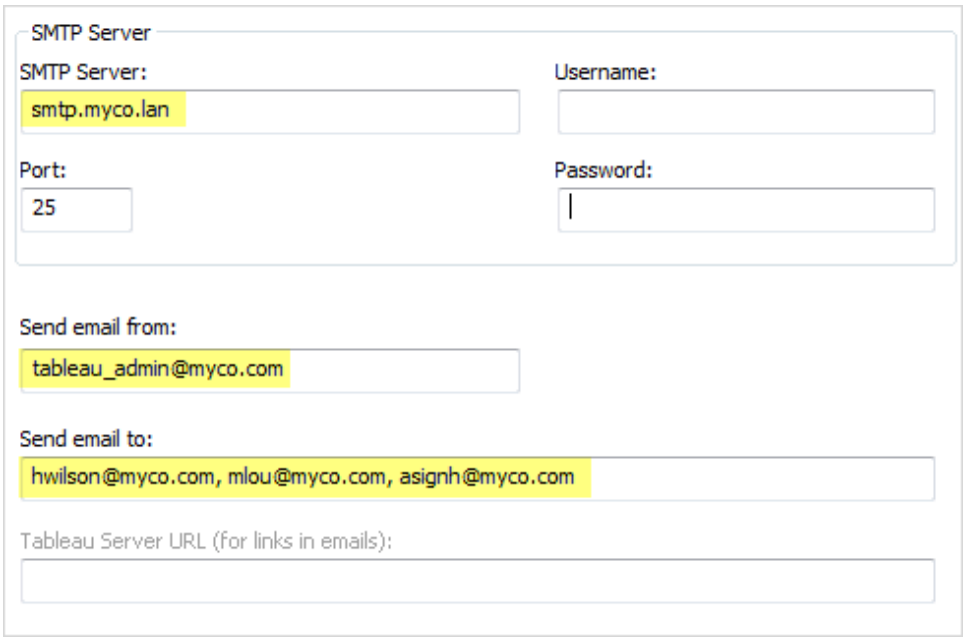
When you configure alerts, Tableau Server sends an email to the recipients under **Send email to** any time the data engine or repository server processes stop or restart. If you are running a single-server installation (all processes on the same machine), DOWN alerts mean that the entire server has stopped and subsequent UP email alerts mean the server is running again. If you are running a distributed installation that's configured for failover (see Configure for Failover) a DOWN alert means that the active repository or data engine instance has failed and the subsequent UP alert means that the standby instance of that process has taken over and is now active.

To configure an email alert:

1. Select **Send email alerts for server health issues**.



2. Under **SMTP Server**, enter the name of your SMTP server. Enter a **Username** and **Password** for your SMTP server account only if it requires one (some do, some do not). The default SMTP port value is 25. Under **Send email from**, enter the email address that will send an alert if there's a system failure. While the email address you enter must have valid syntax (for example, ITalerts@bigco.com or noreply@myco), it does not have to also be an actual email account on Tableau Server.



3. Under **Send email to** enter at least one email address that will receive the alerts. If you enter multiple email addresses, separate them with commas (not semicolons).
4. Click **OK**. When you start the server it will trigger an email alert and this will confirm that you have set up alerts correctly.

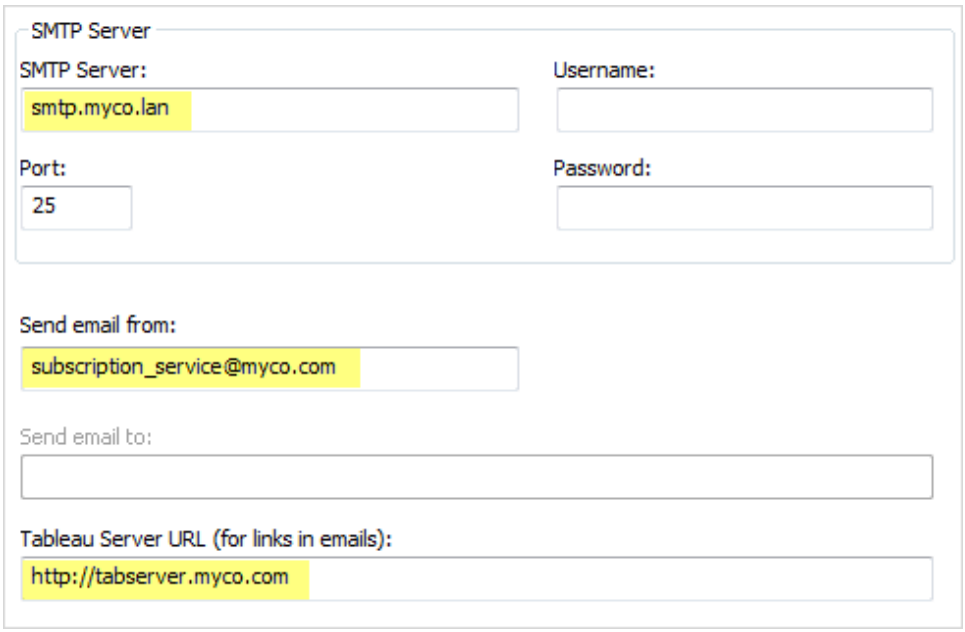
Configure Email Subscriptions

To set up an SMTP server for sending subscriptions:

1. Select **Enable email subscriptions**.

2. Under **SMTP Server**, enter the name of your SMTP server. Enter a **Username** and **Password** for your SMTP server account only if it requires one (some do, some do not). The default SMTP port value is 25. Under **Send email from**, enter the email address that will send subscriptions to Tableau Server users.

   While the email address you enter must have valid syntax (as in `<text>@<text>`, such as `salesteam@bigco.com` or `noreply@myco`), Tableau Server does not require it to be an actual email account (however, some SMTP servers may require it to be an actual email account). You can also override this system-wide **Send email from** address on a per-site basis for subscriptions. See Add or Edit Sites for details.



3. Under **Tableau Server URL**, enter `http://` or `https://`, followed by the name of the Tableau Server. This name will be used for the footer of subscription emails.
4. Click OK.

SSL

You can configure Tableau Server to use Secure Sockets Layer (SSL) encrypted communications on all HTTP traffic. Setting up SSL ensures that access to the web application is secure and that sensitive information passed between the web browser and the server or Tableau Desktop and the server is protected. Steps on how to configure the server for SSL are described in the topic below; however, you must first acquire a certificate from a trusted authority, and then import the certificate files into Tableau Server.

To configure Tableau Server to use SSL:

1. Acquire an Apache SSL certificate from a trusted authority (e.g., Verisign, Thawte, Comodo, GoDaddy, etc.). You can also use an internal certificate issued by your company. Wildcard certificates, which allow you to use SSL with many host names within the same domain, are also supported.

   Some browsers will require additional configuration to accept certificates from certain providers. Refer to the documentation provided by your certificate authority.
2. Place the certificate files in a folder named SSL, parallel to the Tableau Server 8.0 folder. For example:

   ```
   C:\Program Files (x86)\Tableau\Tableau Server\SSL
   ```

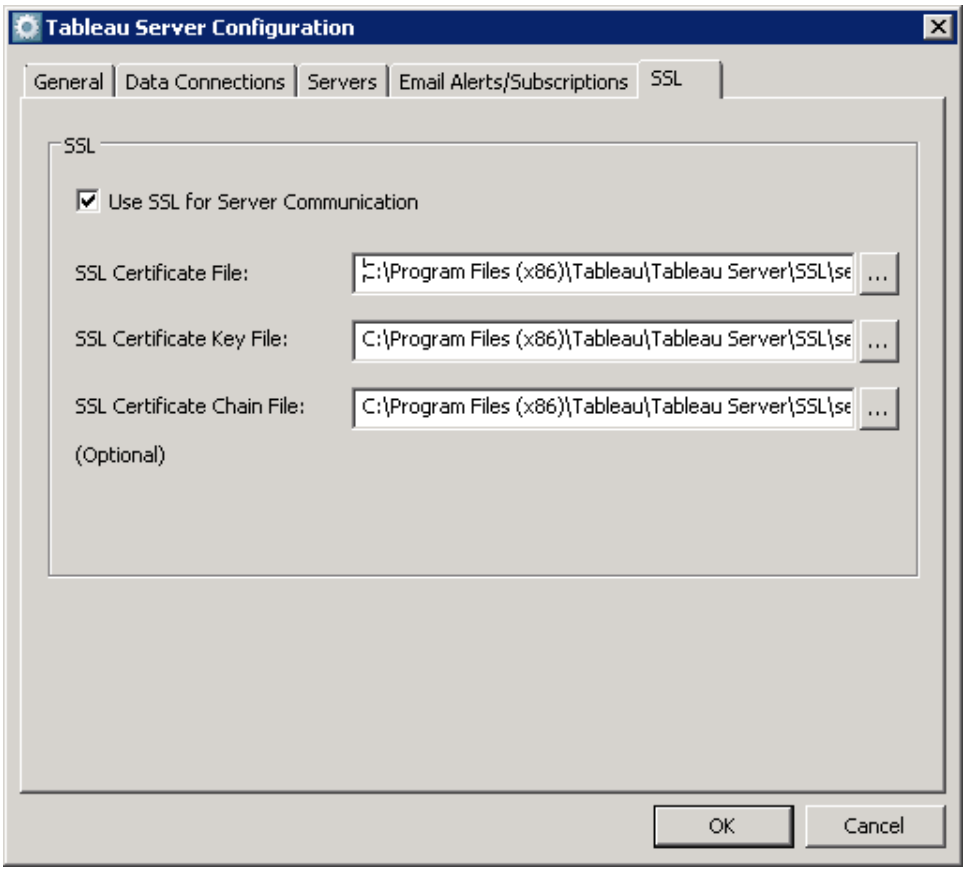
This location gives the account that's running Tableau Server the necessary permissions for the files.

3. Open the Tableau Server Configuration Utility by selecting **Start > All Programs > Tableau Server 8.0 > Configure Tableau Server** on the Start menu.
4. In the Configuration Tableau Server dialog box, select the **SSL** tab.
5. Select **Use SSL for Server Communication** and provide the location for each of the following certificate files:

   SSL Certificate File—Must be a valid PEM-encoded x509 certificate with the extension .crt

   SSL Certificate Key File—Must be a valid RSA or DSA key that is not password protected with the file extension .key

   SSL Certificate Chain File (Optional)—Some certificate providers issue two certificates for Apache. The second certificate is a chain file, which is a concatenation of all the certificates that form the certificate chain for the server certificate. All certificates in the file must be x509 PEM-encoded and the file must have a .crt extension (not .pem).



6. Click **OK**. The changes will take effect the next time the server is restarted.

When the server is configured for SSL, it accepts requests to the non-SSL port (default is port 80) and automatically redirects to the SSL port 443. SSL errors are logged in the install directory at the following location. Use this log to troubleshoot validation and encryption issues.

```
C:\ProgramData\Tableau\Tableau
Server\data\tabsvc\logs\httpd\error.log
```

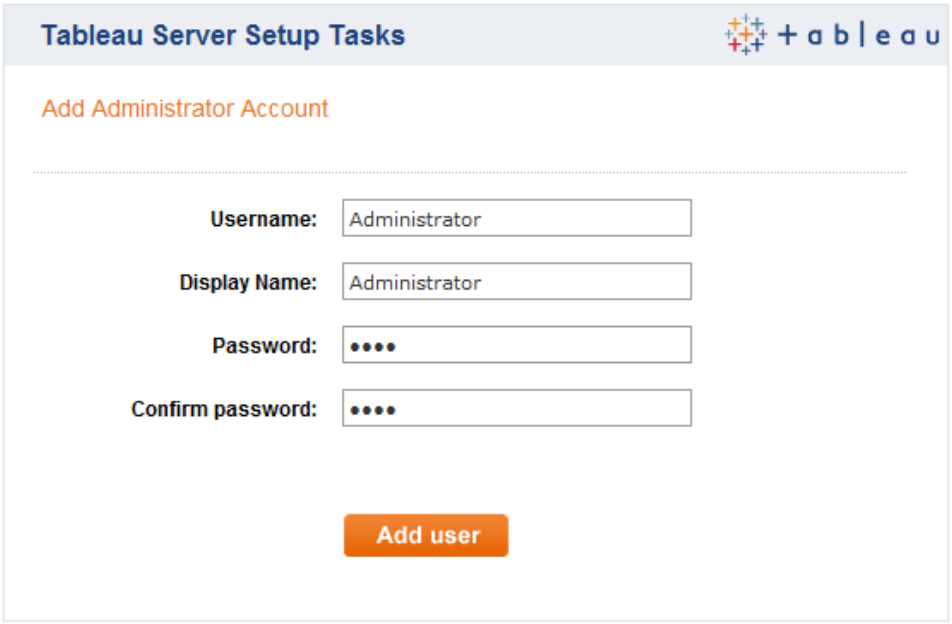
> Tableau Server only supports port 443 as the secure port. It cannot run on a machine where any other application is using port 443.

Add an Administrator Account

The final step in activating Tableau Server is to add an administrator account. The administrator will have all access to the server including the ability to manage users, groups, and projects. Adding an administrator account differs depending on whether you are using Active Directory or local authentication.

Active Directory

If you are using Active Directory, type the Username and Password for an existing Active Directory user who will be the administrator. Then click **Add user**.



Note:

If the administrator account is in the same domain as the server simply type the username without the domain. Otherwise you should include the fully qualified domain name. For example, test.lan\username.

Local Authentication

If you are using Local Authentication, create an administrative account by typing a Username, Display Name, and a Password (twice) of your choosing. Then click **Add user**.

Reconfigure the Server

Entering your Tableau Server configuration settings is part of Setup, but you can open the Configuration dialog box after Setup to make changes. See the steps below for details. You can also use the tabadmin command line tool to make configuration changes. Regardless of how you make the change, the new settings are written to the configuration file tabsvc.yml, which is located in the config directory.

Note: *You cannot switch between Active Directory and Local Authentication. These options can only be configured during Setup.*

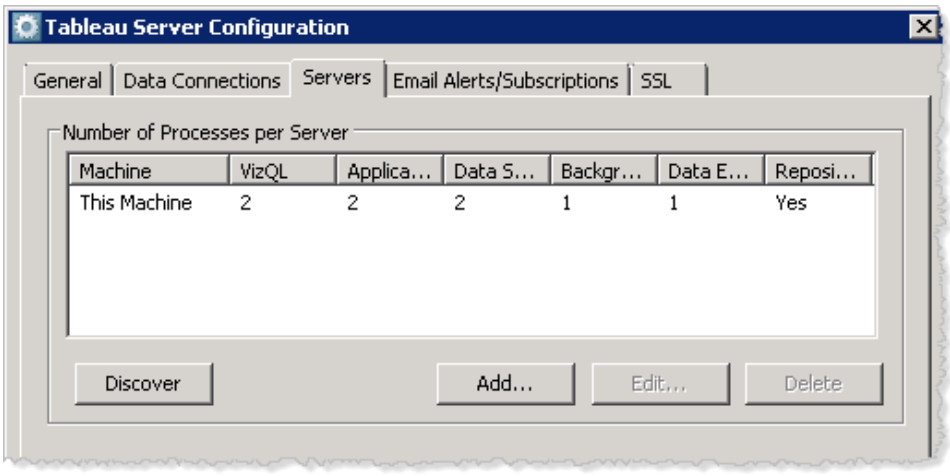
To change a setting in the Tableau Server Configuration dialog box, do the following:

1. Stop the server by selecting **All Programs > Tableau Server 8.0 > Stop Tableau Server** on the Windows Start menu.
2. Next, select **Configure Tableau Server** on the Windows Start menu.
3. If you are using an Active Directory account for the server's Run As User account, enter its password on the **General** tab.
4. Make your configuration change.
5. Click OK.
6. Start the server by selecting **All Programs > Tableau Server 8.0 > Start Tableau Server** on the Windows Start menu.

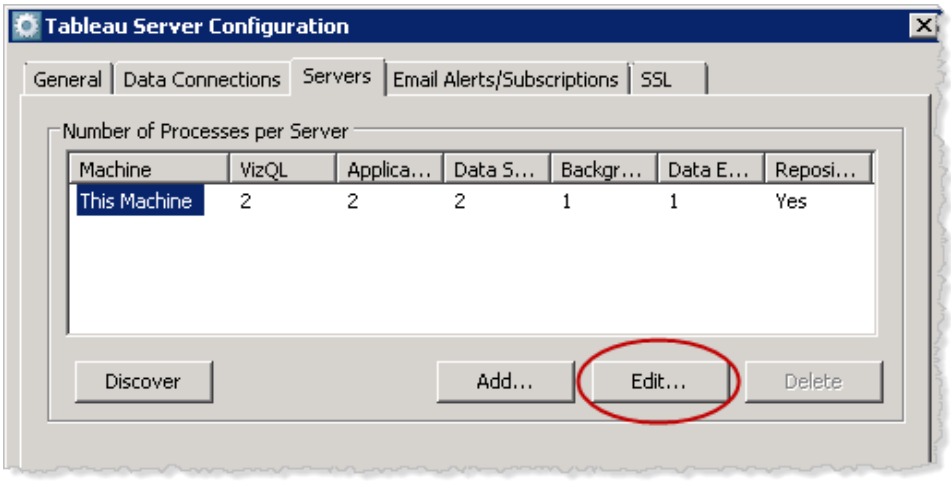
Reconfigure Processes

To change how processes are configured for a single server installation, follow the steps below. If you are changing how processes are configured for a worker, refer to Install and Configure Worker Servers.

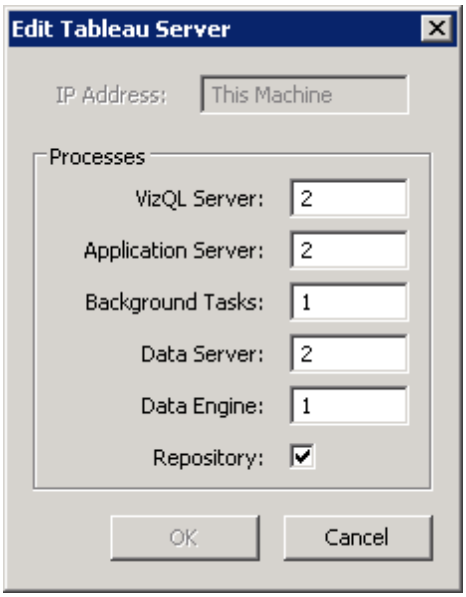
1. Open the Tableau Server Configuration dialog box from the Start menu by navigating to **All Programs > Tableau Server 8.0 > Configure Tableau Server**.
2. Enter your **Password**, if necessary, on the **General** tab then click the **Servers** tab:



3. Highlight `This Machine` and click **Edit**:



4. The Edit Tableau Server dialog box is where you change the number of processes:



5. You can run up to eight instances of the VizQL, application server, data server, or background processes—although this limit can be changed if necessary. See About the Server Process Limit for more information. Also, for Tableau Server to function, there must always be one active instance of the data engine and the repository. For steps on how to move them to another machine, see Move the Data Engine and Repository Processes. For steps on how to configure standby instances of them, refer to High Availability.

   After you make your changes, click OK, then click OK again to exit the Configuration dialog box.

The Tableau Server Processes

There are six Tableau Server processes whose default configuration you can change to achieve different results. The topics Improve Server Performance and High Availability describe some of the approaches you can take. High-level status for each process is displayed on the server's Maintenance page and more detailed information related to some of the processes—such as the background process—is in the Administrative Views.

Process	File Name	Purpose	Multi-Threaded?	Architecture	Performance Characteristics
application server	wgserver.exe	Handles the web application, supports browsing and searching	Yes	32-bit	Only consumes noticeable resources during infrequent operations like publishing a workbook with an extract, or generating a static image for a view. Its

Process	File Name	Purpose	Multi-Threaded?	Architecture	Performance Characteristics
					load can be created by browser-based interaction and by tabcmd.
background	backgrounder.exe	Executes server tasks, including extract refreshes, 'Run Now' tasks, and tasks initiated from tabcmd	No	32-bit	A single-threaded process where multiple processes can be run on any or all machines in the cluster to expand capacity. The backgrounder normally doesn't consume much process memory, but it can consume CPU, I/O, or network resources based on the nature of the workload presented to it. For example, performing large extract refreshes can use network bandwidth to retrieve data. CPU resources can be consumed by data retrieval or complex tabcmd tasks.
data engine	tdeserver64.exe tdeserver.exe	Stores data extracts and answers queries	Yes	64-bit 32-bit	The data engine's workload is generated by requests from the VizQL Server process. It is the component that loads extracts into memory and performs queries against them. Memory consumption is

Process	File Name	Purpose	Multi-Threaded?	Architecture	Performance Characteristics
					primarily based on the size of the data extracts being loaded. The 64-bit binary is used as the default on 64-bit operating systems. The data engine is multi-threaded to handle multiple requests at a time. Under high load it can consume CPU, I/O, and network resources, all of which can be a performance bottleneck under load. At high load, a single instance of the data engine can consume all CPU resources in order to process requests.
data server	dataserver.exe	Handles connections to Tableau Server data sources	Yes	32-bit	Because it's a proxy, it's normally only bound by network, but it can be bound by CPU with enough simultaneous user sessions. Its load is generated by browser- and Tableau Desktop-based interaction and extract refresh jobs for Tableau Server data sources.
repository	postgres.exe	Tableau Server's data-	-	-	Normally consumes

Process	File Name	Purpose	Multi-Threaded?	Architecture	Performance Characteristics
		base, stores workbook and user meta-data			few resources. It can become a bot-tleneck in rare cases for very large deployments (thou-sands of users) while performing operations such as viewing all work-books by user or changing per-missions.
VizQL Server	vizqlserver.exe	Loads and renders views, com-putes and executes que-ries	Yes	32-bit	Consumes notice-able resources dur-ing view loading and interactive use from a web browser. Can be CPU bound, I/O bound, or network bound. Process load can only be created by browser-based interaction. Can run out of proc-ess memory.

About the Server Process Limit

The wgserver, vizqlserver, and backgrounder server processes are engineered to be multi-threaded and multi-processed. A single process instance can run over 16 threads. By default, Tableau Server installs with up to two instances of each server process. If the default settings aren't sufficient, you can change them to up to eight instances either during Setup (for upgrades only) or after Setup, using the Configuration dialog box. Eight instances of a process is the default upper limit. If your machine has enough RAM and CPU cores, you can change the upper limit using the `service.max_procs` tabadmin setting. For each process instance, Tableau recommends that the machine running the process have at least 1 GB of RAM and 1 logical CPU core.

To change the maximum number of processes allowed:

1. After Setup, stop the server.
2. Still in the Tableau Server bin directory, enter the following command, where `number` is

the maximum number of process instances you want to allow:

```
tabadmin set service.max_procs number
```

For example:

```
tabadmin set service.max_procs 16
```

3. Start the server so the changes can take effect.

Upgrade to 8.0

Use the following topics to upgrade your Tableau Server software to version 8.0. If you are upgrading from a version earlier than 7.0, please refer to the Tableau Knowledge Base.

Pre-Upgrade Checklist

Here are items you should locate and steps you should perform before you upgrade Tableau Server to version 8.0.x.

Credentials, Setup Files, and Customizations

Before you upgrade, make sure you have the following:

- **User account credentials**: For each machine you're upgrading, you need credentials for a user account with local admin permissions.
- **Run As account credentials**: Confirm that you have the user name and password for Tableau Server's Run As account. If you are using NT AUTHORITY\NetworkService (the default), no password is required.
- **Setup files**: In addition to having the .exe for the upgrade you're about to perform, you should locate or re-download the Setup .exe for the server version you currently have in production (see Downloading Tableau Products). If something unexpected happens during the upgrade, this can help you recover more quickly.

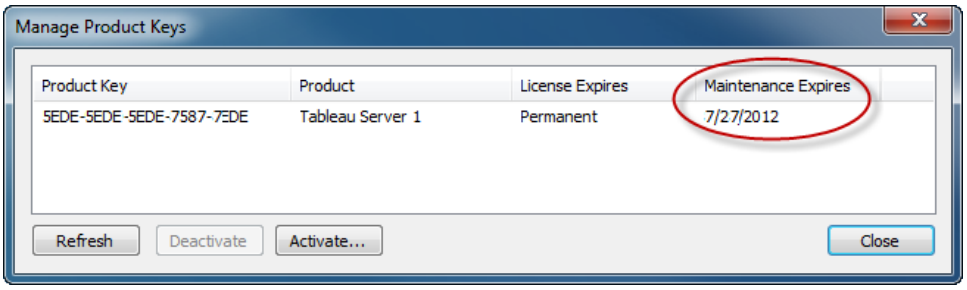
While Tableau retains configuration settings during an upgrade, it's a best practice to also note any customizations you've made so that you can verify them later. These include configuring SSL, changing Tableau's default port and time out values, as well as using custom logos. Also, if you added your current Tableau Server version to your Windows PATH environment variable, you will need to update that entry after upgrading so that it refers to the newer version of Tableau Server.

Check Your Product Maintenance Status

If you attempt to upgrade Tableau Server from a server whose maintenance has expired, the result will be an unlicensed instance of Tableau Server.

To see whether your server's maintenance has expired:

- Select **Start** > **All Programs** > **Tableau Server** > **Manage Product Keys** and look under the **Maintenance Expires** column.



If your maintenance has expired, contact Tableau Customer Support. Reactivating the product key will be part of Setup. See Activate Tableau for details. If your server doesn't have internet access, refer to Activate Tableau Offline.

Create a "Clean" Backup

In addition to your regular Tableau Server backups, it's a best practice to create a backup just prior to upgrading. Before you create the backup, run the tabadmin cleanup command to remove non-essential files from your backup. See Running Cleanup and Back Up the Tableau Data for steps.

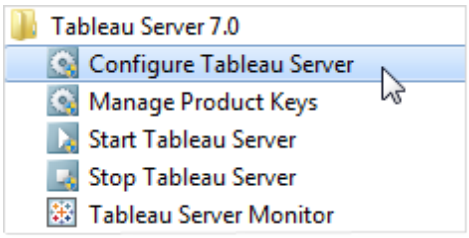
Distributed Installations Only: Whether to Remove Workers Before Creating the Backup

The Tableau backup file (*.tsbak*) includes configuration information as well as data. Therefore, a backup of a distributed installation of Tableau Server will include configuration information about the workers, including their IP addresses. If you don't want this information as part of your backup (for example, because you are migrating workers to new hardware as part of your upgrade), you can do one of two things: remove the workers from the Tableau Server configuration before creating the backup, or plan on using the `--no-config` option when you restore the backup file to your new installation. Note that with the latter option, no configuration information is restored—including the primary Tableau Server's.

If you are running a distributed installation of Tableau Server and have a worker running Windows XP, you must remove it from the configuration before upgrading. Windows XP is not a supported platform in version 8.0.

To delete a worker from your Tableau Server configuration:

1. Stop the server on the primary Tableau Server.
2. On the primary server, open the configuration utility by selecting **Tableau Server <version>** > **Configure Tableau Server** on the Start menu.



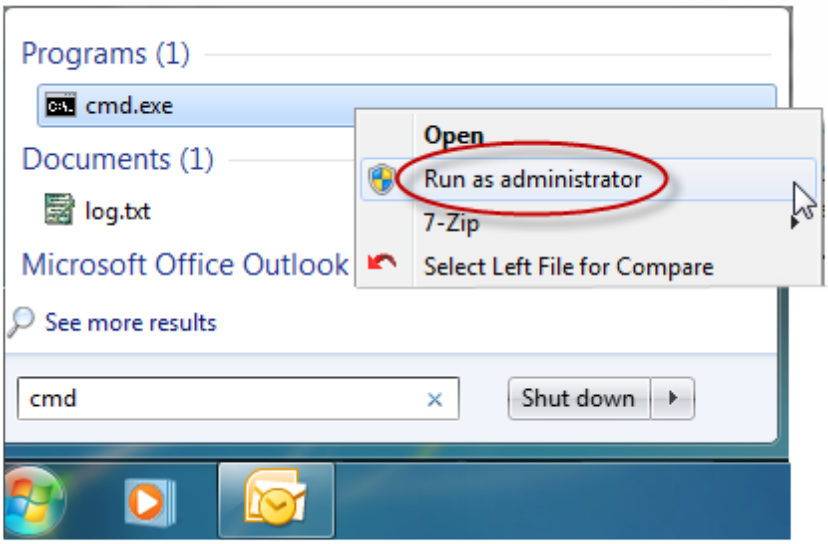
3. In the Configuration dialog box, select the **Servers** tab.
4. If the worker is hosting extracts and/or the repository, move those services onto another machine. See Move the Data Engine and Repository Processes for steps.
5. Next, highlight the worker and click **Delete**.
6. Click OK.
7. Start the server.

Running Cleanup

Running the tabadmin cleanup command removes files from the Tableau Server system that you don't need in your backup file. You should run cleanup once with the server running, which allows it to act on the Tableau database, and once with the server stopped, which allows it to remove log files.

To run tabadmin cleanup:

1. Open a command prompt as an administrator:



2. Navigate to your Tableau Server bin directory. For example:

```
cd "C:\Program Files (x86)\Tableau\Tableau Server\7.0\bin"
```

3. Confirm that the server is running:

```
tabadmin status
```

4. Run cleanup by typing the following:

   ```
   tabadmin cleanup
   ```

5. Stop the server:

   ```
   tabadmin stop
   ```

6. Run cleanup again:

   ```
   tabadmin cleanup
   ```

   Keep the server stopped for creating a backup (next).

Create the Backup File

The tabadmin backup command creates a *.tsbak* file containing data from your repository, data extracts, and server configuration. After you create the file, store it on a separate computer. See Back Up the Tableau Data for steps.

> ***Distributed installations only***: If you removed workers from your server configuration prior to creating your backup and you are upgrading from 8.0.x to 8.0.x, you can now add the workers back to your configuration. Upgrading the primary Tableau Server will push out updates to the workers. Otherwise, if you are upgrading from version 7.0 to 8.0.x, leave the workers off the configuration. See Upgrade to 8.0 for details.

Upgrade to 8.0

After you've completed the Pre-Upgrade Checklist, upgrade your existing Tableau Server to version 8.0 by following the procedure below. If you are migrating to new hardware as part of your upgrade, refer to Migrate to New Hardware instead.

1. Use Add/Remove Programs on your Tableau Server (or primary Tableau Server, if you have a distributed installation), to uninstall the earlier version.

   Uninstalling removes the server software but leaves your data and configuration settings intact.

2. Install Tableau Server. If you have a distributed installation, this step is on your primary Tableau Server.

   Tableau Server Setup will handle importing the data and configuration settings from your earlier version.

Move the Data Engine and Repository Processes

If you need to delete a worker from the Tableau Server configuration and that worker is hosting the only instance of the repository or the data engine (which hosts extracts), you must first move the process onto another machine. This is because there must always be one active instance of the repository and data engine processes.

To move the data engine or repository processes:

1. If you haven't done so already, stop the primary Tableau server and open the Tableau Server Configuration dialog box (**Start > Tableau Server 8.0 > Configure Tableau Server**) on the primary Tableau Server.
2. On the **Servers** tab, highlight the IP address of the machine onto which you want to move the process. It can be another worker or the primary (`This Machine`).
3. Click **Edit**.
4. In the Edit Tableau Server dialog box, select the check box for the process you are moving: either **Data Engine**, **Repository** or both, and click OK.
5. Click OK in the Tableau Server Configuration dialog box.
6. Start the primary Tableau server so that the changes can take effect.
7. Stop the server and open the Tableau Server Configuration dialog box.
8. On the **Servers** tab, highlight the IP address of the worker from which you are removing the process and click **Edit.**
9. Clear the check box for the process you moved and click OK.
10. Click OK again and start the primary server so that the changes can take effect.

If you are performing this procedure as part of deleting a worker from the Tableau Server configuration (as described in the Pre-Upgrade Checklist) stop the server again before proceeding.

Migrate to New Hardware

Use the following procedure to migrate Tableau Server from one machine to another. Specifically, these steps describe how to move Tableau Server data and configuration settings from your in-production machine to a new machine where Tableau Server version 8.0 is installed. Before you start, make sure you have followed the steps in the Pre-Upgrade Checklist, including creating a .tsbak file.

1. Install Tableau Server on the new machine.
2. Copy your *.tsbak* file to the bin folder on your new Tableau Server (for example, C:\Program Files (x86)\Tableau\Tableau Server\8.0\bin).
3. Next, stop Tableau Server.
4. Restore your in-production data and configuration information to your new Tableau Server installation by typing `tabadmin restore <filename>`, where `<filename>` is the name of the *.tsbak* file. For example:

   ```
   tabadmin restore mybackup.tsbak
   ```

   Or, to restore only the data from your in-production Tableau Server and no configuration information, type the following:

   ```
   tabadmin restore --no-config mybackup.tsbak
   ```

5. Start the server.

6. **Distributed installations only**: Run the Tableau worker installer on all the additional machines you want to add to your Tableau Server cluster. See Install and Configure Worker Servers for steps.
7. After you have tested the new installation, you must deactivate the earlier version of Tableau Server before you uninstall it. To deactivate the earlier version:

   - Select **Start** > **All Programs** > **Tableau Server** > **Manage Product Keys**.

   - For each product key, select the product key and click **Deactivate**.

   > If you do not have an internet connection, you are prompted to create an offline return file to complete the deactivation process. See Activate Tableau Offline for steps. After you create the offline return file, email it to activation@tableausoftware.com. After the offline return file is processed, another file is sent back to you to run on your machine to remove the server product key and complete the deactivation process.

Distributed Environments

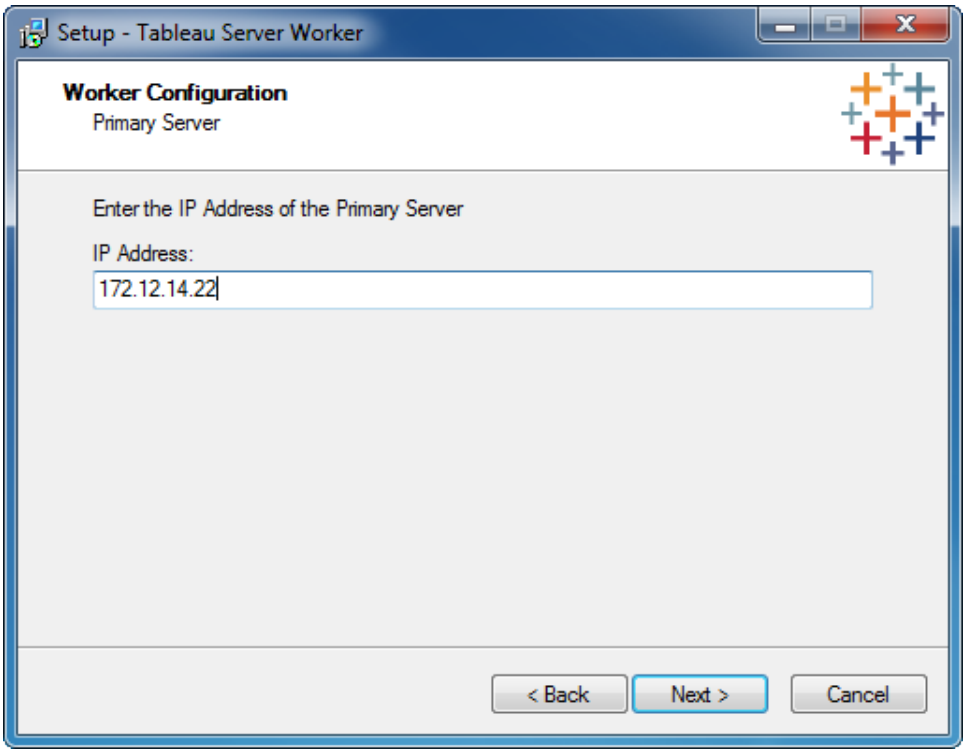
Use the topics below to learn more about running a distributed installation of Tableau Server:

Install and Configure Worker Servers

After you complete the initial configuration, you can set up Tableau Server to run on multiple machines. This is called a distributed installation. Running a distributed installation uses additional ports on the primary Tableau Server and requires that certain ports be available for binding during Setup on the Tableau worker server. See TCP/IP Ports for more information.

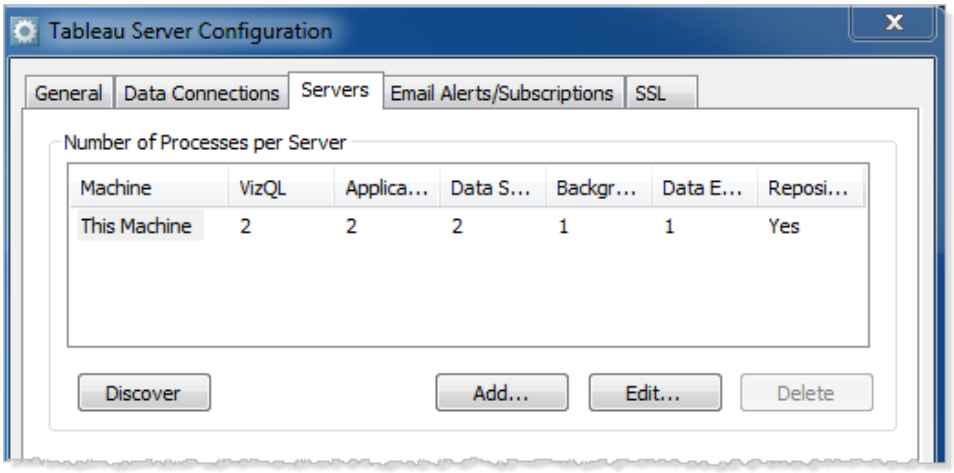
> All machines in a distributed environment must be members of the same domain. Also, the server's Run As User account, which is specified on the primary Tableau Server, must be a domain account in this same domain.

1. Make sure you've installed Tableau Server on the primary machine.
2. Stop the server on the primary machine (see Tableau Server Monitor to learn how).
3. Download the Tableau Server Worker software from the Tableau Customer Account Center.
4. Run the Tableau Server Worker Software installer on all additional machines that you want to add to the Tableau Server cluster. During installation you will be asked to provide the IP address of the primary server.



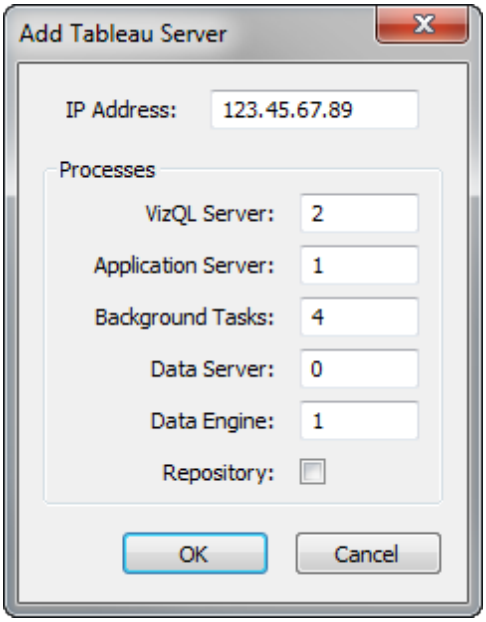
If you have a worker running Windows 7 with Windows Firewall enabled, refer to the Tableau Knowledge Base before proceeding.

5. Once the Worker software is installed on worker machines, and with the primary Tableau Server still stopped, return to the primary server and open the configuration utility by selecting **Tableau Server 8.0 > Configure Tableau Server** on the Start menu.
6. In the Configuration Utility, enter your password on the **General** tab then select the **Servers** tab and click **Add**.



7. In the next dialog box, type the **IP Address** for one of the worker machines and specify

the number of **VizQL**, **Application Server**, **Data Server**, and **Background** processes to allocate to the machine. You can assign up to eight instances of a process to a worker (or primary) server.



By default, the repository and data engine are hosted on the primary server; however, you can select the **Data Engine** and **Repository** checkboxes to use this server for extract storage (data engine) and the repository—or as the standby server for same. See High Availability for more information.

8. Click **OK**. It may take several minutes for the updates to complete.
9. Repeat these steps for each machine you want to add to the distributed environment. When you're finished adding workers, click **OK** again to save the changes, then start the server on the primary machine.

Database Drivers

The installers for Tableau Server and Tableau Server Workers automatically install drivers for Oracle and Oracle Essbase databases. If you plan to publish workbooks and data sources that connect to other databases, you will need to make sure that both your primary and worker machines have the corresponding drivers.

Workers running VizQL, application server, data server, or backgrounder processes need these database drivers. For example, if you have a worker machine dedicated as a VizQL server and another machine dedicated to extract storage, you only need to install drivers on the VizQL server machine.

Server process	Requires data-base driver?
VizQL	yes

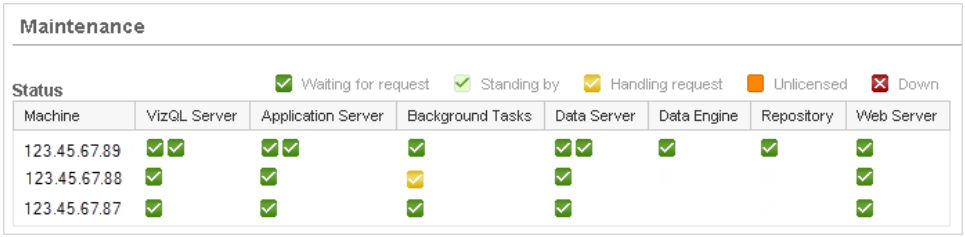
Server process	Requires data-base driver?
Application server	yes
Data server	yes
Backgrounder	yes
Data engine (extract storage)	no
Repository	no

Maintain a Distributed Environment

After you set up a primary and one or more worker machines for a distributed installation, you can perform all subsequent configuration and updates from the primary server, using the command line tools and configuration utility on the primary server machine. Updates will be pushed to the workers automatically.

If the primary server changes its IP address, you will need to re-install all of the worker machines.

You can monitor the status of the distributed machines on the server Maintenance page. See Server Maintenance to learn more about maintaining the server.



High Availability

Use the links below to learn more about Tableau Server's support for high availability:

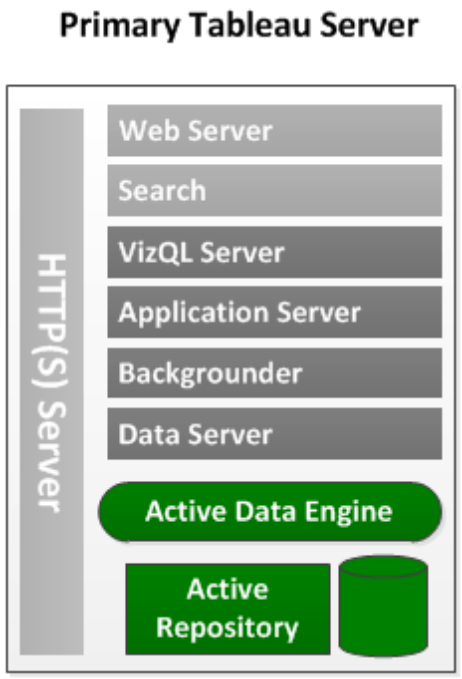
Understanding High Availability

If you're configuring a Tableau Server system for high availability, the steps you perform are all about building in redundancy, thus reducing your potential downtime. The three areas that require redundancy are the data engine process, the repository process, and the primary Tableau Server (gateway). Because there must always be one active instance of each of these, configuring the cluster is a multi-phased procedure that requires the primary Tableau Server to be stopped and restarted at certain points so that settings can take effect. For exact steps, see Configure for Failover and Configure a Highly Available Gateway. See High Availability Requirements as well.

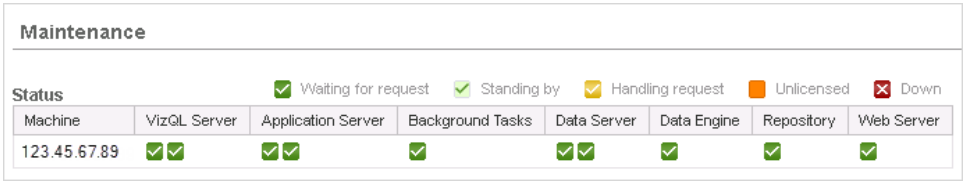
The topics below summarize how your server system topology evolves as you configure it for high availability. The minimum configuration for high availability is a three-node system. This includes a lightweight gateway that routes requests and two workers that host the main processes. You can increase reliability of the system by adding a fourth computer to serve as a backup gateway.

A Single Server System

After you install the primary Tableau Server, it is running at least one instance of all server processes. This is the most basic configuration of Tableau Server. It has no redundancy.



Here's what the Status table on the Maintenance page typically looks like for a single-server system:

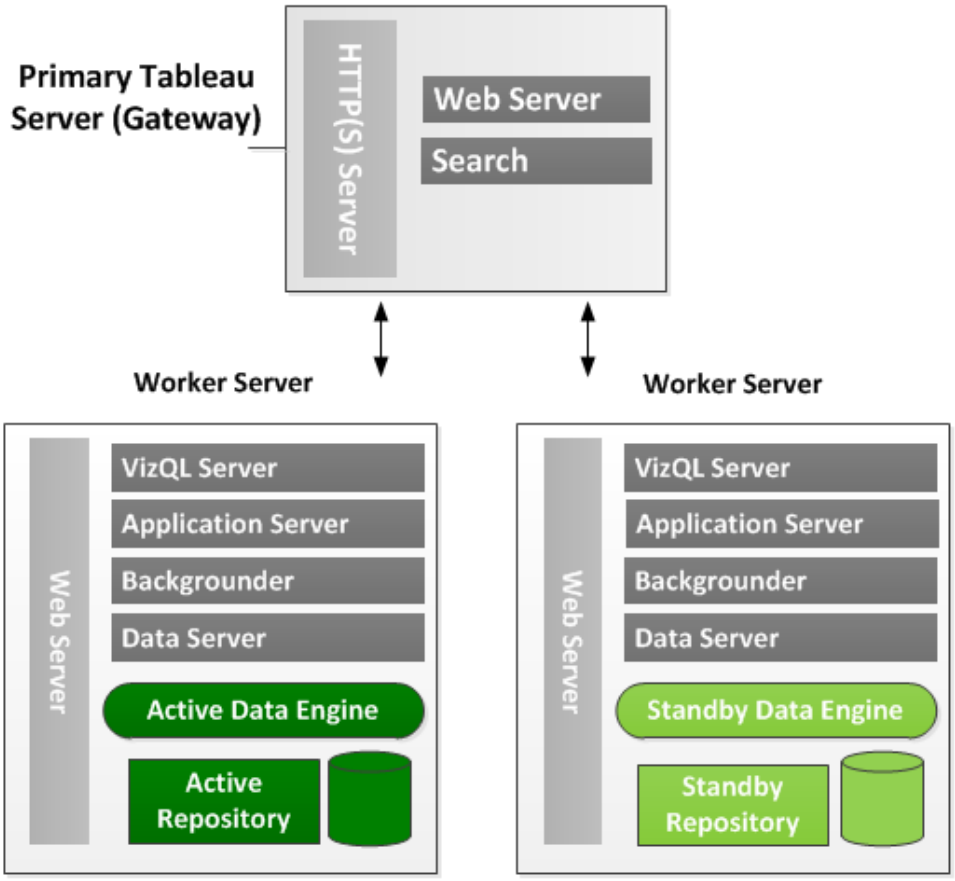


To build in redundancy, you need to add additional servers to host the active and standby data engine and repository processes. In addition, to reduce the system's vulnerability, the gateway should be isolated on its own node, ideally running as few of the server processes as possible.

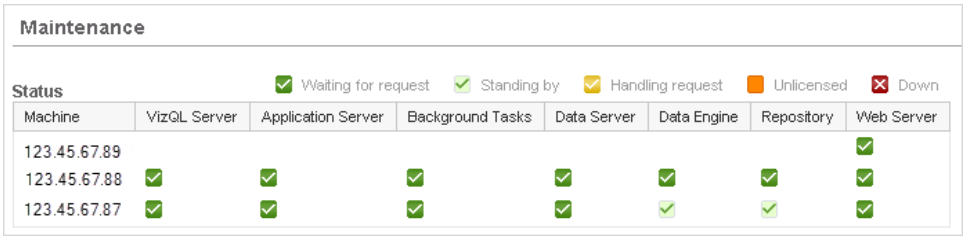
A Three-Node System

The next step in configuring for failover support and high availability is to install Tableau Server on two workers and add them to the primary's configuration one at a time. Assigning the data engine and the repository processes takes two steps because: 1) there must always be one active instance of each process, and 2) there cannot be more than two total. Processes also need to be removed from the primary Tableau Server.

Exactly how to add the workers and remove the processes from the primary is described in Configure for Failover. After you go through those steps, you have a three-node system:



The Status table on the Maintenance page looks similar to the following:



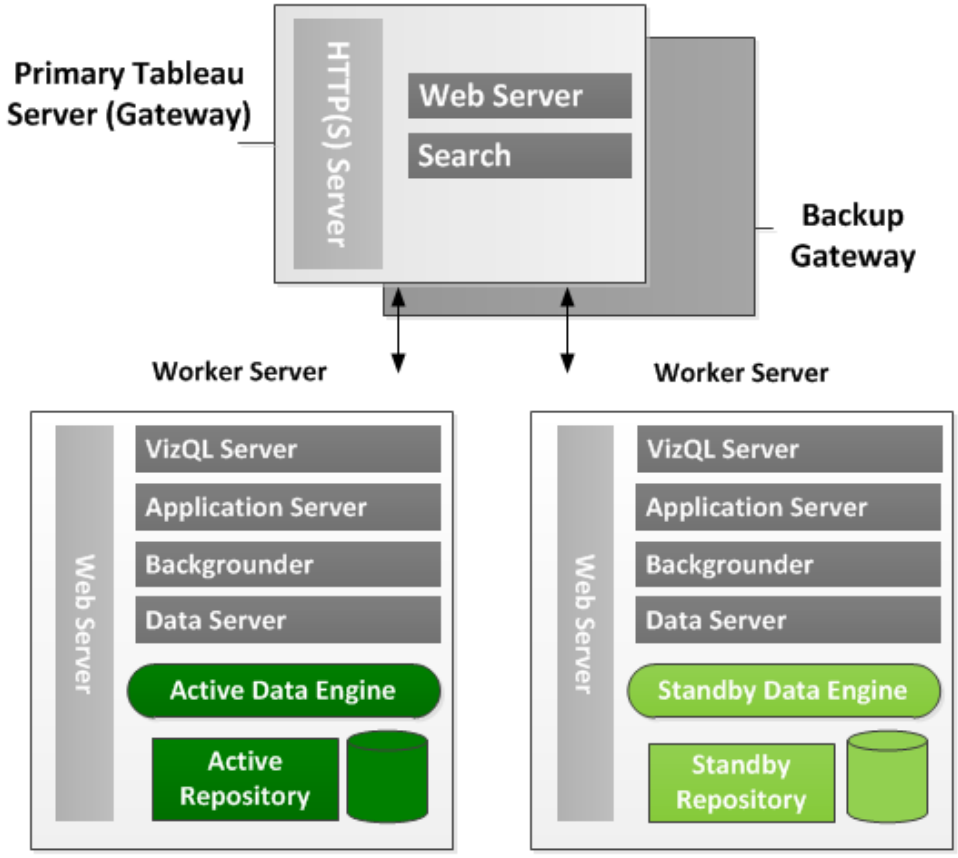
As you can see, the node running the primary only has the web server on it. Its purpose is to route requests to the two workers. In this configuration, if your active worker fails, the standby worker automatically becomes active and handles all requests from the gateway.

However, in a three-node system there is still a single point of failure: the gateway. You can mitigate risks in this area by creating a backup gateway. With a few manual steps, you can activate the backup gateway if the primary gateway fails.

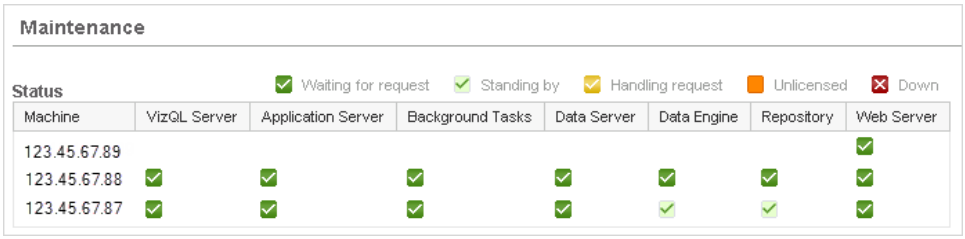
Adding a Backup Gateway

Adding a backup gateway provides a safeguard for your system. The backup gateway is an additional server added to the system to be ready if your primary gateway fails. While it is not an active server, after you complete the first set of steps in Configure a Highly Available Gateway, it is ready to be activated.

Here's what the system looks like with a backup gateway:



The Status table for the above configuration looks the same as for a three-node system. If the primary gateway fails and you perform the steps for the backup gateway to take over, your system is back online using the new gateway:



High Availability Requirements

Before you start to configure a cluster for failover and high availability, make sure you meet the following requirements.

Hardware

With the exception of the gateway, the systems you use for failover and high availability must meet the requirements described in Before you install... but do not need to be identical:

- **Failover—three computers**: To configure a cluster that provides failover support for the data engine and repository processes, you need three computers or VMs: one for the primary Tableau Server and the other two for the Tableau workers.
- **High availability—four computers**: To configure for high availability, you need the three computers or VMs described above plus an additional one to be the backup gateway for your primary Tableau Server gateway.
- **Gateway computers**: If you configure for high availability, the primary Tableau Server gateway and the backup gateway may be running few or no Tableau Server processes. Therefore, the computers that run the gateway do not need as many cores as the ones running your worker servers. You will, however, need adequate disk space for backups. The gateway is temporarily used during the database backup and restore processes.

Networking and Ports

As with any distributed system, the computers or VMs you use need to be able to communicate with one another. See TCP/IP Ports for a list of ports that must be available on the gateways and workers.

Best Practices

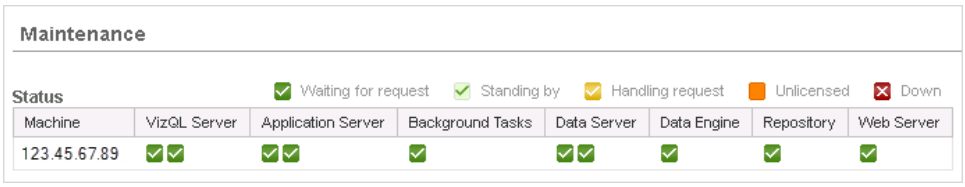
Here are some things to keep in mind before you start to install and configure:

- **IP addresses**: Note the IP addresses of each computer or VM you'll be working with. You will need to provide them during Tableau Worker Setup and configuration.
- **CNAME record**: If you're configuring for high availability, make sure your primary Tableau Server (gateway) and backup gateway have the same CNAME record so that your Tableau Server users have a smooth experience if one gateway fails and you configure the other to take over.
- **User account credentials**: For each machine you're upgrading, you need credentials for a user account with local admin permissions. If you're configuring for high availability, the Run As account you use for your primary Tableau Server gateway must be the same as the one you use for your backup Tableau Server gateway.
- **Backup**: It's a best practice to create a backup prior to making significant system changes. See Back Up the Tableau Data for steps.

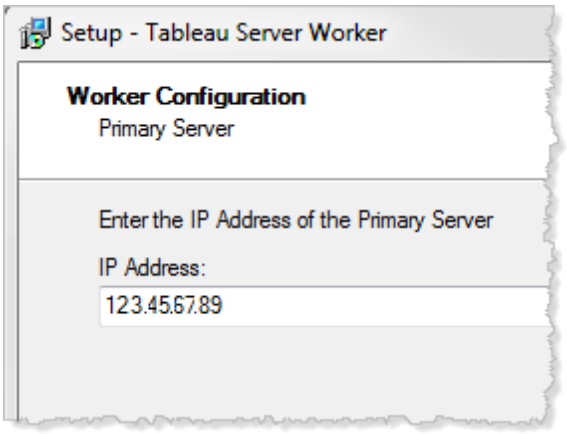
Configure for Failover

Do the following to configure a three-computer cluster that provides failover support:

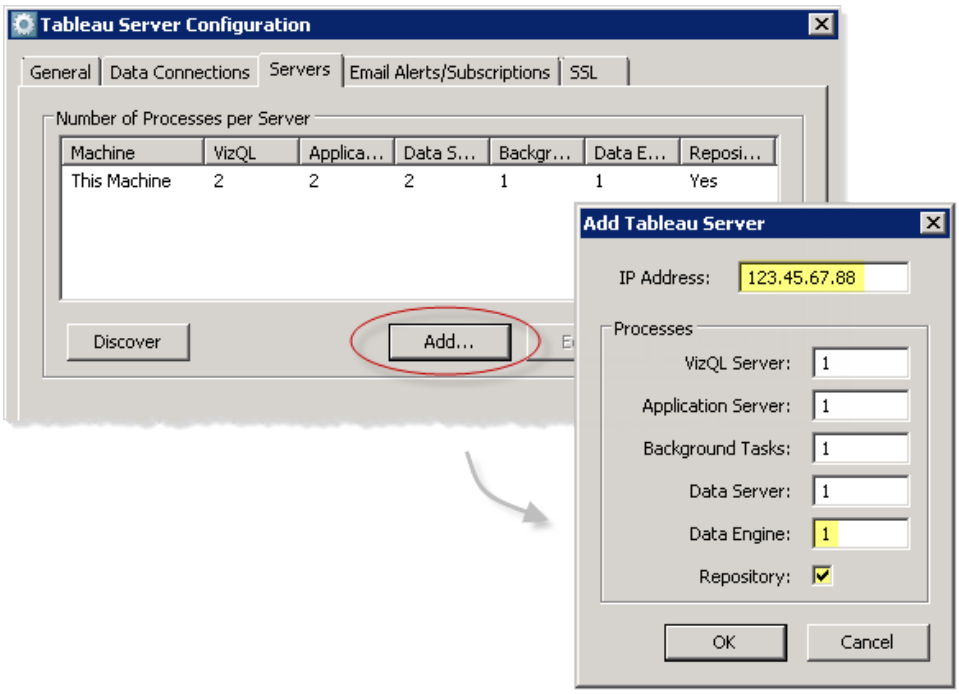
1. Install Tableau Server on your primary computer.
2. After Setup completes, check the Status table on the Maintenance page. All the processes should have a green "waiting for request" status:



3. Stop the server on the primary.
4. Next, run Tableau Worker Setup on the two additional computers or VMs that will provide failover support. During Worker Setup, you will need to provide the IPv4 address of the primary Tableau Server:

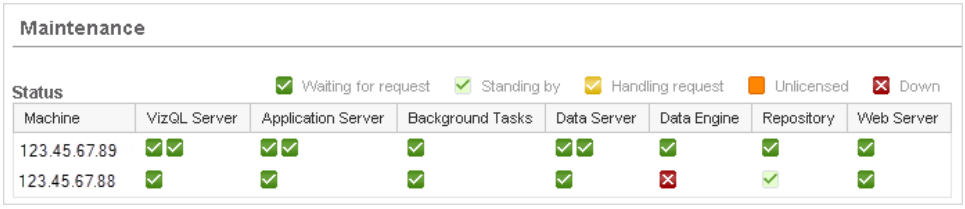


5. With the primary server still stopped, open its Configuration dialog box: **Start** > **All Programs** > **Tableau Server** > **Configure Tableau Server**. On the General tab enter the Run As account password.
6. On the Servers tab, click **Add** to add a worker.
7. Enter the IPv4 address of the worker, enter `1` for **Data Engine** and select the **Repository** check box:

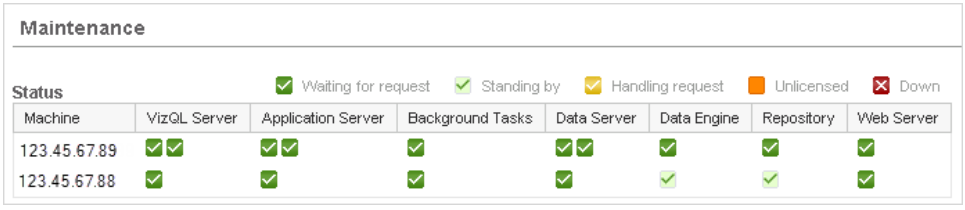


If you want the worker to run other server processes, enter the number of instances you want to run, such as `1` or `2`. The maximum number is eight of each.

8. Click OK to close the Add Worker dialog box then click OK again to close the Configuration dialog box.
9. Start the server on the primary.The repository and data engine processes may initially have the red "down" icon as data is copied to the new repository and/or data engine. The amount of time needed to copy the data will vary, depending on the size of the data:



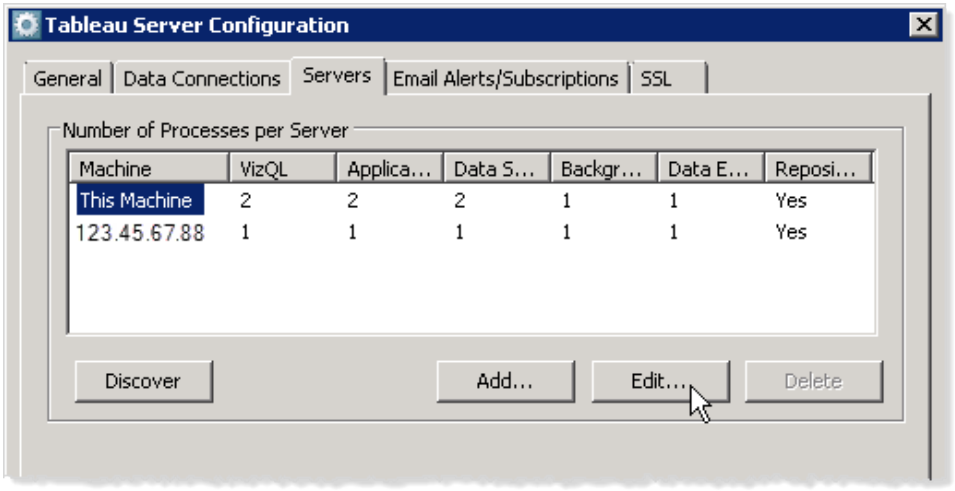
After the data has been replicated, the Status table on the Maintenance page should look similar to the following:



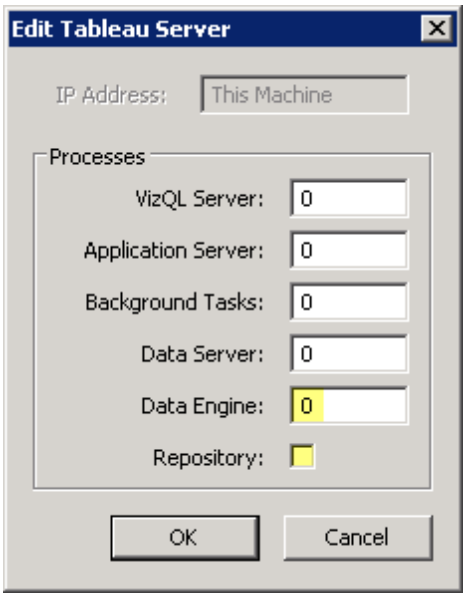
The worker you just added is running standby instances of the repository and data engine processes.

Next, you will remove processes from the primary and add a second worker to run them.

10. Stop the server on the primary and open its Configuration dialog box again. On the General tab enter your password.
11. Select the Servers tab, highlight **This Machine** (which is the primary Tableau Server), and click Edit.

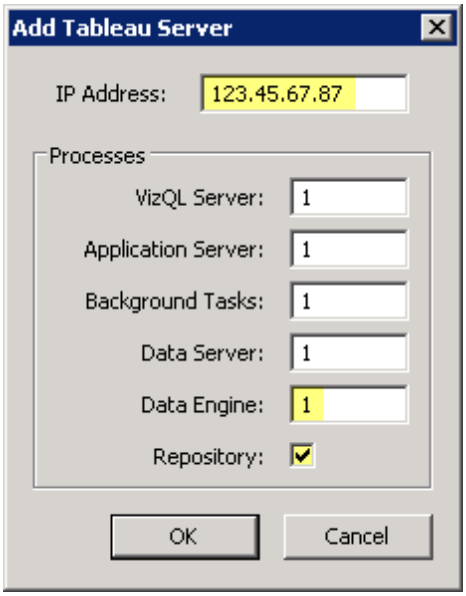


12. In the Edit Tableau Server dialog box, set **Data Engine** to `0` and clear the **Repository** check box. If you want the primary Tableau Server to run nothing but Apache (so, no Tableau Server processes), you can remove the remaining processes from it by entering 0 in each text box:



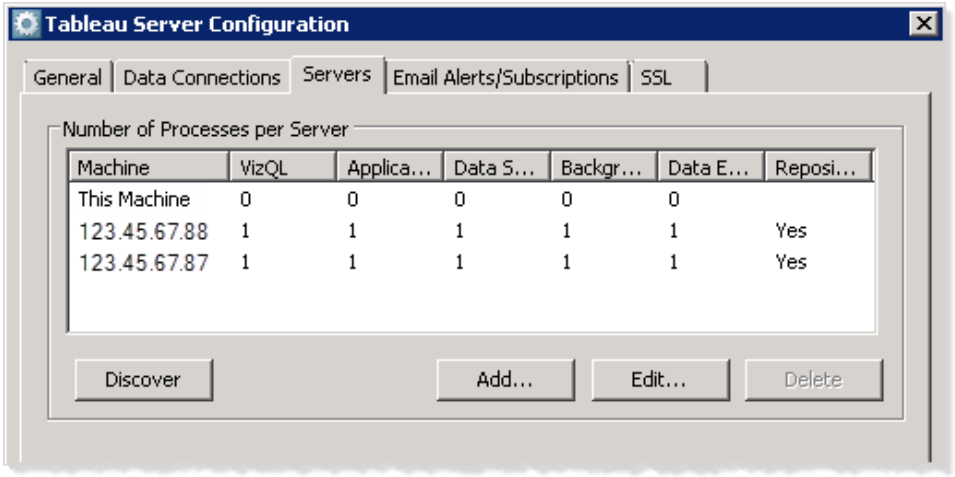
13. Click OK.
14. In the Tableau Server Configuration dialog box click **Add** to add a second worker.

15. In the Add Tableau Server dialog box, enter the IP address of the second worker, set **Data Engine** to 1, select the **Repository** check box, and enter the number of instances you want to run, such as `1` or `2`. The maximum number is eight of each.



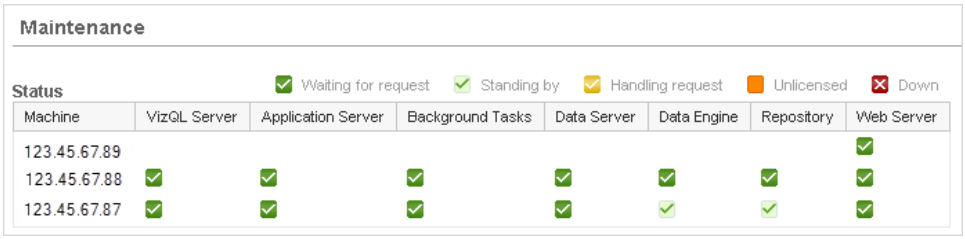
You do not need to specify which worker is active and which is standby for the data engine and repository.

16. Click OK. The Tableau Server Configuration dialog should now look similar to the following:



17. You can also set up email alerts so that you're notified of server failures or changes in status for your data engine and repository processes. To do this, click the Email Alerts/Subscriptions tab in the Configuration dialog box and follow the steps in Configure Email Alerts.
18. Click OK.

19. Start the server on the primary (it may take a few minutes for your changes to take effect). Your system is now configured to provide failover support for the data engine and repository processes. The Status table on the Maintenance page should look similar to the following:



A light green check mark means a process is standing by, ready to take over if the active process (dark green check mark) should fail.

Configure a Highly Available Gateway

Before you follow the procedures in this topic, follow the steps in Configure for Failover. After going through those steps, you will have two worker servers that are providing failover support and a gateway (your primary Tableau Server). Your gateway can be running nothing but Apache (no server processes) or it can be running any server process except for the data engine and repository processes. Those must be run by the workers.

The first procedure below describes how to create a backup of your gateway. The second procedure walks you through what to do if your current gateway fails.

Creating a Backup Gateway

Do the following to create a backup gateway:

1. Stop the server on your primary Tableau Server (referred to as the gateway going forward).
2. On the gateway computer, open a command prompt as an administrator and navigate to the Tableau Server bin directory:

   32-bit: `C:\Program Files\Tableau\Tableau Server\8.0\bin`
   64-bit: `C:\Program Files (x86)\Tableau\Tableau Server\8.0\bin`

3. Enter the following command, where `IP_address1` is the current gateway's IP address and `IP_address2` is the backup gateway's IP address:

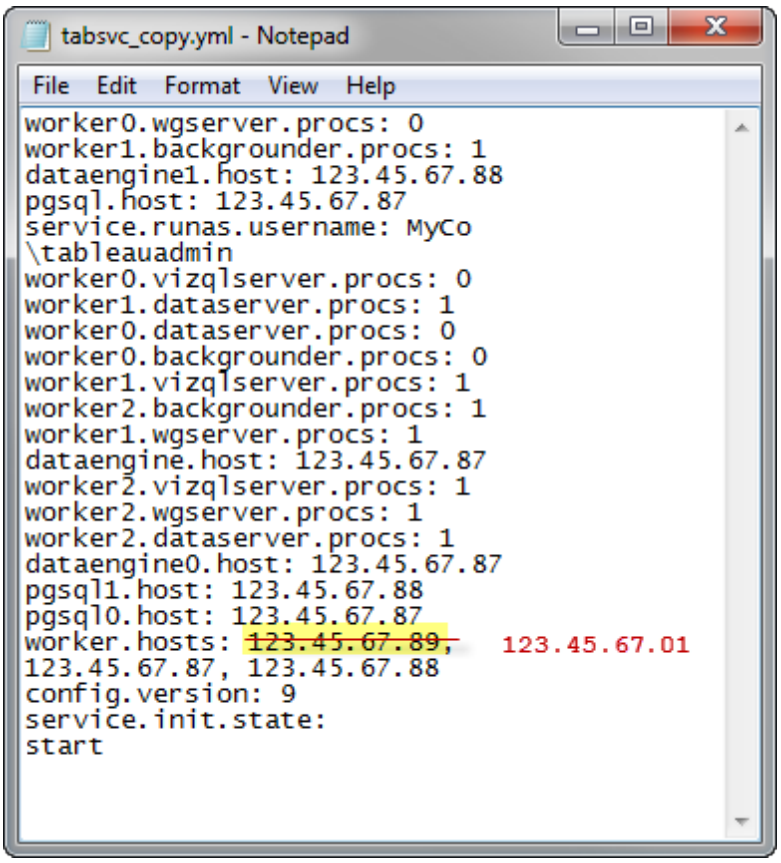
   ```
   tabadmin failovergateway --primary IP_address1 --secondary IP_address2
   ```

4. Next, create a copy of the gateway's tabsvc.yml file (located in ProgramData\Tableau\Tableau Server\config) and put the copy in a temporary location on your backup gateway computer.

   The tabsvc.yml file contains server configuration settings. It gets written to when you change your configuration settings in the Tableau Server Configuration dialog box or via

tabadmin. If tabsvc.yml changes, you will need to update the copy of tabsvc.yml on your backup gateway.

5. On your backup gateway computer, open the tabsvc.yml file and replace the IP address for the gateway on the `worker.hosts` line with the IP address for the backup gateway (the computer you're currently on):



6. On your backup gateway computer, install Tableau Server. Use the same Run As account and configuration settings that you used when you ran Tableau Server Setup on your gateway.
7. After Setup completes, stop the server on the backup gateway computer.
8. Still on your backup gateway computer, enter the following command to disable its Tableau Server service:

   ```
   sc config tabsvc start= disabled
   ```

   You've finished creating a backup gateway. See the next set of steps for what to do if your current gateway fails. If you are working in a test environment, this would be a good time to test your configuration by powering down your current gateway to simulate a system failure..

Configuring the Backup Gateway

Follow this second set of steps in the event of a gateway failure. All steps should be performed on the backup gateway computer.

1. On your backup gateway computer, use the tabsvc.yml file you edited in step 5 of the previous procedure to overwrite the locally installed one in ProgramData\Tableau\Tableau Server\config.
2. Open a command prompt as an administrator and navigate to the Tableau Server bin directory:

   32-bit: `C:\Program Files\Tableau\Tableau Server\8.0\bin`
   64-bit: `C:\Program Files (x86)\Tableau\Tableau Server\8.0\bin`

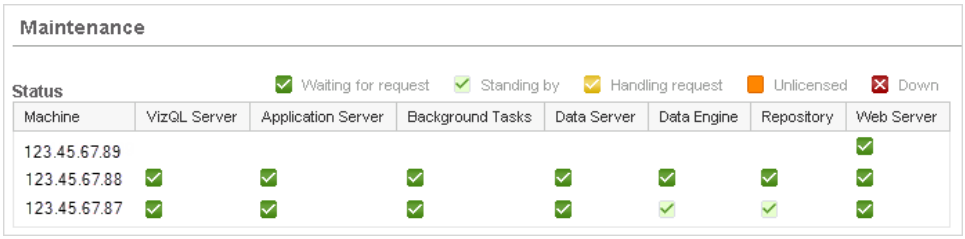
3. Enter the following command, where `IP_address2` is the IP address of your backup gateway (soon to be your new gateway) and `IP_address1` is the IP address of your former gateway (soon to be your backup):

   ```
   tabadmin failovergateway --primary IP_address2 --secondary IP_address1
   ```

4. Enter the following command:

   ```
   sc config tabsvc start= auto
   ```

5. Start the server. Your backup gateway is now your primary gateway. When you look at the Status table on the Maintenance page, you should notice that the IP address for the gateway has changed:



6. For your former primary gateway to now act as your backup gateway, you will need to do the following:
   - Use Add/Remove Programs to remove Tableau Server from your former primary gateway. At the end of the Uninstall program you will receive a backup error, which you can ignore.
   - Delete the Tableau folders under Program Files (x86) and ProgramData on your former primary gateway.
   - Repeat the steps in this topic starting with step 4 under “Creating a Backup Gateway.”

Work with the Server

Refer to the following topics as you use the Tableau Server user interface to administer your installation:

Users and Licenses

Everyone who needs to access Tableau Server, whether it's to publish, browse, or administer, must be added as a user. In addition, users must be assigned a license level.

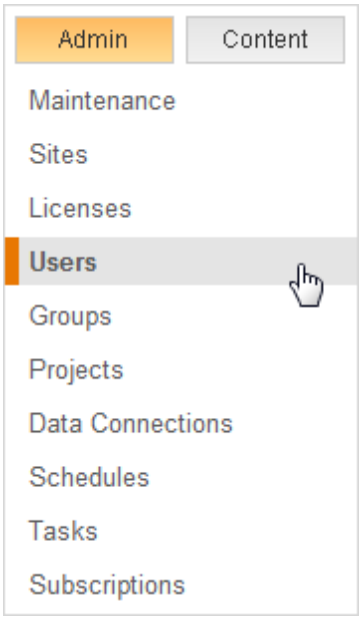
Users

Everyone who needs to access Tableau Server—whether it's to publish, browse, or administer—must be added as a user. If Tableau Server is running multiple sites, the **All Users** page is where system administrators do this. Otherwise, if Tableau Server is in single site mode, system and site administrators can add users on the **Users** page.

Once users have been added, you can edit and delete them, add or remove them from sites, and assign them license levels and user rights. See the topics below for more information.

Add Users

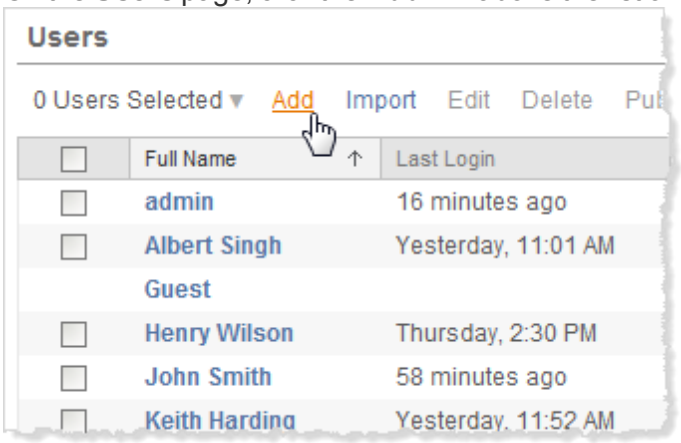
Both system administrators and site administrators with the correct permissions can add users from the **Users** page:



There are two ways you can add users from this page: One at a time (described below) or in batches using the **Import** command, which relies on a CSV file (described in Import Users from a CSV File).

To add a user:

1. On the **Users** page, click the **Add** link above the list of users:



2. Enter a **Username**.
   - **Local authentication**: If the server is configured for local authentication, using an email address for the **Username** is the best way to avoid username collisions (for example, *jsmith@myemail.com* instead of *jsmith*). After you enter a **Username**, click **Add User**.
   - **Active Directory**: If you are adding a user that is from the same Active Directory domain that the server is running on, you can type the **Username** without the domain. The server's domain will be assumed.

     If there is a two-way trust set up between the server's domain and another domain, you can add users from both domains. The first time you add a user from the "non-server domain," use the fully-qualified domain name with the username. Subsequent users can be added using the domain's nickname. For example, assuming a "non-server domain" of *mybiz.lan*, enter the first user from that domain as *user1@mybiz.lan* or *mybiz.lan\user1*. The next user can be entered using the domain's nickname, such as *user2@mybiz* or *mybiz\user2*.

     Note: Be sure not to enter the user's **Full Name** in this field as it can cause errors during the importing process.
3. **Local authentication only**, provide the following:
   - **Full Name**—Type a display name for the user (e.g., *John Smith*).
   - **Password**—Type a password for the user.
   - **Confirm**—Retype the password.
4. **License Level**: Select a license level. See Licenses and User Rights and Permissions to learn more.
5. **User Rights**: Select whether the user can publish workbooks and assign administrator

rights. Refer to Allow or Deny User Rights to learn more.

6. Click **Add**.

Note for multi-site servers:

Site administrators can edit an existing user's account as long as the user is only a member of sites that the site administrator also controls. For example, if User Joe is a member of Site A and Site B and the site administrator is only an administrator of Site B, the site administrator can't edit Joe's Full Name or reset his password.

Import Users from a CSV File

If you are importing users through a CSV file, see the following topics for format requirements, multi-site server considerations, and how to do the import.

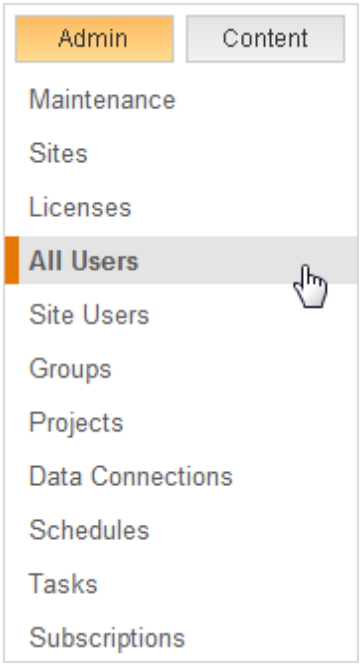
Requirements

- The CSV file must be saved in UTF-8 format.
- Character encodings other than UTF-8, such as BIG-5, must be converted. You can do this via a "Save As".
- Do not use column headers. If you use column headers (`Username`, `Password`, etc.), Tableau Server will attempt to import them as the literal credentials for the first user in the file.
- The following two columns are always required:
  - `Username`
  - `Password`: If Tableau Server is configured to use Active Directory user authentication, there must be a `Password` column, but the column itself should be empty. If the server is using local authentication, you must provide passwords for new users. See also "Multi-Site Mode and Where you Import From" for other information.
- The CSV file can also have the following additional columns, in the order shown below (after the `Username` and `Password` columns):
  - `Full Name`
  - `License Level` (Interactor, Viewer, or Unlicensed)
  - `Administrator` (System, Site, or None)
  - `Publisher` (yes/true/1 or no/false/0)
- The order of the columns is what determines how things are imported. The first column is treated as `Username`, the second as `Password`, the third as `Full Name`, etc., regardless of the columns' contents.

Multi-Site Mode and Where You Import From

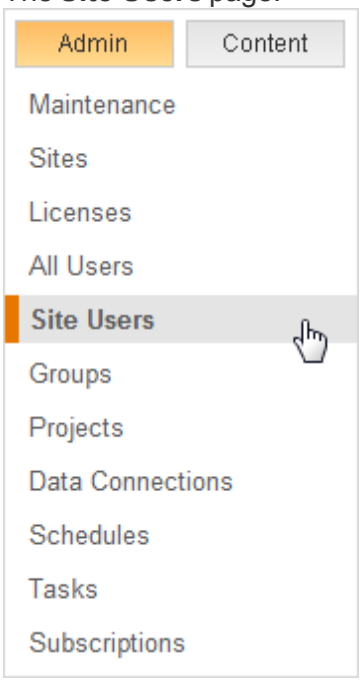
If the server is running multiple sites and you are a system administrator, there are two pages from where you can do a CSV user import. Each has different capabilities where existing server user accounts are concerned.

All Users page: This page displays if a server is running multiple sites and only system administrators can access it.

- 

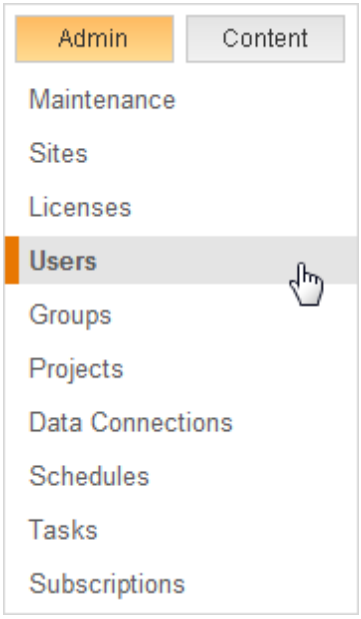
CSV imports from here allow you to update existing server user accounts, in addition to adding new ones. For example, if you do a CSV import that has a new password for each existing user, their passwords will be reset.

The **Site Users** page:

- 

When a system administrator is working from here, they have the same capabilities as a site administrator. This means they can add new user accounts with CSV imports and, if existing users are part of the import, the **Password** and **Full Name** fields must either match or be left blank. If new passwords or full names are used, the import will fail.

If you are a site administrator on a server with multiple sites, you perform CSV user imports from the **Users** page.

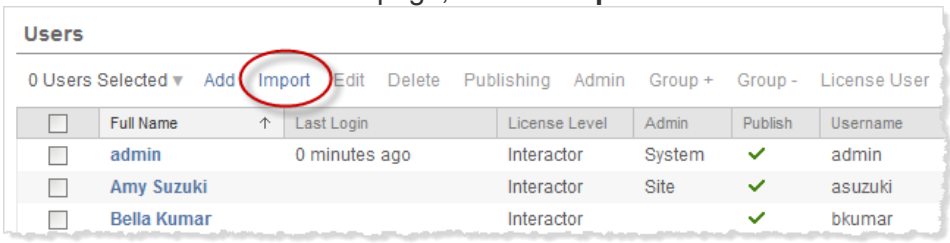


A user can belong to more than one site on the same server system, but they must use the same credentials for each site. This becomes important when you are adding users who are new to your site but perhaps not to the server (in other words, they're a member of a different site on the server). If you think this may be the case, try leaving the `Password` column blank (but keep the required `Password` column header). If the server is configured for local authentication and a new site user is also new to the server system, you'll see a message in the CSV import window telling you to provide a password for the user.

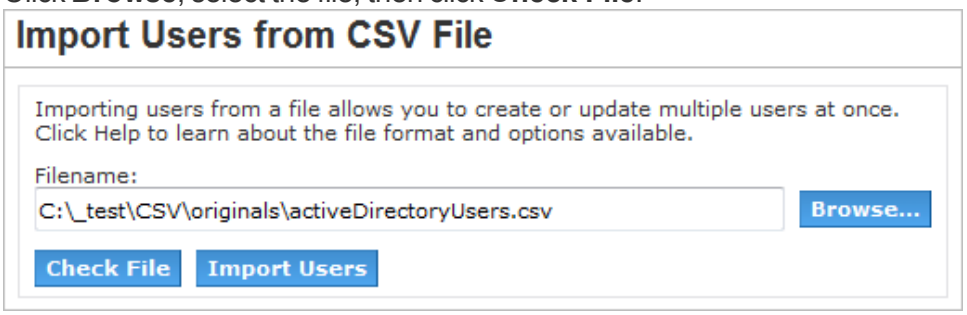
Adding Users from a CSV File

To add users from a CSV file:

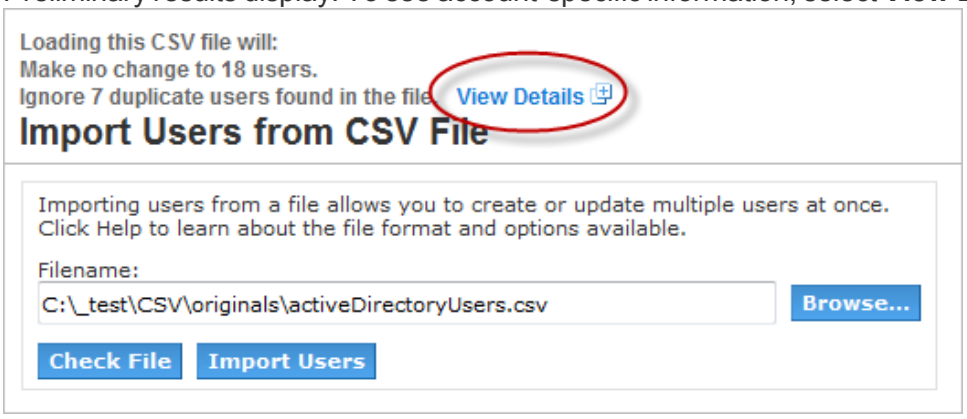
1. From the **Users** or **All Users** page, click the **Import** link:



Click **Browse**, select the file, then click **Check File**:

2. 

Preliminary results display. To see account-specific information, select **View Details**:

3. 

4. To continue, click **Import Users** then click Exit in the final dialog box. To reimport the file or select a different file, click **Browse** again.

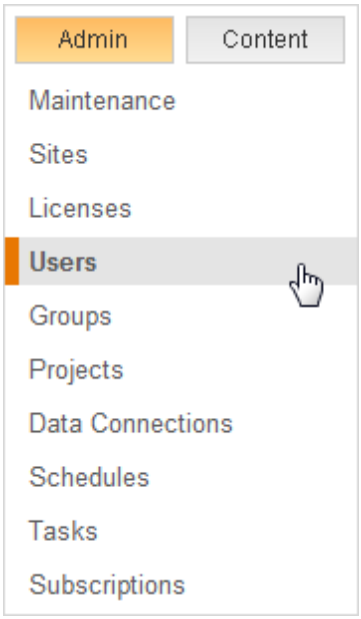
After you import the CSV file, assign the users to a site.

Add Users to a Group

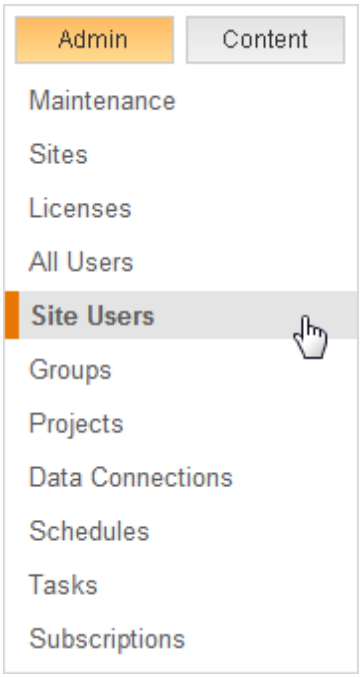
One way to make it easier to manage users is to group them. That way you can assign permissions to an entire group rather than each individual user. To add a user to a group, the group must already exist. See Creating Groups for more information.

To add a user to a group:

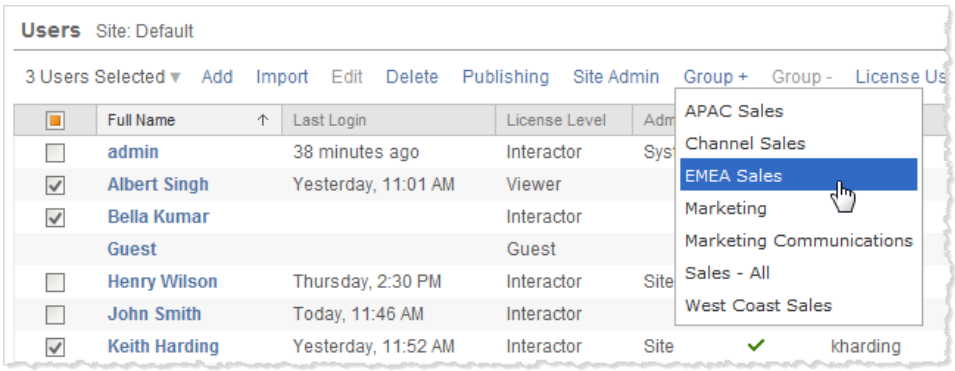
1. On the Admin tab, select the Users page:



   If you are the system administrator for a multi-site server, you'll need to do this on a per-site basis using the Site Users page:



2. Select one or more users.
3. Click the **Group +** link above the user list, then select a group to add the users to:

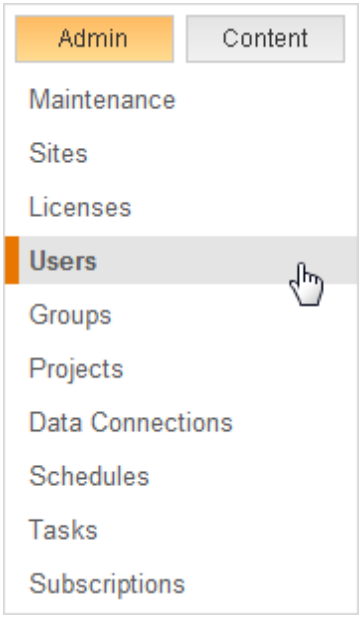


View, Edit and Delete Users

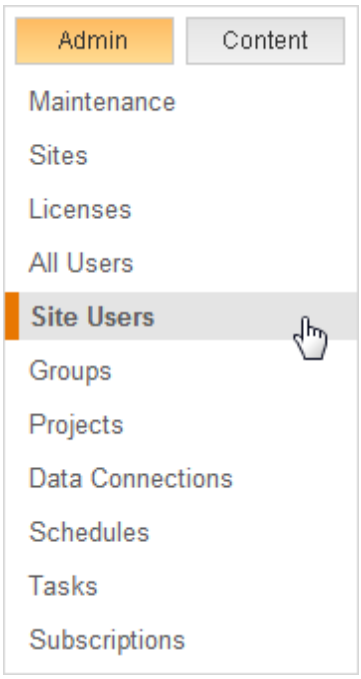
Use this topic to learn how to view, edit, and delete Tableau Server users.

View Users

To view of a list of all users on the server, click **Users** on the Admin tab:



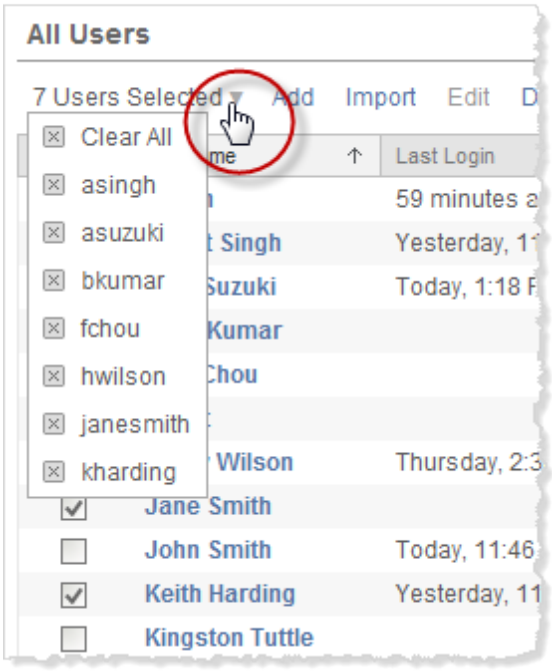
If Tableau Server is running multiple sites, **All Users** lists all users on the server system, and **Site Users** displays all users for the site you're currently logged into:



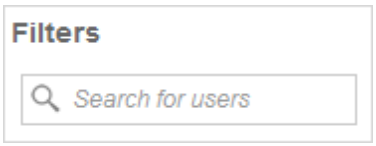
Note:

By default, this list of users is private and can only be seen by administrators. You can make the list of users public by enabling Public User List, in the Settings area of the Maintenance page. If the server is running multiple sites, enabling this setting will only show users the names of users on their site.

Users may be listed across multiple pages. As you select users in the list they are added to a quick list in the upper left corner. The quick list lets you see how many users you have selected and helps you easily remove users from the selection. Click the "x" button next to the username in the quick list to remove someone from the selection.



You can also use the **Search** box under **Filters** on the left to quickly find a specific user in the list.



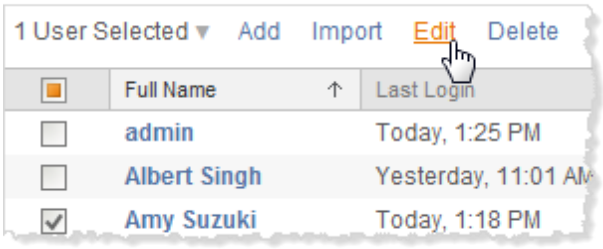
Type all or part of the user's name and press Enter. You can use an asterisk (*) character as a wildcard in the search. For example, searching for `John*` will return all names that start with *John*.

Edit Users

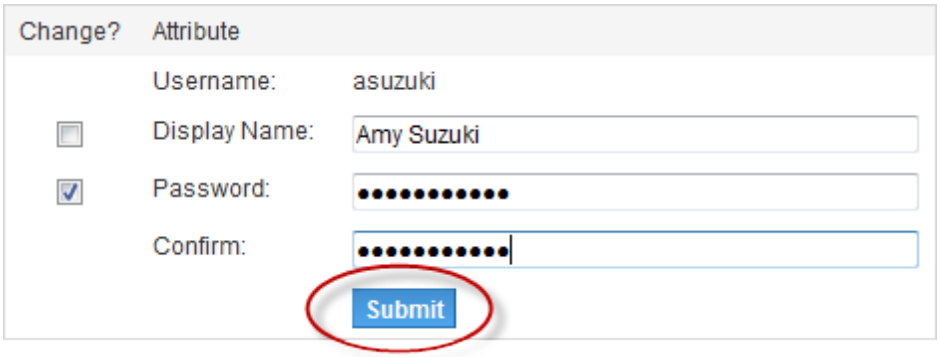
If the server is configured to use the internal user management system (Local Authentication), you can edit the **Display Name** and **Password**for users after they have been added. If you are making many changes it is easiest to import from a CSV file, see Add Users.

To edit user information:

1. Select a single user in the user list.
2. Click the **Edit** link at the top of the list.



3. Type a new **Display Name** and **Password** into the corresponding text boxes.
4. Click **Submit**.



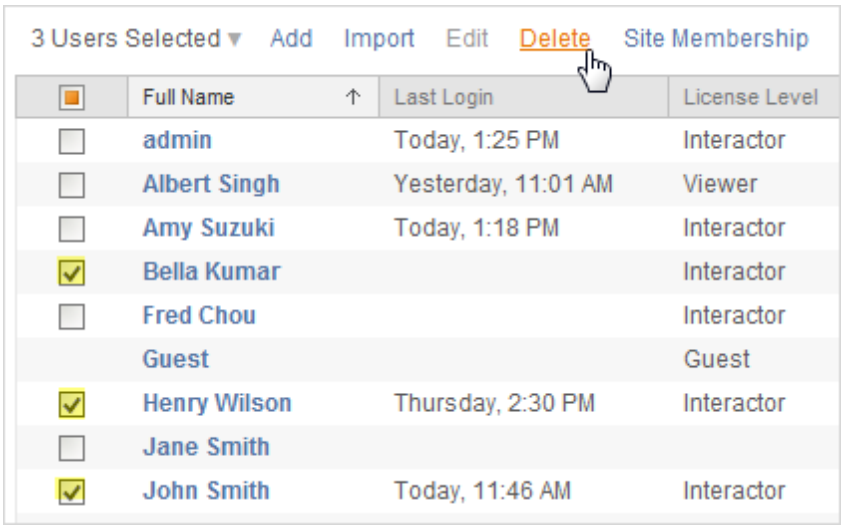
Note for multi-site servers:

Site administrators can edit an existing user's account as long as the user is only a member of sites that the site administrator also controls. For example, if User Joe is a member of Site A and Site B and the site administrator is only an administrator of Site B, the site administrator can't edit Joe's Full Name or reset his password.

Delete Users

You can only remove a user from Tableau Server if the user does not own any content (workbooks, data sources, etc.). If you use the following procedure to delete a user who owns content, the user will be Unlicensed instead of removed. To delete users:

1. Select one or more users to delete.
2. Click **Delete** at the top of the list.



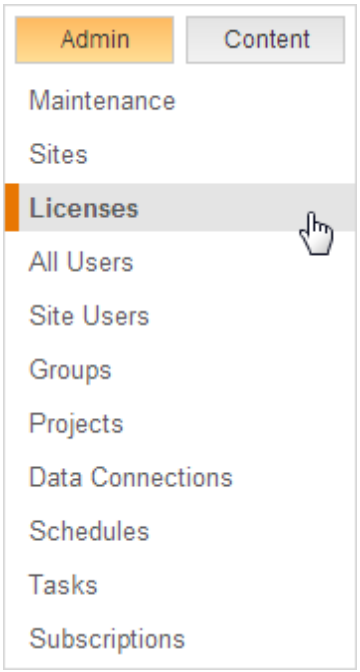
3. Click **Yes** in the dialog that appears.

Licenses and User Rights

Use the topics below to learn more about licensing and user rights.

About License Levels

To open the Licenses page, click the **Licenses** link on the Admin tab:



All users on Tableau Server must be assigned a license level, even if that level is Unlicensed. Tableau Server license levels do not correspond to the Tableau Server named user licenses you purchased from Tableau (if you are using user-based instead of core-based server

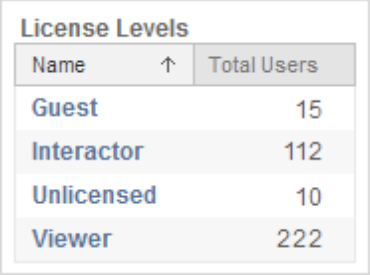
licensing). Those licenses allow you to have a certain number of users on the server. Tableau Server license levels allow administrators to control how much access users have on the server.

Here are the license levels you can assign:

License Level	Description
Unlicensed	The user cannot log in to the server. All users are added as unlicensed by default.
Viewer	The user can log in and see published views on the server but cannot interact with the views. Users with this level can only be given permission to view, add comments, and view comments, they cannot interact with quick filters or sort data in a view.
Interactor	The user can log in, browse the server, and interact with the published views. It's important to note that specific views, workbooks, and projects may have been added with permissions that restrict a user's capabilities. Permission settings can be edited by the workbook author or an administrator.
Guest	The guest license level is available to allow users without an account on the server see and interact with an embedded view. When enabled, the user can load a webpage containing an embedded visualization without logging in. This option is only avail-

License Level	Description
	able with a core-based server.

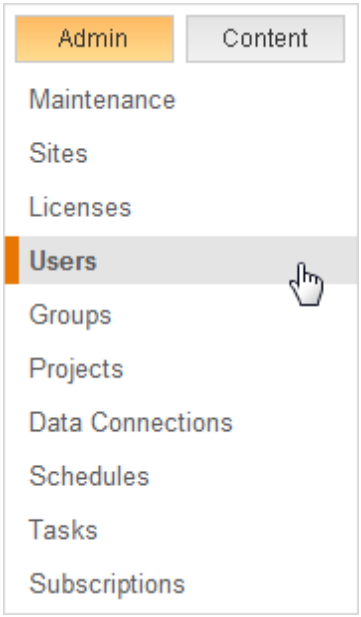
You can review how these levels have been distributed on the Licenses page:



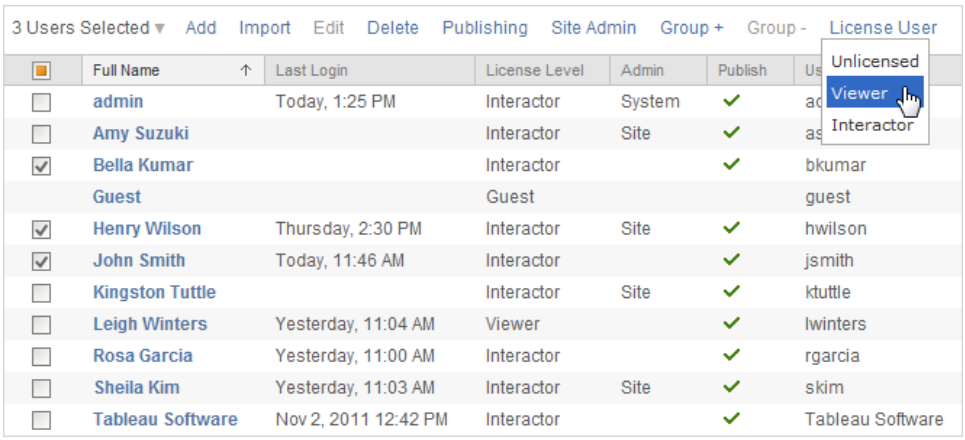
Assign License Levels

To assign license levels to users:

1. Log in to Tableau Server using your administrator user name and password.
2. On the Admin tab, click **Users**.



3. Select one or more users you want to assign license levels to.
4. Click the **License User** at the top of the list.
5. Select either **Unlicensed**, **Viewer**, or **Interactor** for the selected users.



The **License Level** column in the list of users is updated to reflect the changes.

About User Rights

In addition to the license levels, what you can do on Tableau Server is also affected by your user rights:

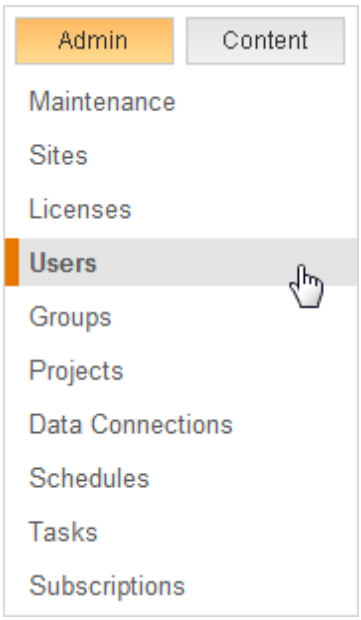
User Right	Description
Publish	Allows the user to connect to Tableau Server from Tableau Professional so that she can publish and download workbooks and data sources.
Admin	Makes the user an administrator. There are two types of administrators: Site administrators and system administrators. • **Site administrators** can manage groups, projects, workbooks, and data connections. By default, site administrators can also add users and assign user rights and license levels but a system administrator can disable that (see sites_edit.htm). • **System administrators** have all the rights of a site administrator, plus they can license unlicensed users, control whether site administrators can add users, create additional system administrators, and they can administer the server itself. This

User Right	Description
	includes handling maintenance, settings, schedules, and the search index. The Admin right can only be assigned to users with the Interactor license level and the Publish right.

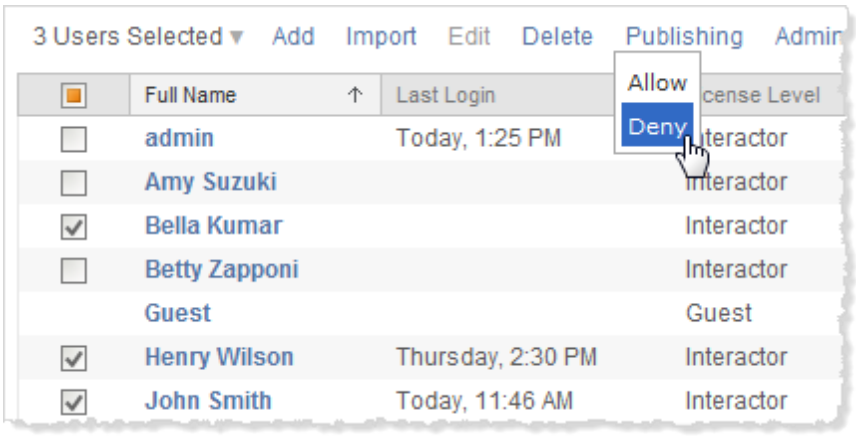
Allow or Deny User Rights

To allow or deny rights for one or more users:

1. Log into Tableau Server using your administrator user name and password.
2. On the Admin tab, click **Users**.



3. Select one or more users you want to assign user rights to.
4. Click the **Publishing** or **Admin** links at the top of the list.
5. Select **Allow** or **Deny** to change the Publishing right for the select user(s).



6. From **Admin**, select **System**, **Site**, or **None** to change the Admin right for the selected users. The Admin and Publish columns in the list of users are updated to reflect the changes.

Work with Permissions

What you can do with views, workbooks, projects, and data sources on Tableau Server is controlled by both your license level (specified by an administrator) and the permissions set by the author of the view or data source.

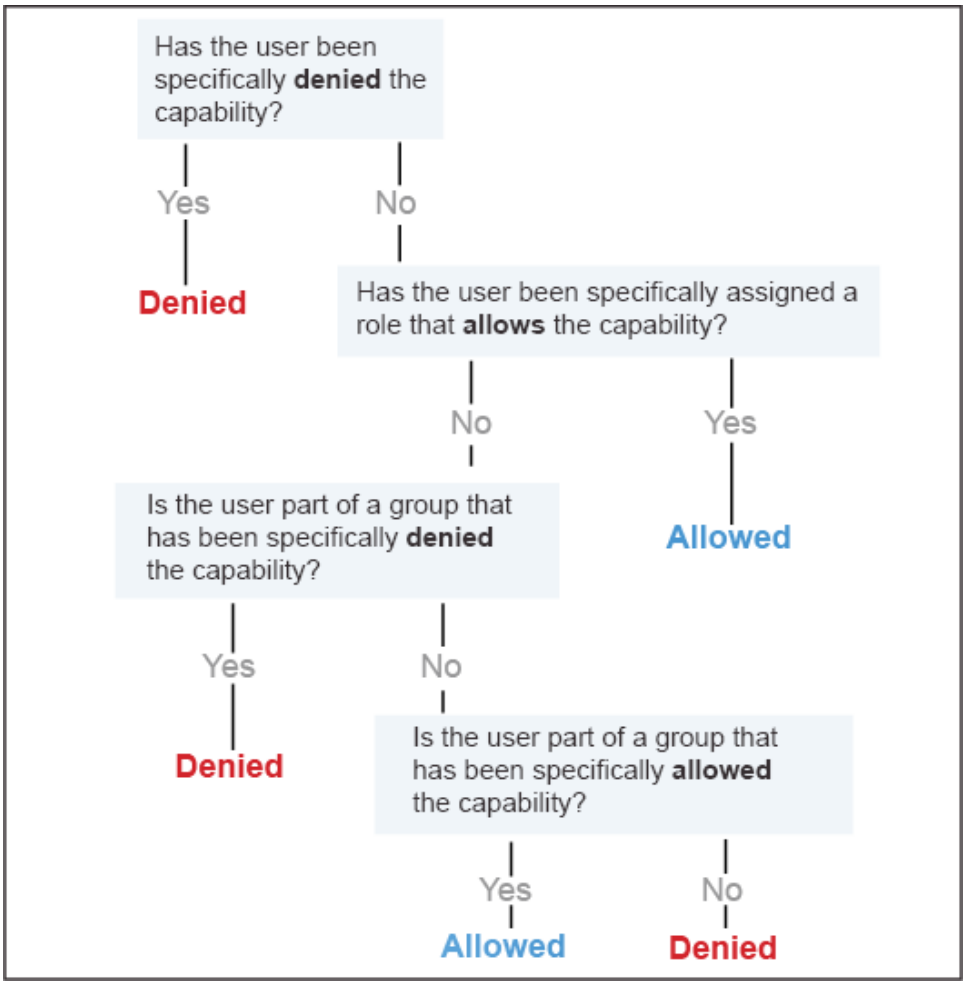
You can change permissions for an item if you have an Interactor license level and at least one of the following is true:

- You are the owner of the workbook or data source (you published it to the server).
- You have been assigned the Set Permissions permission.
- You have been assigned the Project Leader permission for the project that contains the item.
- You have been granted the Admin right.

See the following topics for more information:

How Permissions are Set

The diagram below illustrates how permissions are evaluated.

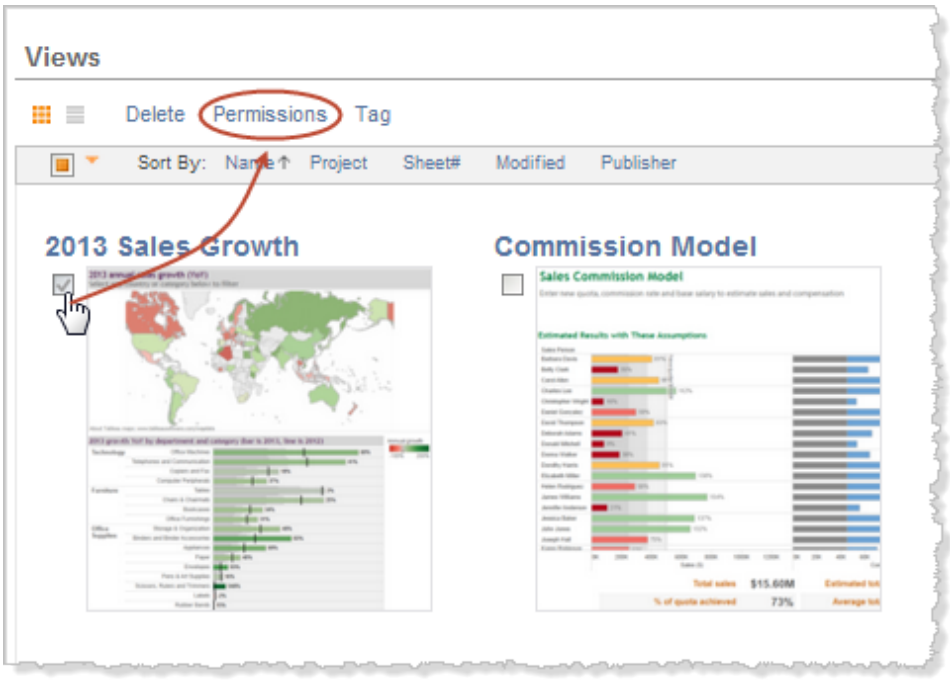


If a workbook is configured to show sheets as tabs, all views inherit the workbook permissions even if different permissions are specified on an individual view.

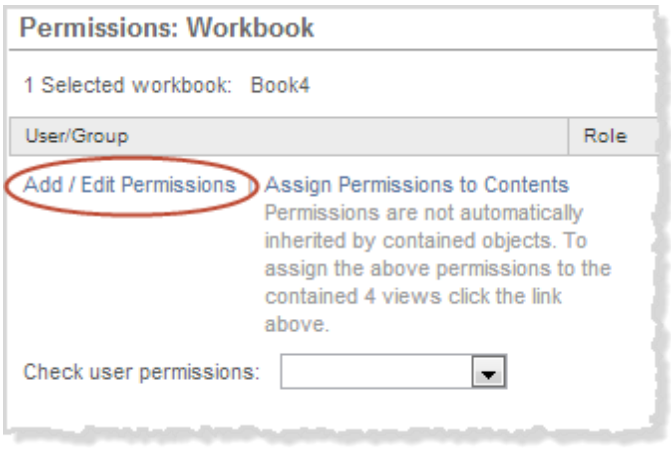
Set Permissions for Workbooks and Views

Follow the steps below to set permissions for a workbook or a view.

1. From a page that displays one or more workbooks, or one or more views, click to select one or more workbooks or views, then click **Permissions**:

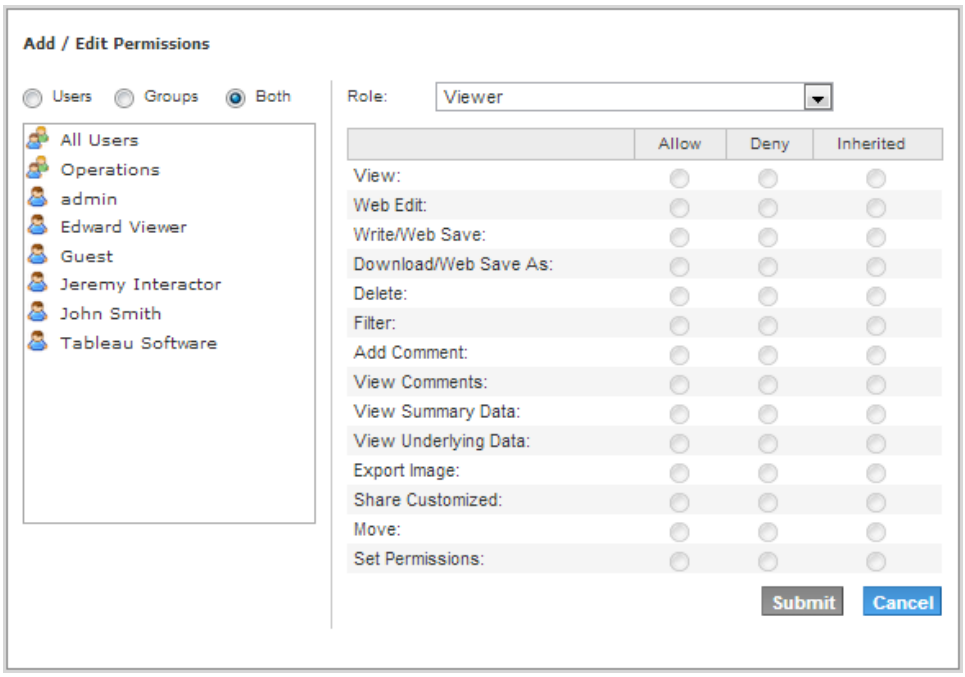


2. Click **Add/Edit Permissions** in the Permissions: Workbook or Permissions: View page:



> The **Assign Permissions to Contents** option is shown for workbooks but not for views.

3. In the Add/Edit Permissions window, select a user or group from the listing on the left:



You can configure the list to show users, groups, or both.

4. Select a predefined role from the **Role** drop-down menu, or specify individual permissions in the area below. The list of permissions and the predefined roles vary a bit depending on whether you are setting permissions for a workbook or a view. See Permissions for a table that defines the various permissions and what items they apply to.

   The available roles for workbooks and views are:

Role	Applies to...	Description
Viewer	workbooks views	Allows the user or group to view the workbook or view on the server.
Interactor	workbooks views	Allows the user or group to view the workbook or view on the server, edit workbook views, apply filters, view underlying data, export images, and export data. All other permissions are inherited from the user's or group's project permissions.
Editor	workbooks views	Allows all permissions to the user or group
Data Source Connector	views	Allows the user or group to connect to the data source on the server. This permission is relevant for views when accessing a view that connects to a data source.

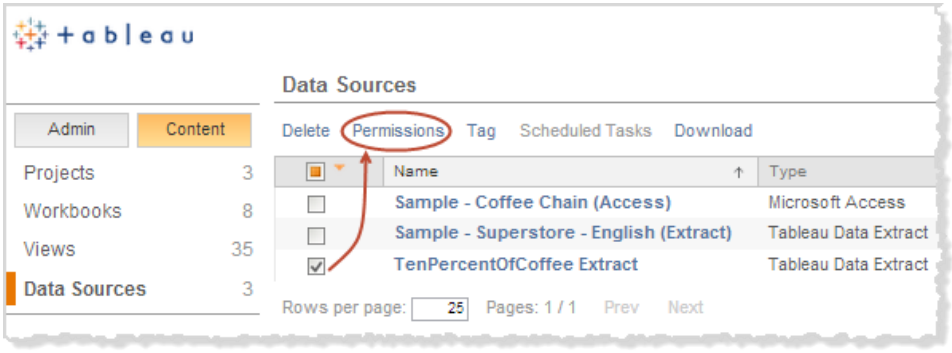
Data Source Editor	views	Allows the user or group to connect to data sources on the server. Also to publish, edit, download, delete, and set permissions for a data source, and schedule refreshes for data sources you publish. This permission is relevant for views when accessing a view that connects to a data source.

5. You can configure permissions for one user or group, or for multiple users and groups. When you are finished, click **Submit**.

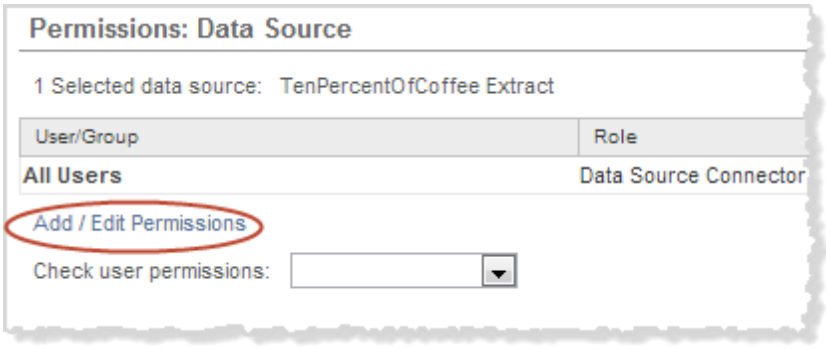
Set Permissions for a Data Source

Follow the steps below to set permissions for a data source.

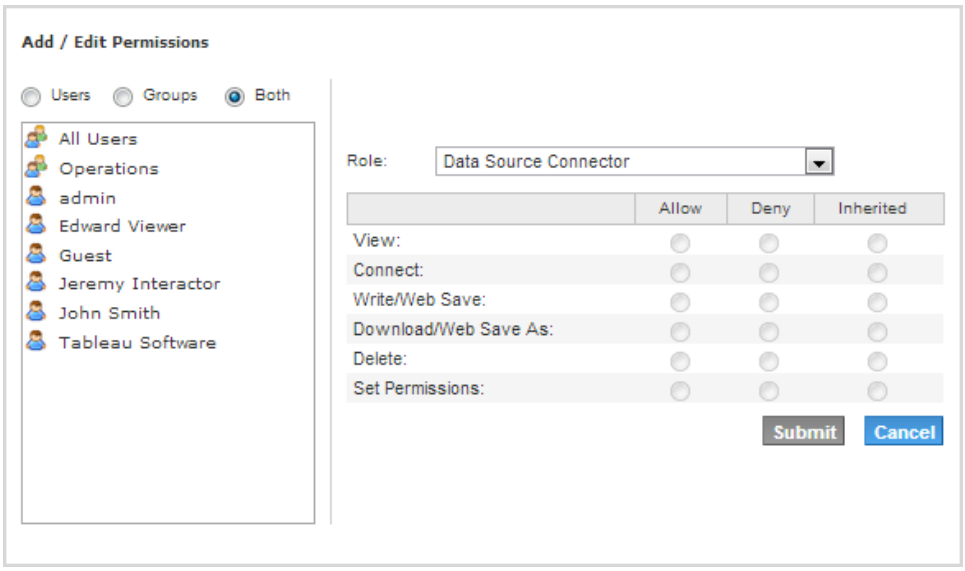
1. From the Data Sources page, click to select one or more data sources, then click **Permissions**.



2. Click **Add/Edit Permissions** in the Permissions: Data Source page:



3. In the Add/Edit Permissions window, select a user or group from the listing on the left:



4. Select a predefined role from the **Role** drop-down menu, or specify individual permissions in the area below. See Permissions for a table that defines the various permissions and what items they apply to.

   The available roles for data sources are:

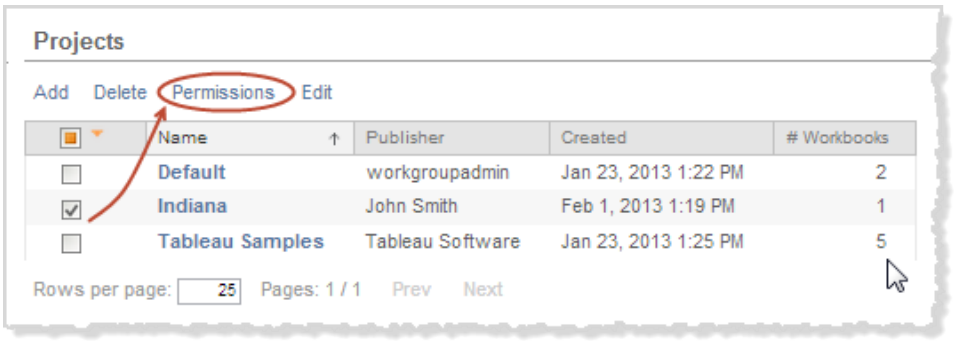
Role	Description
Data Source Connector	Allows the user or group to connect to the data source on the server.
Data Source Editor	Allows the user or group to connect to data sources on the server. Also to publish, edit, download, delete, and set permissions for a data source, and schedule refreshes for data sources you publish.

5. You can configure permissions for one user or group, or for multiple users and groups. When you are finished, click **Submit**.

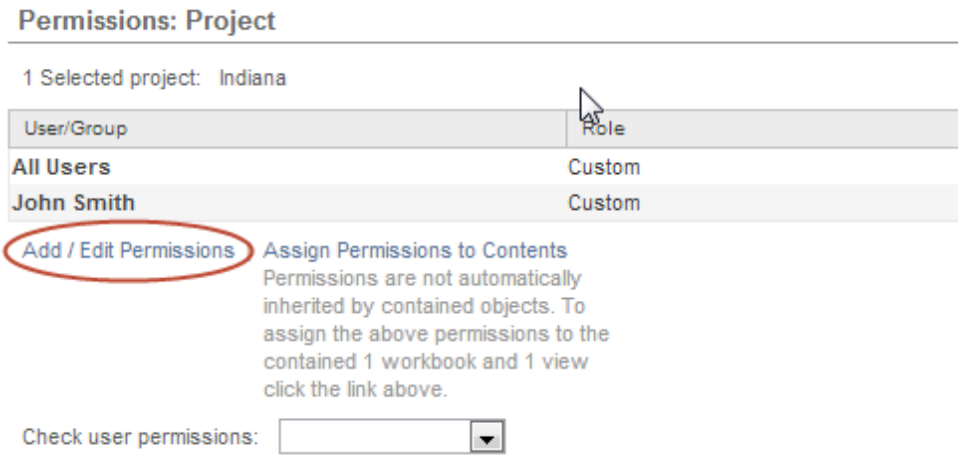
Set Permissions for a Project

Administrators and Project Leaders can specify project permissions. When you create a new project, it has the same permissions as the **Default** project. You can set permissions for the project to allow or deny individual users and groups permission to access the project. To specify project permissions:

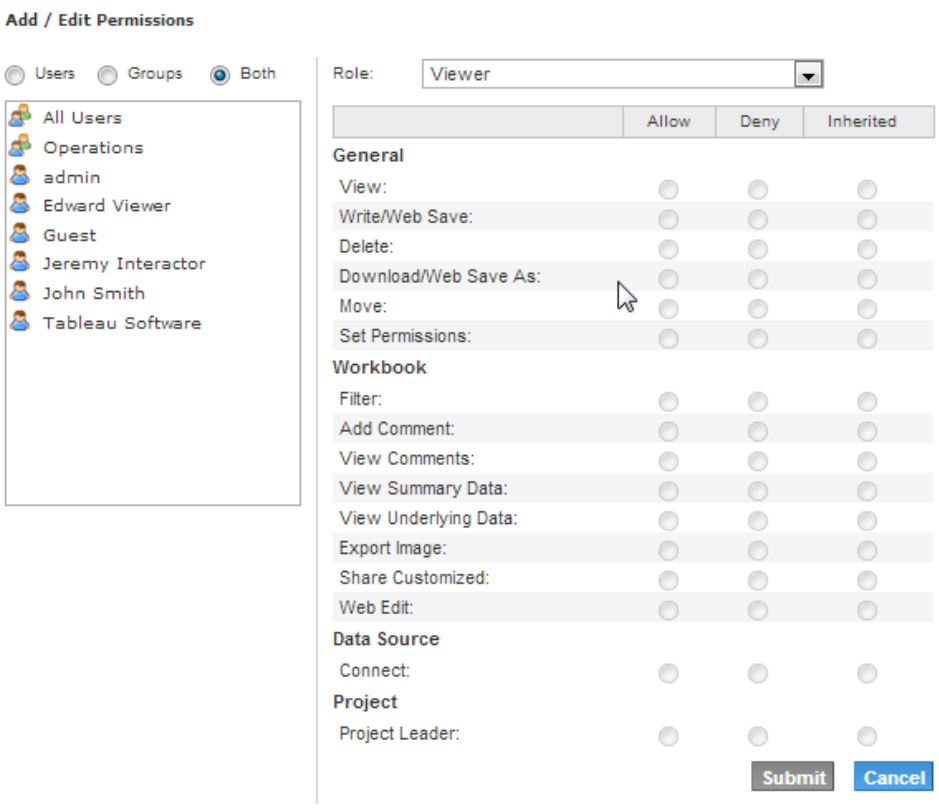
1. Click **Admin**, and the **Projects**.
2. Click to select one or more projects, then click **Permissions**:



3. Click **Add/Edit Permissions** in the Permissions: Project page:



4. In the Add/Edit Permissions window, select a user or group from the listing on the left:



You can configure the list to show users, groups, or both.

5. Select a predefined role from the **Role** drop-down menu, or specify individual permissions in the area below. See Permissions for a table that defines the various permissions and what items they apply to.

   The available roles for projects are:

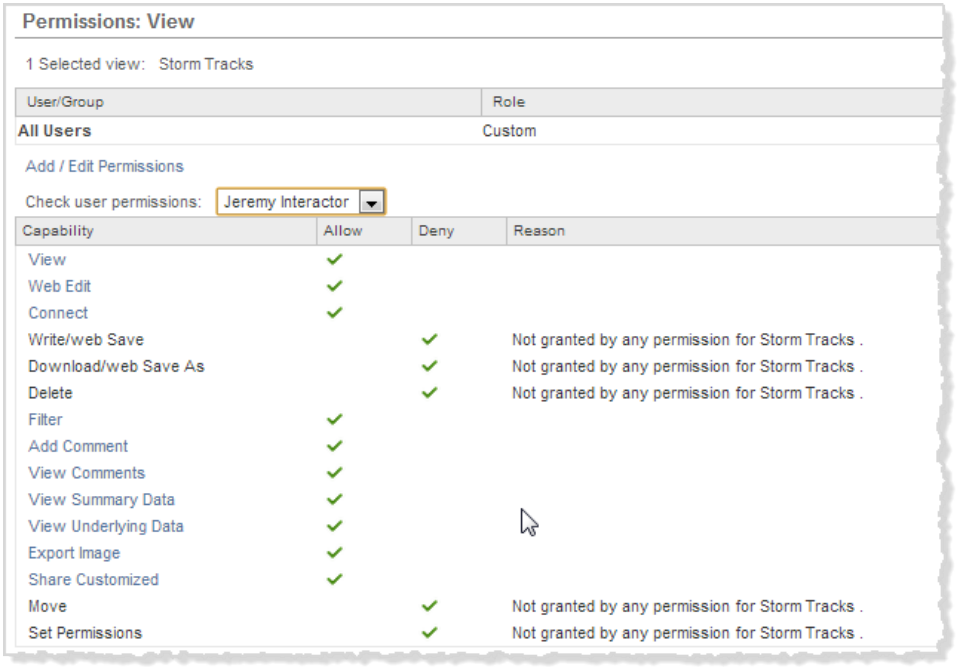
Role	Description
Viewer	Allows the user or group to view the workbooks and views in the project.
Interactor	Allows the user or group to view the workbooks and views in the project, edit workbook views, apply filters, view underlying data, export images, and export data.
Editor	Allows all permissions to the user or group
Data Source Connector	Allows the user or group to connect to data sources in the project.
Data Source Editor	Allows the user or group to connect to data sources in the project. Also to publish, edit, download, delete, and set per-

	missions for a data source, and schedule refreshes for data sources you publish. This permission is relevant for views when accessing a view that connects to a data source.
Project Leader	Allows the user or group to set permissions for all items in a project.
Publisher	Allows the user or group all permissions needed for publishing workbooks to the server.

The permissions you specify apply to the project itself. Any explicit permissions that are set on the workbooks, views, and data sources in the project are not affected. You have, however, the option to assign the project permissions to all of the workbooks, views, and data sources in the project. In that case, those permissions override the existing permissions on the workbooks and views. For example, say there are several workbooks that have each been published with custom permissions and you group the workbooks into a new project with a new permission set. You can apply the new permissions to each of the workbooks by clicking **Assign Permissions to Contents** on the Permission page.

Check Current Permissions

At any time, you can see a user's permissions for a particular view, workbook, project, or data source. From any page where you can set permissions, select a user from the **Check user permissions** drop-down list.



The permissions shown are specific to the view, workbook, data source, or project you have selected.

Permissions

Administrators and other authorized users can allow or deny permissions on actions that users can perform in Tableau Server. Permissions can also be set in Tableau Desktop when publishing a workbook or data source to Tableau Server.

Administrators always have full control of all assets on Tableau Server, and site administrators have full control of all assets on a site. If you publish a workbook or data source to Tableau Server, you are the publisher of that asset, and you retain full control over that asset.

The following table shows which permissions apply to which items in Tableau Server, and describes the actions users can perform with each permission.

Permission	Affects...	When allowed, users can...
View	workbooks data sources views projects	View the item on Tableau Server. A user who accesses a view that connects to a data source must have both View permission for the workbook and Connect permission for the data source.
Web Edit	workbooks views projects	Edit views in workbooks. See Who Can Edit and Create Views . Permissions for worksheets (views) in a workbook are copied (overwritten) from the workbook's permissions when you publish a workbook from Tableau Desktop. They are also copied when you click **Assign Permissions to Contents** on the Permissions: Workbook page. If you select **Show sheets as tabs** when saving a workbook, the permissions for all worksheets (views) in the workbook are overwritten by the permissions for the workbook, but only until such time as tabs are disabled. Special Consideration for the **All Users** group: In the interest of protecting a publisher's content from being overwritten by another user (either via publishing from Tableau Desktop or saving a web-edited workbook on Tableau Server), whenever a user publishes into a project where the **All Users** group has permissions, the **Write/Web Save As** permission for the **All Users** group is changed from Allow to Inherit by default. You can manually modify this permission by following the steps in Set Permissions for Workbooks and Views to change this from Inherit to Allow.
Write/Web Save	workbooks data sources	Overwrite the item on the server. When allowed, the user can re-publish a workbook or data source from Tab-

Permission	Affects...	When allowed, users can...
	views projects	leau Desktop, thereby becoming the publisher and gaining all permissions. Subsequently, the original publisher's access to the workbook is determined by that user's group permissions--and any further permissions the new publisher decides to set. This permission also determines the user's or group's ability to overwrite a workbook after editing it on the server. See Who Can Edit and Create Views .
Download/Web Save As	workbooks data sources projects	When allowed, a user can download the item from the server, and also save an edited workbook as a new workbook on the server. See Download Workbooks and Who Can Edit and Create Views .
Delete	workbooks data sources views projects	Delete the item.
Filter	workbooks views projects	Modify quick filters, keep only filters, and exclude data. See Comment on Views.
Add Comment	workbooks views projects	Add comments to views in a workbook.
View Comments	workbooks views projects	View the comments associated with the views in a workbook.
View Summary Data	workbooks views projects	View the aggregated data in a view, or in the user’s selection within the view, and download that data as a text file.
View Underlying Data	workbooks views projects	View the raw data behind each row in a view, as restricted by any marks the user has selected, and download the data as a text file.
Export Image	workbooks views	Export each view as an image.See Export Views.

Permission	Affects...	When allowed, users can...
	projects	
Share Customized	workbooks views projects	Make saved customizations to a view available for others to see. Users can create custom views using the **Remember my changes** option in Tableau Server. See Customized Views.
Move	workbooks views projects	Move workbooks between projects.
Set Permissions	workbooks data sources views projects	Specify permissions for the item. For workbooks, this permission extends to the views in a workbook.
Connect	data sources views projects	Connect to the data source. A user who accesses a view that connects to a data source must have both View permissions for the view and Connect permission for the data source.
Project Leader	projects	Set permissions for all items in a project.

Groups and Projects

Groups and Projects are available on Tableau Server to help you organize your workbooks and the people using the server.

Groups

You can organize users on the Tableau Server into groups to make it easier to assign permissions to multiple people at once. You can either create groups locally on the server or import from Active Directory. You can create and delete groups on the Groups page which shows a list of all groups on the server or site, if the server hosts multiple sites.

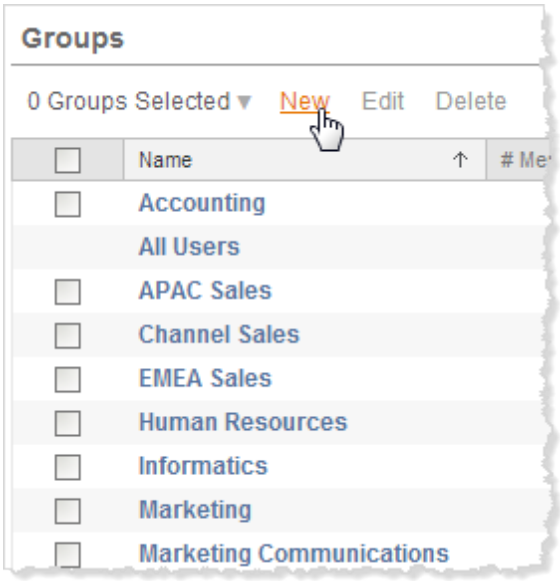
Create Groups

Depending on how the server has been configured you can add groups using the internal user management system (local authentication) or you can import from Active Directory.

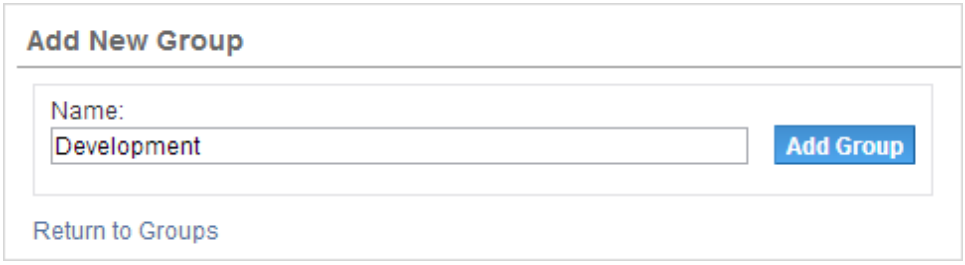
Create a Local Group

A local group is one that's created on Tableau Server using the internal user management system. After you create a group you can add and remove users. To create a local group:

1. Click **New** at the top of the list of groups.



2. Type a name for the group and click **Add Group**:

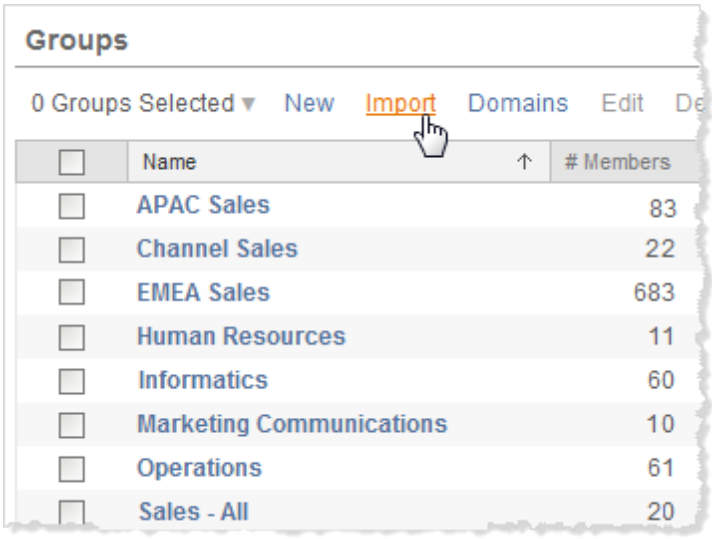


3. Click **Return to Groups** to return to the list of groups.

Create a Group via Active Directory

Groups can also be imported from Active Directory. When you import Active Directory groups, a matching group is created on the server and a user is created on the server for each member of the group. Each user is unlicensed and does not have permission to publish. If the user already exists on the server, he or she is added to the new group and his or her permissions are not changed. See Licenses and User Rights to learn more about license levels and user rights

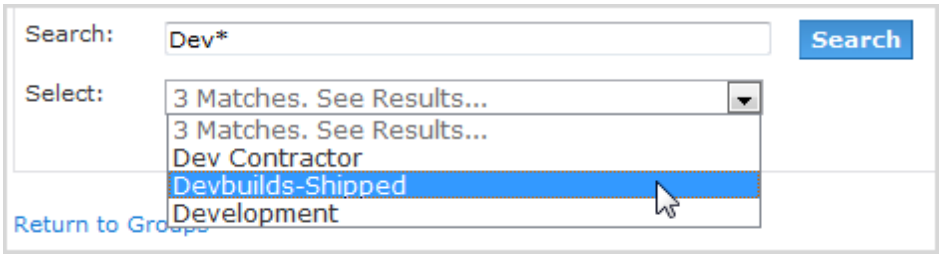
1. Click **Import Active Directory Group** at the bottom of the list of groups.



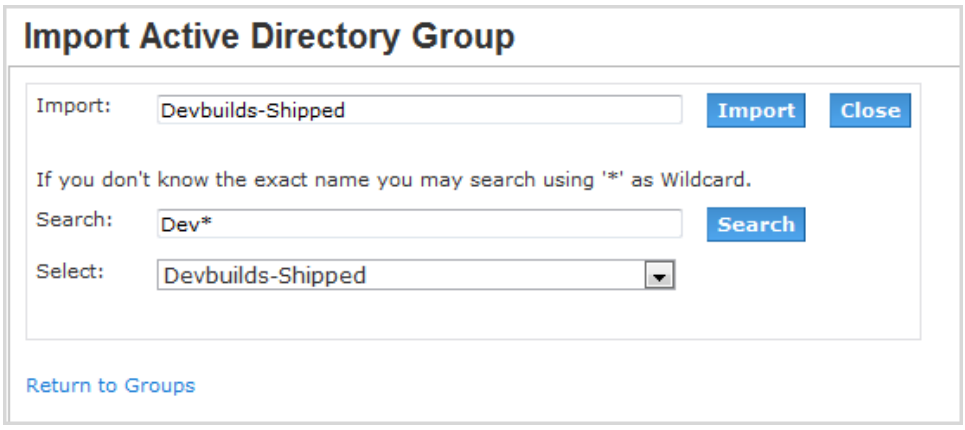
2. Type the name of the Active Directory group you want to import and click **Import**

3. If you don't know the exact name of the group you can find it by typing all or part of the group name into the Search text box. Then click **Search**. You can use the asterisk symbol (*) as a wildcard.

4. Select the group from the list of search results.



5. The group name is automatically added to the Import text box. Click **Import** to add the group to Tableau Server.

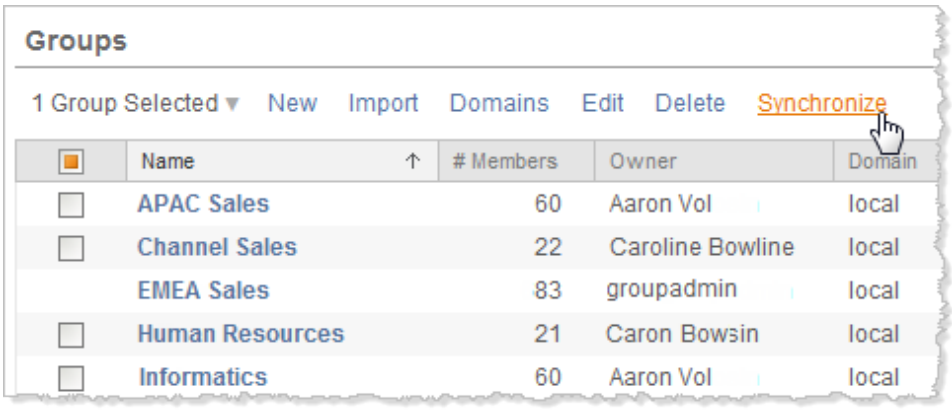


You cannot change the name of groups imported from Active Directory. The group name can only be changed in Active Directory.

Synchronize an Active Directory Group

At anytime, you can synchronize an Active Directory group with Tableau Server so that any new users in Active Directory are also added to the server. You can synchronize individual groups or multiple groups at once.

1. On the Groups page, select one or more groups.
2. Click **Synchronize**.

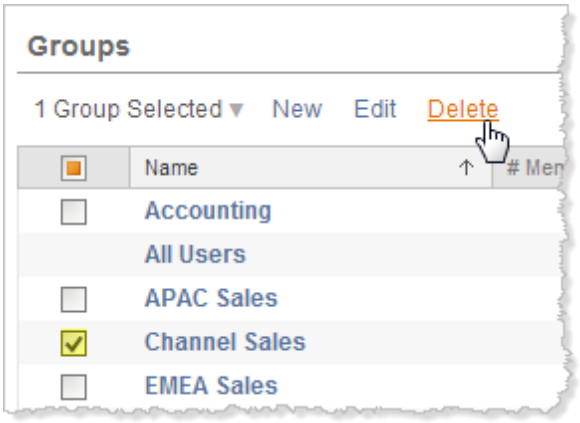


If you are adding a group that is from the same Active Directory domain that the server is running on you can simply type the group name. In addition, if there is a two way trust set up between the domain the server is using and another domain, you can add groups from both domains. The first time you add a group from a different domain than the one the server is using, you must include the fully qualified domain name with the group name. For example, *domain.lan\group* or *group@domain.lan*. Any subsequent groups can be added using the domain's nickname. See Modify Domain Names to learn more about managing domain names.

Delete Groups

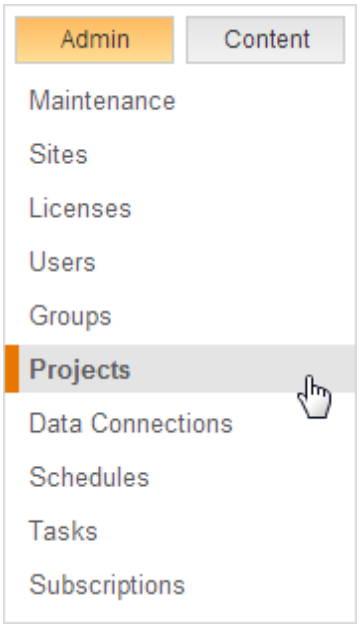
You can delete any group from the server. When you delete a group, the users are removed from the group but they are not deleted from the server.

1. On the Groups page, select one or more groups to delete.
2. Click **Delete** above the groups list:



Projects

A project is a collection of related workbooks. As the administrator, there are two places where you will see **Projects** listed: under the **Content** tab and under the **Admin** tab. If you want to create new projects, assign permissions, or delete projects, use the **Projects** page under the **Admin** tab:



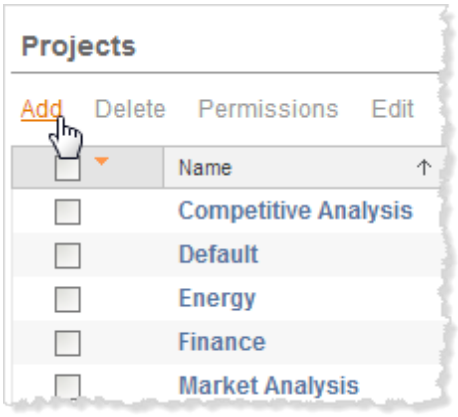
While only administrators can create new projects, users and groups can be assigned the Project Leader permission. This permission lets a user or group specify project permissions

and move workbooks into projects. See the topics below for procedures and more information on working with projects:

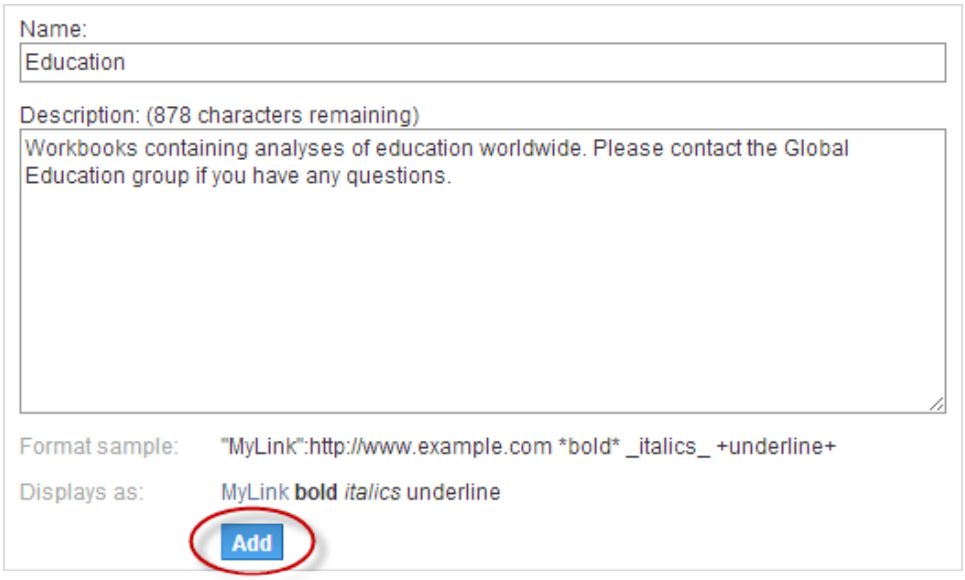
Add Projects

To add one or more projects:

1. Click the **Add Project** link at the bottom of the project list.



2. Type a name and description for the project and click **Add**. You can include formatting and hyperlinks in the project description.



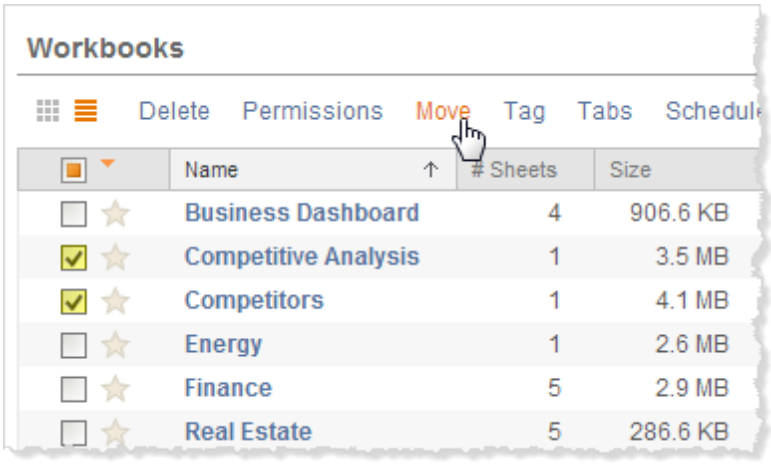
Move Workbooks into Projects

All workbooks must be part of a project. By default, workbooks are added to the **Default** project. After you've created your own projects you can move workbooks into them from any list of workbooks. You can move workbooks into projects if you have an Interactor license level and at least one of the following is true:

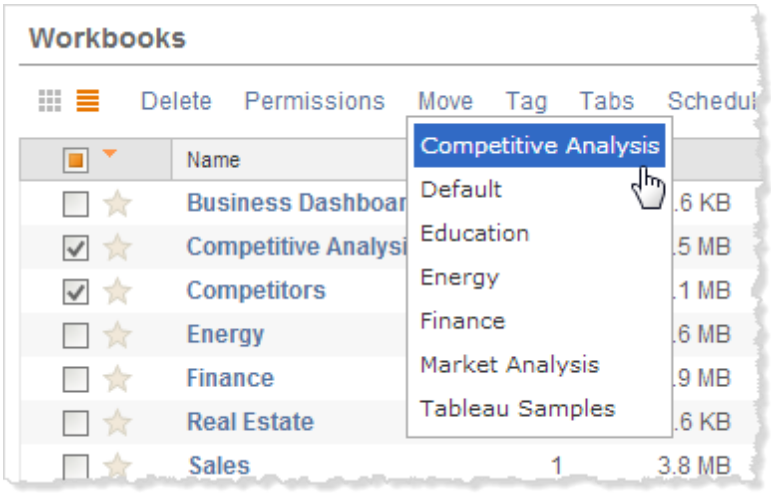
- You have been given permission to Write to the project.
- You have been given Project Leader permission for the project.
- You have been granted the Admin right.

To move a workbook into a project:

1. Select one or more workbooks and click the **Move** link at the top of the list of workbooks.



2. Select a project to move the workbook into.

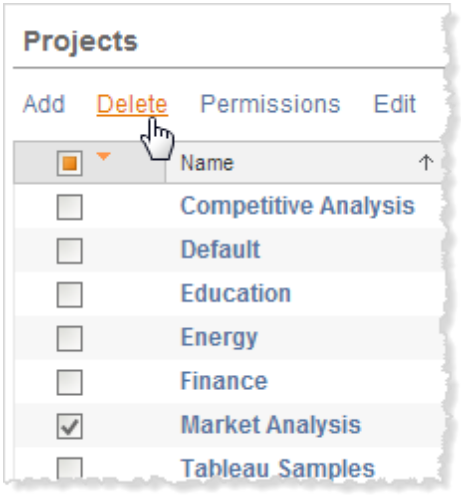


Because all workbooks must be part of a project, you can remove a workbook from a project by moving it to the Default project. Each workbook can only be part of a single project.

Delete Projects

Only administrators can delete projects. When you delete a project, all of the workbooks and views that are part of the project are also deleted from the server. To delete a project:

1. Select the project in the project list.
2. Click **Delete** above the Projects list.



3. Click **Yes** in the confirmation dialog box.

The **Default** project cannot be deleted, even if you have been granted permission to delete this project.

Schedule Refreshes & Subscriptions

Extract refreshes and subscription deliveries are tasks performed by Tableau Server and schedules control when those tasks are run.

As the server administrator you have the highest level of control over server tasks and schedules, however, there are two types of tasks that Tableau Server users can schedule. Workbook authors can schedule extract refreshes when they publish a workbook or a data source with an extract and Tableau Server users can subscribe to views, which are delivered on a schedule. As the administrator, you can adjust an extract or subscription schedule, create new schedules and refresh tasks, and delete them. You also control whether workbook authors are allowed to schedule (see Enable Scheduling), and you are in control of whether the server is configured to send subscriptions (see Manage Subscriptions). Any changes you make to an extract's schedule are reflected in the Tableau Desktop Schedule dialog box the next time the author publishes. Similarly, if you create a new subscription schedule or delete an existing one, it's reflected in the schedule choices a Tableau Server user sees when they subscribe to a view.

See the topics below for more information:

About Extracts and Schedules

Tableau Desktop allows authors to create a data extract, which is a copy or a subset of data from the original data source. Workbooks that use data extracts are generally faster than those that use live database connections because the extracted data is imported into Tableau's built-

in, fast data engine. Extracts can also increase functionality. Once a workbook with an extract has been published, the extract resides on Tableau Server. There are several ways that extracts on Tableau Server can be refreshed, both from Tableau Server and from Tableau Desktop.

Refreshing Tableau Server extracts from Tableau Server:

- **User interface**: As the administrator, you can use options in the Tableau Server user interface to change or reassign exract refresh schedules, regardless of whether a workbook or data source with an extract was given a refresh schedule at the time it was published. Any changes you make in Tableau Server are reflected in the Schedule dialog box in Tableau Desktop when the workbook or data source is published again. You can also refresh an extract immediately, using the **Run Now** option. See Manage Refresh Tasks and Create or Modify a Schedule for details. Note that before you can create refresh schedules you must enable scheduling on the server. See Enable Scheduling to learn more.
- **tabcmd command line utility**: The tabcmd command line utility provides a `refreshextracts` command which you can use from the command line or incorporate into your own script. See Automate Refresh Tasks for more information.

Refreshing Tableau Server extracts from Tableau Desktop:

- **At publish time**: When an author publishes a workbook or data source that uses an extract, he or she can assign it to a recurring refresh schedule on Tableau Server. At the interval the author chooses, Tableau Server refreshes all the data in the extract. Authors can also optionally define an incremental update for an extract, where they identify a column in the extract that has a numerical value, such as a timestamp. (Specifically, the value must be an integer, up to 18 digits long. Dates and datetime are both valid.) Tableau uses this column to identify new rows that need to be added to your extract. This is called an incremental refresh and it can also be scheduled. See Schedules in the Tableau Desktop help for more information.
- **User interface**: You can use Tableau Desktop's **Refresh from Source** and **Append from File** options to upload an addition to or refresh an extract on Tableau Server. You may want to do this if Tableau Server doesn't have sufficient credentials to refresh data from the original data source. See Updating Extracts on Tableau Server in the Tableau Desktop online help for details on how to upload.
- **Data Extract command line utility**: The Data Extract command line utility installs with Tableau Desktop. You can use it to upload an addition to an extract on Tableau Server or refresh it. See Tableau Data Extract Command Line Utility in the Tableau Desktop online help for more information on how to upload.

Enable Scheduling

Before you can schedule an extract refresh, scheduling must be enabled on the server. After you enable scheduling, you can add workbooks and data sources to schedules, create and edit

schedules, manage scheduled tasks, and change schedule settings to allow publishers to assign workbooks to schedules. This setting does not affect scheduling for subscriptions.

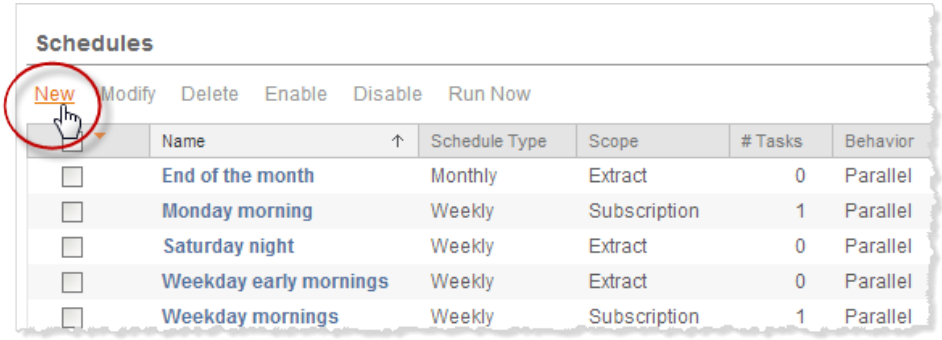
To enable scheduling, select the **Scheduling** checkbox under **Settings** on the Server Maintenance page:

> Because database passwords may be required to refresh the extract, you must enable **Embedded Credentials** in order to allow scheduling.

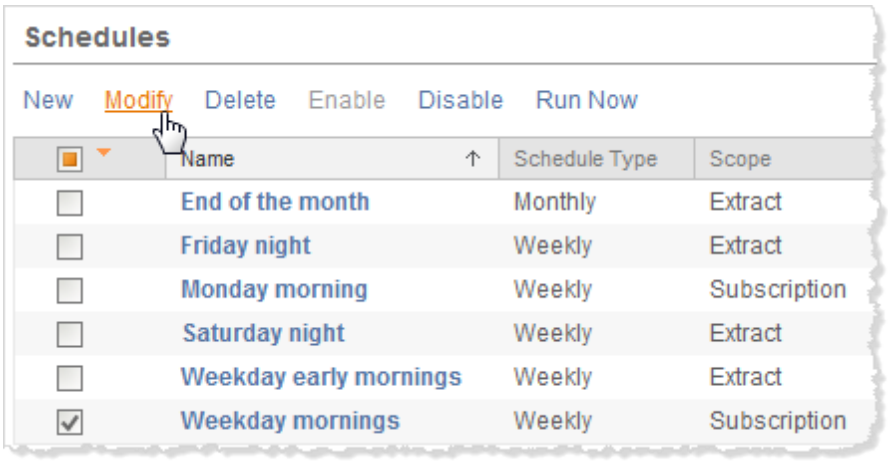
Create or Modify a Schedule

The Schedules page shows a list of schedules, including their name, type, what they're for (scope), number of tasks, behavior (concurrent or serial processing), and when they are scheduled to run.

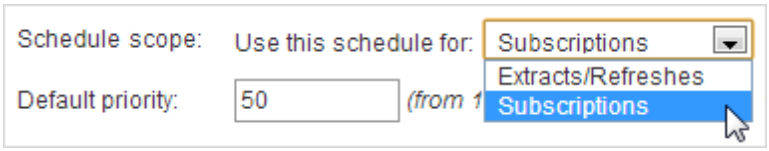
1. To create a new schedule, click **New**:



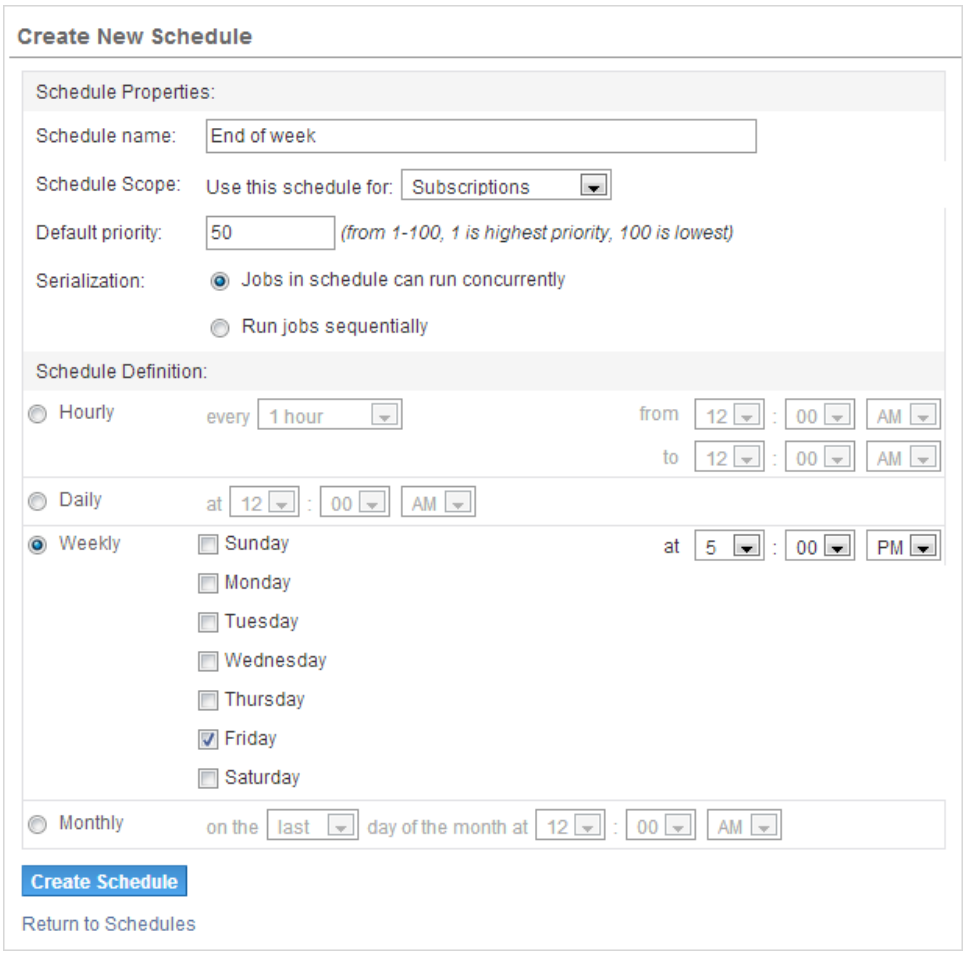
2. To modify an existing schedule, select it then click **Modify**:



3. Specify a descriptive **Name** for the schedule (e.g., Every Saturday Morning, End of the Month).
4. Choose a **Schedule scope**, which is what the schedule will handle—either extract refreshes or delivering subscriptions.



5. Optionally define a **Default Priority** from 0 to 100. This is the priority that will be assigned to the tasks by default. If two tasks are pending in the queue, the one with the highest priority runs first. See Manage Refresh Tasks to learn more about modifying a task's priority.
6. Choose whether the jobs in the schedule will run at the same time (concurrently, the default) or one after the other (sequentially).
7. Finish defining or editing the schedule. You can define an hourly, daily, weekly, or monthly schedule.

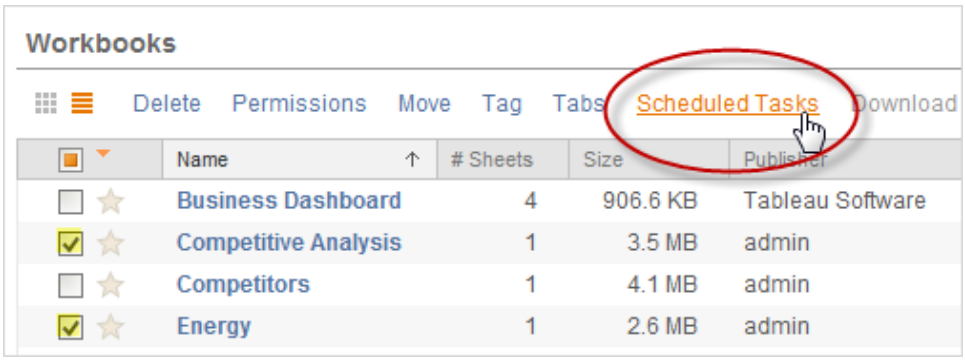


8. Click **Create Schedule** if it's a new schedule or **Save Schedule** if you modified an existing schedule.

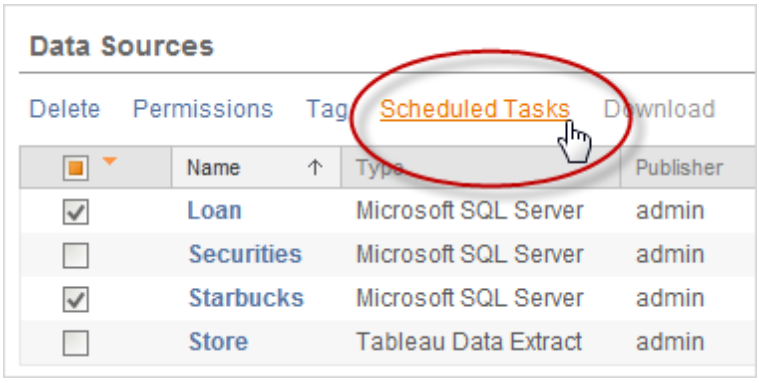
Add a Data Source or Workbook to a Schedule

Once you've Enable Scheduling you can add a workbook to a schedule from the Workbooks list, or you can add a data source to a schedule from the Data Sources list. By default Tableau Server has three built in schedules for refreshing extracts. You can also create your own. See Create or Modify a Schedule for details.

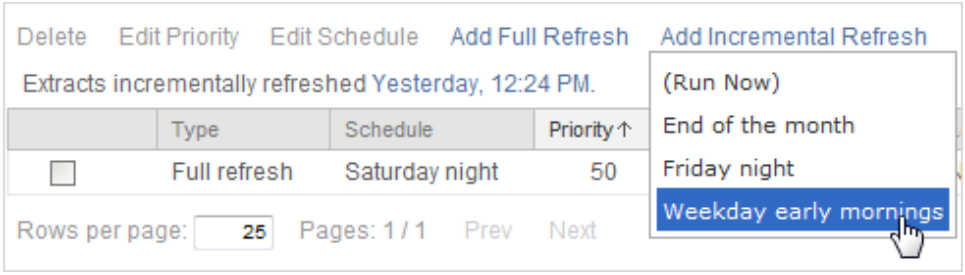
1. If you are scheduling a workbook for an extract refresh, select one or more on the Workbooks page and click **Scheduled Tasks**:



If you are scheduling a data source for an extract refresh, select one or more on the Data Sources page and click **Scheduled Tasks**:



2. Select either **Add Full Refresh** or **Add Incremental Refresh**, then select a schedule from the list:

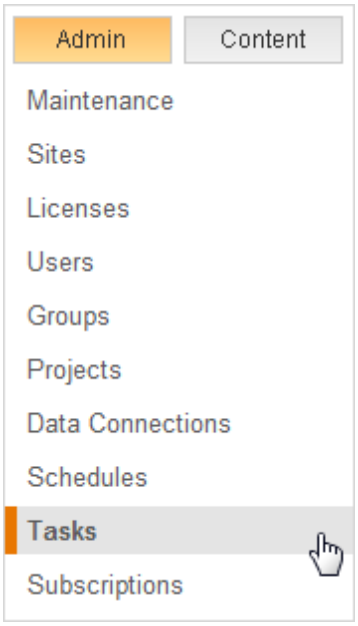


Add Full Refresh is only available if the selected data source connects to an extract. **Add Incremental Refresh** is only available if the selected data source connects to an extract data source for which you've defined an incremental refresh. Tableau Server cannot refresh data sources that connect to local file data sources on a mapped drive. Update the connection to use the full path to the data source.

Manage Refresh Tasks

The Tasks page displays all the full and incremental extract refresh tasks being handled by the server. System and site administrators can use this page to change a task's priority, move it to

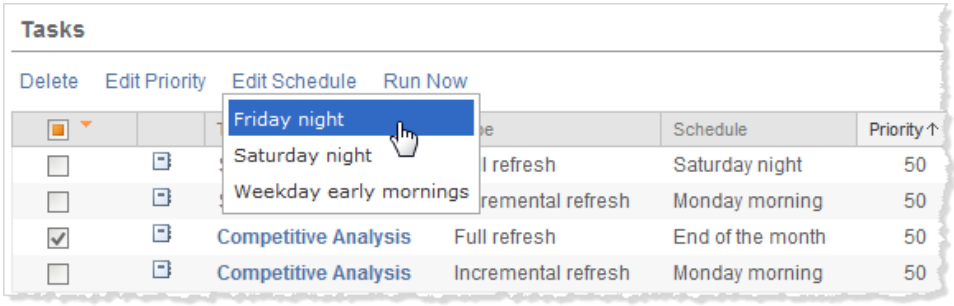
different schedule, run it, or delete it. You can open the Tasks page by clicking **Tasks** on the Admin tab:



Edit a Task's Schedule

Move an extract refresh task from one schedule to another by doing the following:

1. On the Tasks page select one or more tasks to modify.
2. Click **Edit Schedule**. Select a new schedule from the list of schedules:



You can also delete and run tasks by selecting one or more tasks in the list and selecting an option on the toolbar.

Run a Task Now

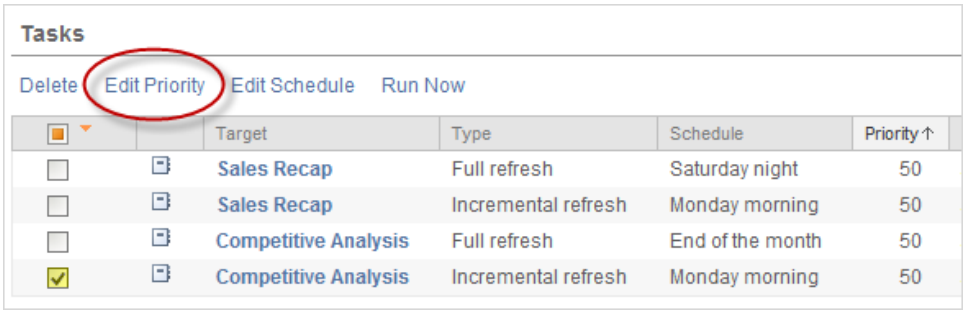
You can force an immediate refresh of a task, such as an extract refresh task, using the **Run Now** option.

1. On the Tasks page, select a task to run.
2. Click **Run Now**.

Change a Task's Priority

To change the priority of an extract refresh task:

1. On the Tasks page select one or more tasks to modify.
2. Click **Edit Priority**.



3. Type a new priority from 0 to 100 and click **Submit**.

Automate Refresh Tasks

You can associate extract refresh tasks with schedules in Tableau Server to automate refreshing data extracts. You can also automate extract refreshes using tabcmd, a command line utility that ships with Tableau Server and can be installed on a separate machine from Tableau Server. In particular, you can use the `refreshextracts` command in combination with other commands in your own script. For example:

```
    tabcmd login - http://mytabserver -u jsmith -p P@ssw0rd!
refreshextracts --datasource salesq4
```

Manage Subscriptions

A subscription is a view or workbook on Tableau Server that users can receive a snapshot of via email. When they click the snapshot in their email, the view or workbook opens on Tableau Server. See the topics below for more information.

Requirements

For Tableau Server users to receive subscriptions, the following things need to be in place:

- **Email settings configuration:** As the system administrator, you configure the basic SMTP server settings for subscriptions on the **Email Alerts/Subscriptions** tab in the Configuration dialog box, which displays during Setup. This is the "from account" Tableau Server uses to email subscriptions to server users. You can access this tab after

Setup as well. See Reconfigure the Server and Configure Email Subscriptions for steps.

- **Credentials embedded or not required**: From Tableau Server's perspective, a subscription includes a workbook, data, and a schedule. To deliver the data piece, Tableau Server needs to be able to access the data with no end-user involvement. This can be accomplished by using either a workbook with embedded database credentials, a Tableau Server data source, or by using data that doesn't require credentials, such as a file that's included with the workbook at publish time. Workbooks that prompt for credentials for live database connections can't be subscribed to.
- **User requirements**: If a user can see a view or workbook onTableau Server and it has the subscription icon (✉) in the upper right corner, he or she can subscribe to it. The ability to see a view or workbook is controlled by the **View** permission. A user must also have an email address. If Tableau Server doesn't already have an email address for a subscribing user, it prompts for one at subscription sign-up time. Users can change their delivery options, unsubscribe, or update their email address on their User Preferences page.
- **No trusted authentication**: If Tableau Server is configured for trusted authentication, subscriptions are disabled. Trusted authentication, in combination with Tableau's Local Authentication, creates a "login free" yet authenticated experience for end-users. To create this same experience and use subscriptions, use Active Directory (with Enable Automatic Login) as the user authentication type instead. You choose the user authentication type during Setup. See Configure the Server for details.

Additional Subscription Settings

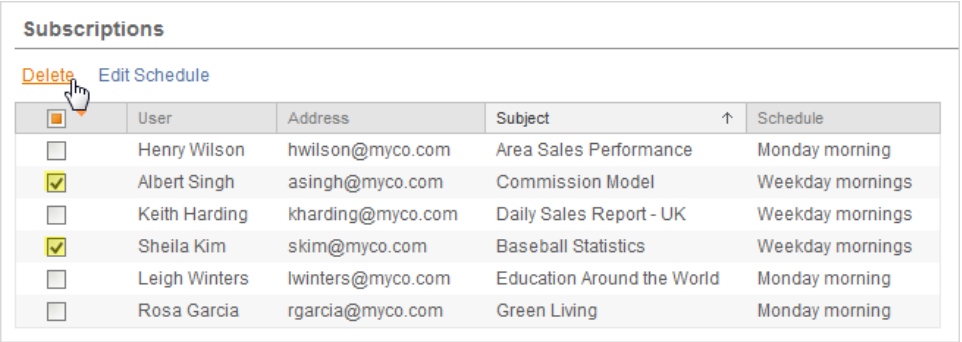
As long as subscriptions are configured on the **Email Alerts/Subscriptions** tab and Tableau Server is using its default settings, server users can subscribe to the views and workbooks they see. To prevent users from subscribing or to customize their subscription experience, here's where to go:

- **Sites page**: By default, subscriptions are enabled for every site, but you can use the Sites page to disable subscriptions on a per-site basis or to customize it. For example you can enter a custom **From address** for subscriptions instead of the one you specified in the Configuration dialog box. You can also create your own footer for the subscription emails your users receive.
- **Schedules page**: Your users will need at least one subscription schedule to choose when they subscribe. Tableau provides two by default. As the system administrator, you can create additional schedules or remove the default ones. See Create or Modify a Schedule for details.
- **Subscriptions page**: This page lists all the subscriptions on the server or, if you're a site administrator, on the site. System administrators can use this page to change a server user's subscription schedule or delete their subscription. See the topics below for details.

For steps on how to test whether you've configured subscriptions correctly, see Test Your Subscription Configuration. If you're experiencing an issue with subscriptions, see Troubleshoot Subscriptions.

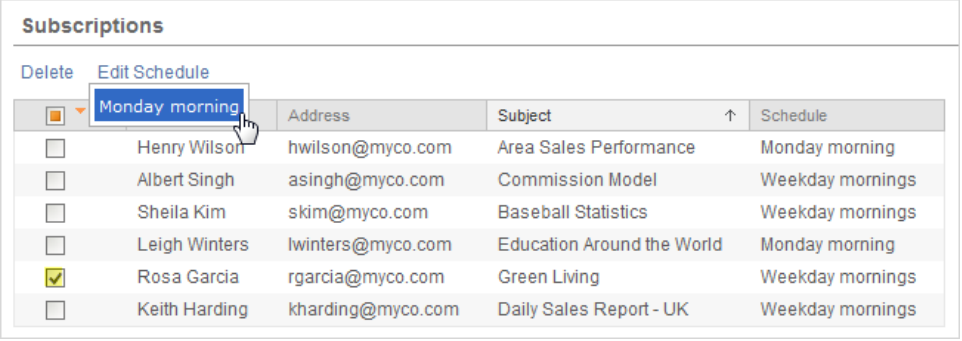
Delete a Subscription

To delete a subscription, select the subscription you want to remove and click **Delete**:



Edit a Subscription Schedule

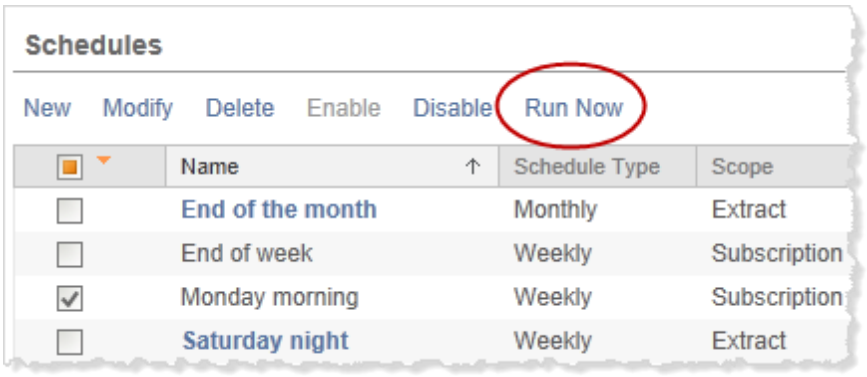
To change the schedule for a subscription, select the subscription, click Edit Schedule and select a schedule:



Test Your Subscription Configuration

As the administrator, you can test whether you've correctly configured subscriptions by doing the following:

1. Subscribe to a view.
2. On the Schedules page, select the schedule that contains your subscription.
3. Click **Run Now**:



4. In a few moments, your subscription should appear in your email inbox.

Troubleshoot Subscriptions

"The view snapshot in this email could not be properly rendered."

If you receive a subscription with this error message, there could be several reasons:

- **Missing credentials**: Some views are published with embedded credentials.You may receive the above error if the embedded credentials are now out-of-date, or if the view was republished without the embedded credentials.
- **Database temporarily down**: If the view has a live database connection and the database was temporarily down when the subscription was being generated, you might receive the above error.
- **Background process timeout**: By default, the background process that handles subscriptions times out after 30 minutes. In the majority of cases, this is plenty of time. However, if the background process is handling an extraordinarily large and complex dashboard, that may not be enough time. You can check the Background Tasks admin view to see if that's the case. To increase the timeout threshold, use the tabadmin option subscriptions.timeout.

Can't Subscribe

If you can see a view on Tableau Server and it has a subscription icon (✉) in the upper right corner, you can subscribe to it.

Two things need to be in place for you to subscribe to a view: Tableau Server needs to be correctly configured (described in Manage Subscriptions) and the view you're subscribing to must either have embedded credentials for its data source or not rely on credentials at all. Examples of the latter include a workbook that connects to an extract that isn't being refreshed, or a workbook whose data is in a file that was included with the workbook at publish time. Embedding credentials is a step that happens in Tableau Desktop (see the Tableau Desktop help for details).

No Subscription Icon

It's possible to see a view on Tableau Server but be unable to subscribe to it. This happens for views with live database connections, where you're prompted for your database credentials when you first click the view. A subscription includes a view (or workbook), data, and a schedule. To deliver the data piece, Tableau Server either needs embedded database credentials or

data that doesn't require credentials. Where live database connections are concerned, Tableau Server doesn't have the credentials, only the individual users do. This is why you can only subscribe to views that either don't require credentials or have them embedded.

You may also be able to see a view but be unable to subscribe to it (no subscription icon) if Tableau Server is configured for trusted authentication. See Subscription Requirements for more information.

Receiving Invalid or "Broken" Subscriptions

If you configured subscriptions on test or development instances of Tableau Server in addition to your in-production instance, disable subscriptions on your non-production instances. Keeping subscriptions enabled on all instances can result in your users receiving subscriptions that appear to be valid, but which don't work, or receiving subscriptions even though they've unsubscribed from the view or workbook.

Subscriptions not Arriving ("Error sending email. Can't send command to SMTP host.")

You may see the above error in Windows Event Viewer if subscriptions appear to be sent (according to the Background Tasks admin view), yet subscriptions aren't arriving, and your SMTP server is using encrypted (SSL) sessions. Subscriptions are only supported for unencrypted SMTP connections. The solution is to use an unencrypted SMTP server.

Sites

Use the Sites page to create independent sites for different organizations or groups on a single server system. Each site's workbooks, data, and user lists are isolated from those of other sites. As the system administrator, only you can see every site and perform actions such as creating sites and making system-wide changes. See the topics below for more information:

Work with Sites

The topics below describe aspects of working with multiple sites such as which type of authentication is used, as well as things you should know about user licenses, and administrator roles.

Authentication and Login Credentials

All sites on a server use the same Server Run As account and user authentication mode. You choose both of these settings when you install Tableau Server. See General for more information.

Users who belong to more than one site on the same server system use the same credentials for each site. For example, if Jane Smith has a username of jsmith and a password of "MyPassword" on Site A, she uses those same credentials on Site B. When she logs into Tableau Server, she'll be able to choose which site she wants to access.

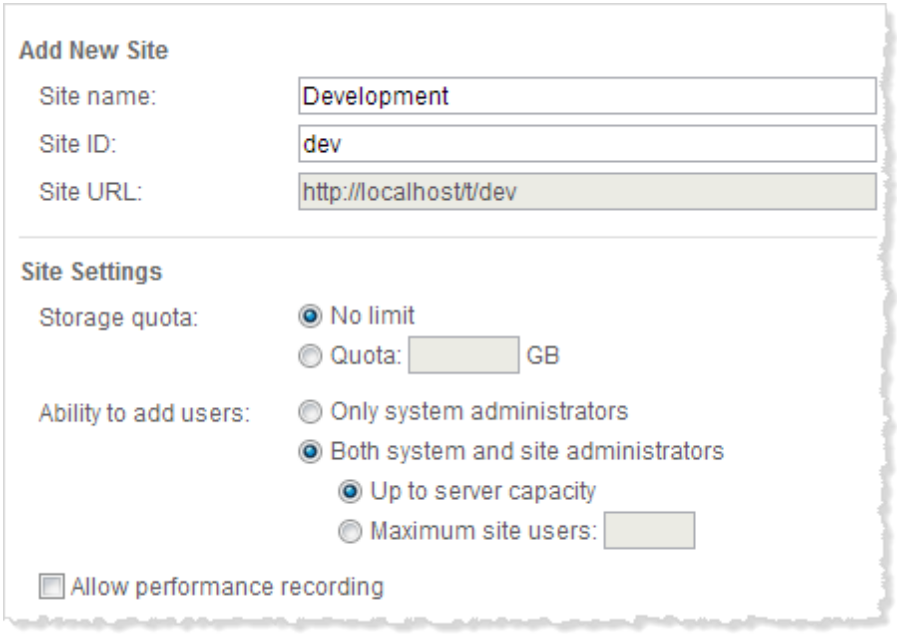
The Default Site

To help you transition smoothly from a single- to multi-site server system, Tableau Server installs with a site named Default. If you're running in single-site mode, you don't need to explicitly use Default, it happens automatically. However, if you add one or more sites, Default becomes one of the sites you can log into when you log into Tableau Server. Default differs from sites that you add to the system in the following ways:

- It can never be deleted but, just like sites that you add, it can be renamed.
- It stores the samples and data connections that ship with Tableau Server.
- The URL used for Default has no corresponding web folder named “default”. For example, the URL for a view named Profits on a site named Sales is `http://localhost/t/sales/views/profits`. The URL for this same view on the Default site would be `http://localhost/views/profits`.

The Site and System Administrator Roles

There are two types of administrators in Tableau Server, system administrators and site administrators. System administrators can control whether site administrators can add and remove users in the Add New Site (or Edit Site) dialog box:



If **Only system administrators** is selected, site administrators cannot add or remove site users. However, they can still manage groups, projects, workbooks, and data connections within their site. If **Both system and site administrators** is selected (the default), site administrators can do all of the above, and add or remove users.

Licensing and User Limits

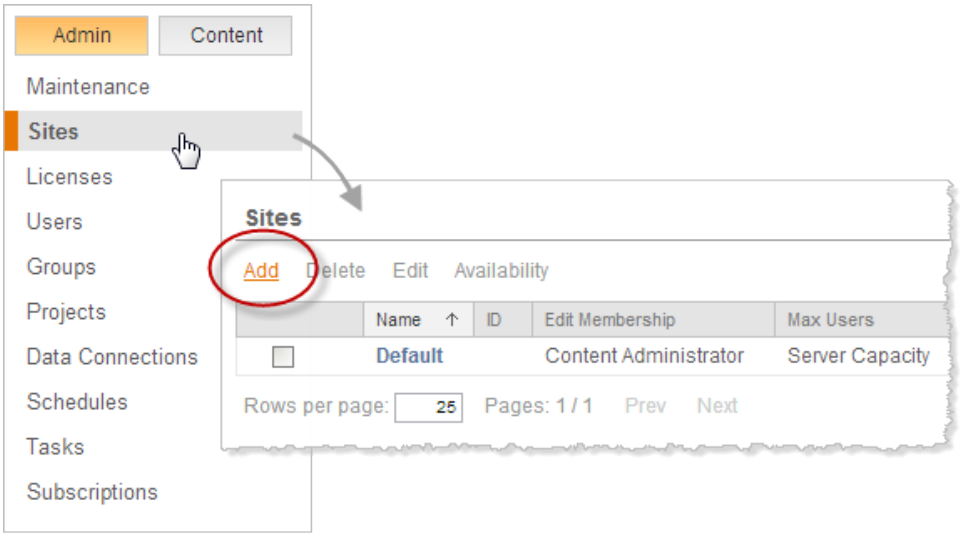
Users can belong to multiple sites, with different user rights and license levels on each site. A user who belongs to several sites, however, does not need a license for each site. Each server user only needs one license.

System administrators can use the **Maximum site users <n>** setting to specify a user limit for a site. Only licensed users are counted; system administrators are excluded. For example, if a site has 90 licensed users, 20 unlicensed users, and one system administrator, the user count is 90. If **Maximum site users** is set to **100**, 10 more licensed users can be added.

Add or Edit Sites

If you are a system administrator, you can add a site to Tableau Server, or edit an existing one, by doing the following:

1. Open the Sites page by clicking Sites under Admin, then click **Add**:

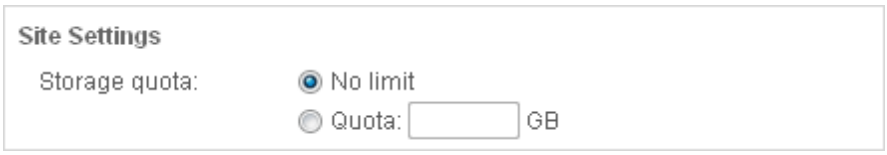


   Or, if you are editing a site, select the site you want to change and click **Edit**. If you have not added sites to Tableau Server, there will be just a single site to select: Default.

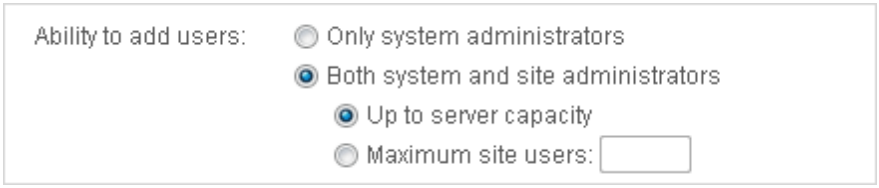
2. Enter a **Site name** and **Site ID** for the site (if you are editing the Default site, you cannot change the **Site ID**):

   The "t" in the URL (for example, http://localhost/**t**/wsales) cannot be changed. In multi-site server systems, it appears in the URL for sites other than the Default site.

3. Workbooks, extracts, and data sources all consume storage space on the server. Select either **No limit** or **Quota**, and enter the number of GB you want as a limit. If you set a quota and the site exceeds it, publishers will be prevented from uploading new content until the site is under the limit again. System administrators can track where the site is relative to its quota using the **Storage Quota** and **% Quota Used** columns on the Sites page.



4. Next, select whether only you, the system administrator, can add and remove users (**Only system administrators**) or if it can be done by both types of administrators (**Both system and site administrators**).



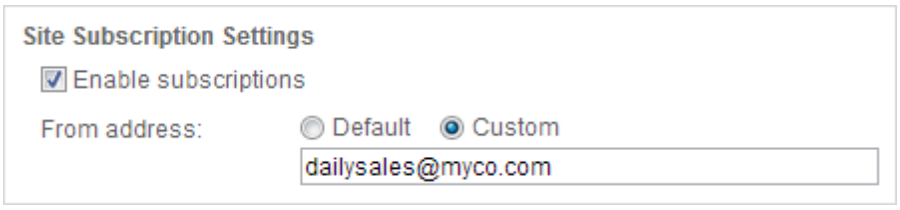
If you are allowing site administrators to add users, specify how many users they can add to the site by selecting one of the following:

- **Up to server capacity**: For a server with user-based licensing, the limit is the number of available server seat licenses. For a server with core-based licensing, there is no limit to the number of users that can be added.
- **Maximum site users <n>**: Allows a site administrator to add users up to a limit you specify. See Working with Sites for information on licensing and user limits.

5. Select **Allow performance recording** to permit your site users to collect metrics on how workbooks perform, such as how quickly they load, etc.

In addition to having this check box selected for the site, to initiate recording, users must add a parameter to the workbook's URL. See Create a Performance Recording for more information.

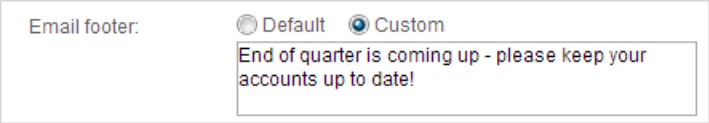
6. Under Site Subscription Settings, keep **Enable subscriptions** selected if you want site users to be able to subscribe to views. This option is only visible if you have also configured subscription settings in the Configuration dialog box.



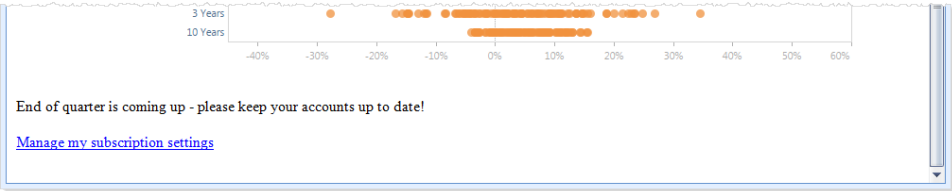
You can also enter a custom **From address** for the subscriptions. While the address you enter should use valid email address syntax (such as `bizdev@bigco.com` or `noreply@sales`),Tableau Server does not require it to correspond to a real email account (some SMTP servers may require it to be an actual address, however).

7. By **Email footer**, select **Custom** and enter any text you want to appear above the

Tableau Server URL in subscription footers. For example, if you enter this text:

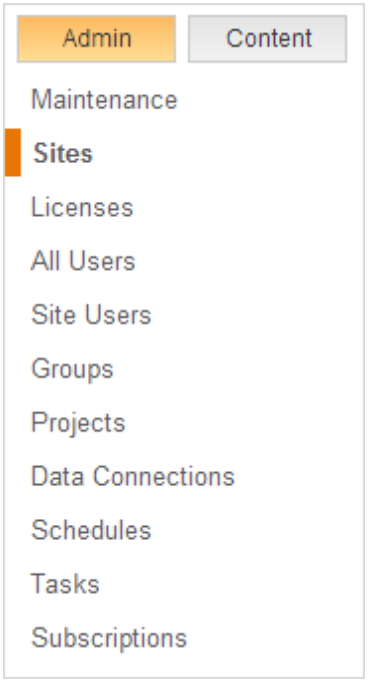


The email footer will look similar to the following:



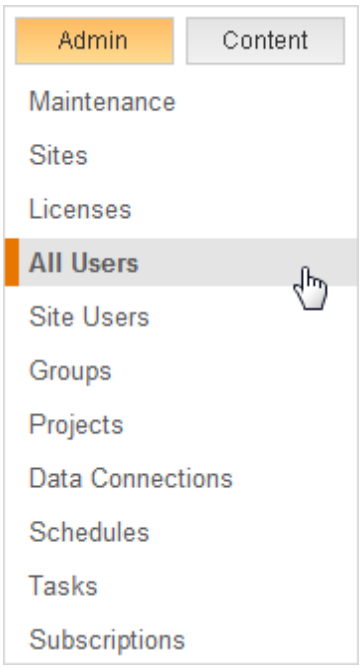
8. Click OK.

If you are adding your first site to Tableau Server, the Admin tab changes. **Users** is now **All Users**, because it pertains to all users on the server, and a **Site Users** category appears.

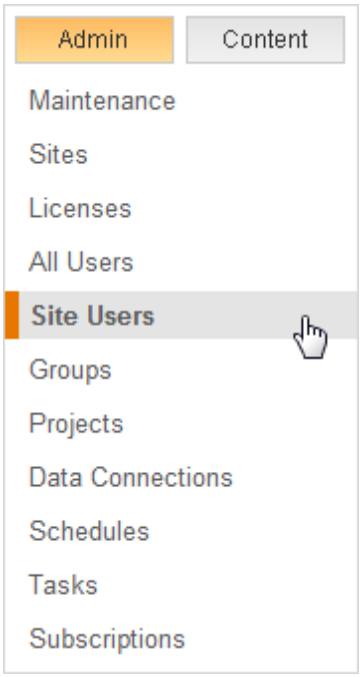


Add Users to a Site

Once you add a site to Tableau Server, it becomes a multi-site system and what was formerly the **Users** page becomes two pages: **All Users** and **Site Users**. As the system administrator, only you can access the **All Users** page, which applies to the entire server system. It's the only place where you can add users to multiple sites all at once, remove users, and if the server is using local authentication, reset user passwords.



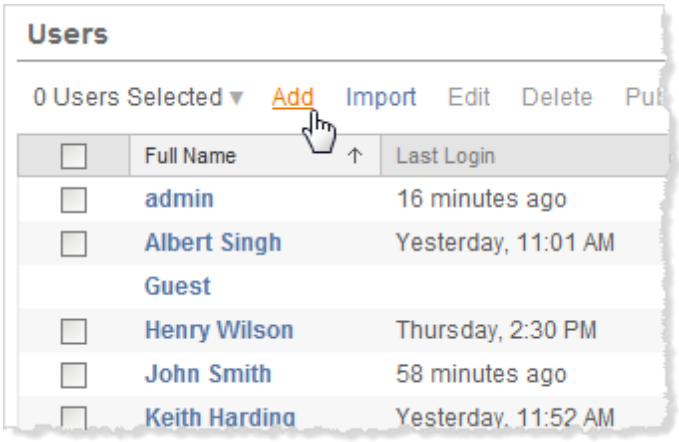
The **Site Users** page is an easy way to quickly see which users are on the site you're currently logged into. You can add users from here, but they will only be added to that site.



The following procedure describes how to add users from **All Users**. There are two approaches you can take: One at a time (described below) or in batches using the **Import** command, which relies on a CSV file (described in Import Users from a CSV File).

To add a user:

1. From the **All Users** page, click the **Add** link at the top of the list of users.



2. Enter a **Username**:
    - **Local authentication**—If the server is using local authentication, using an email address for the username is the best way to avoid username collisions (for example, jsmith@myco.com instead of jsmith).
    - **Active Directory**— If you are adding a user that is from the same Active Directory domain that the server is running on, you can type the **Username** without the domain. The server's domain will be assumed.

      If there is a two-way trust set up between the server's domain and another domain, you can add users from both domains. The first time you add a user from the "non-server domain," use the fully-qualified domain name with the username. Subsequent users can be added using the domain's nickname. For example, assuming a "non-server domain" of *mybiz.lan*, enter the first user from that domain as *user1@mybiz.lan* or *mybiz.lan\user1*. The next user can be entered using the domain's nickname, such as *user2@mybiz* or *mybiz\user2*.

      Note: Be sure not to enter the user's **Full Name** in this field as it can cause errors during the importing process.
3. If the server is using local authentication, provide the following:
    - **Full Name**—Type a display name for the user (e.g., John Smith).
    - **Password**—Type a password for the user.
    - **Confirm**—Retype the password.
4. **Site Membership**—Select which site(s) the user should be a member of. The site you are logged in to is selected by default.
5. **License Level and User Rights**—Select a license level, Admin role, and whether the user can publish workbooks and data sources. A user who belongs to multiple sites can have different license levels and user rights on each site. See About License Levels, Permissions, and About User Rights to learn more.

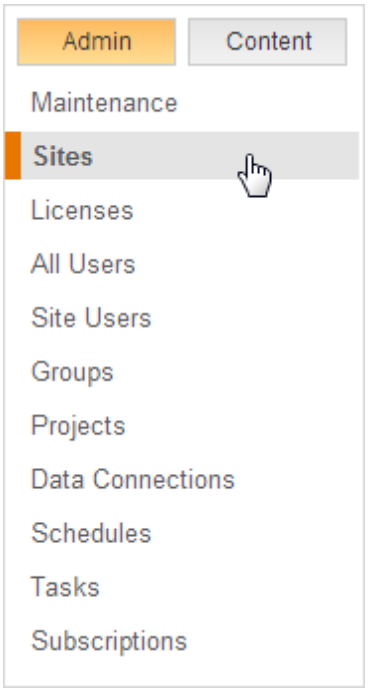
6. Click **Add User**.

Delete Sites

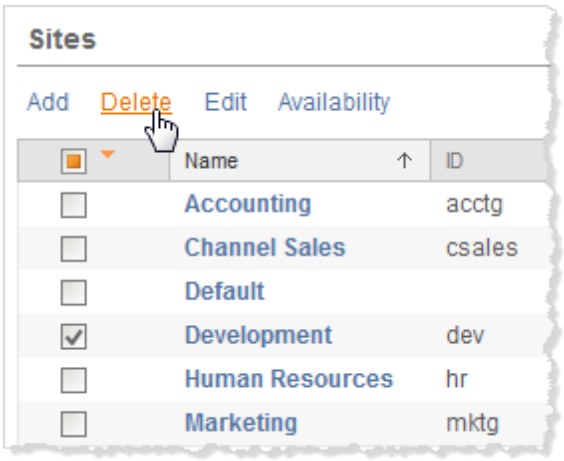
System administrators can delete sites that have been added to Tableau Server. Deleting a site also removes workbooks and data sources that were published to the site, as well as users. If a user belongs to additional sites, they will not be removed. To permanently remove a user, you need to use the All Users page.

To delete a site:

1. Open the Sites page under Server:



2. Select the site you want to remove and click **Delete**:



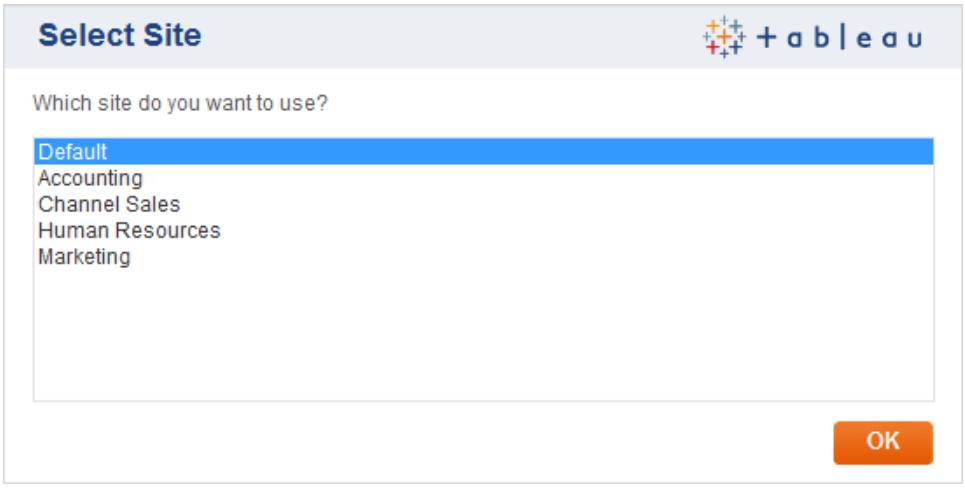
3. Click **Yes** in the confirmation dialog box that appears.

Multi-Site Navigation

Here are some tips on how to navigate from site to site and identify which site you're using.

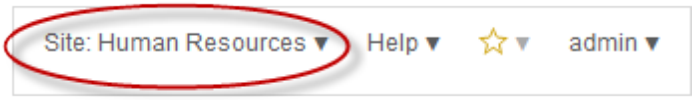
Site Login

If you are a member of multiple sites, at server login you are prompted to choose a site:

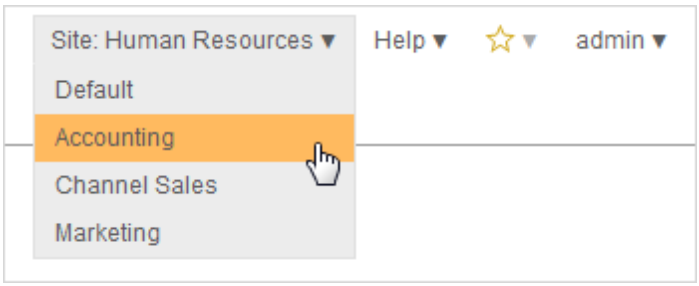


Navigating to Other Sites

If you belong to multiple sites, you'll see a Site menu at the top of the page:

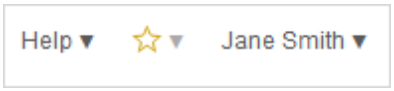


To log in to a different site, click the Site menu and select the site:

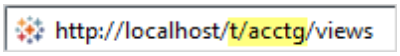


Identifying Your Logged In Site

If the server is running multiple sites yet you only belong to one site, you are not prompted to choose your site at server login. After you log in, you do not see a Site menu at the top of the page:



However, your web browser URL will show a **t** followed by the **site ID** for your site:



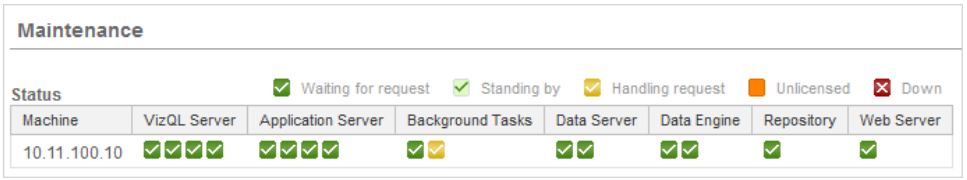
If the server isn't running multiple sites, your web browser URL will look similar to this (no **t**, no **site ID**). If you see this, you are using Tableau's built-in site, which is named Default.

Server Maintenance

As a system administrator, you will want to check the status of the server, analyze and monitor the activity on the server, manage scheduled tasks, or perform certain maintenance activities such as rebuilding the search index. In addition, there are several settings that you may want to specify to customize the user experience for people using the server. You can do all of these tasks from the Maintenance page.

View Server Process Status

You can use the Status table on the Maintenance page to view the state of Tableau processes on each Tableau server:



For information about unlicensed status for a VizQL Server process, see Handle an Unlicensed VizQL Server Process.

To display a machine-readable version of the above information, from the Maintenance page, replace the word status in your URL with systeminfo (for example, http://jsmith/admin/-systeminfo). A web page similar to the following appears:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <systeminfo>
  - <machines>
    - <machine name="Primary">
        <repository worker="Primary:8060" status="OK" />
        <dataengine worker="Primary:27042" status="OK" />
        <dataengine worker="Primary:27043" status="OK" />
        <serverwebapplication worker="Primary:8001" status="OK" />
        <serverwebapplication worker="Primary:8002" status="OK" />
        <serverwebapplication worker="Primary:8003" status="OK" />
        <serverwebapplication worker="Primary:8004" status="OK" />
        <vizqlserver worker="Primary:9100" status="OK" />
        <vizqlserver worker="Primary:9101" status="OK" />
        <vizqlserver worker="Primary:9102" status="OK" />
        <vizqlserver worker="Primary:9103" status="OK" />
        <dataserver worker="Primary:9700" status="OK" />
        <dataserver worker="Primary:9701" status="OK" />
        <backgroundtasks worker="Primary:0" status="OK" />
        <backgroundtasks worker="Primary:1" status="Busy" />
        <webserver worker="Primary:80" status="OK" />
      </machine>
    </machines>
    <service status="OK" />
  </systeminfo>
```

The three types of status for a Tableau service are OK, Busy, and Down.

Access Status Remotely

As the Tableau administrator, only you can see the tools on the Maintenance page, including the Status table. You can, however, make the machine-readable version of the Status table available to non-admin users and to computers other than the one that's hosting Tableau Server—for example, as part of a remote monitoring process. To grant remote access to Tableau Service status:

1. On the computer running the primary Tableau Server, open the Tableau Server config file:

   `ProgramData\Tableau\Tableau Server\config\tabsvc.yml`

2. Add the line `wgserver.systeminfo.allow_referrer_ips: <IP address>` to tabsvc. yml, where `<IP address>` is the IP address of the computer you'd like to add. If you are granting service status access to multiple computers, use commas (no spaces) to separate each IP address. For example:

```
wgserver.systeminfo.allow_referrer_ips: 123.45.67.89,123.45.67.88
vizqlserver.extract.connection.class: dataengine
worker0.vizqlserver.procs: 4
service.runas.username: MYCO\jsmith
vizqlserver.extract.type: internal
config.version: 4
wgserver.authenticate: activedirectory
worker0.wgserver.procs: 4
wgserver.sspi.ntlm: true
service.init.state: start
```

3. Save and close tabsvc.yml.
4. Open a command prompt as an administrator and type:

   32-bit:`cd "C:\Program Files\Tableau\Tableau Server\8.0\bin"`

   64-bit:`cd "C:\Program Files (x86)\Tableau\Tableau Server\8.0\bin"`
5. Then use the following command to restart the Tableau Server processes:

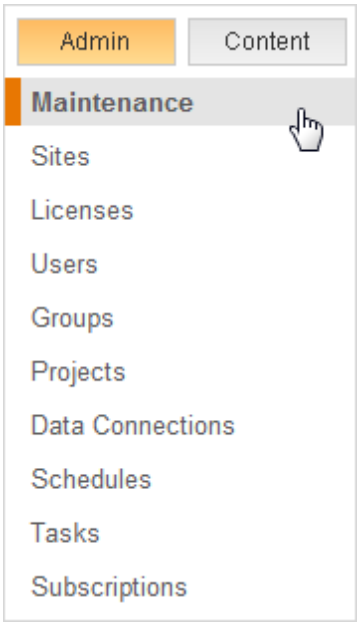
   ```
   tabadmin restart
   ```

   Now, users at computers whose IP addresses are added to tabsvc.yml can view Tableau process status by entering the URL `http://<server>/admin/systeminfo` in a browser or from a command line (for example, `curl http://jsmith/admin/systeminfo`). This functionality can also be used as part of an automated remote monitoring process.

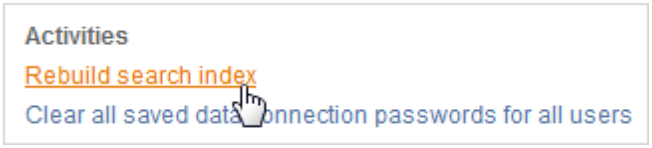
Rebuild the Search Index

If for any reason the search index stops returning the correct results or is missing results, you may need to rebuild the search index. Additionally, you should rebuild the search index if the indexer goes down for an extended period of time.

1. To rebuild the search index, on the Admin tab, click Maintenance:



2. Click **Rebuild search index** to begin.

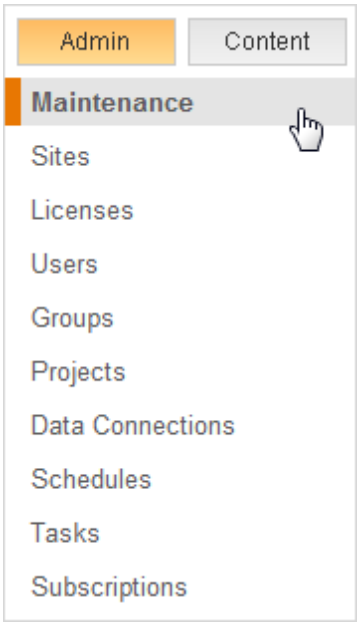


Clear Saved Data Connection Passwords

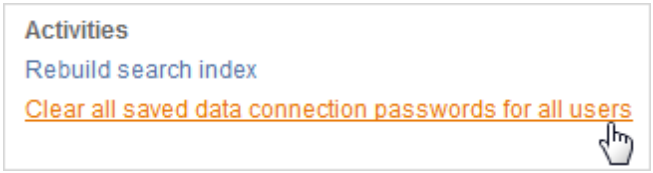
As the administrator, if you enable the Saved Passwords setting, server users can save data source passwords across multiple visits and browsers. You can reset all of the passwords for all Tableau Server users, which forces them to log in to the data sources the next time they visit a view that requires database authentication. Server users can also clear their saved data connection passwords on an individual basis using their User Preferences page.

To clear saved data connection passwords for all server users:

1. Click the Maintenance link in the Administration section on the left side of the page:



2. Under Activities, click **Clear all saved data connection passwords for all users**.



Maintenance Settings

The following settings are available in the Settings section of the Maintenance page on the server:

Setting	Description
Embedded Credentials	Allows publishers to attach passwords to published workbooks that will automatically authenticate web users to connect to data sources. The passwords are attached to workbooks and are only accessible on server. That is, when the workbook is opened in Tableau Desktop, users will still need to enter a user name and password to connect to the data source. When this setting is turned off, all existing embedded passwords are saved but are not used for authentication. That way if you turn the setting back on, users don't have to re-embed the passwords.
Scheduling	Allows publishers to assign workbooks to schedules. This option is only available if **Embedded Credentials** is enabled. When this setting is enabled, publishers will see scheduling options in the Publish dialog box.

Public User List	Allows web users to see a list of all users on the system. When this setting is enabled, a link to a list of all users is added to the left navigation bar. This is useful if your user list is not private and you want to let web users browse by user. When you browse by user, you can see all workbooks, tags, and comments associated with a selected user.
Saved Passwords	Allows users to save data source passwords across multiple visits and browsers. By default users can choose to "Remember my password until I log out," which lets them save their password during a single browser session. When the **Saved Passwords** setting is selected a user can instead choose to Remember my password, which saves the password across multiple visits and browsers so users will be automatically authenticated regardless of the computer they are using. You, as an administrator, can clear all saved passwords at any time. In addition, users can clear their own saved passwords.
Enable Guest	Allows users to view and interact with embedded views without having to log into a Tableau Server account. Permission can be assigned to the Guest User account to control the interactivity allowed for each view. This option is only available if you have a core-based server license. Also, it cannot be used with **Enable Automatic Login**, an option you can select during Setup (learn why). If **Enable Automatic Login** is selected, **Enable Guest** is grayed out.
Default start page	Takes you to the server's current default start page for all users. See Set the Default Start Page for All Users for steps on how to change it. Individual users will be able to override this setting (see Your User Preferences Page for details).
Default language and locale	Controls the language used for the server user interface and the locale used for views. Individual users can override this setting on their User Preferences page. Also, web browser settings are the first thing that's used to determine which language and locale are used. See Language and Locale for more information.
Reset all settings to their default values	Any server settings that have been changed since Setup are returned to their original state.

Set the Default Start Page for All Users

By default, Tableau Server installs with the Views page as the default start page for all users. As the administrator, you can change this to another page that all users have access to, such as the Workbooks page. Individual users will be able to override your setting (see Your User Preferences Page for details).

To set the default start page for all users:

1. Navigate to the page you want to be the default page.
2. Click your name on the upper right corner of the page.

3. Select **Make this the start page for all users**.

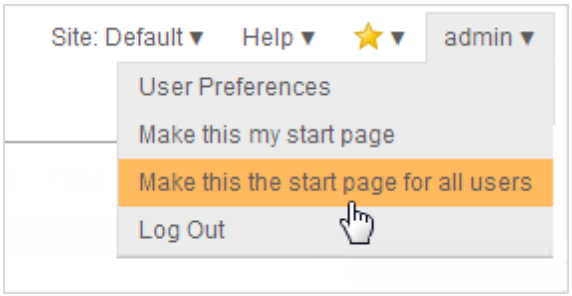
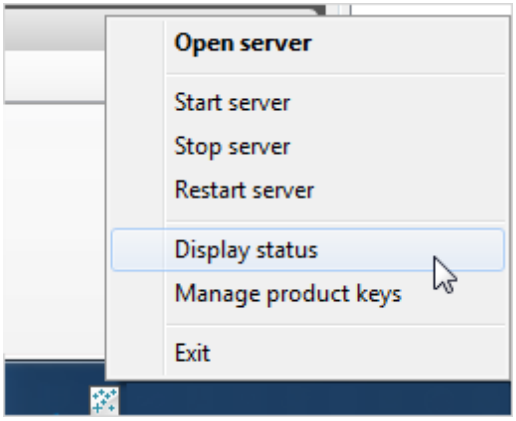


Tableau Server Monitor

Tableau Server Monitor is installed as part of Tableau Server and can be accessed in the Windows system tray.



Using this tool you can start and stop the server, open Tableau Server, and display server status.

Open the Server

This command launches Tableau Server in your web browser. This is an easy way to access the web application and the associated maintenance tools.

Start/Stop the Server

You can start and stop the server using these commands. When you stop the server you make it unavailable to all of your users and terminate any sessions that are currently in progress. If someone is publishing a workbook when the server is stopped, the process is abandoned. As a result, only some of the worksheets in the workbook may be published to the server. Because stopping the server can be very disruptive to your users, be sure to warn them prior to this operation or plan maintenance during non-business hours.

Restart the Server

This command restarts the server. While the server is restarting it will be unavailable to all users. Be sure to warn your users of the outage prior to this operation. You will need to restart the server if you make changes to the Tableau Server configuration.

Display Status

This command opens a screen tip containing the status of each process. For more detailed status, use the Maintenance page.

Manage Product Keys

This command opens the product key manager where you can add and remove product keys.

Exit

This command closes Tableau Server Monitor. It does not stop Tableau Server. You can re-open the application by selecting **All Programs > Tableau Server 8.0 > Tableau Server Monitor** on the Windows Start menu.

Data Sources

A Tableau Server data source is a reusable connection to data. It can include a data extract or information for a pass-through connection to a live, relational database. It can also include a layer of customizations, such as calculations, groups, or sets. Server users with the appropriate permissions can use a data source to create a workbook from scratch, on the server. See Create a Workbook for details. Administrators will perform two main tasks on the Data Sources page:

- **Edit and view data source permissions:** These include which users or groups can connect to data sources, modify them, and download them. See Setting permissions for a data source for more information.
- **Schedule data source extracts for refresh:** If a data source includes an extract, you can assign the extract to a refresh schedule. See Scheduling Tasks for more information.

Although both of the above tasks can be performed in Tableau Desktop by the person who published the data source, administrators can change the settings as well. You can also use the Data Sources page to remove a data source or add tags to it. See the topics below for more information.

Manage Data Sources

For users to work with Tableau Server data sources, they need to have the appropriate permissions for the data source. For data sources that are proxy connections, you should also be aware of how users will be authenticating to the database, and whether you have the appropriate drivers installed on Tableau Server. For more information, see the topics below.

Set Permissions for a Data Source

Database Drivers

Data Security

About Tableau Data Server

Tableau's data server is a server component that lets you centrally manage and store Tableau Server data sources. A data source is a reusable connection to data. The data can be located either in Tableau's data engine, as an extract, or in a live, relational database (cubes are not supported). In the latter case, the information stored in the data source is for a pass-through connection. The data source can also include customizations you've made at the field-level in Tableau Desktop, such as calculations, dimension aliases, groups, or sets.

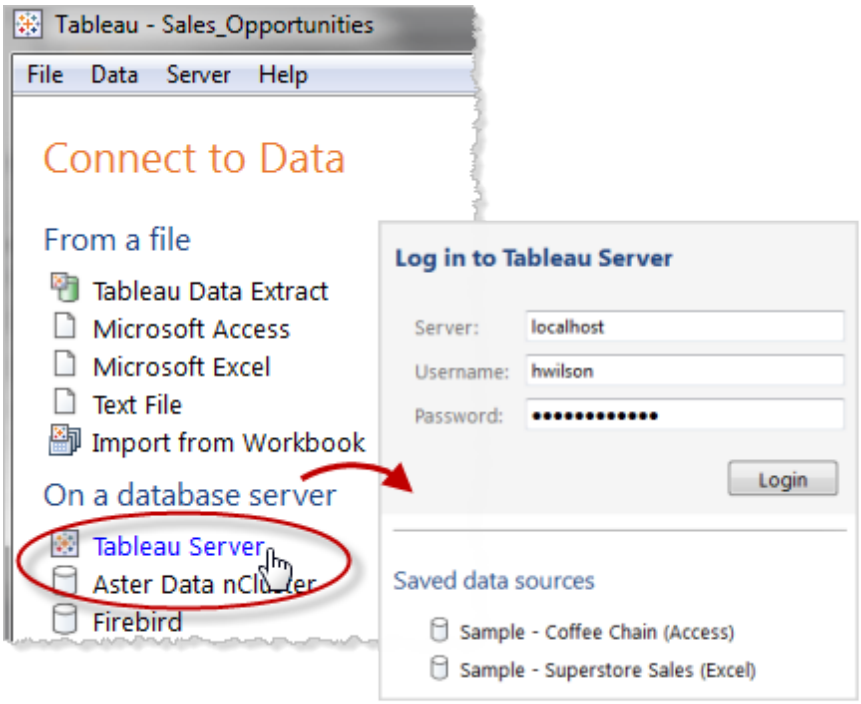
For administrators, there are many advantages to using Tableau Server data sources. Because one data source extract can be used by many workbooks, you save on server space and processing time. Extract refreshes can be scheduled per-extract instead of per-workbook, and when a workbook using a Tableau Server data source is downloaded, the data extract stays on the server, resulting in less network traffic. Finally, if a database driver is required for a connection, you only have to install the driver once, on Tableau Server, instead of multiple times, on all your users' desktops.

To use the data server, all authors have to do is connect to data in Tableau Desktop, either by creating an extract or a connection to a live relational database, and publish it to Tableau Server. Once published, these reusable data sources and the server contain everything workbook authors need to quickly connect to data and start authoring.

If you are running a distributed installation of Tableau Server and expect data sources to be heavily used, there are several ways you can optimize your server deployment. See Distributed Environments for more information.

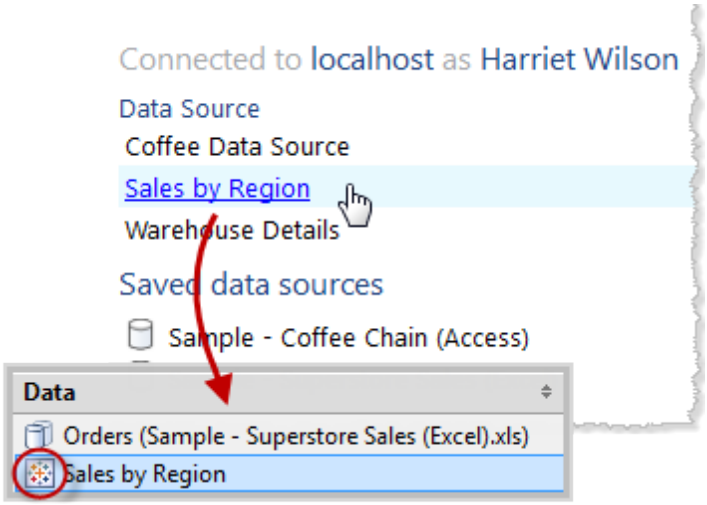
Use Data Sources

If you're a workbook author, using a Tableau Server data source is simply a matter of connecting to it from Tableau Desktop. On the Connect to Data page in Tableau Desktop, click **Tableau Server**, then provide your server credentials:



After you log in to Tableau Server, data sources available to you are listed on the right. To see a data source, the person who published it had to set the Connect permission to **Allow** for you as a user. By default, all Tableau Server users have this permission.

Select a data source and it will load in the Data window in the workbook. Tableau Server data sources have a Tableau icon instead of a database icon:



For more information about creating and using data sources, see the Tableau Desktop online help.

Troubleshoot Data Sources

For users to work with Tableau Server data sources, up to three things need to be in place:

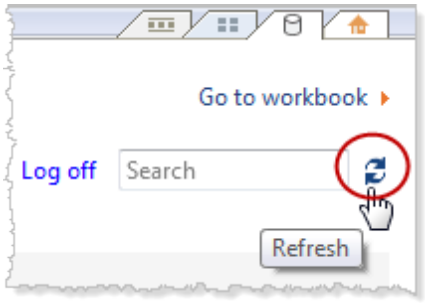
- **Permissions for the data source:** Anyone connecting to a data source must have the **Connect** and **View** permissions for it. This also applies to users accessing views that connect to data sources. Anyone publishing and modifying data sources must be licensed to Publish and also have the **Write/Save As** and **Download/Web Save As** permissions. See Work with Permissions and Set Permissions for a Data Source for more information.
- **Ability to authenticate to the database:** There are several ways you can connect to data in Tableau and control who has access to what. Basically, whichever entity is connecting to the database must be able to authenticate. The entity could be Tableau Server performing an extract refresh. It could be a Tableau Desktop user connecting to a data source that then connects to a live database. It could also be a Tableau Server user who's accessing a view that connects to a live database. Refer to Data Security to learn more about your options.
- **Database drivers:** If the person who created and published the data source in Tableau Desktop needed to install additional database drivers, you may need to install them on Tableau Server as well. If you are running a distributed installation of Tableau Server where, for example, the data server process is running on a worker server, any required database drivers must be installed there as well as on the primary server. Other processes require drivers as well. See Database Drivers for more information.

Data Source Error Messages

Here are some errors that workbook authors and other users may encounter as they work with data sources and views:

Permission to access this Tableau Server data source denied: Connecting to a data source requires the Connect permission. See Work with Permissions and Set Permissions for a Data Source for more information.

Data source not found: Someone working with a view may see this error if a data source is removed from Tableau Server or if their Connect to Data page needs to be updated. To update the Connect to Data page in Tableau Desktop, click the Refresh icon:



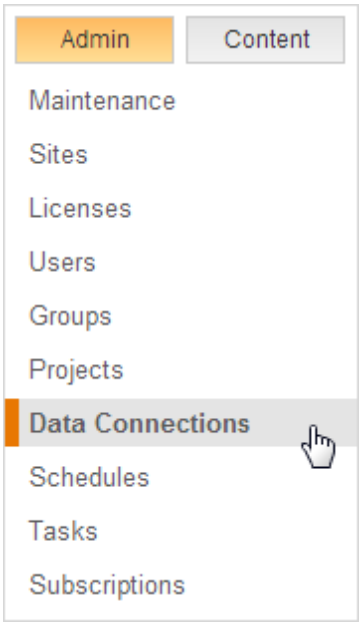
Unable to connect to this Tableau Server data source: This error may appear if the connection information for the data source has changed—for example, as a result of the database server name changing. Look at the Data Connection information for the data source and confirm that it has the correct settings.

Unable to list Tableau Server data sources: This error may occur if a user is trying to access Tableau Server data sources and there are connectivity issues between Tableau Server and Tableau Desktop.

Can't connect with a cube data source: Connections to cube data sources (such as MSAS) are not supported. The data must be either an extract or a live connection to a relational database.

Data Connections

Every workbook that is published to the server contains one or more connections. These connections are listed under the Admin tab, on the Data Connections page:

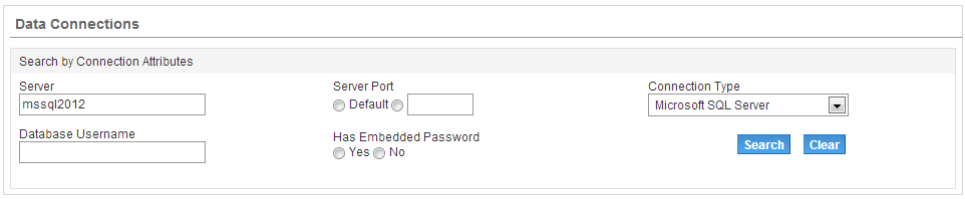


The Difference Between Data Connections and Data Sources

Data connections are different from data sources in that each connection is associated with a single workbook and describes the attributes required for connecting to a data source (e.g., server name, database name, etc.). That means if you have three workbooks that connect to the same data source, you will still have three connections listed on the connections page.

Searching for Data Connections

The Search area at the top of the Data Connections page helps you find connections by database server name, username, port, general connection type, and by whether or not the database credentials are embedded. To use this area to search for a connection, fill in one or more areas and click Search:



Which connections can I edit?

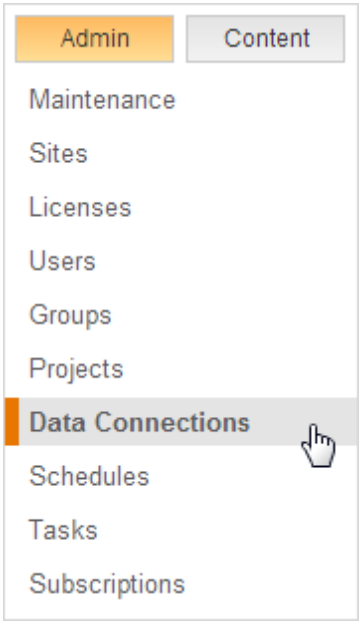
You can edit connection information for live database connections and for extracts that need to be refreshed by Tableau Server. For example, you may have a large number of workbooks that connect to a database on a specific database server. If the name of the server changes, you can update all of the workbooks at once so they reference the new server name. Another example is if a workbook connects to a database using a specific user name and password. You can quickly update all of the workbooks to use a different set of credentials. For steps on how to edit data connections, see the topic below.

Edit Data Connections

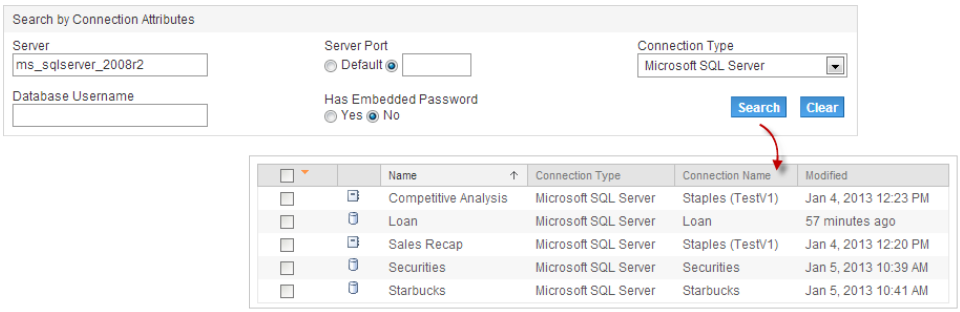
Use the Data Connections page to manage the connection information for all of the workbooks published to the server or a site. To modify connection attributes:

1. If you are running multiple sites on the server, log into the site that has the data connections you want to modify.

   Navigate to the Data Connections page.

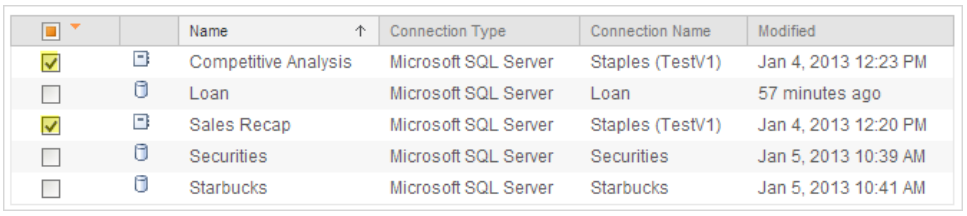


2. Use the search box at the top of the list of connections to find the connections to edit. You can search by **Server**, **Connection Type**, the **Server Port**, **Database User Name**, and whether it **Has Embedded Password**.

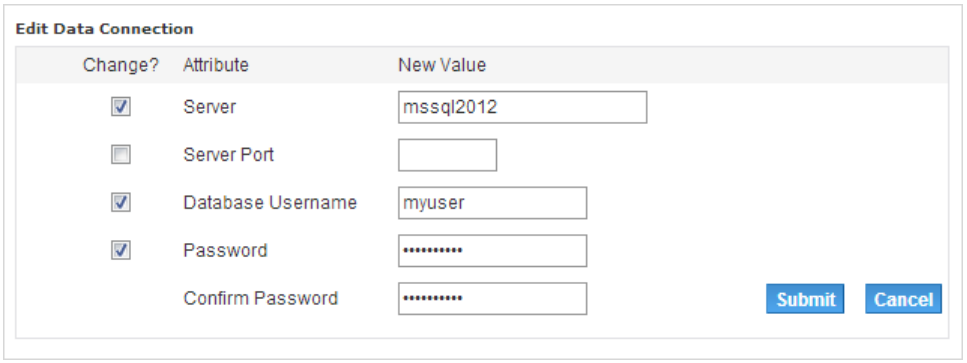


The values you type into the **Server** and **Database User Name** fields are treated as regular expressions.

3. Select the connections to modify in the list of search results:



4. Type in a new value for one or more of the connection attributes. If a database or database driver doesn't support connecting via an IP address, the value entered for **Server** must be the database name. All attributes selected under the **Change?** column will be updated. If you select the **Change?** checkbox and leave the **New Value** field blank, the attribute will be set to blank as well.



5. Click **Submit**.
6. Refresh the server page (press F5) for your changes to take effect.

Monitor Progress

A monitor dialog opens automatically where you can watch the progress of the changes. If you close the monitoring dialog box, the modifications will run in the background until completed. Tableau Server will make as many changes as possible. Any failures will be skipped but will not impede other changes from being made. For example, if you try to change the server name

and add a password to several connections, the server names will be changed, and the passwords on workbooks will be changed, but because you cannot add a password to a data source, the passwords for the data sources will not be changed.

There is an administrative view that allows you review details for completed and pending tasks. See Background Tasks to learn more.

Customize the Server

You can customize how Tableau Server looks to personalize it for your company or group. For example, you can change the name that appears in screen tips and messages, and you can change the logo that appears on most server pages.

You can also customize how users can interact with the server. For example, you can allow workbook publishers to embed their data source credentials so that when people click a published view with a connection to a live data source they get immediate access to the view and don't have to supply their database credentials first.

You can also control which language is used for the server user interface and which locale is used for views.

See the following topics for more information on customizing Tableau Server:

Change the Name or Logo

You can customize the following aspects of Tableau Server's look and feel:

Change the Name

You can customize Tableau Server's look and feel by customizing the name that appears in screen tips and messages. To change the name that appears in screen tips and messages:

1. Open a command prompt as an administrator and type the following:

   32-bit: `cd "C:\Program Files\Tableau\Tableau Server\8.0\bin"`

   64-bit: `cd "C:\Program Files (x86)\Tableau\Tableau Server\8.0\bin"`
2. Change the name by typing the following:

   ```
   tabadmin customize name "new_name"
   ```

   In the above line, replace "new_name" with the text that you want to appear as the name on the server. Example: `tabadmin customize name "Company Server"`
3. Restart the server for the change to take effect by typing:

   ```
   tabadmin restart
   ```

Change the Logo

You can customize Tableau Server's look and feel by customizing the logo that appears on the Tableau Server login page and in the left column of most pages. Two general logo sizes are supported: a large logo size (up to 160 x 160 px), which you implement by running the `tabadmin customize logo` command, and a small logo size (up to 32 x 32 px), which you implement using the `tabadmin customize smalllogo` command. If an image is larger than

these two sizes, it will appear to be clipped. The image file you use should be in GIF, JPEG, or PNG format.

To change the logo:

1. Open a command prompt as an administrator and type the following:

   32-bit: `cd "C:\Program Files\Tableau\Tableau Server\8.0\bin"`

   64-bit: `cd "C:\Program Files (x86)\Tableau\Tableau Server\8.0\bin"`
2. Change the logo by typing the following for a "large size" logo (up to 160 x 160 px, but not smaller than 32 x 32 px):

   ```
   tabadmin customize logo "C:\My Pictures\logo.png"
   ```

   If your logo is 32 x 32 px or smaller, use the following command:

   ```
   tabadmin customize smalllogo "C:\My Pictures\logo.png"
   ```
3. Restart the server for the change to take effect by typing:

   ```
   tabadmin restart
   ```

Restore the Default Name or Logo

You can restore Tableau Server's default look and feel by doing the following:

1. Open a command prompt as an administrator and type the following:

   32-bit: `cd "C:\Program Files\Tableau\Tableau Server\8.0\bin"`

   64-bit: `cd "C:\Program Files (x86)\Tableau\Tableau Server\8.0\bin"`
2. Change the logo by typing the following:

   ```
   tabadmin customize <parameter> -d
   ```

   In the above line, replace `<parameter>` with what you want to restore, either `name` or `logo`.
3. Restart the server for the change to take effect by typing:

   ```
   tabadmin restart
   ```

Language and Locale

Tableau Server is localized into several languages and has language and locale settings that you can configure on a per-user (see Your User Preferences Page) and system-wide basis (see Maintenance Settings). The **Language** setting controls user interface (UI) items such as menus and messages. The **Locale** setting controls items in views such as number formatting and currency.

Default Settings

Tableau Server obtains its default language setting during Setup. If the host computer is set to a language Tableau Server supports, it installs with that language. If it's not a supported language, Tableau Server installs in English.

How Language and Locale are Determined

Another influence on which language and locale display when a user clicks a view is the user's web browser. If a server user has not specified a **Language** setting on their User Account page, and their web browser is set to a language that Tableau Server supports, the browser's language will be used—even if Tableau Server itself is set to a different language.

Here's an example: Assume that Tableau Server has a system-wide setting of English as the **Language** for all users. Server user Claude does not have a language specified on his Tableau Server User Account page. Claude's browser uses German (Germany) for its language/locale.

When Claude logs in to Tableau Server, the server UI displays in German and when he clicks View A, it's using the Germany locale for numbers and currency. If Claude had set his user account **Language** and **Locale** to French (France), the UI and view would have been displayed in French. His user account setting supercedes those of his web browser, and both of those have precedence over Tableau Server's system-wide setting.

Another setting to be aware of is the **Locale** setting in Tableau Desktop (**File** > **Workbook Locale**). This setting determines the locale of the data in the view, such as which currency is listed or how numbers are formatted. By default, **Locale** in Tableau Desktop is set to **Automatic**. However, an author can override that by selecting a specific locale. Using the above example, if the author of View A set **Locale** to **Greek (Greece)**, certain aspects of the data in View A would display using the Greek (Greece) locale.

Here are the settings Tableau uses to determine language and locale, in the following order of precedence:

1. Workbook locale (set in Tableau Desktop)
2. Tableau Server User Account language/locale settings
3. Web browser language/locale
4. Tableau Server Maintenance page language/locale settings
5. Host computer's language/locale settings

Administrative Views

Tableau Server comes with several views for adminstrators, to help monitor activity on Tableau Server. The views are located in the Analysis table on the server's Maintenance page:

Analysis

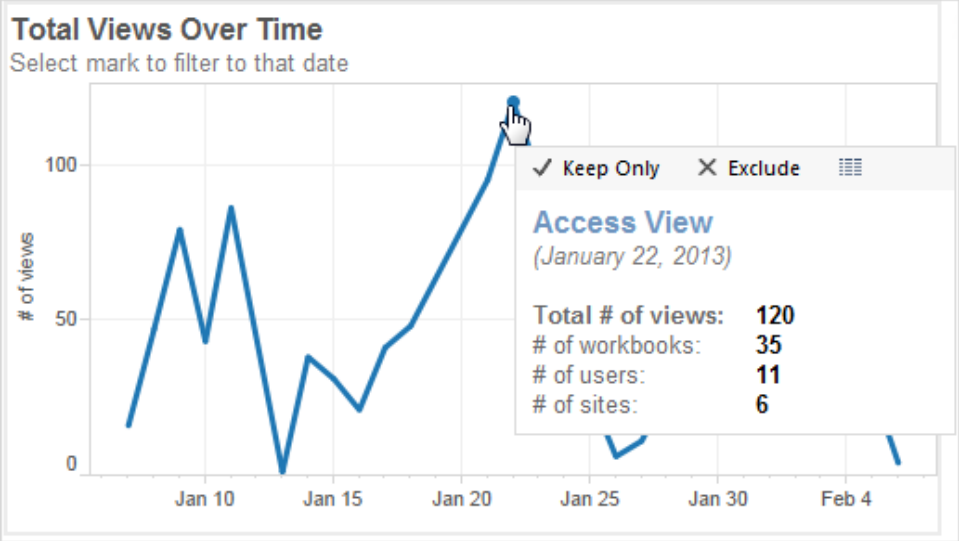
Click on the views below to display the corresponding analysis.

View	Analysis
Server Activity	A dashboard view showing recent activity on the server
User Activity	A view describing user activity, including logon time, hostname, and idle time
View Performance History	A view describing server activity broken down by view
Background Tasks	A view showing completed and pending task details
Space Usage	A dashboard view showing the space used by published workbooks and data sources
Customized Views	A dashboard view showing utilization of customized views

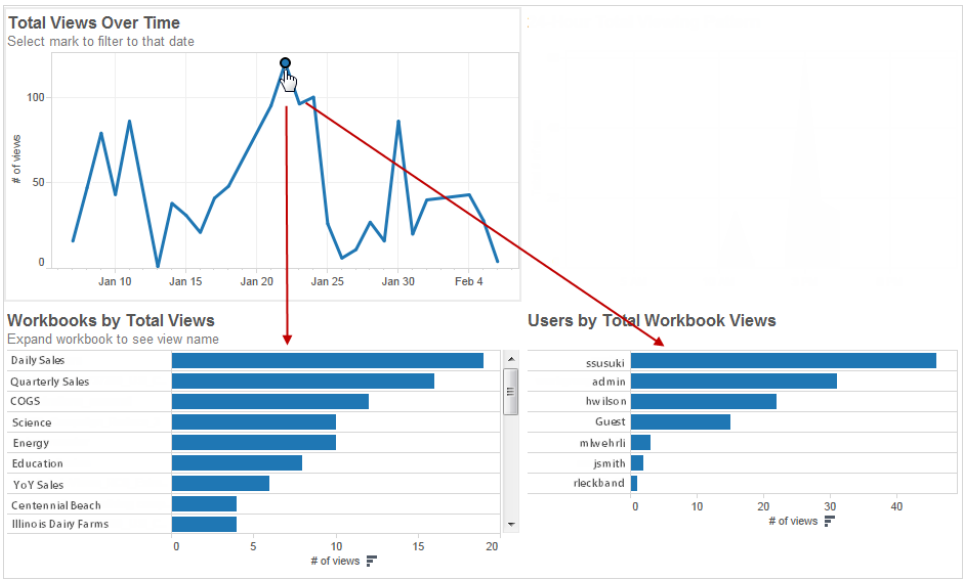
See the following topics for more information:

Server Activity

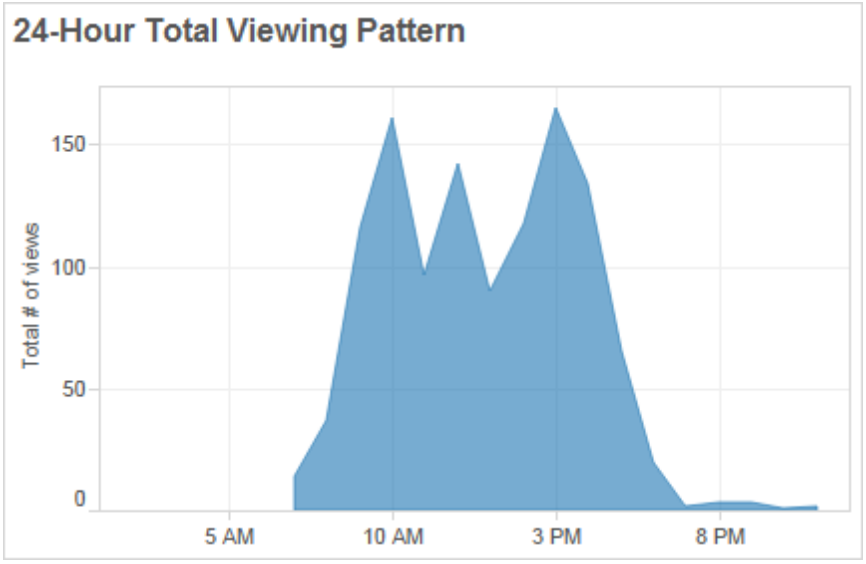
The Server Activity administrative view gives you a snapshot of Tableau Server activity within the past 30 days. In **Total Views Over Time**, you can hover your mouse over any point in the line and see a tooltip that shows you how many views were opened that day, along with other information:



Click the line and the bar charts below update to show you which workbooks were being viewed that day and who was doing the most viewing:



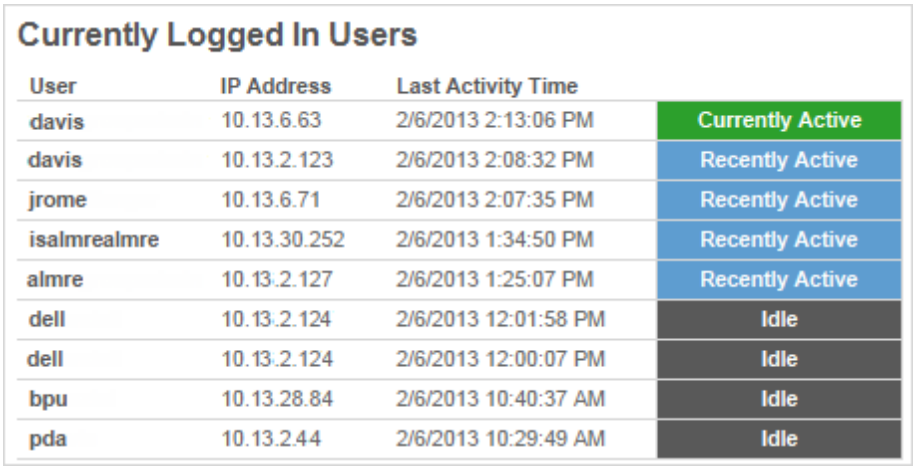
Selecting a mark in **Total Views Over Time** also filters the **24-Hour Total Viewing Pattern**, showing the viewing pattern for a particular day. If no marks are selected in **Total Views Over Time**, **24-Hour Total Viewing Pattern** sums the data in **Total Views Over Time** and displays it in a 24-hour time period so that you can see typical patterns over the course of a day:



User Activity

The User Activity view can help you gauge how heavily your Tableau Server installation is being used and whether you may need to buy additional licenses. Specifically, this view shows you who is logged into Tableau Server, from where, and when they last interacted with the server.

If a user is logged in from multiple browsers, that will be displayed as well. For example, if a user logs in once from Internet Explorer and once from Mozilla Firefox, their name appears twice. If a user logs in twice from Mozilla Firefox, their name appears once.

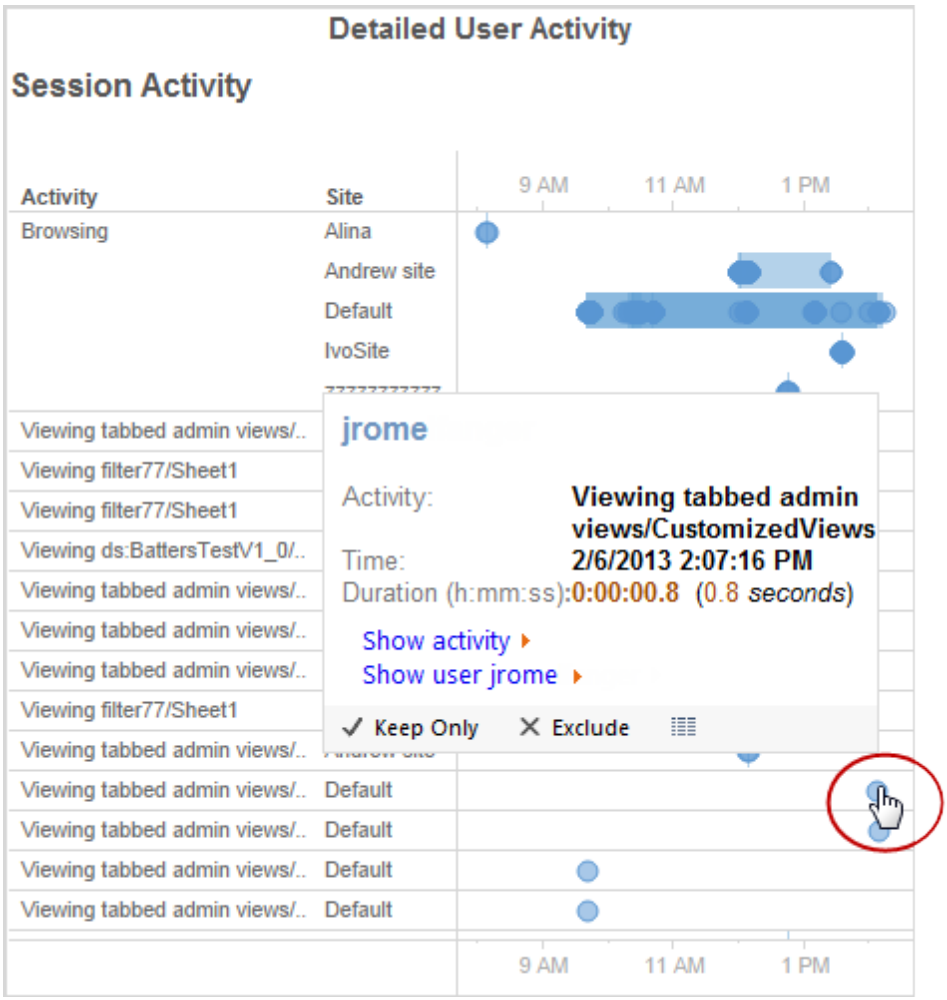


Currently Logged In Users

User	IP Address	Last Activity Time	
davis	10.13.6.63	2/6/2013 2:13:06 PM	Currently Active
davis	10.13.2.123	2/6/2013 2:08:32 PM	Recently Active
jrome	10.13.6.71	2/6/2013 2:07:35 PM	Recently Active
isalmrealmre	10.13.30.252	2/6/2013 1:34:50 PM	Recently Active
almre	10.13.2.127	2/6/2013 1:25:07 PM	Recently Active
dell	10.13.2.124	2/6/2013 12:01:58 PM	Idle
dell	10.13.2.124	2/6/2013 12:00:07 PM	Idle
bpu	10.13.28.84	2/6/2013 10:40:37 AM	Idle
pda	10.13.2.44	2/6/2013 10:29:49 AM	Idle

Currently Active means that the user interacted with the server during the past five minutes. **Recently Active** indicates that the user was active between the last five to 15 minutes, and **Idle** means there's been no activity from the user for the last 15 minutes. By default, after four hours of inactivity, users are logged off of Tableau Server. You can change this setting by using the tabadmin tabadmin set options option.

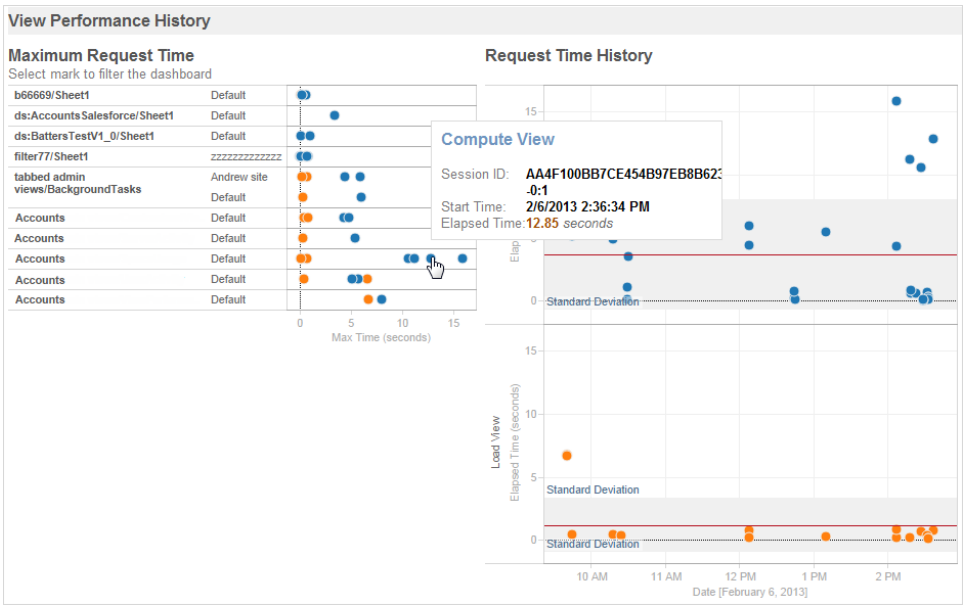
In the **Detailed User Activity** view, circles indicate an action, such as logging into the server or filtering a view. Bars span the total time period over which there was activity. To learn more, just hover over an area and a tooltip appears:



Performance History

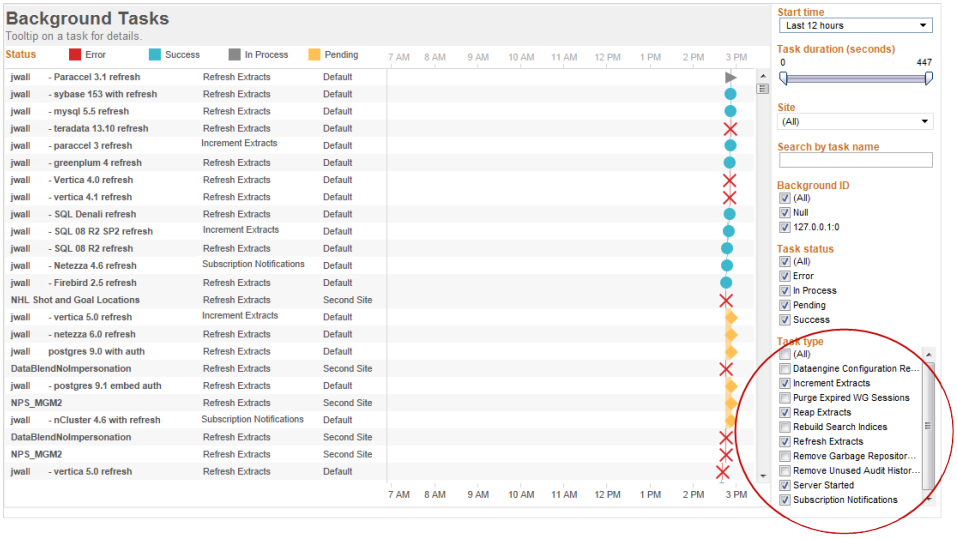
Use View Performance History to see which views are the most expensive in terms of server performance.

There are two different requests associated with views: Initial load requests, in orange, and compute requests, in blue. The latter are anything that causes Tableau Server to recompute what the user is seeing. This includes reload requests, and selecting and filtering items within a view. Outlier marks represent requests with the biggest impact on server performance:



Background Tasks

The Background Tasks view displays tasks that the server runs. The most common tasks are those associated with user actions. These are selected by default under **Task Type**:



Tasks can have a status of successful completion, error, in process, or pending:

Icon	Description
	**Error**—Server was unable to complete the task.
	**Success**—Server completed the task.
	**In process**—Server is currently completing the task.
	**Pending**—A task that the server has not yet started.

For details on a task, hover over its icon:

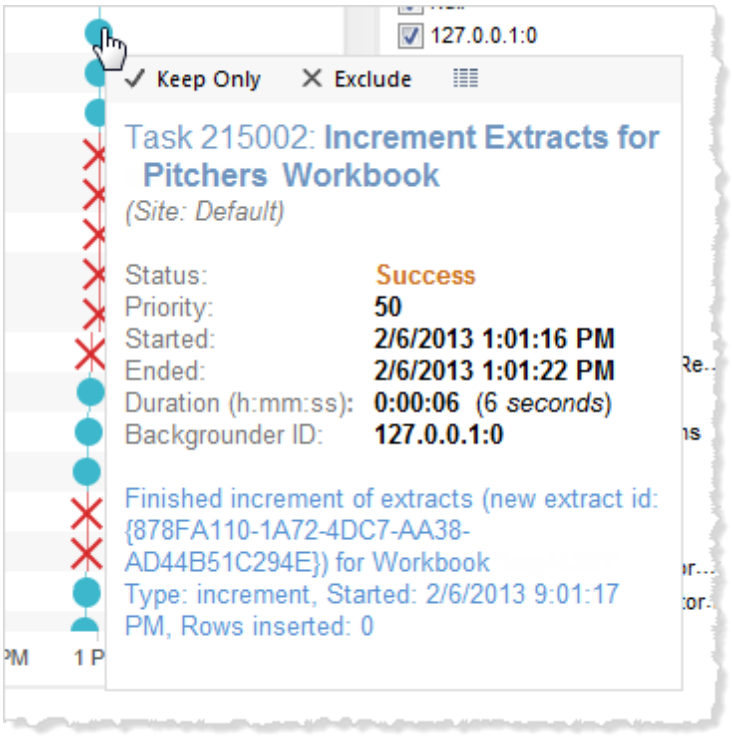
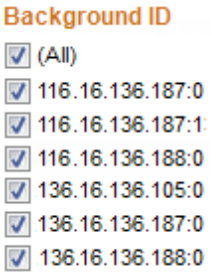


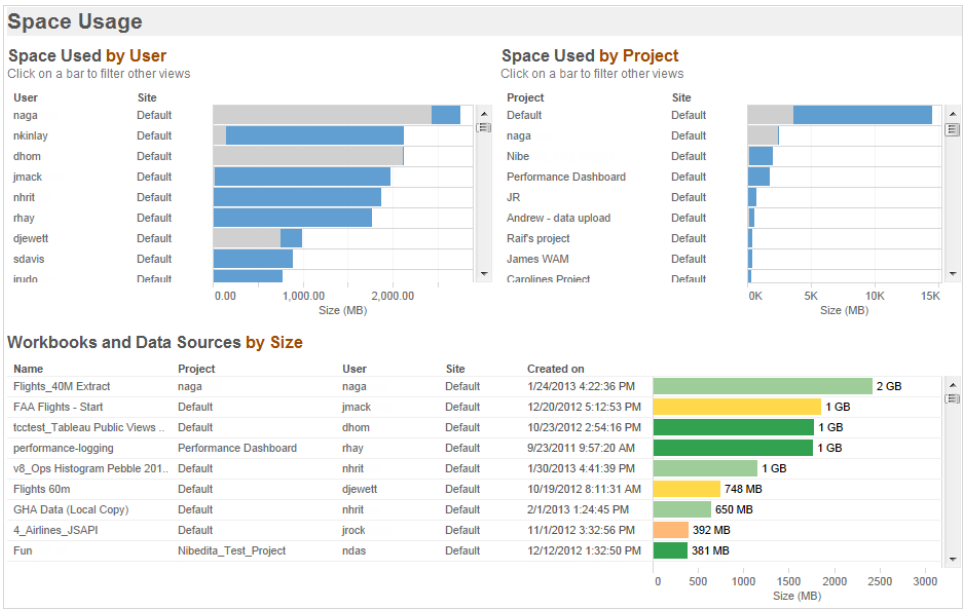
Tableau Server can run multiple background processes in parallel. The IP addresses under **Background ID** in the Background Tasks view show you which machines are assigned to run background processes:



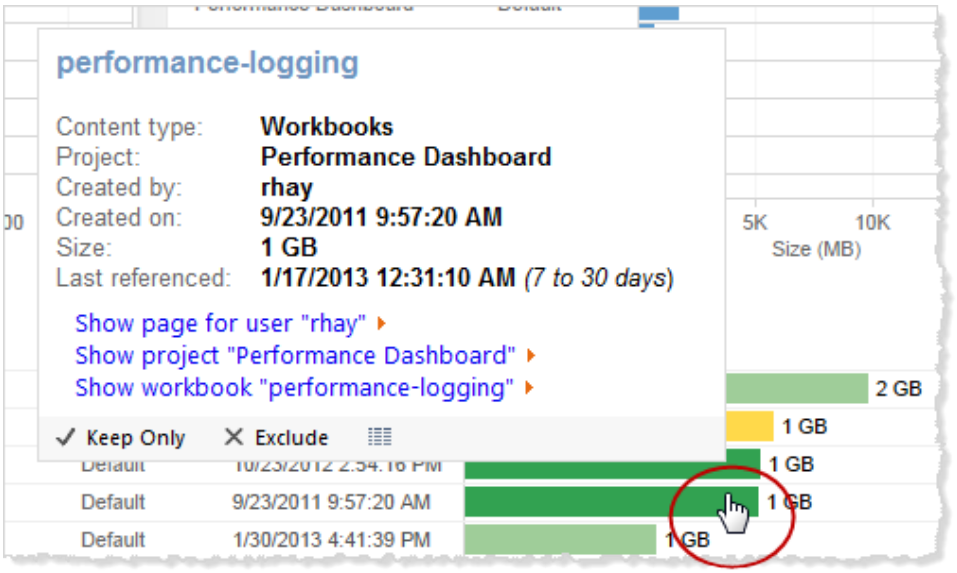
A multi-core machine running more than one background process will be listed with `<IP address>:0` for the first process, `<IP address>:1` for the second, and so on.

Space Usage

The Space Usage view can help you identify which workbooks and data sources are taking up the most disk space on your server. Disk space usage is displayed by user, project, and by the size of the workbook or data source and is rounded down to the nearest number:



Move your cursor over any size bar to display usage details:

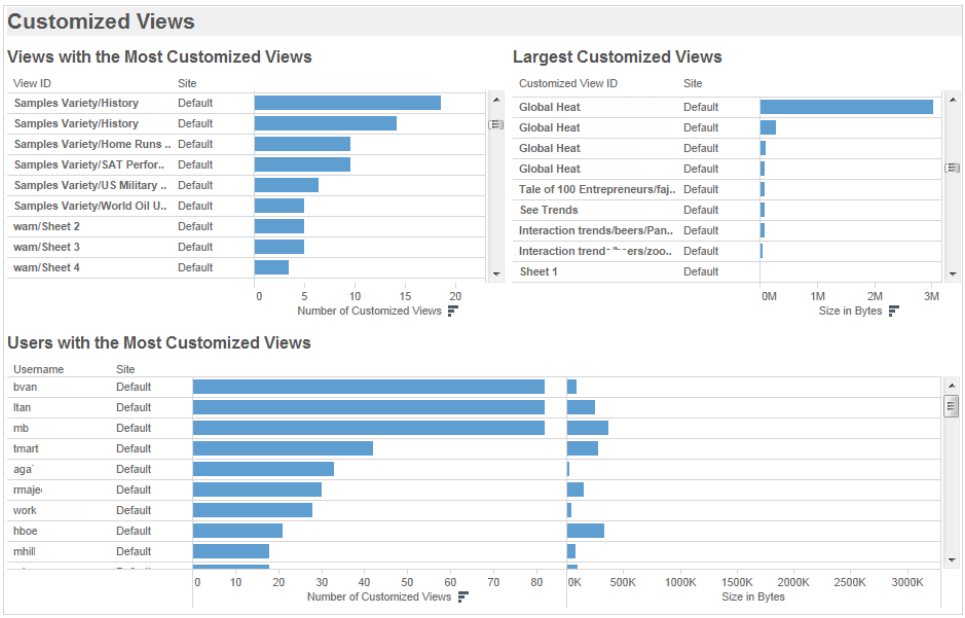


You can also drill into the links in the tooltip. For example, you can go to the user details for the user, or see the workbook.

Customized Views

People working with views can use the Remember my changes option to save their customized views and publishers can allow or prevent the sharing of customized views.

The Customized Views administrative view lists all the views on the server that have been customized with **Remember my changes**. It can be used as one indicator of a view's popularity or importance:



Create Custom Administrative Views

In addition to the pre-built administrative views available on the Maintenance page on the Server, you can use Tableau Desktop to query and build your own analyses of server activity. The Tableau Server repository has several database views set up that you can connect to and query. The tabadmin option auditing.enabled controls whether Tableau Server collects historical user activity and other information in the repository. It is enabled by default. The tabadmin option wgserver.audit_history_expiration_days controls how many days of event history are kept in the repository. By default, this is set to 183 days. One thing to note is that collecting historical events does impact the size of Tableau Server's backup file (.tsbak).

To access these views you must first use the command line tool to enable external access to the Tableau Server database. Next, you need to connect to and query the Tableau Server database.

Enabling Access to the Tableau Server database

The Tableau Server repository has several database views set up that you can connect to and query as part of building your own analyses of Tableau Server activity. To access these views you must first use the tabadmin command line utility to enable external access to the database.

1. Open a command prompt as an administrator and type:

   32-bit: `cd "C:\Program Files\Tableau\Tableau Server\8.0\bin"`

   64-bit: `cd "C:\Program Files (x86)\Tableau\Tableau Server\8.0\bin"`

2. Next, use the following command to enable external access to the database for the user "tableau" with a password that you specify.

   ```
   tabadmin dbpass [password]
   ```

Substitute the `[password]` option with your own password. For example:

```
tabadmin dbpass P@ssw0rD!
```

3. Restart the server.

After you've enabled external access to the database you can connect to and query the database. Follow the steps in Connecting to the Tableau Server Database to connect.

To disable external access later, run `tabadmin dbpass --disable` then restart the server.

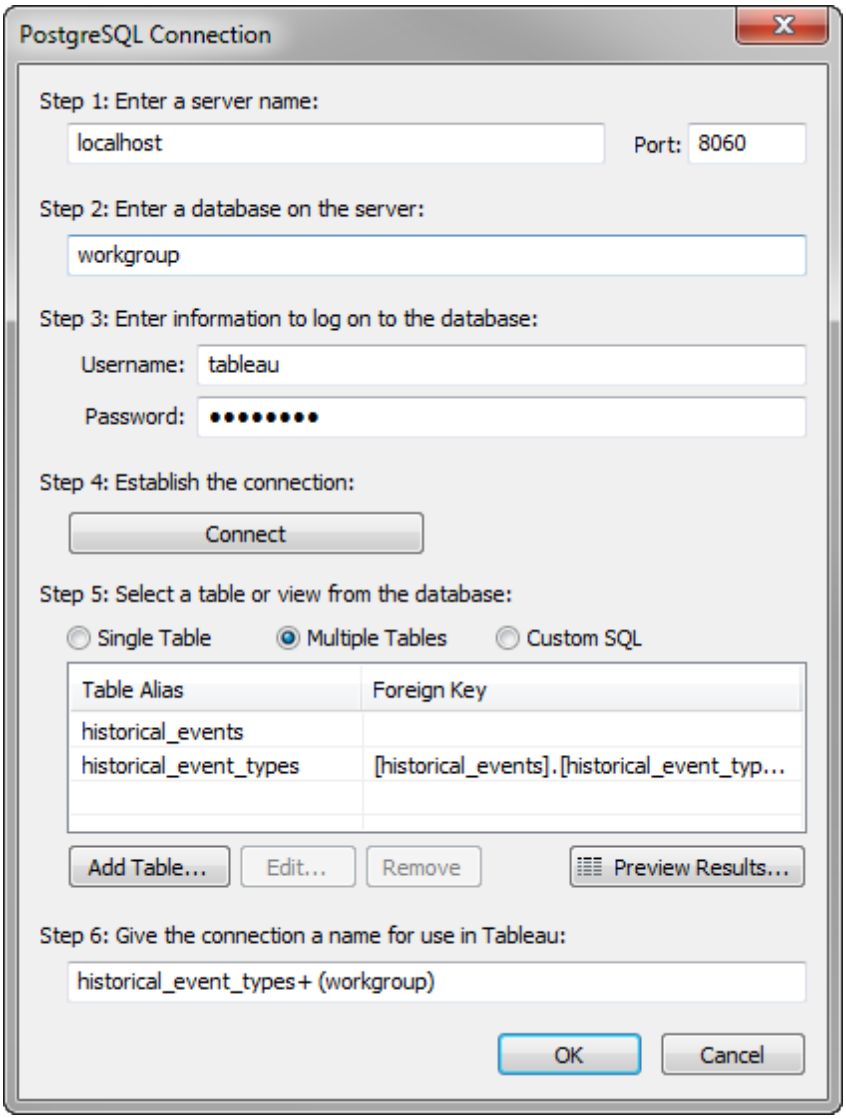
Connecting to the Tableau Server Database

After you enable external access to the Tableau Server database, follow the steps below to connect to and query the database.

1. In Tableau Desktop select **Data > Connect to Data**, then select **PostgreSQL** as the database to connect to. You may need to install the PostgreSQL database drivers. You can download drivers from www.tableausoftware.com/drivers.
2. In the PostgreSQL Connection dialog box, type the name or URL for Tableau Server. If you have a distributed server installation and a worker is hosting the repository, enter the name of the worker instead.

   You should connect using the port you have set up for the pgsql.port, which is 8060 by default. For more information about ports, see TCP/IP Ports.
3. Type `workgroup` as the database to connect to.
4. Connect using the following username and password:

   Username: tableau
   Password: The password you specified when you enabled access to the Tableau Server database.
5. Click **Connect**.
6. Select one or more tables to connect to. The "tableau" user has access to all of the tables the start with an underscore. For example, you can connect to **_background_tasks** and **_datasources**. The tables that begin with **historical_** point to **hist_** tables. The **hist_** tables include information about server users that isn't currently presented in the User Activity view.



7. Click **OK**.

Edit and Create Views

Users with appropriate credentials can edit existing workbooks and create new workbooks in Tableau Server.

Who Can Edit and Create Views

Administrators can always edit existing workbooks or create new workbooks. Other users should be provisioned as follows:

- The user must have the Interactor license level.
- The user must have the Publish user right.

Administrators assign a license level and user rights when they create or modify users. See Users. Also see Licenses and User Rights.

Permissions must also be set on the workbook:

- The **Web Edit** permission determines whether the user can edit workbook views.
- The **Download/Download/Web Save As** permission determines whether the user sees the **Save** and **Save As** commands while in edit mode. This permission also determines whether users can save their work as a new workbook.
- The **Write/Web Save** permission determines whether users can overwrite a workbook on the server.

Note that to enable a user to overwrite a workbook you must grant all three of these permissions: **Web Edit** to enter edit mode, **Download/Web Save As** to make the **Save** and **Save As** commands available, and **Write/Web Save** to actually be able to overwrite the workbook.

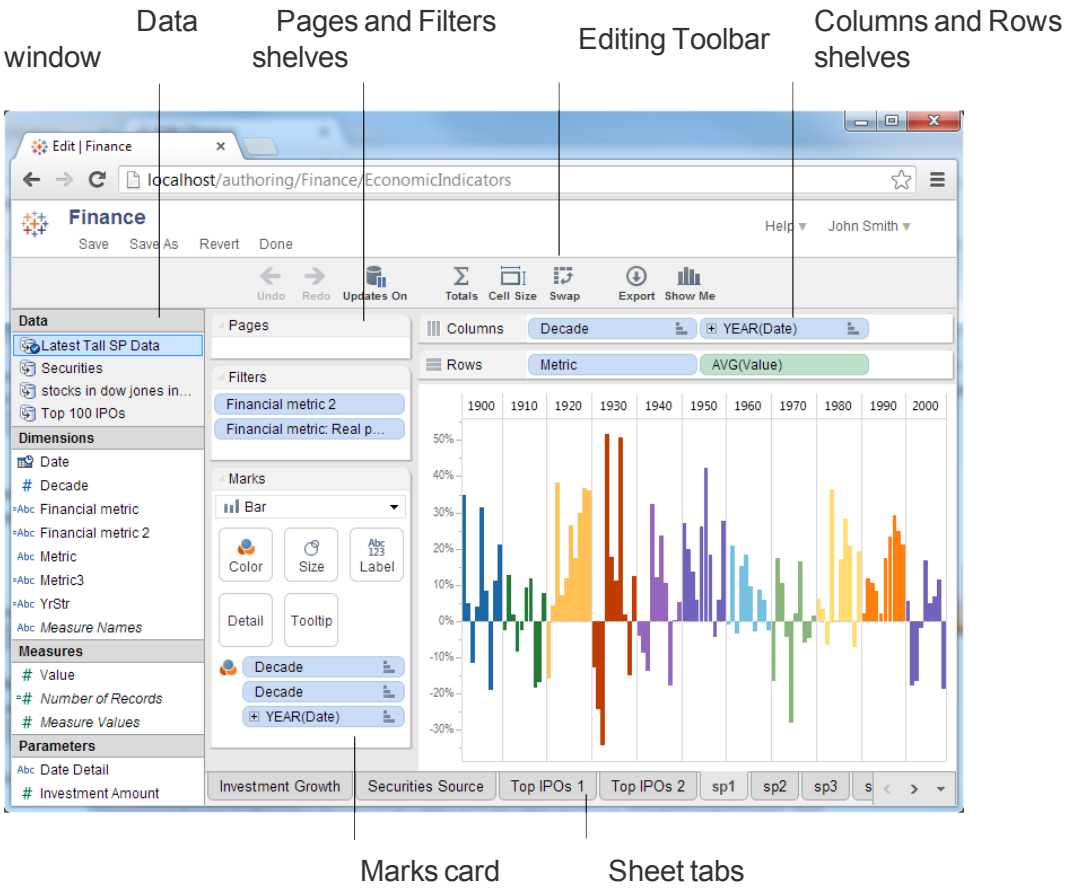
You can find out if you are authorized to edit views on a workbook by opening a view and then checking to see if there is an **Edit** command above the viewer area:

View permissions are copied from workbook permissions when you upload a workbook from Tableau Desktop. They are also copied when you click **Assign Permissions to Contents** on the Permissions: Workbook page. When you select **Show sheets as tabs** upon saving a workbook, the view permissions are overridden by the workbook permissions, until such time as tabs are disabled.

Open a View for Editing

Navigate to a view and click the **Edit** button in a view to enter edit mode. You can also click edit in the tooltip for a view.

When you enter edit mode, your screen changes to show an editing environment similar to Tableau Desktop, with a Data Window, a Marks card, Pages and Filters shelves, and Columns and Rows shelves:



You can edit the current view along with any other views in the workbook that are available. For details on the options and actions available when you edit workbook views, see Edit a View. If the workbook has a live database connection, you are prompted to authenticate.

If you are editing a workbook that contains multiple sheets, the available sheets are displayed along the bottom of the browser window. You can edit any of the other views in the workbook, but not dashboards. You can also choose the New Sheet tab to begin creating a new view:

When you are finished editing a view, click either **Save** or **Save As** above the viewer area:

- **Save** overwrites the original workbook, replacing any views you have edited.
- **Save As** creates a new workbook.

You launch edit mode from a view, but when you save your work, you save a complete workbook, including any other views you may or may not have edited. You cannot save your original workbook with both the original version of a view and the edited version. If you want to keep both versions, save your changes using **Save As** to create a new workbook.

To discard edits and return to the last saved version of a view, click **Revert**.

To exit edit mode, click **Done**. You can also exit edit mode by clicking the Tableau logo in the upper left corner of the screen. If you have unsaved changes, you are prompted to save them. If you opt to not save changes, the unsaved changes are still present when you return to edit mode for the view, for as long as you remain logged in.

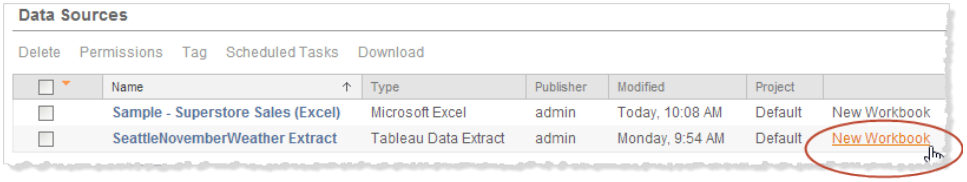
Create a Workbook

Users with the Interactor license level can create new workbooks. Such users are able to see data sources in the top-level list of navigation links:

There must also be at least one Data Source on the server, which you can use to create the new workbook.

To create a new workbook:

1. Click **Data Sources** to open the Data Source page, listing all the data sources that have been published to the server.
2. Click the **New Workbook** option at the right end of a data source listing to create a new workbook with that data source:



Your screen changes to show an editing environment similar to Tableau Desktop, with a Data Window, a Marks card, Pages and Filter shelves, and Rows and Column shelves.

You can click at the bottom of the window to create additional sheets in the workbook:

For details on the options and actions available when you edit in Tableau Server, see Edit a View.

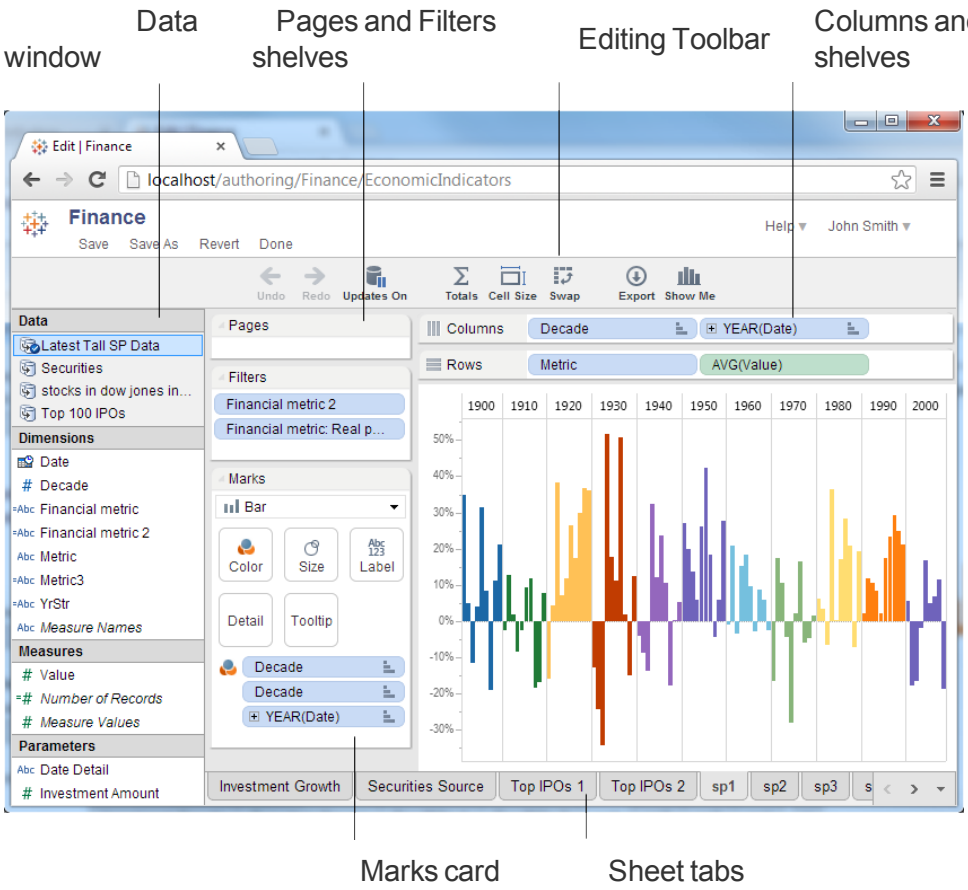
3. When you are finished editing a view, choose **Save** or **Save As** above the viewer area. Tableau prompts you to specify an existing project name and a new workbook name.

   Click **Revert** to return to the last saved version of the view.

4. Click **Done** to exit edit mode. You can also exit edit mode by clicking the Tableau logo in the upper left corner of the screen. If you have unsaved changes, you are prompted to save them. If you opt to not save changes, the unsaved changes are still present when you return to edit mode for the view, for as long as you remain logged in.

Edit a View

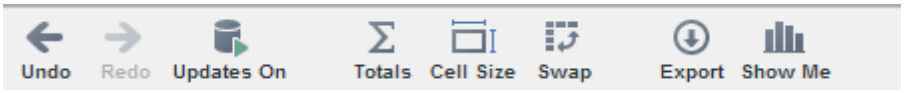
The web page where you edit an existing view resembles the workspace where you build visualizations in Tableau Desktop, with a Data Window, a Marks card, Pages and Filters shelves, and Columns and Rows shelves:



You can edit the views in a dashboard online, but you cannot edit the dashboard structure itself.

Click one of the following for more information.

Toolbar



When you are editing a view, the available toolbar buttons are:

- **Undo/Redo**

  You can undo and redo an action or series of actions. You can undo almost any action in Tableau by choosing the Undo button on the toolbar, or redo almost any action by choosing the Redo button.

- **Updates On**

  When you place a field on a shelf, Tableau generates the view by querying the data source. If updates seem slow when editing the view, you can turn off updates while making a series of edits, then turn them on again.

- **Totals**

  You can automatically compute grand totals and subtotals for the data in a view. Select Totals on the toolbar to see four options:

  - **Show Column Grand Totals**: Adds a row showing totals for all columns in the view.
  - **Show Row Grand Totals**: Adds a column showing totals for all rows in the view.
  - **Add All Subtotals**: Inserts subtotal rows and/or columns in the view, if you have multiple dimensions in a column or row.
  - **Remove All Subtotals**: Removes subtotal rows or columns.

- **Cell Size**

  Use the options under **Cell Size** to change the proportions of your view. The commands have different effects depending on the type of visualization. Try different commands to see their effect.

- **Swap**

  Moves the fields on the Rows shelf to the Columns shelf and vice versa.

- **Export**

  Use the options under Export to capture parts of your view for use in other applications.

  - **Image**: Displays the view or dashboard as an image in a new browser tab.
  - **Data**: Displays the data from the view in a new browser window with two tabs:

Summary, showing aggregated data for the fields shown in the view, and **Underlying**, showing underlying data for the selected marks in the visualization. If the new window does not open, you may need to disable your browser's popup blocker.

- **Crosstab**: Saves the underlying data for the selected marks in the visualization to a CSV (comma-separated values) file which can then be opened in Microsoft Excel.
- **PDF**: Opens the current view as a PDF in a new browser window. From there you can save it to a file. If the new window does not open, you may need to disable your browser's popup blocker.

- **Show Me**

  Opens a window showing a range of visualization types available in Tableau. The range of possible visualizations for a view depends on the type of data that is used in the view. If a visualization type is possible with the data in your view, you can select it. You can also hover over a visualization type to see what types of fields are required before it can be selected for the view.

Data Window

At the top of the Data window is a list of available data sources for the workbook. If you are editing an existing workbook, there may be multiple data sources. Select a data source to see the dimensions and measures for that data source. If you are creating a new workbook, you see just the data source from which you created the workbook.

All data sources contain fields. These fields appear below the list of data sources in the Data window. Dimensions and measures always appear, other field types appear if they are present in the data source:

- **Dimensions** are fields that contain discrete qualitative data. Examples of dimensions include dates, customer names, and customer segments.
- **Measures** are fields that contain numerical data that can be aggregated. Examples of measures include sales, profit, number of employees, temperature, frequency, and pressure.
- **Sets** are custom fields that define a subset of data based on some conditions. A set may be based on a computed condition, which updates as the data changes, or a constant list of values. Sets may be present in workbooks that you edit, but you cannot create sets.
- **Parameters** are dynamic values that can replace constant values in calculations, filters, and reference lines. Parameters may be present in workbooks that you edit, but you cannot create parameters.

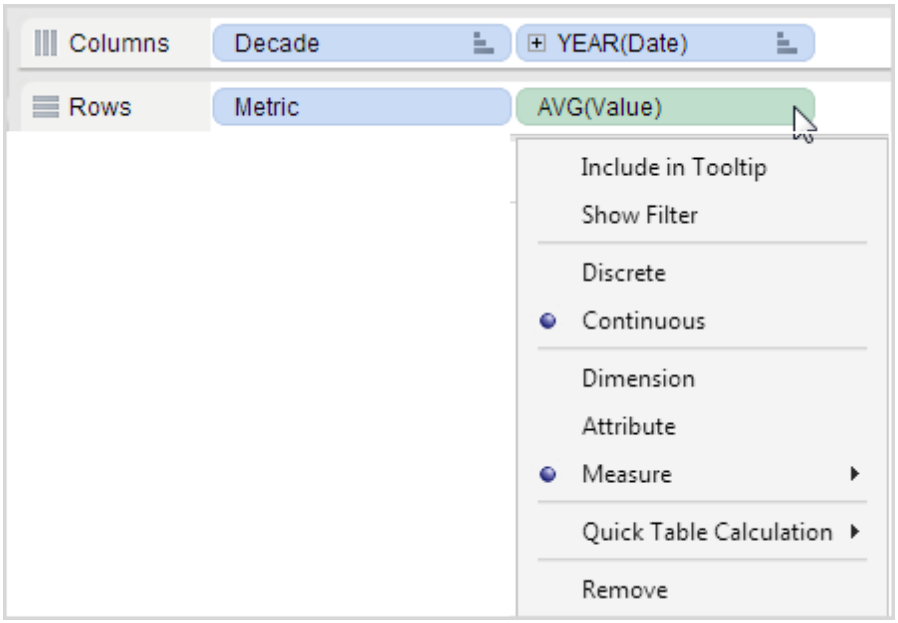
To build visualizations, you drag fields from the Data window to the Rows and Columns shelves, the Marks card, or one of the other available shelves. For a demonstration, see Build a View.

Columns and Rows Shelves

Drag fields to the Columns shelf to create the columns of a table, or to the Rows shelf to create the rows of a table. You can drag multiple fields to either shelf.

Discrete values (typically, dimensions) are displayed in blue on the Columns and Row shelves; continuous values (typically, measures) are displayed in green.

At the right end of any field you place on the Columns or Rows shelf is a drop down menu that you can use to configure the dimension or measure:



The options that are available depend on the type of field. The complete list of options includes:

- **Include in Tooltip**

  By default, all fields on the Columns and Rows shelf are included in the tooltips that appear when you move your mouse over one or more marks in the view. Un-check this option to remove a field from tooltips.

- **Show Filter**

  Choose this option to add a filter for this field to the view. Users will then be able to specify which data to include and exclude for this dimension or measure.

- **Discrete/Continuous**

  Use these options to convert a continuous range of values into a set of discrete values, or a discrete set into a continuous range.

- **Dimension/Attribute/Measure**

Use this range of options to convert a dimension to a measure or a measure to a dimension.

You can also define the option as an Attribute, which returns the value of the given expression if it only has a single value for all rows in the group, and otherwise displays an asterisk (*) character. Null values are ignored.

- **Quick Table Calculation**

  Provides a set of options for redefining the meaning of the marks for the value.

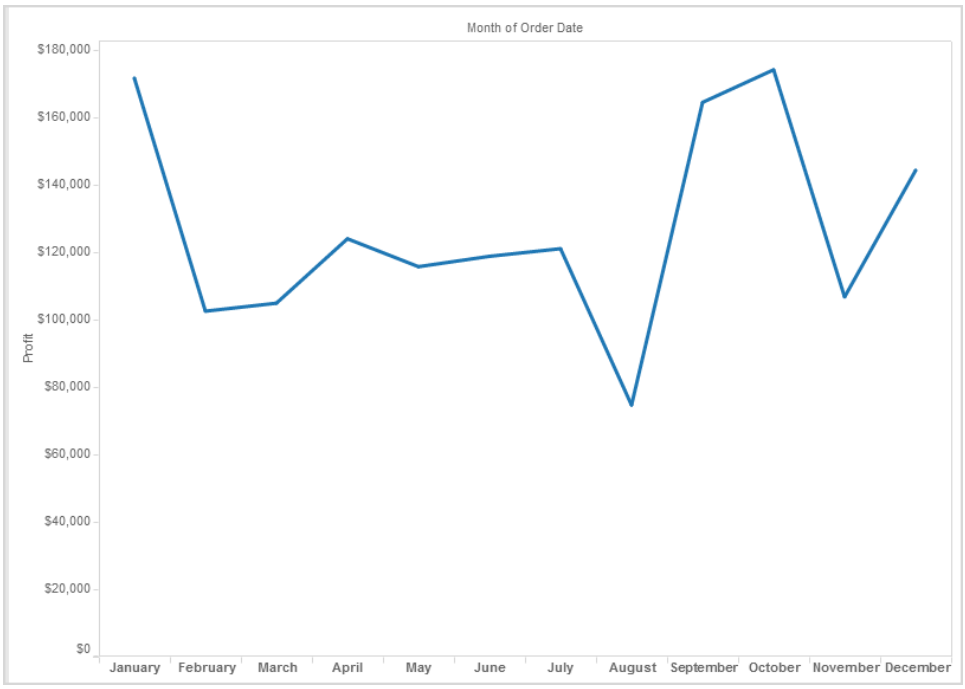
- **Remove**

  Removes the value from the Columns or Rows shelf.

Options for Date Dimension

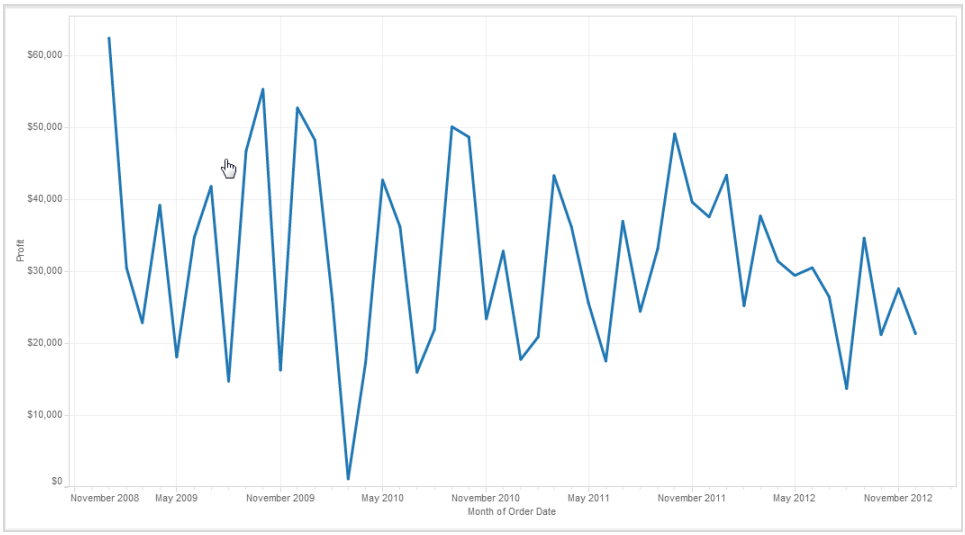
An additional set of options is available for date dimensions:

Choose one of the options from the upper group to define the granularity of the data as discrete values. For example, if you choose **Month** your view will combine the data for each named month in your data across the full range of years:



There are exactly twelve marks in the data--one for each month. The November mark combines the data from November 2008, November 2009, etc.

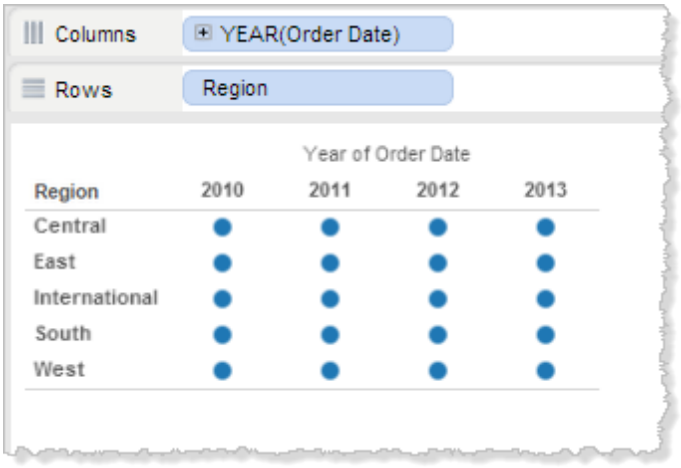
Choose one of the options from the lower group to define the granularity of the data as continuous values. For example, if you choose Month your view will show your data sequentially, over the range of available months.



In this case there are 48 marks in the data--one for each month since November 2008.

The Marks Card

When you drag fields to the view, the resulting data points are displayed using marks. Each mark represents the intersection of all of the dimensions in the view. For example, in a view with **Region** and **Year** dimensions, there is a mark for every combination of those two field--for example, East 2011, East 2012, West 2011, West 2012, etc.:

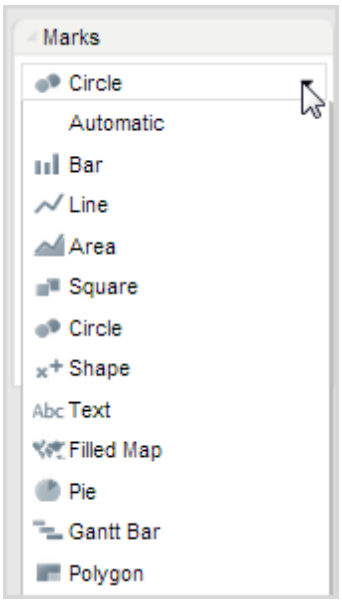


Marks can be displayed in many different ways including lines, shapes, bars, maps, and so on. You can show additional information about your data using mark properties such as color, size, shape, and labels. The type of mark you use and the mark properties are controlled by the Marks card. Drag fields to the Marks card to show more data. For example, you can enhance the view above by dragging **Profit** to Color. With this additional information, it becomes apparent that the International region is consistently more profitable than other regions:

Use the drop-down menu in the Marks card to specify the type of mark to show. Then use the drop-down menus for the individual fields on the Marks card to add more information to the view and to control the color, shape, size, labels, and number of marks in the view.

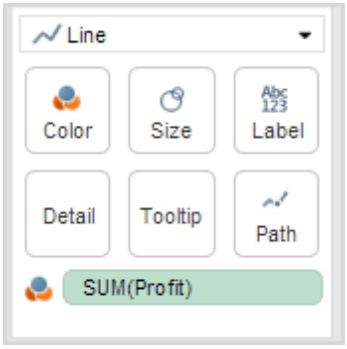
Mark Types

Mark types are available from the Mark card drop-down menu.



Mark Properties

With the marks properties on the Marks card, you can control the colors, size, shape, etc., of the marks in the view. Drag fields to one of the property panes to encode the marks using your data.



The number of properties available, and the actual properties themselves, vary from mark type to mark type. For example, the Shape property is only available with the Shape mark type, and the Angle property is only available with the Pie mark type.

The properties are:

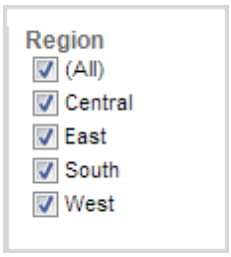
Property	Description
Color	Encodes data by assigning different colors to the marks in a data view based on the values of a field.
Size	Separates marks according to the members in the dimension, and assigns a unique size to each member. Because size has an inherent order to it (small to big), categorical sizes work best for ordered data like years or quarters.

Label/Text	Encodes data by assigning text labels to the marks. When working with a text table, this property is called Text, and shows the numbers associated with a data view.
Detail	When you place a dimension on the Rows or Columns shelf, the categorical members create table headers. The headers represent levels of detail because they separate the data source rows into specific categories. You can identify each category by the member name. The Detail property also allows you to separate the marks in a data view according to the members (levels of detail) of a dimension. However, unlike the Rows and Columns shelf, this property does not modify the table structure.
Tooltip	Adds the dimension to the tooltip for each mark.
Path	Allows you to encode data by connecting marks using a particular drawing order. You can path-encode your data using either a dimension or a measure. When you place a dimension on the Path shelf, Tableau connects the marks according to the members in the dimension. If the dimension is a date, the drawing order is given by the date order. If the dimension holds words such as customer names or product types, the drawing order is given by the order of the members in the data source. When you place a measure on the Path shelf, Tableau connects the marks according to the values of the measure. The Path property is available only when you select the line or polygon mark type from the Mark menu.
Shape	Separates the marks according to the members in the dimension, and assigns a unique shape to each member.

The Filters Shelf

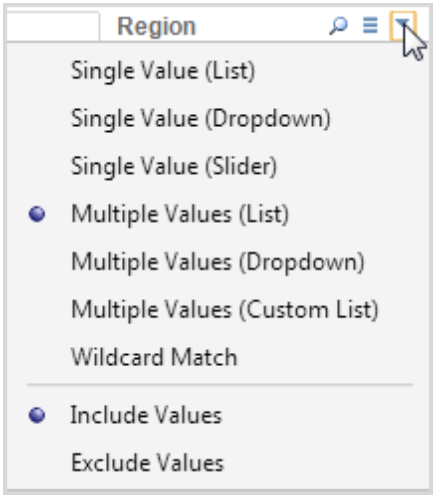
Use the Filters shelf to specify which data to include and exclude for a dimension or measure. For example, you might want to analyze the profit for each customer segment, but only for certain shipping containers and delivery times. By placing the Container dimension on the Filters shelf you can specify which containers to include. Similarly, you can put the Delivery Date field on the Filters shelf to define which delivery times to include.

When you drag a dimension or measure to the Filters shelf, Tableau automatically inserts a filter control into the view for selecting the values to display. For example:



For dimensions, the filter control shows discrete values, as above. For measures, the control shows a continuous range:

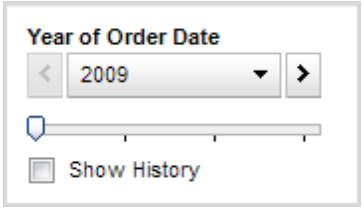
Hover your mouse to the right of the title for the filter control to specify how values in the control are to be displayed:



The Pages Shelf

Drag a dimension or measure to the Pages Shelf to break a view into a series of pages so you can better analyze how a specific field affects the rest of the data in a view. Dragging a dimension to the Pages shelf is like adding a new row for each member in the dimension. Dragging a measure to the Pages shelf automatically converts the measure into a discrete measure that can be broken into individual pages.

When you drag a dimension or measure to the Pages shelf, Tableau automatically inserts a control into the view to let you navigate the pages in your view. For example:



You can manually advance through the sequence of pages in any of the following ways:

- Use the drop-down menu to select a value.
- Use the forward and back buttons on either side of the drop-down list to navigate through the pages one at a time.
- Use the Page slider to quickly scroll forward and backward in the sequence of pages.

Select **Show History** to show marks from previous pages in addition to marks for the current page.

Tooltips

Place your cursor over a mark in the view to see the tooltip for that mark.

Tooltips provide information on the values of dimensions and measures for the selected mark:

Tooltips also provide these options:

- **Keep Only**

  Exclude all marks from the view except this one.

- **Exclude**

  Exclude this mark only.

- **Group Members**

  Choose the paperclip icon to create a new group, which is a dimension, from the selected mark. Typically, you would select multiple marks and then create a group. For example, if you have a dimension Region with values North, South, East and West, you could select South and West and then create a group from them.

- **View Data**

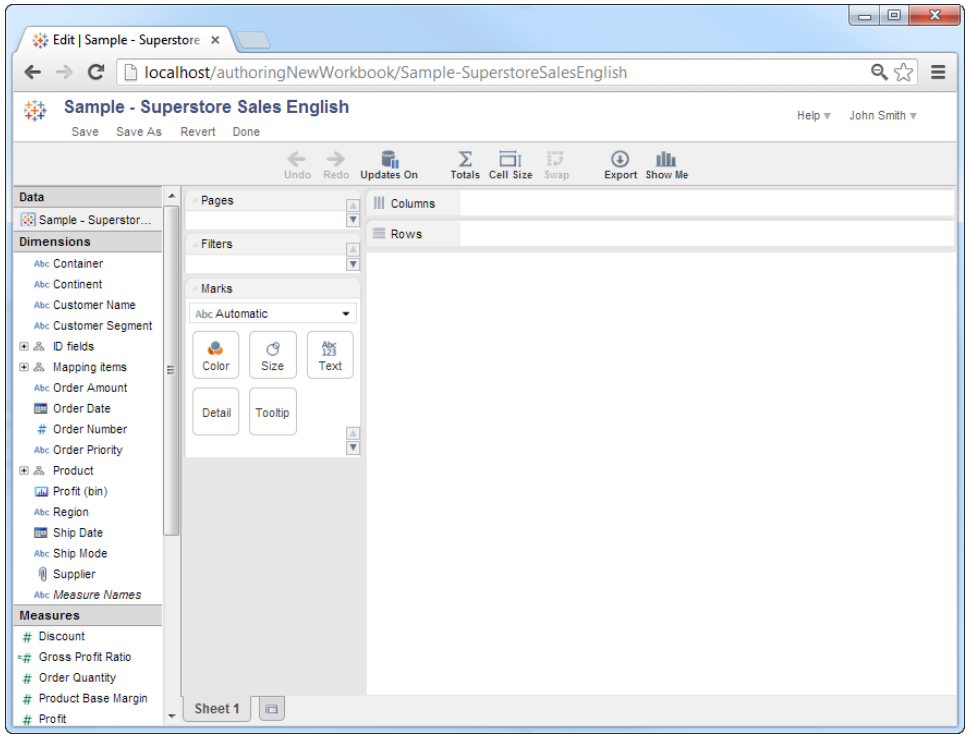
  Choose the table icon to open a new browser window to display two tabs: **Summary**, which shows only data for the current mark, and **Underlying**, which shows data for the entire view.

Build a View

You can build a new view either by creating a new sheet in an existing workbook or by creating a new workbook. This topic shows the actions you perform to build a view.

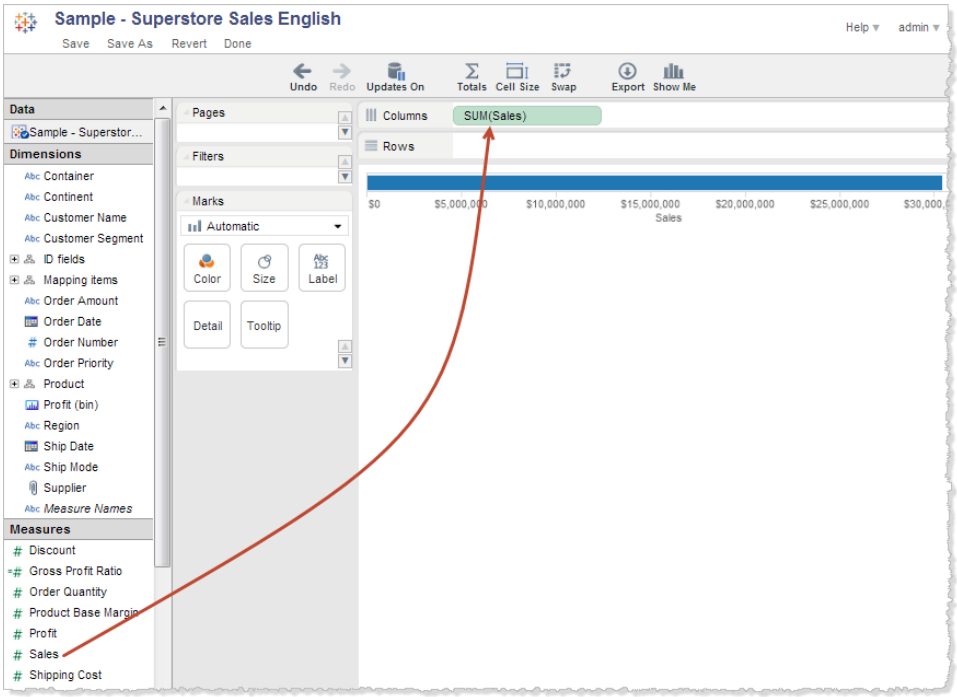
The following procedure shows how to get started creating a simple visualization. It uses the **Sample – Superstore Sales** sample data source, and incorporates information about sales by category and region. If you have the **Sample – Superstore Sales** sample data source available, you can perform the actions in the procedure.

When you create a new view, it initially looks like this:

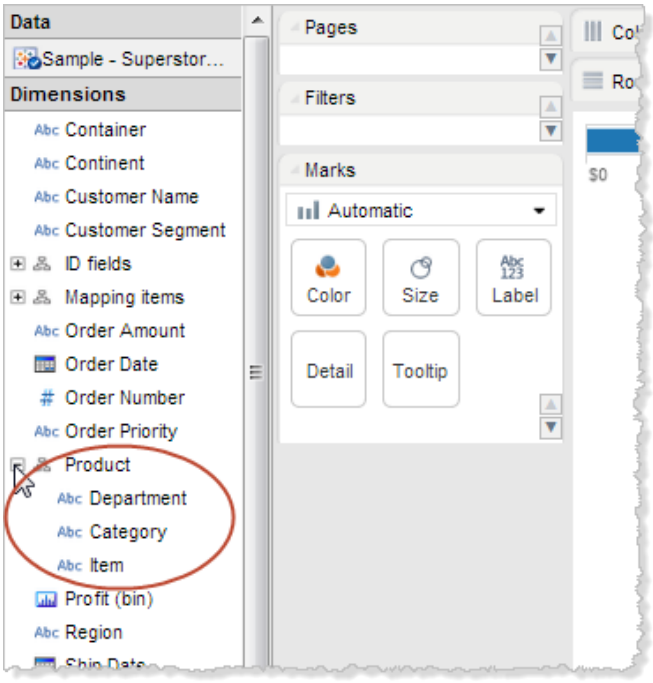


Follow these steps to build a view.

1. Drag **Sales** from the Measures section of the Data Window to the Columns shelf:



2. Under Dimensions in the Data Window, expand the **Product** hierarchy dimension to show the three subcategories: **Department**, **Category**, and **Item**.



3. Drag **Category** to the Rows shelf:

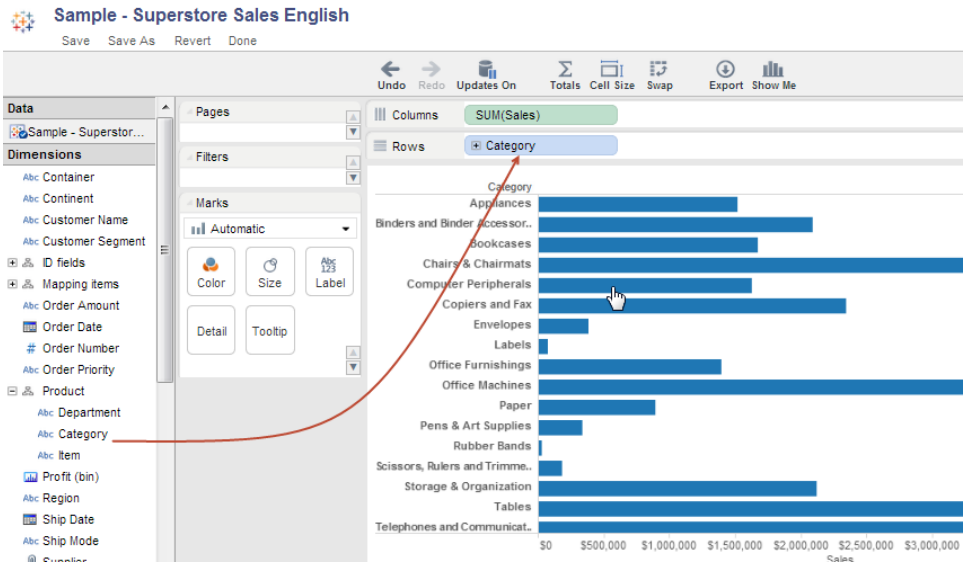
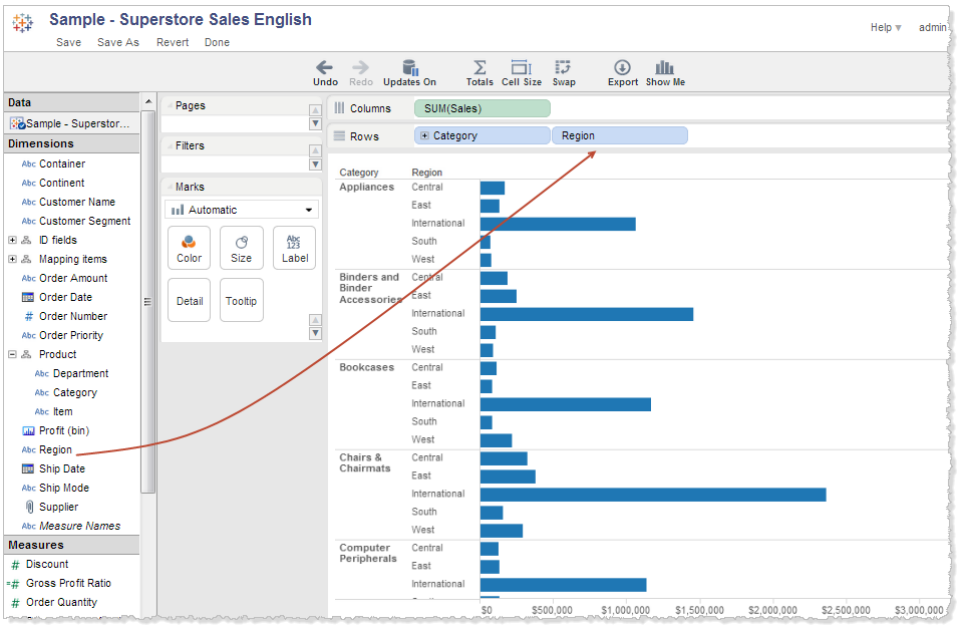


Tableau now has enough information to create a visualization, in this case a horizontal bar chart.

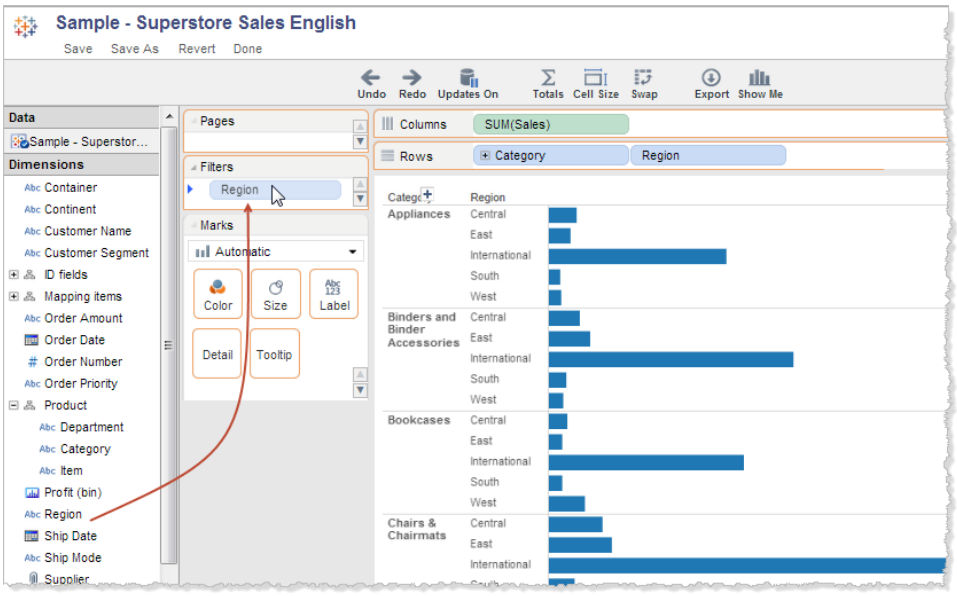
4. Drag the **Region** dimension to the Rows shelf:



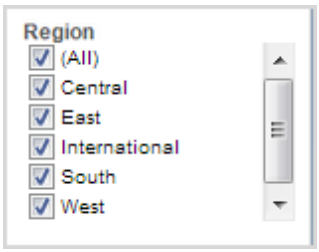
The visualization now contains another layer of data--the categories are broken out by region.

But notice that the majority of sales are in the International region. Suppose you only wanted to view and compare sales in the US. You can accomplish this by filtering out International sales.

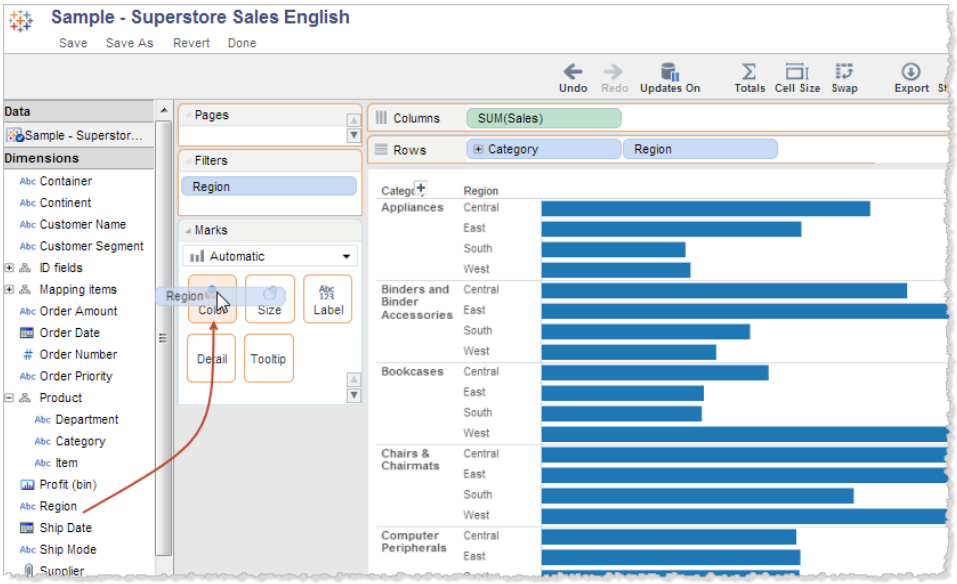
5. Drag the **Region** dimension from the Data Window to the Filters shelf. As you hover over the Filters shelf, you see a small triangle at the left of the field. This indicates that you can now drop **Region** on the shelf.



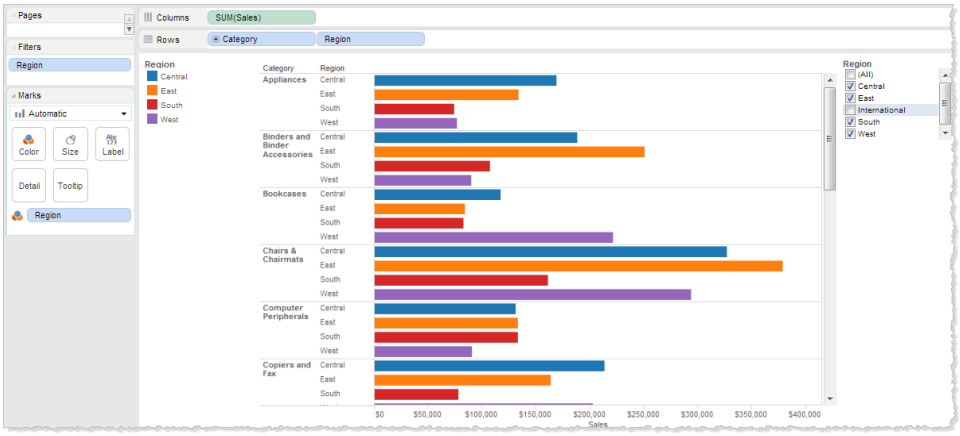
At the right edge of the page, there is now a Filter control:



6. Clear the **International** region check box.
7. You can enhance the visualization using color. Drag **Region** to the Color property on the Marks card:

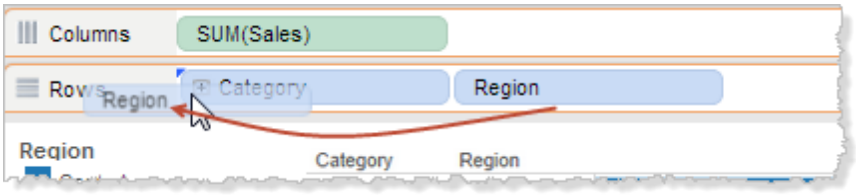


You now have a useful visualization that allows you to compare sales of different product categories across regions:

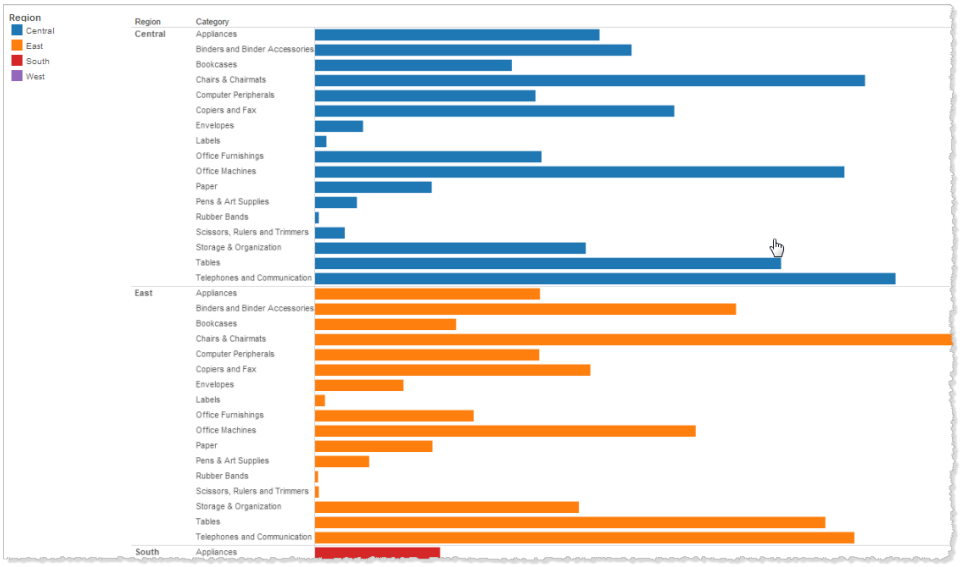


8. Maybe you'd prefer a view that lets you more easily compare sales across regions. Drag

Region on the Rows shelf and drop it in front of **Category**:



The visualization now makes it easier to compare regions:

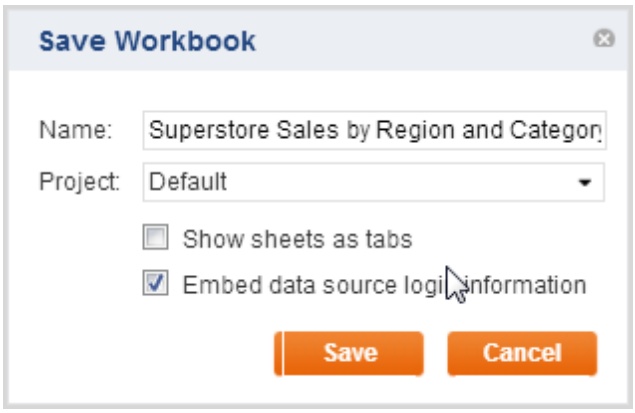


9. You decide that the you'd rather go back to the previous version before saving your visualization to a workbook.

   Click **Undo** in the Toolbar.

10. Click **Save** to save the workbook. In the **Save Workbook** dialog box, specify a workbook name: Superstore Sales by Region and Category.

    Because there is only a single sheet in your workbook, you can ignore the **Show sheets as tabs** option. Select **Embed data source login information** if you want to assure that users who don't have an account on the database are able to see the view. When you're finished, click **Save**:



Security

There are four main components to security in Tableau Server:

Authentication

Authentication establishes a user's identity. This is done to prevent unauthorized access to Tableau Server and allow for a personalized user experience. Tableau Server supports three types of authentication:

- **Active Directory**: Authenticates Tableau Server users based on their Windows credentials.
- **Local Authentication**: Uses the internal authentication mechanism provided with Tableau Server.
- **Trusted Authentication**: Handles authentication through a trust relationship between Tableau Server and one or more web servers.

Whether to use Active Directory or Local Authentication is a choice you make during Tableau Server Setup. After Setup, you can't switch between the two. To switch authentication types, uninstall Tableau Server (your data will be preserved) and rerun Setup.

Active Directory

When Active Directory is used for user authentication, all usernames and passwords are managed by Active Directory. When a user enters their credentials at the Tableau Server login, Tableau passes them to the Active Directory server. It does not participate in the authentication process—although it does store usernames (but not passwords) in its repository.

With Active Directory user authentication, administrators can also automatically log in users based their current Windows credentials (Enable Automatic Login). This means that the user's

credentials are being passed from their local computer, not from another system or portal that they may have logged in to.

For example, if a user logs into their local computer as 'MSmith' and then logs into a SharePoint portal as 'Mary', the credentials passed to Tableau Server will be for 'MSmith'. To use the credentials from the SharePoint site ('Mary') for automatic login, the SharePoint portal must use the Tableau web part with trusted authentication.

Administrators can synchronize groups with Active Directory either manually or programmatically, using tabcmd. See Synchronize an Active Directory Group and `syncgroup group-name` in tabcmd Commands for more information.

Local Authentication

When Local Authentication is used for user authentication, Tableau Server manages users, groups, passwords and the entire authentication process. User lists can easily be imported to the Tableau Server and most user management functions can be performed programmatically through tabcmd. Users can either manually login by entering their credentials when prompted or, when accessing content in a portal, via transparent trusted authentication.

Trusted Authentication

Trusted authentication means that you have set up a trusted relationship between Tableau Server and one or more web servers. For example, you may have your corporate wiki use trusted authentication to show dashboards to employees who are already signed onto the wiki, without requiring another login.

When Tableau Server receives requests from a trusted web server it assumes that the web server has already handled whatever authentication is necessary. Tableau Server receives the request with a redeemable token or ticket and presents the user with a personalized view which takes into consideration the user's role and permissions.

See Trusted Authentication for information on how to set up trusted authentication at your site.

Authorization

Authorization is what a user can access and do once he or she is authenticated. In Tableau, authorization is handled by the following:

- **Roles and permissions**: Define specific capabilities that users can or cannot perform on certain objects in Tableau. A role is a set of permissions that administrators can use as-is or customize. See Work with Permissions for details.
- **Licensing and user rights**: Control the maximum set of permissions that a user can have. See Licenses and User RightsLicenses and User Rights and Allow or Deny User Rights.

While the above items control which actions a user can perform and on what, they do not control which data will appear inside a view. The data a user sees is controlled by your data security choices.

Initial Permissions

The initial permissions for a project are copied from the Default project. The initial permissions for a workbook are copied from the permissions for its project. The initial permissions for a view are copied from the permissions of its workbook. This is a one-time copy of the parent's permissions. Changes to the parent's permissions are not automatically applied to the children unless the new permissions are actively assigned to the contents.

Any item can have permissions that differ from the parent. For example, a group might not have permission to see Project X, but it may have permission to see a view that's published to Project X. Tableau Server does not support hierarchical object permissions; however, it does provide an inheritance model for users and groups. If a user does not have a permission explicitly set to Allow or Deny, the setting will be inherited from the groups the user belongs to.

Permissions and the Default Project

If Tableau Server is deployed in an open environment where knowledge and information sharing is key, then you should consider setting the permissions for the Default project to include the **All Users** group, with its role set to **Interactor**. Users will be able to automatically publish to and consume content from new projects.

If Tableau Server is deployed in a restrictive environment where data security and access control is key, then consider emptying the permissions for the Default project: Delete the permissions for all users and groups. Users and groups will need to be explicitly granted permission to publish and consume content in new projects.

Data Security

Tableau provides several ways for you to control which users can see which data. For data sources that connect to live databases, you can also control whether users are prompted to provide database credentials when they click a published view. The following three options work together to achieve different results:

- **Database login account:** When you create a data source that connects to a live database, you choose between authenticating to the database through Windows NT or through the database's built-in security mechanism.
- **Authentication mode:** When you publish a data source or a workbook with a live database connection, you can choose an **Authentication mode**. Which modes are available depends on what you choose above.
- **User filters:** You can set filters in a workbook or data source that control which data a person sees in a published view, based on their Tableau Server login account.

The table below outlines some dependencies with the above options:

Database Connection Options		*Data Security Questions*		
Database login account uses...	**Authentication mode**	**Is database security possible per Tableau Server user?**	**Are user filters the only way to restrict which data each user sees?**	**Are web caches shared among users?**
Window NT Integrated Security (Windows Authentication)	*Server Run As account*	No	Yes	Yes
	Impersonate via server Run As account	Yes	No*	No
Username and Password	*Prompt user*: Viewers are prompted for their database credentials when they click a view. Credentials can be saved.	Yes	No	No
	Embedded credentials: The workbook or data source publisher can embed their database credentials.	No	Yes	Yes
	Impersonate via embedded password: Database credentials with impersonate permission are embedded.	Yes	No*	No

* Because it can create unexpected results, Tableau recommends that you not use this authentication mode with user filters.

User filters, the embedded credentials option and the impersonation modes have similar effects—when users click a view, they are not prompted for database credentials and they see only the data that pertains to them. However, user filters are applied in the workbook by authors, and the impersonation authentication modes rely on security policies defined by administrators in the database itself.

Some of the options described above require configuration steps that must happen during Tableau Server Setup or before you publish a workbook or data source. See the following topics for more information:

- Run As User
- SQL Server Impersonation
- Embedded Credentials
- Saved Passwords

Network Security

There are three main network interfaces in Tableau Server:

- **Client to Tableau Server**: The client can be a web browser, Tableau Desktop, or the tabcmd utility.
- **Tableau Server to your database(s)**: To refresh data extracts or handle live database connections, Tableau Server needs to communicate with your database(s).
- **Server component communication**: This applies to distributed deployments only.

Client to Tableau Server

A Tableau Server client can be a web browser, Tableau Desktop, or tabcmd. Communications between Tableau Server and its clients use standard HTTP requests and responses. Tableau Server can also be configured for HTTPS (see SSL). When Tableau Server is configured for SSL, all content and communications between clients are encrypted and use the HTTPS protocol.

Passwords are communicated from browsers and tabcmd to Tableau Server using public/private key encryption. Tableau Server sends a public key to the browser, which uses the key to encrypt the password for transmission. Each encrypted transmission uses a key one time before it is discarded. This means that passwords are always secured regardless of the use of SSL.

Tableau Server to Your Database

Tableau Server makes dynamic connections to databases to process result sets and refresh extracts. It uses native drivers to connect to databases whenever possible and relies on a generic ODBC adapter when native drivers are unavailable. All communications to the database are routed through these drivers. As such, configuring the driver to communicate on non-standard ports or provide transport encryption is part of the native driver installation. This type of configuration is transparent to Tableau.

Server Component Communication

There are two aspects to communication between Tableau Server components in a distributed server installation: trust and transmission. Each server in a Tableau cluster uses a stringent

trust model to ensure that it is receiving valid requests from other servers in the cluster. The primary server is the only machine in the cluster that accepts requests from 3rd parties (clients), all other machines in the cluster only accept requests from other trusted members of the cluster. Trust is established by a whitelist of IP address, port, and protocol. If any of these are invalid, the request is ignored. All members of the cluster can communicate with each other. With the exception of license validation and accessing the repository, transmission of all internal communication is performed via HTTP.

When passwords are transmitted within the cluster, a key is used to encrypt the passwords transmitted between Tableau Server components (for example, between the application server and the VizQL server processes). Each encrypted transmission uses a key one time before it is discarded.

Performance

Every server environment is unique and there are many variables that impact performance. Variables include hardware details, such as disk speed, memory, and cores; the number of servers in your deployment; network traffic; usage factors such as workbook complexity, concurrent user activity, and data caching; Tableau Server configuration settings, such as how many of each server process you're running; and data considerations—such as data volume, database type, and database configuration. Because of this complexity, there is no single formula for improving server performance, but there are some basic guidelines you can follow. Use the topics below for more information:

General Performance Guidelines

Hardware and Software

Use a 64-bit operating system: Although Tableau Server runs well on 32-bit Microsoft operating systems, for the best performance, choose a 64-bit operating system. It ensures that the 64-bit version of the Tableau data engine is used. It also increases the capacity of the 32-bit processes because they get access to more main memory.

Add more cores and memory: Regardless of whether you're running Tableau Server on one machine or several, the general rule is that more CPU cores and more RAM will give you better performance. Make sure you meet Tableau Server's recommended hardware and software requirements and see When to Add Workers & Reconfigure to assess whether you should add additional machines. If you are running Tableau Server in a virtual environment, use your VM's best practices for vCPU allocation in relation to the number of physical CPU cores on the VM host.

Configuration

Schedule refreshes for off-peak hours: Backup tasks tend to stall other background tasks until the backup is completed. Use the Background Tasks administrative view to see your refresh and backup task schedules.Your refresh tasks should be scheduled for off-peak hours that don't overlap with your backup window.

Look at caching: Caching helps Tableau Server respond to client requests quickly, especially for views that connect to live databases. Confirm that **Refresh Less Often** on the Data Connections tab of the Configuration dialog box is selected.

Consider changing two session memory settings:

- **VizQL session timeout limit**: The default VizQL session timeout limit is 30 minutes. Even if a VizQL session is idle, it is still consuming memory and CPU cycles. If you can make do with a lower limit, use tabadmin to change the `vizqlserver.session.expiry.timeout` setting .
- **VizQL clear session**: By default, VizQL sessions are kept in memory even when a user navigates away from a view. This consumes a good deal of session memory. Instead, you can end sessions when users move away from a view by changing the value of the `vizqlserver.clear_session_on_unload` setting to `true` (default is `false`).

Assess your process configuration: Tableau Server is divided into six different components called server processes. While their default configuration is designed to work for a broad range of scenarios, you can also reconfigure them to achieve different performance goals. Specifically, you can control on which machines the processes run and how many are run. See Improve Server Performance for guidelines for one-, two-, and three-machine deployments.

When to Add Workers & Reconfigure

Tableau Server can scale up and out as your needs and requirements evolve. Here are some guidelines to help you figure out whether it's time to add more nodes to your system, reconfigure the server, or both:

- **More than 100 concurrent users**: If your deployment is user-intensive (>100 simultaneous viewers), it's important to have enough VizQL processes—but not so many that they exceed your hardware's capacity to handle them. Also, enabling the Tableau Server Guest User account can increase the number of potential simultaneous viewers beyond the user list you may think you have. The administrative view User Activity can help you gauge this. For tips on how to configure or scale your deployment, see Improve Server Performance.
- **Heavy use of extracts**: Extracts can consume a lot of memory and CPU resources. There's no one measurement that qualifies a site as extract-intensive. Having just a few, extremely large extracts could put your site in this category, as would having very many small extracts. Extract heavy sites benefit from isolating the data engine process on its own machine. Refer to Improve Server Performance for guidelines.
- **Frequent extract refreshes**: Refreshing an extract is a CPU-intensive task. Sites where extracts are frequently refreshed (for example, several times a day) are often helped by more emphasis on the background process, which handles refresh tasks. Use the Background Tasks administrative view to see your current refresh rate, then see Improve Server Performance for details on how to scale.
- **Troubleshooting performance**: If views are slow to load or server performance is

generally slow, there could be several causes. See General Performance Guidelines as well as Improve Server Performance.

- **Downtime potential**: If your server system is considered mission critical and requires a high level of availability, you can configure it so there's redundancy for the server processes that handle extracts and the repository. Refer to High Availability for details.

Improve Server Performance

Use the topics below for guidance on how to improve the performance of deployments that are extract-intensive, user-intensive, or both:

What's your goal?

Optimizing for Extracts

The data engine process stores extracts and answers queries; the background process refreshes extracts. Because both are demanding of CPU resources, the best approach to improving performance for an extract-intensive deployment is to isolate these two processes from one another, and from the other server processes. This may take three machines. If you don't have three machines to work with, there are still strategies you can use (see the deployment examples below).

Optimizing for Users and Viewing

The VizQL server process handles loading and rendering views for Tableau Server users. If you are trying to optimize your deployment for a high number of users and a lot of view

interaction, this is the process you should focus on. If you can run additional VizQL Server processes on additional machines instead of on the same machine, do so. You'll see better performance since the VizQL Server processes will have fewer processes to compete with for system resources.

How Many Processes to Run

In general, more processes means more processing power and better performance, but too many can overwhelm your system or even decrease server performance. You can run up to 8 instances each of the VizQL Server, application server, data server, or background processes on any given machine—but it's likely that 8 aren't necessary (learn more). As you consider how many instances of each process to run, pay special attention to the VizQL Server process and the background process.

VizQL Server Process

The VizQL Server process is multi-threaded and multi-processed on a machine. There can be over 16 threads running on a single machine.

Caching is something to keep in mind with the VizQL Server process. Each instance has its own cache, and caching affects server performance. This is because a cached view loads faster than a view that's being requested for the first time. Too many unique caches makes it less likely that a request will be handled by a process that already has the result in its cache. As you look at the suggested VizQL ranges in this topic, start with the low end of the range and see how views perform before you increase to the maximum number.

Here are some general, conservative guidelines for determining the minimum and maximum number of VizQL Server processes to run:

- **Minimum number per deployment**: 1 VizQL Server process for every 100 concurrent viewers using your deployment.
- **Maximum number per machine**: The machine's RAM divided by 4 (64-bit) or 2 (32-bit), with an upper limit of 8 per machine.

  Note: The deployment examples later in this topic may use slightly different formulas to address specific performance goals.

As an example, a single 64-bit machine with 32 GB of RAM that's handling 300 concurrent viewers should run a minimum of 3 VizQL Server processes and a maximum of 8. If you have two such machines to work with, you should have at least 3 VizQL Server processes across both machines, with each machine being a capable of running up to 8 VizQL Server processes each. A single 32-bit machine with 8 GB of RAM that's handling 200 concurrent viewers can run 2-4 VizQL Server processes. If this same machine is handling 500 concurrent viewers, it's probably time to add more hardware.

Background Process

A single background process can consume 100% of a single CPU core, and sometimes even more for certain tasks. As a result of this, the total number of instances you should run depends on the machine's available cores—as well as on what you're trying to improve. The deployment examples below uses `N` to represent the machine's total number of cores, and each suggests a

different strategy where the background process is concerned. When in doubt, start with the low end of the suggested range and assess performance before increasing the number.

Data Engine and Repository Processes

There are scenarios where the data engine process should be isolated on its own node—such as if you are trying to improve an extract-intensive deployment and you want to emphasize querying more than extract refreshes. The deployment examples provide specifics. Because the data engine stores real-time data, transferring it is a multi-phased procedure. Move the Data Engine and Repository Processes describes how to do it.

Another reason to isolate the data engine (and/or the repository) is to minimize your deployment's potential for downtime. See High Availability for details. Unless you're configuring for high availability, the repository can usually remain on the primary Tableau Server.

Where to Configure Processes

You configure the type and number of processes any machine is running using the Tableau Server Configuration dialog box. If you are adding new machines as part of your reconfiguration, they must already have Tableau Worker software installed on them. Refer to Install and Configure Worker Servers for steps.

If you are reconfiguring the processes on your primary or standalone Tableau Server, see Reconfigure Processes.

Optimizing the Extracts and Workbooks

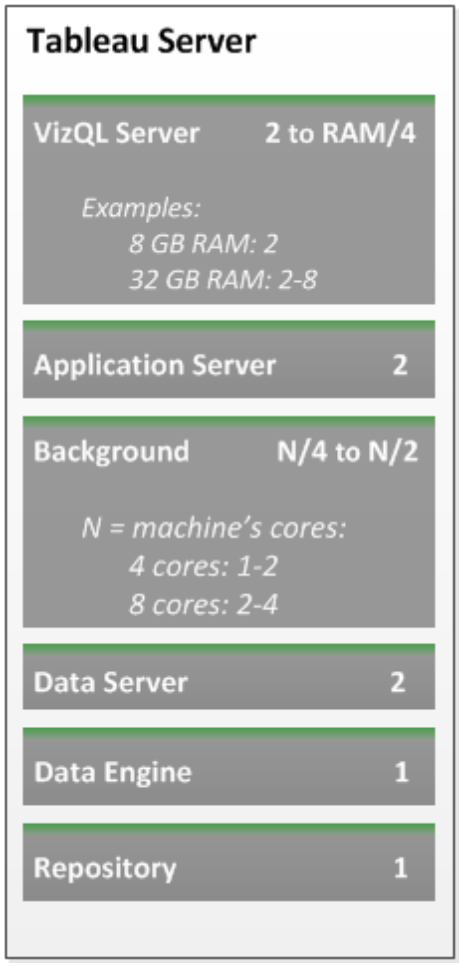
Fast server performance with extracts is partly a function of the extracts and workbooks themselves. Workbook authors can help improve server performance by keeping the extract's data set short, through filtering or aggregating, and narrow, by hiding unused fields. Use the Tableau Desktop options **Hide All Unused Fields** and **Aggregate data for visible dimensions** to do this. For steps, see Creating an Extract (Tableau Desktop help). For general tips on building well-performing workbooks, search for *"performance"* in the Tableau Desktop help. To see how workbooks perform after they've been published to Tableau Server you can create a performance recording. See Create a Performance Recording for details.

Assessing View Responsiveness

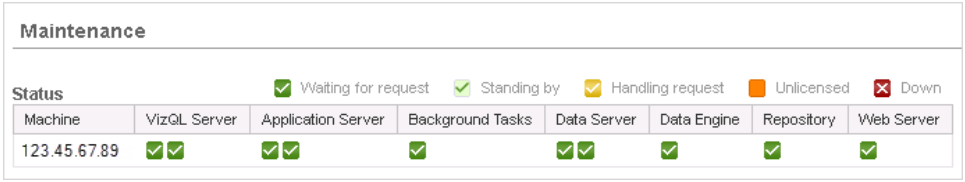
When a user opens a view, the components of the view are first retrieved and interpreted, then displayed in the user's web browser. For most views, the display rendering phase occurs in the user's web browser and in most cases, this yields the fastest results and highest level of interactive responsiveness. Handling most interactions in the client web browser reduces bandwidth and eliminates round-trip request latencies. If a view is very complex, Tableau Server handles the rendering phase on the server instead of in the client web browser—because that generally results in the best performance. If you find that views aren't as responsive as you'd like, you can test and change the threshold that causes views to be rendered by the server instead of in the client web browser. See About Client-Side Rendering for more information.

One-Machine Example: Extracts

A Tableau Server installation with heavy extract usage can run on a single 64-bit machine configured as follows:



The above configuration would look like the following on the Tableau Server Maintenance page:



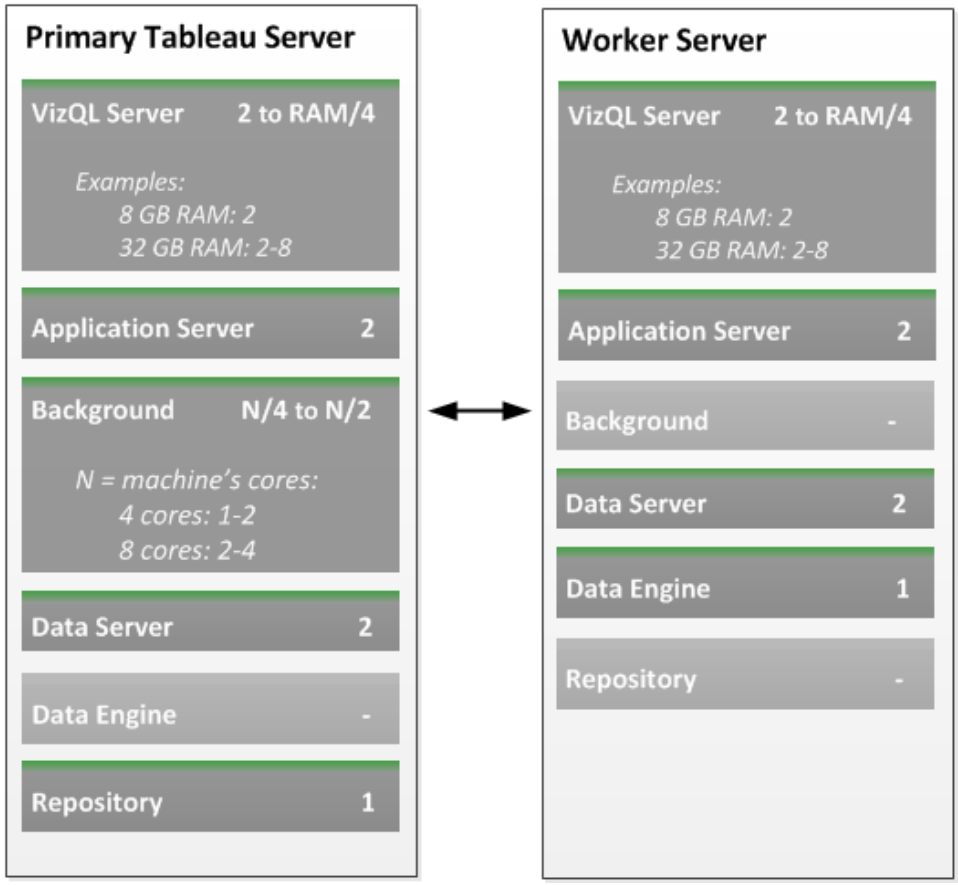
Configuration Notes:

- Run at least 2 VizQL server processes and determine the maximum number to run by dividing the machine's RAM by 4.
- Calculate the least number of background processes to run by taking the machine's total number of cores and divide it by 4. To determine the maximum number, divide by 2.

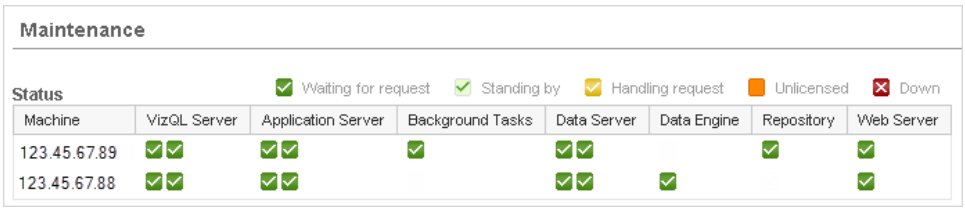
- Both the background and data engine processes are CPU-intensive and the above configuration balances them.
- Scheduling extract refreshes for off-peak times helps the data engine and background processes not compete with one another for system resources.
- Because you are not trying to optimize for a high number of concurrent users here, start off with 2 VizQL Server processes and observe how views perform before you increase the number.

Two-Machine Example: Extracts

Here's how you can configure a two-machine Tableau Server deployment so that it can handle heavy extract usage. The most important thing to note in this example is that the data engine process is isolated from the background processes. Both servers are running on 64-bit operating systems:



With the above configuration, the Status table on the Maintenance page would look like this:

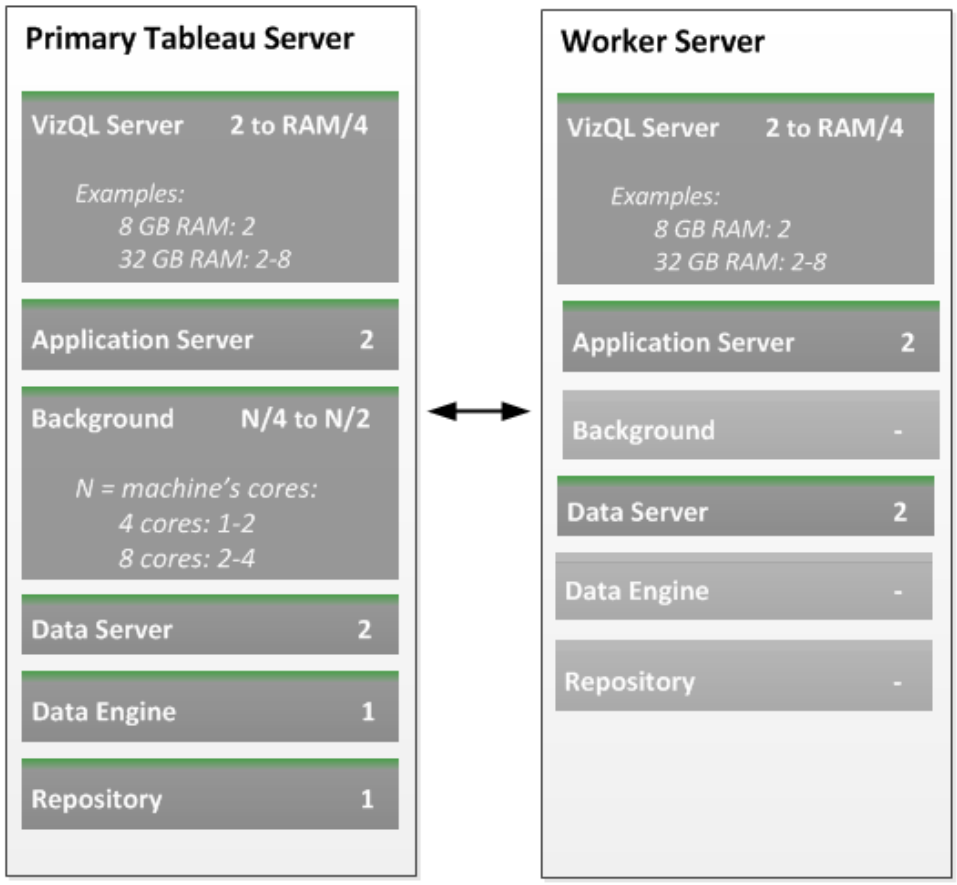


Configuration Notes:

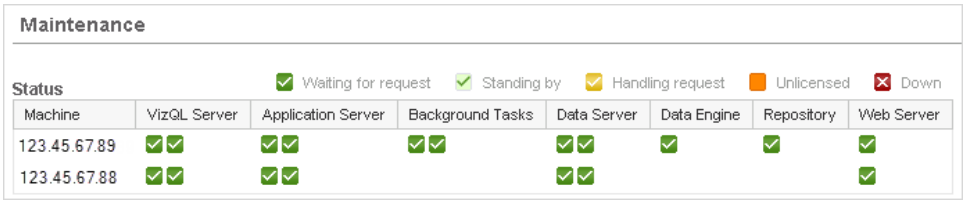
- Once you go from a one- to two-machine deployment, your first server becomes the primary Tableau Server. In the Status table it gets a value of Gateway.
- Run at least 2 VizQL server processes on each machine and determine the maximum number to run by dividing the machine's RAM by 4.
- To figure out the minimum number of background processes to run on the primary Tableau Server, take the machine's total number of cores and divide it by 4. For the maximum number, divide by 2.
- Moving the data engine from the primary Tableau Server to the worker is a multi-phased procedure. See Move the Data Engine and Repository Processes for steps.

Two-Machine Example: Viewing

A two-machine deployment with light extract usage and heavier viewing can be configured as follows:



The Status table for the above configuration would look like this:



Configuration Notes:

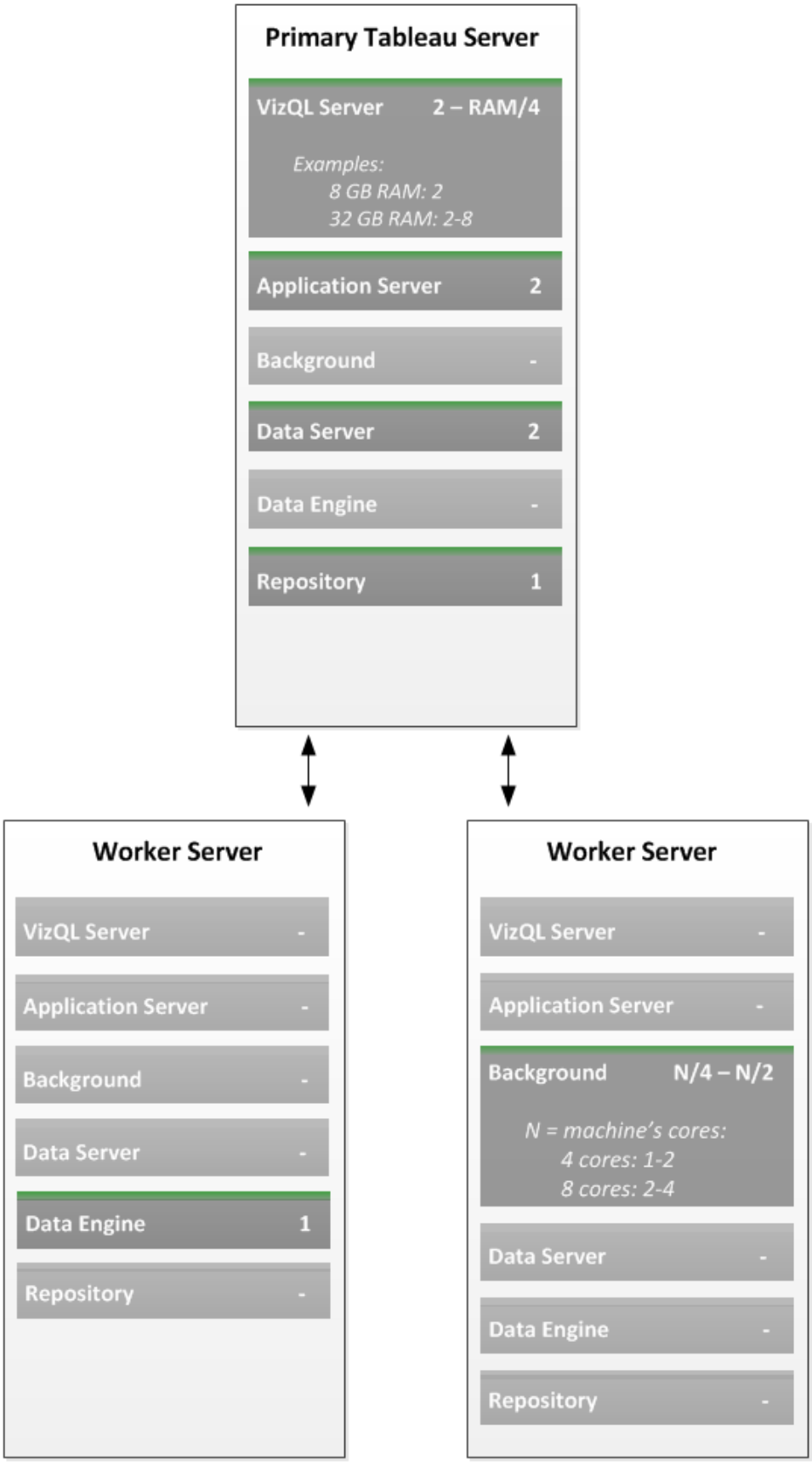
- Run at least 2 VizQL server processes on each ma2chine and determine the maximum number to run by dividing the machine's RAM by 4.
- A minimum of 2 background processes should be run on the primary Tableau Server. The maximum number you should run is equal to the machine's total number of cores.
- In a deployment where extracts are refreshed infrequently, the data engine and background processes can be on the same machine as the other processes.
- If extract refresh jobs will be only run during off hours, many background processes can be placed on each machine to maximize their parallelism.
- The number of machines in the cluster is solely determined how many VizQL Server

processes are needed to support the number of concurrent viewers.

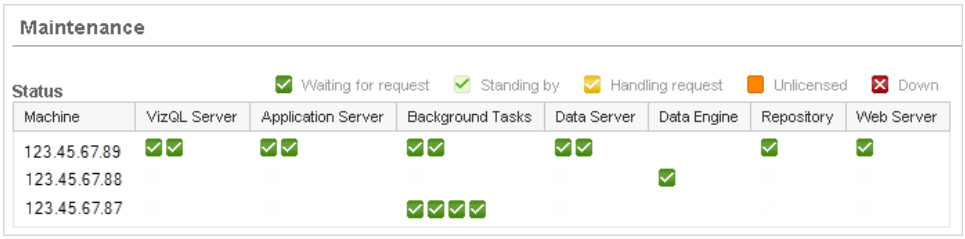
Each additional machine you add to support viewing can be configured like the second machine (the worker server) above.

Three-Machine Example: Extracts & Viewing

A three-machine configuration is the recommended minimum number of machines to achieve the best performance if you have both a high amount of extract refreshing and usage, and a high number of concurrent users. Each of these machines is running a 64-bit operating system:


Primary Tableau Server
VizQL Server 2 – RAM/4
Examples:
8 GB RAM: 2
32 GB RAM: 2-8
Application Server 2
Background -
Data Server 2
Data Engine -
Repository 1
Worker Server
VizQL Server -
Application Server -
Background -
Data Server -
Data Engine 1
Repository -
Worker Server
VizQL Server -
Application Server -
Background N/4 – N/2
N = machine's cores:
4 cores: 1-2
8 cores: 2-4
Data Server -
Data Engine -
Repository -

Here's the Status table for the above configuration:



Configuration Notes:

- Calculate the number of VizQL Server processes to run on the primary Tableau Server, by taking the machine's RAM and divide by 4.
- The background processes are on their own machine so that their work does not compete with that of the other processes. Because the machine is dedicated to background processes and they can consume 100% of the CPU resources, the low end of the suggested range equals the total number of cores. Depending on the size of the data being refreshed, it's possible for some deployments to run up to twice as many background processes than cores and still obtain parallel speed-up.
- Because the data engine process can consume all CPU resources on a machine, it is isolated on its own machine.
- The user loads for the application server and data server processes can typically be handled by 1 process each but they are set to 2 to provide redundancy.
- Under most conditions, the primary Tableau Server and the data engine will not be a bottleneck for the system's overall throughput as long as sufficient CPU cycles exist for them. To increase viewing capacity, add machines dedicated to the VizQL Server process. To increase capacity for refreshing extracts, add machines dedicated to the background process.

If you increase the number of VizQL instances and memory becomes an issue, there is a setting you can change. Refer to VizQL 'Out of Memory' Error for more information.

About Client-Side Rendering

Before a view's marks and data are displayed in a client web browser, they are retrieved, interpreted, and rendered. Tableau Server can perform this process in the client web browser or on the server. Client-side rendering is the default mode because handling the rendering and all interaction on the server can result in more network data transfer and round-trip delays. With client-side rendering, most view interactions are faster, because they are interpreted and rendered right there in the client.

Some views, however, are more efficiently rendered on the server where there's more computing power. Server-side rendering makes sense for a view that is complex to the extent that image files take up significantly less bandwidth than the data used to create the images. Also, because tablets usually have much slower performance than PCs, they can handle less view complexity. There are cases where a view opened from a PC's web browser might be client-rendered but the same view opened from a tablet's web browser is server-rendered.

Tableau Server is configured to automatically handle all of these situations using The Threshold Calculation as the trigger for rendering a view on the server instead of in the web browser. As the administrator, you can test or fine tune this setting for both PCs and tablets. See the topics below for more information.

Requirements

- **Supported browsers**: Client-side rendering is supported in Internet Explorer version 9.0 or higher, Firefox, Chrome, and Safari. All of these web browsers include the HTML 5 `<canvas>` element, which is used by client-side rendering.
- **Polygons, custom shapes, and the page history feature**: If a view uses polygons, custom shapes, or the page history feature, server-side rendering is performed, even if client-side rendering is otherwise enabled.

The Threshold Calculation

When client-side rendering is enabled, Tableau Server uses a calculation to determine the view's complexity. If the complexity value exceeds 100 (for PC browsers) or 20 (for tablet browsers), the view is rendered on the server instead of in the web browser. Here's the calculation:

```
(# of marks) + 3(# of headers) + 3(# of annotations) + 3(# of reference lines) = view complexity
```

For example, if you have a view with 2,000 marks, 150 headers (you can sometimes determine this by adding the number of rows and columns in a view), 1 annotation, and 1 reference line, your equation would be:

```
2,000 + 3(150) + 3(1) + 3(1) = 2,456
```

Now take the current threshold value and divide it by 100, then multiply it by 5,000 (dividing the threshold by 100 is a normalization and multiplying by 5,000 is Tableau's scaling factor). Assuming a current threshold value of 100, the equation would be as follows:

```
100/100 * 5,000 = 5,000
```

Compare the two sums. Knowing that 5,000 represents a complexity of 100, you can see that 2,456 represents about half the complexity (49). Therefore, to force server-side rendering for this particular view on a PC browser, you would need to set that threshold to 48. Keep in mind that interactions such as filtering may change the complexity of the view, and a session may switch rendering modes whenever the view's complexity changes.

See the topics below for details on how to test and configure client-side rendering.

Test with the URL Parameter

Tableau Server is configured to perform client-side rendering by default, as long as the requirements are met. To test server-side rendering on a session basis, type `?:render=false` at the end of the view's URL. For example:

```
http://localhost/views/Supplies/MyView?:render=false
```

If client-side rendering is disabled on Tableau Server, enter `?:render=true` to enable it for the session:

```
http://localhost/views/Supplies/MyView?:render=true
```

You can also test particular complexity thresholds on individual views to see if it's appropriate to adjust the server-wide threshold for your server and network conditions. For example, you may find that lower complexity (such as 80) or higher complexity (such as 120) tipping points result in more responsiveness to user interactions. To test a threshold, you can keep the server's default configuration (client-side-rendering enabled) and enter the test threshold number at the end of the view's URL. For example:

```
http://localhost/views/Supplies/MyView?:render=80
```

Configure with the tabadmin set Options

You can use the tabadmin options `vizqlserver.browser.render` to disable or enable client-side rendering and `vizqlserver.browser.render_threshold` and `vizqlserver.browser.render_threshold_mobile` to change the thresholds for client-side rendering. See tabadmin set options for details.

Create a Performance Recording

With the Performance Recording feature in Tableau, you can record performance information about key events as you interact with workbooks. You then view performance metrics in a performance workbook that Tableau creates automatically. The steps you follow to create and view performance recording vary somewhat between Tableau Desktop and Tableau Server. However, the resulting performance workbooks have the same format in both Tableau Desktop and Tableau Server.

Use performance workbooks to analyze and troubleshoot performance issues pertaining to different events that are known to affect performance, including:

- Query execution
- Geocoding
- Connections to data sources
- Layout computations
- Extract generation
- Data blending
- Server rendering

Tableau support may request that you create performance workbooks as they assist you with diagnosing performance issues.

To Create a Performance Recording in Tableau Server

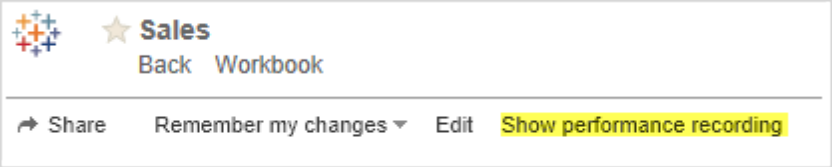
The server administrator determines whether to enable performance recording at the site level. By default, performance recording is not enabled in the default site, or in any site you create. To enable performance recording for a site, follow these steps:

1. Choose the Admin button in Tableau Server.
2. Choose Site.
3. Select a site.
4. Choose Edit.
5. In the Edit Site dialog box, select Allow Performance Recording.
6. Choose OK.

You start performance recording for a specific view by adding ?:record_performance=yes to the url. For example:

http://localhost/views/Variety/BaseballStatistics?:record_performance=yes

The visual confirmation that recording has started is a **Show Performance Recording** command in the toolbar:



Choose **Show Performance Recording** to open a performance workbook, which is an up-to-the-minute snapshot of performance data. You can continue taking additional snapshots as you continue working with the view; the performance data is cumulative. Once you page away from the view, or remove ?:record_performance=yes from the url, recording stops.

Interpret a Performance Recording

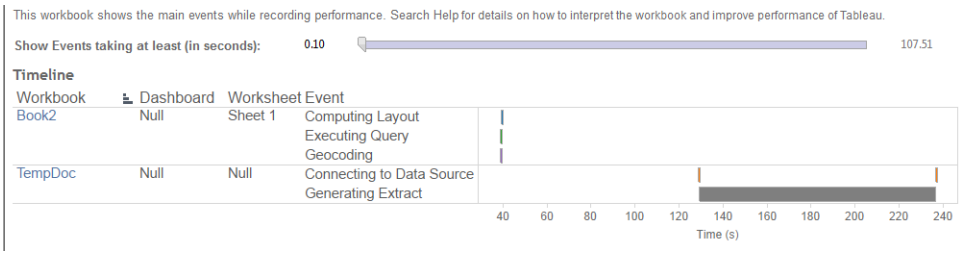
A performance recording workbook is a Tableau dashboard that contains three views: **Timeline**, **Events**, and **Query**.

For information on how to create a performance recording in Tableau Server, see Create a Performance Recording.

Timeline

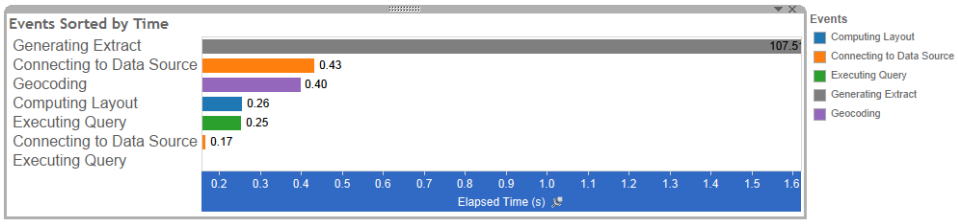
The uppermost view in a performance recording dashboard shows the events that occurred during recording, arranged chronologically from left to right. The bottom axis shows elapsed time since recording began.

In the Timeline view, the **Workbook**, **Dashboard**, and **Worksheet** columns identify the context for events. The **Event** column identifies the nature of the event, and the final column shows each event's duration and how it compares chronologically to other recorded events:



Events

The middle view in a performance recording workbook shows the events, sorted by duration (greatest to least). This can help you identify where to look first if you want to speed up your workbook.



Different colors indicate different types of events. The range of events that can be recording is:

- Computing layouts.

  If layouts are taking too long, consider simplifying your workbook.
- Connecting to data source.

  Slow connections could be due to network issues or issues with the database server.
- Executing query.

  If queries are taking too long, consult your database server's documentation.
- Generating extract.

  To speed up extract generation, consider only importing some data from the original data source. For example, you can filter on specific data fields, or create a sample based on a specified number of rows or percentage of the data.
- Geocoding.

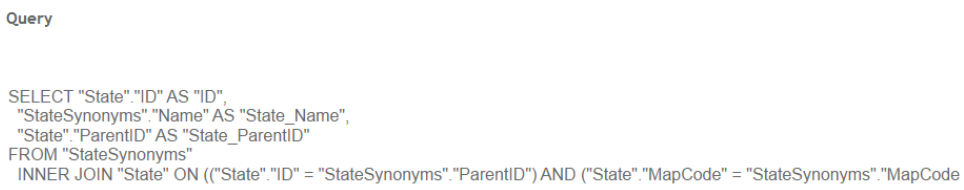
  To speed up geocoding performance, try using less data or filtering out data.
- Blending data.

  To speed up data blending, try using less data or filtering out data.
- Server rendering.

  You can speed up server rendering by running additional VizQL Server processes on additional machines.

Query

If you click on an **Executing Query** event in either the **Timeline** or **Events** section of a performance recording dashboard , the text for that query is displayed in the Query section. For example:



Sometimes the query is truncated and you'll need to look in the Tableau log to find the full query. Most database servers can give you advice about how to optimize a query by adding indexes or other techniques. See your database server documentation for details.

Embed Views

You can embed views from Tableau Server into webpages, blogs, wikis, web applications, and intranet portals. The embedded views blend seamlessly into your webpages and are interactive. The views update as the underlying data changes or the workbooks are updated on the server. Embedded views follow the same licensing and permission restrictions used on the server. Generally, people loading a webpage with an embedded view must also have an account on Tableau Server. If you have a core-based license you can alternatively select Enable Guest, which allows users to load the view without logging in. There are three ways you can embed views:

- **Use the Share embed code as-is:** The Share link in the upper left corner of each view provides automatically-generated embed code. All you have to do is copy the code and paste it into your webpage.
- **Write your own embed code:** You can enhance the default embed code Tableau provides or you can build your own code. Either way you can add parameters that control the toolbar, tabs, and more.
- **Use the Tableau JavaScript API:** You can use Tableau JavaScript objects in your own web application code. See JavaScript API for details.

> For users to successfully authenticate when they click an embedded view, their browsers must be configured to allow third-party cookies.

See the following topics for details on embedding views:

Writing Embed Code

If you're writing your own embed code, you can take one of two approaches:

- **Use Tableau JavaScript:** This is the preferred approach. Just use the Share embed code as the starting point for your own code, adding or editing object parameters that control the toolbar, tabs, and more. The default embed code, which relies on a Tableau JavaScript file, is also the only way to control the load order of multiple embedded views.
- **Specify the View URL:** As with earlier versions of Tableau, you can embed a view using an Iframe or Image tag, where the source is the raw URL for the view. You may want to do this if you can't use JavaScript at your web site. There may also be situations where all you can specify is an URL—such as if you're embedding a view using SharePoint's Page Viewer Web Part.

Regardless of the approach you take, you must define a width and height if you are embedding a view.

Tableau JavaScript

Here's an example of the embed code you get by default when you click Share:

```
<script type="text/javascript" src="http-
://myserver/javascripts/api/viz_v1.js"></script>
<div class="tableauPlaceholder" style="width:800; height:600;">
<object class="tableauViz" width="800" height="600" style="d-
isplay:none;">
   <param name="host_url" value="http://myserver/" />
   <param name="site_root" value="/t/Sales" />
   <param name="name" value="-
MyCoSales/SalesScoreCard/jsmith@myco.com/EastCoastSales" />
   <param name="tabs" value="yes" />
   <param name="toolbar" value="yes" /></object></div>
```

The source for the `<script>` tag is the URL for the Tableau Server JavaScript file, viz_v1.js. The JavaScript file handles assembling the full URL of the view that's displayed for your users. The `name` and `site_root` object parameters are the only required parameters; all other parameters are optional. For examples, see the List of Embed Parameters and the "Script Tag Examples" in the Examples section.

View URL as the Source

Here's an example of embedding the same view using an IFrame, where the source is the URL for the view:

```
<iframe src="http-
://myserver/t/Sale-
s/MyCoSales/SalesScoreCard?:embed=yes&:tabs=yes&:toolbar=yes"
width="800" height="600"></iframe>
```

You must specify the `embed` URL parameter and can optionally include parameters that control the toolbar and revert options, among others. You can also add filters to the URL that

control the specific data that shows when a view is loaded. For examples, see the List of Embed Parameters and the "Iframe Tag Examples" in the Examples section.

List of Embed Parameters

You can embed a view using either an Iframe tag, which uses URL parameters, or a JavaScript tag, which uses object parameters. The following table lists both sets of parameters and how to use them:

Object Parameter	URL Parameter	Values	Description	Examples
customViews	:customViews	no	Hides the "Remember my changes" option.	<param name="customViews" value="no"/>http://-tabserver/views/Date-Time/Date-Calcs?:embed=yes&:customViews=no
-	:embed	yes	Required for URL parameter. Hides the top navigation area, making the view blend into your web page better.	http://tabserver/views/Date-Time/Date-Calcs?:embed=yes
filter	-	string	Customizes what is displayed when the view opens.	<param name="filter" value="Team=Blue"/>

Object Parameter	URL Parameter	Values	Description	Examples
			Filtering by URL parameters is also possible.Refer to the Iframe tag examples in Add Filters and Filter on Multiple Fields.	
	:format	pdf; png	Displays a view as a PDF or .png file.	http://-tabserver/views/Sales/Q2?:format=pdf
host_url	-	string	The server name as it appears in the URL.	<param name="host_url" value="http-://myserver.bigco.com/"> <param name="host_url" value="http-://localhost/">
linktarget	:linktarget	string	The target window name for external hyperlinks.	<param name="linktarget" value="_blank"/> http://tabserver/views/Date-Time/Date-Calcs?:embed=yes&:linktarget=_blank
load-order	-	number	When multiple views	<param name="load-order" value="2"/>

Object Parameter	URL Parameter	Values	Description	Examples
			are embedded, the default load order is the order in which the views are listed. Use this setting to override that order. Negative numbers are allowed.	
name	-	string	Required for object parameter. Workbook and sheet name and optionally, a custom view (username@domain/	<param name="name" value="-MyCoSales/Sales"/> <param name="name" value="-MyCo-Sales/Sales/jsmith@myco.com/EastCoastSales"/>

Object Parameter	URL Parameter	Values	Description	Examples
			[custom view name]).	
path	-	string	For trusted authentication only, cannot be used with the "ticket" parameter. Overrides value of the "name" parameter and is used as the URL. See the Trusted Authentication examples.	<param name="path" value="trusted/123456789/-views/workbookQ4/SalesQ4"/> http://-tab-leauserver/trusted/123456789/views/workbookQ4/SalesQ
-	:refresh		Rerenders the page. See Refresh Data for details.	http://tabserver/views/Date-Time/Date-Calcs?:embed=yes&:refresh
-	:revert	all;	Returns the item	http://tabserver/views/Date-Time/Date-Calcs?:embed=yes&:revert=all

Object Parameter	URL Parameter	Values	Description	Examples
		filters; sorts; axes; shelves	to its original state.	
site_root	-	string	Required. The site name. The Default site value is null (value=""). If your server is multi-site and you want to use trusted authentication, see the Trusted Authentication examples.	<param name="site_root" value="/t/Sales"/> <param name="site_root" value=""/>
tabs	:tabs	yes;	Displays or hides	<param name="tabs" value="yes"/>

Object Parameter	URL Parameter	Values	Description	Examples
		no	tabs.	
ticket	-	number	For trusted authentication only, cannot be used with the "path" object parameter. Must be used with "name" object to construct the trusted ticket redemption URL. See the Trusted Authentication examples.	<param name="ticket" value="123456789"/> http://-tab-leauserver/trusted/123456789/views/workbookQ4/SalesQ
toolbar	:toolbar	yes; no; top	The toolbar is displayed by default on the bottom when	<param name="toolbar" value="top"/> http://tabserver/views/Date-Time/Date-Calcs?:embed=yes&:toolbar=no

Object Parameter	URL Parameter	Values	Description	Examples
			this parameter is not set. When `no` the toolbar is excluded from the embedded view. When `top`, the toolbar is placed above the view.	

Examples

Here are some examples of ways you can customize or work with your embed code:

Add Filters

You can pass filter values so the view opens showing just the data you want. For example, you may want to include a hyperlink from another part of your web application to an embedded sales performance view that only shows a specific region.

Script Tag Example

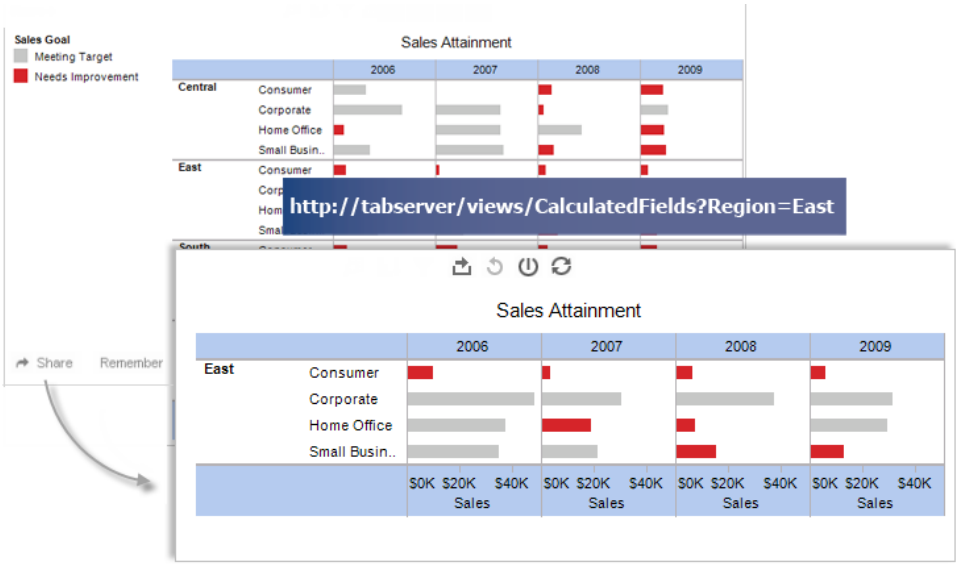
```
<script type="text/javascript" src="http-
://myserver/javascripts/api/viz_v1.js">
</script>
<object class="tableauViz" width="800" height="600" style="d-
isplay:none;">
   <param name="host_url" value="http://myserver/" />
   <param name="site_root" value="" />
   <param name="name" value="Sales/Sales-Performance" />
   <param name="filter" value="Region=East" />
</object>
```

To pass through multiple filters, just separate each value with a comma. For example:

```
<param name="filter" value="Region=East,West" />
```

Iframe Tag Examples

```
<iframe src="http-
://myserver/views/-
CalculatedFields?:embed=yes&Region=East"width="800"
height="600"></iframe>
<iframe src="http://myserver/views/Sales/Sales-Per-
formance?:embed=yes&Region=East,West" width="900px"
height="700px"></iframe>
```



Filter on Multiple Fields

You can pass filters on as many fields as you want, including fields that are not in the original view.

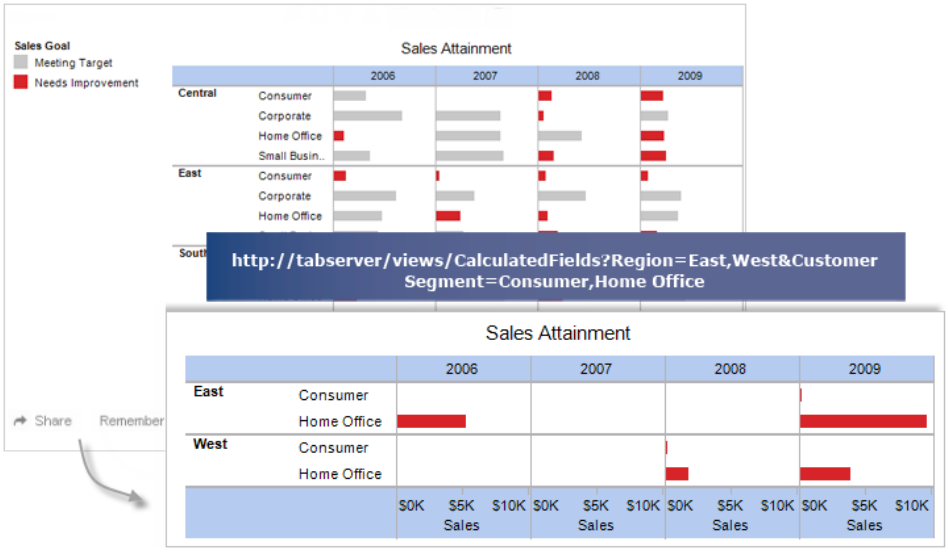
Script Tag Example

```
<script type="text/javascript" src="http-
://myserver/javascripts/api/viz_v1.js">
</script>
<object class="tableauViz" width="800" height="600" style="d-
isplay:none;">
   <param name="host_url" value="http://myserver/" />
   <param name="site_root" value="" />
   <param name="name" value="Sales/Sales-Performance" />
   <param name="filter" value="Region=East,West&Customer Seg-
```

```
ment=Consumer,HomeOffice" />
</object>
```

Iframe Tag Example

```
<iframe src="http-
://myserver/views/CalculatedFields?:embed=yes&Region=East,West&Cu-
stomer Segment=Consumer,Home Office" width="800"
height="600"></iframe>
```



If a filter value contains a special character, such as a comma, replace the character with the URL encoding sequence for \ (backslash, `%5c`) followed by the URL encoding sequence for the special character. The backslash is needed to escape the special character. For example, the URL encoding sequence for `\,` (backslash, comma) is `%5c%2c`.

Filter Dates and Times

If you want to filter on a Date/Time field, include the value using the default Tableau format shown below:

```
yyyy-mm-dd hh:mm:ss
```

The time part uses a 24-hour clock. Many databases store all date values as Datetime fields, so you may need to pass a time value along with your date.

Script Tag Example

```
<script type="text/javascript" src="http-
://myserver/javascripts/api/viz_v1.js"></script>
<object class="tableauViz" width="800" height="600"
```

```
style="display:none;">
   <param name="host_url" value="http://myserver/" />
   <param name="site_root" value="" />
   <param name="name" value="Sales/Sales-Performance" />
   <param name="filter" value="Date=2012-12-01" />
</object>
```

This example filters on both a date field and a datetime field:

```
<param name="filter" value="2012-12-01%2022:18:00" />
```

Iframe Tag Example

```
<iframe src="http://myserver/Sales/SalesPer-
formance?:embed=yes&Date=2008-12-01%2022:18:00" width="800"
height="600"></iframe>
```

To filter multiple dates, separate each date with a comma.

Filter Measures

You can filter measures by including one or more values. There is no support for greater than, less than, or ranges. The example below filters to show only $100 and $200 sales.

Script Tag Example

```
<script type="text/javascript" src="http-
://myserver/javascripts/api/viz_v1.js">
</script>
<object class="tableauViz" width="800" height="600  "style="d-
isplay:none;">
   <param name="host_url" value="http://myserver/" />
   <param name="site_root" value="" />
   <param name="name" value="Sales/Sales-Performance" />
   <param name="filter" value="Profit=100, 200" />
</object>
```

Iframe Tag Example

```
<iframe src="http://myserver/Sales/Sales-Per-
formance?:embed=yes&Profit=100,200" width="800"
height="600"></iframe>
```

Control the Load Order of Multiple Views

You can control the order in which multiple views load for the people working with your views. This feature can only be accessed using embed code that relies on the Tableau JavaScript file.

In the following example, two views are embedded. The second view loads first, followed by the top view. If you embed multiple views and give them all the same load order value, or if you don't specify load order parameters, they are loaded in the order in which they appear on the page.

Script Tag Example

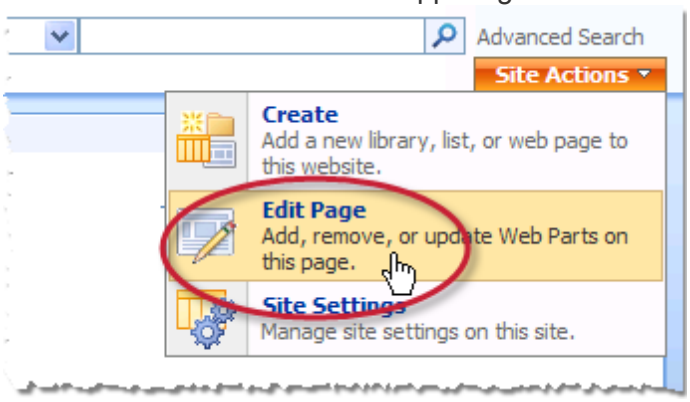
```
<script type="text/javascript" src="http-
://myserver/javascripts/api/viz_v1.js">
</script>
<object class="tableauViz" width="600" height="400" style="d-
isplay:none;">
   <param name="host_url" value="http://myserver/" />
   <param name="site_root" value="" />
   <param name ="name" value="MyCoSales/TopPerformers" />
   <param name="tabs" value="yes" />
   <param name="toolbar" value="yes" />
   <param name="filter" value="Salesperson=Top 5" />
   <param name="load-order" value="0" />
</object>
<script type="text/javascript" src="http-
://myserver/javascripts/api/viz_v1.js">
</script>
<object class="tableauViz" width="600" height="400" style="d-
isplay:none;">
   <param name="host_url" value="http://myserver/" />
   <param name="site_root" value="" />
   <param name="name" value="MyCoSales/SalesScoreCard" />
   <param name="tabs" value="yes" />
   <param name="toolbar" value="yes" />
   <param name="load-order" value="-1" />
</object>
```

Embed Views into SharePoint (Microsoft SSPI)

If both Tableau Server and SharePoint are using Microsoft SSPI, you can embed views using the Page Viewer Web Part. Follow the steps below to embed a view into a SharePoint page.

1. Navigate to the SharePoint page that you want to embed a view into.

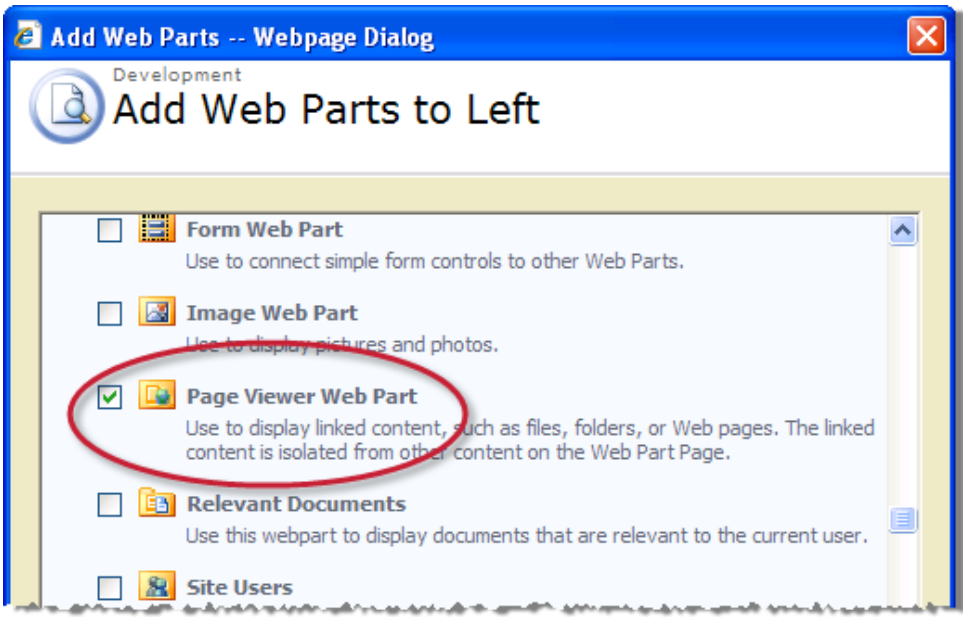
On the Site Actions menu in the upper right corner of the page select **Edit Page**.

2. 

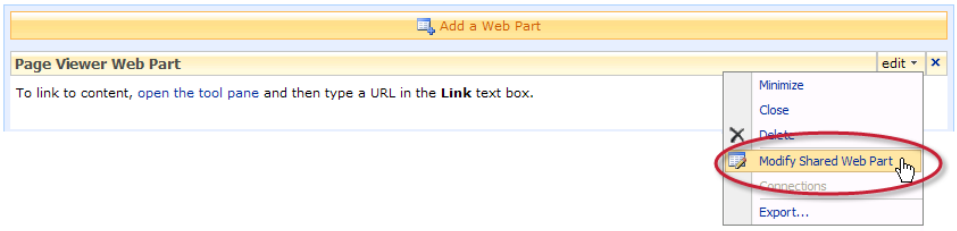
Click the **Add a Web Part** button in the section of the page where you want to embed the view.

3. 

On the page that opens, select the **Page Viewer Web Part** located in the Miscellaneous section and click **Add**.

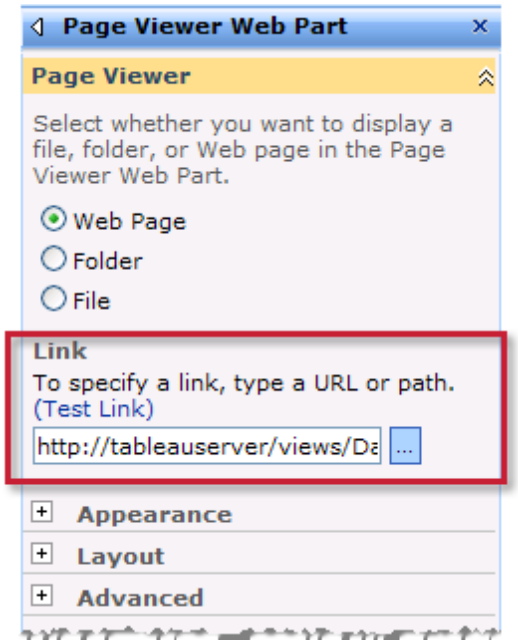
4. 

Back on the SharePoint page select **Modify Shared Web Part** on the Edit menu for the new web part.

5. 

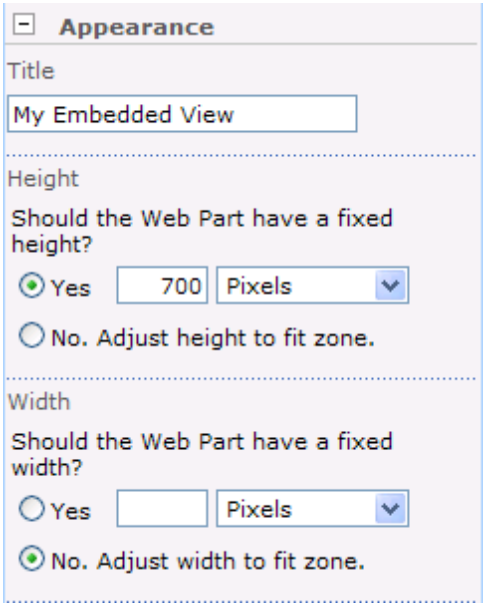
On the right side of the page, you can specify the attributes of the Page View Web Part. Type the URL for the view you want to embed. Use the format specified in Embed Views. For example you may type:

```
http://tableauserver/views/Date-Time/Date-
Calcs?:embed=yes&:toolbar=no
```

6. 

7. Then in the Appearance section you can specify a **Title** of the web part, the **Height**, and **Width**. In general you should specify a fixed height (e.g., 700 Pixels) and adjust the

width to fit to the zone.



8. Click **OK** to apply the changes and exit edit mode.

   The view will be embedded into the web part that you just created. Your users will not need to log in to Tableau Server to see the embedded view, rather they will be automatically authenticated using Microsoft SSPI.

Embed Views into Wikis

You can easily embed a view into a wiki or other web page simply by putting the view inside an `<iframe>` tag.

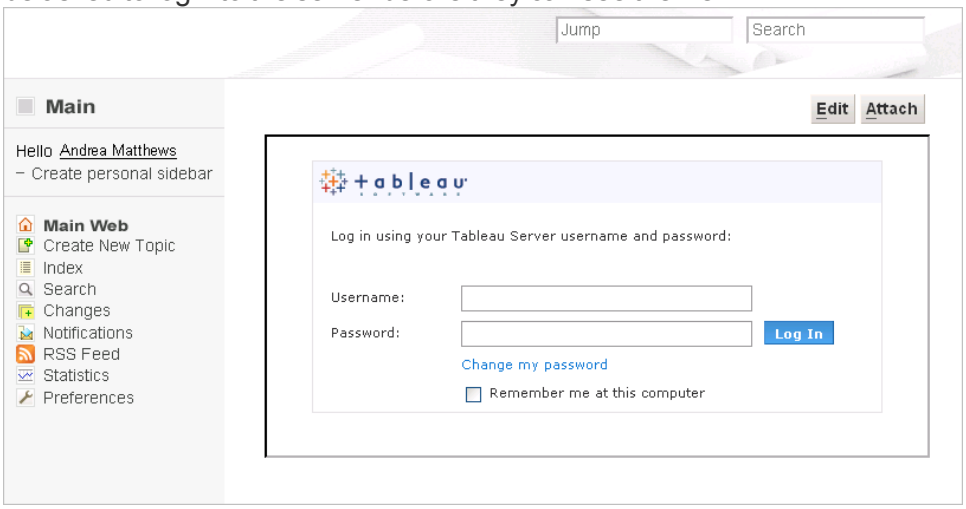
1. Navigate to the wiki page you want to embed a view into.
2. Edit the page and add an `<iframe>` where the source is the URL for the view. For example:

   ```
   <iframe src="http://tableauserver/views/Date-Time/Date-
   Calcs?:embed=yes&:toolbar=no" width="800"
   height="600"></iframe>
   ```

3. Save your changes.

   The view is embedded into the wiki page. If both Tableau Server and the wiki are configured to use Microsoft SSPI, users accessing an embedded view on the wiki will be automatically logged in so they can see the view.

If the server and the wiki are not using the same method for authentication, users will first be asked to log into the server before they can see the view.



Embed Images

In addition to embedding a view into a `<script>` or `<iframe>` tag you can also embed the view as an image. When you embed an image the view is not interactive, however, it is updated every time the page fully reloads. That way the image shows the latest data even if the underlying data changes.

1. Navigate to the page where you want to embed the image.
2. Edit the page and add an `<img>` tag where the source is the URL for the view plus the .png file extension. For example:

```
<img src="http://tableauserver/views/Date-Time/DateCalcs.png"
width="900" height="700">
```

Note:

If both the web page and Tableau Server are using Microsoft SSPI for authentication, then anyone accessing the embedded image will be automatically logged into Tableau Server and be able to see the view. However, if the server and the web page are not using the same authentication method, the image will not display.

Embed Views into SharePoint (Trusted Authentication)

If you are embedding a view into SharePoint but you don't use Microsoft SSPI for authentication, you can set up trusted authentication using the extra web part .dll installed with Tableau Server. Follow the instructions below to install the Tableau Web Part dll and embed a view into a SharePoint page.

1. Locate the TableauEmbeddedView.dll file that is installed with Tableau Server. The file is usually located in:

```
C:\Program Files\Tableau\Tableau Server\8-
.0\extras\embedding\sharepoint\
```

2. Copy the .dll file into the root directory of your SharePoint server. The root directory is usually located at:

   ```
   C:\Inetpub\wwwroot\wss\VirtualDirectories\<port>\bin
   ```

3. In a text editor, open the web.config file located at:

   ```
   C:\Inetpub\wwwroot\wss\VirtualDirectories\<port>\bin
   ```

4. Add the following text to the bottom of the SafeControl section:

   ```
   <SafeControl Assembly="TableauEmbeddedView, Version=1.0.0.0,
   Culture=neutral,PublicKeyToken=9f4da00116c38ec5" Names-
   pace="TableauEmbeddedView" TypeName="*" Safe="True" />
   ```

5. You also need to allow the webpart access to your SharePoint server. You can do this one of the following three ways:

   - Copy the TableauEmbeddedView.dll file into your C:\Windows\assembly folder and delete it from the bin file you copied it into in step 2 above.
   - Reopen the web.config file you opened in step 3 above and find the following line:

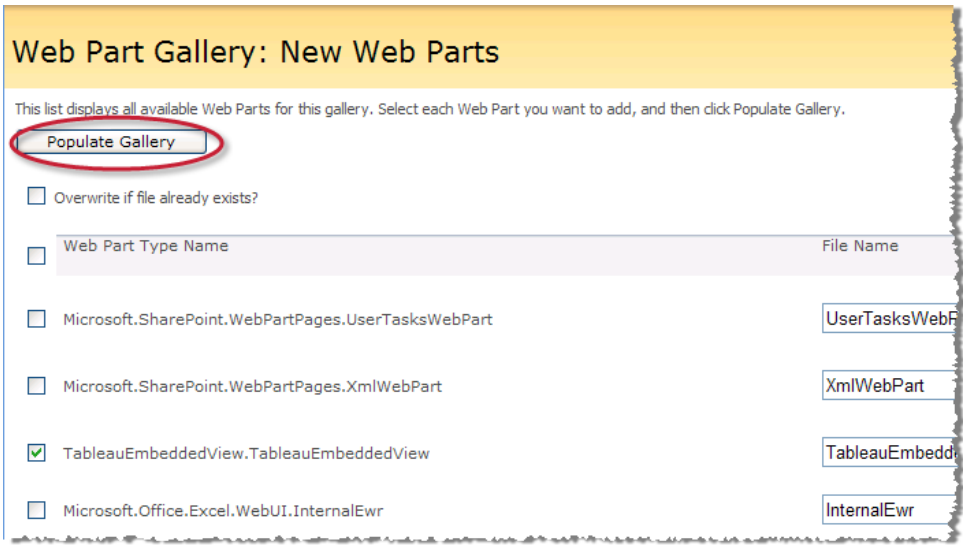
     ```
     <trust level="WSS_minimal" originUrl="" />
     ```

     Change the line above to the following:

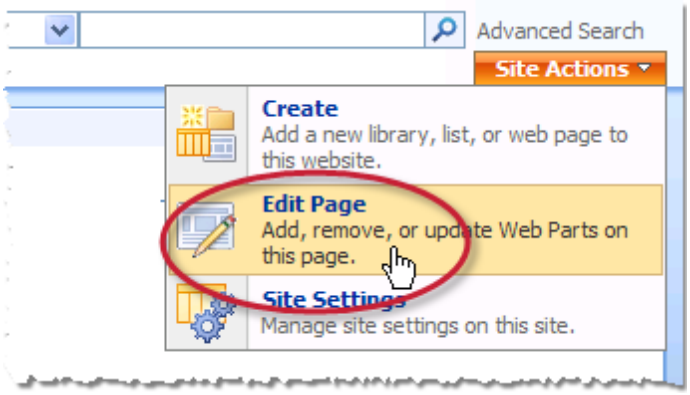
     ```
     <trust level="Full" originUrl="" />
     ```

   - Create a custom trust policy, which will grant full access to the TableauEmbeddedView.dll only. Refer to the Microsoft Technical Article to learn more about how to do this.

6. Open a browser and navigate to: http://<your_sharepoint_server>/_layouts/newdwp.aspx.
7. Select the entry titled TableauEmbeddedView.TableauEmbeddedView and click the **Populate Gallery** button.

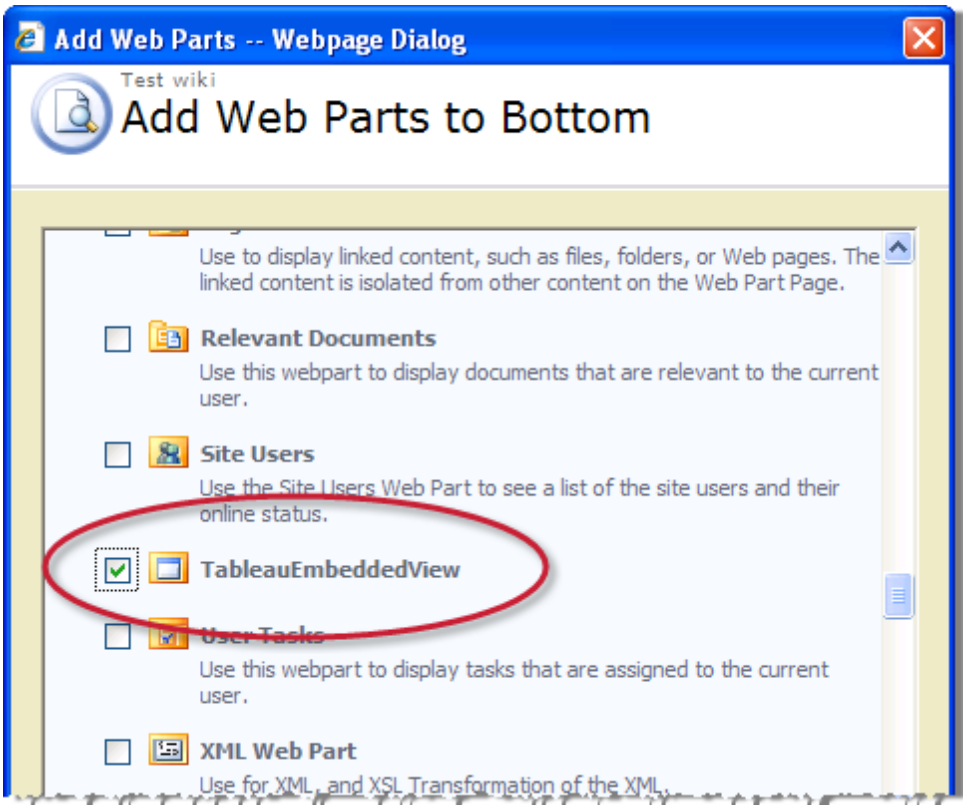


8. Navigate to the SharePoint page that you want to embed a view into.
9. On the Site Actions menu in the upper right corner of the page select **Edit Page**.

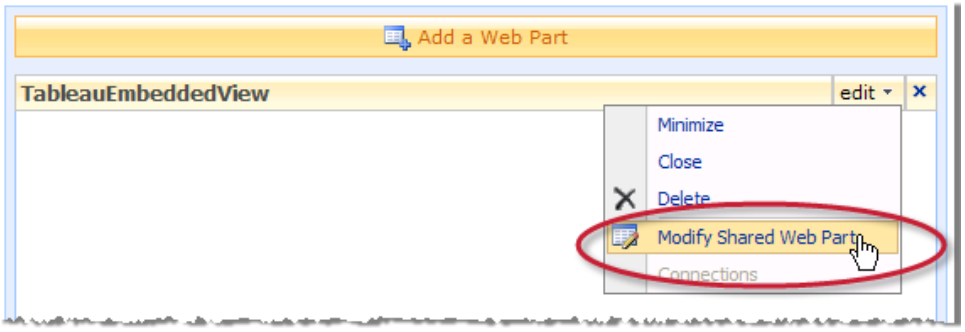


10. Click the **Add a Web Part** button in the section of the page where you want to embed the view.

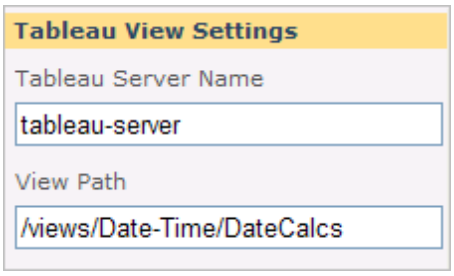
11. On the page that opens, select **TableauEmbeddedView** located in the Miscellaneous section and click **Add**.



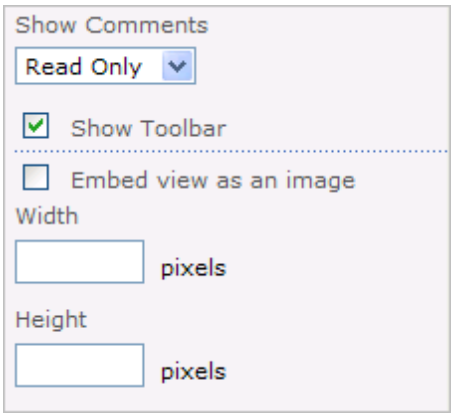
12. Back on the SharePoint page select **Modify Shared Web Part** on the Edit menu for the new web part.



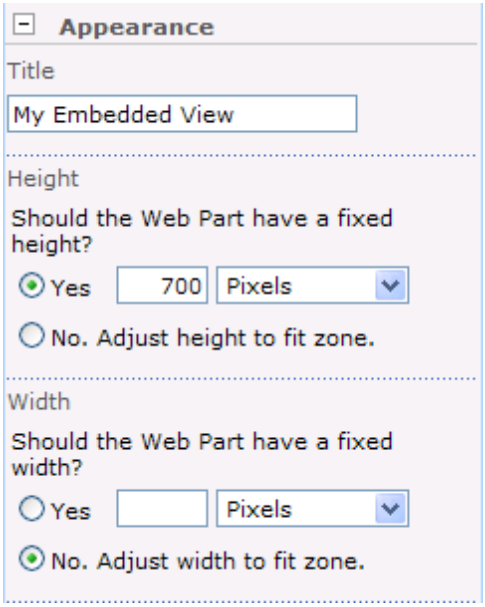
13. On the right side of the page, you can specify the attributes of the TableauEmbeddedView web part. Type the name of your Tableau Server, then type the path to the view you want to embed. For example you may type /views/Date-Time/Date-Calcs..



14. Specify other attributes such as whether you want to show the toolbar, or even if you want embed the view as an image instead of as an interactive view.



15. Then in the **Appearance** section you can specify a **Title** of the web part, the **Height**, and **Width**. In general you should specify a fixed height (e.g., 700 Pixels) and adjust the width to fit to the zone.



16. Click **OK** to apply the changes and exit edit mode.

Now the view is embedded in the page and users who access it will be automatically logged in based on their user name and password for SharePoint. Anyone who accesses an embedded view needs to be a licensed user on Tableau Server and their user name on SharePoint must be the same as their user name on Tableau Server.

> This is an example of embedding views into SharePoint using the provided .dll file. You can also embed views into other types of web applications and even build your own .dll file. See embed_api.htm for more information.

Proxy Servers

Tableau Server can be configured to work with a proxy server. In this type of environment, the proxy server acts as an intermediary between Tableau Server and the clients that are making requests for resources on Tableau Server. There are several ways to configure proxy servers—for example, as forward proxies or reverse proxies. These topics assume that you have already configured your proxy server, and now need to identify your proxy server to Tableau Server.

Use the topics below for more information:

Prepare to Configure for a Proxy Environment

To configure Tableau Server to work with a proxy server, you will need the following information about your proxy server:

- **IP address**: The IP address of the proxy server machine. The address must be in IPv4 format, for example, *123.45.67.89*.
- **FQDN**: The fully-qualified domain name of the proxy server. For example, *bigbox.myco.com*.
- **Non-FQDN**: Any non-fully-qualified domain names for the proxy server. Using the above example, the non-fully-qualified domain name of the proxy server would be *bigbox*.
- **Aliases**: Any aliases for the proxy server. Aliases are designated using CNAMEs (Canonical Name records). An example would be a proxy server with a CNAME of *bigbox.myco.com* and aliases of *ftp.myco.com* and *www.myco.com*.

Configure Tableau to Work with a Proxy Server

After you collect the information described in Prepare to Configure for a Proxy Environment, you can configure Tableau Server to work with a proxy by performing the following steps. For information on the settings below, see tabadmin set options.

1. Stop the server.
2. Still in the Tableau Server bin directory, enter the following command, where `name` is

the canonical (externally-visible) name of the proxy server:

```
tabadmin set gateway.public.host "name"
```

3. By default, Tableau assumes that the proxy server is listening on port 80 for external communications. To designate a different port, enter the following command, where `port_number` is the port:

   ```
   tabadmin set gateway.public.port "port_number"
   ```

4. Now, enter the following command, where `IP_address` is the IP address of the proxy server:

   ```
   tabadmin set gateway.trusted "IP_address"
   ```

   The value for IP_address can be a comma-separated list, for example:

   ```
   tabadmin set gateway.trusted "123.45.67.89, 123.45.67.88,
   123.45.67.87"
   ```

5. In the next command, you will provide any alternate names for the proxy server, such as its fully-qualified domain name, any non-fully-qualified domain names, and any aliases. These are the names a user might type in a browser. Separate each name with a comma:

   ```
   tabadmin set gateway.trusted_hosts "name1, name2, name3"
   ```

   For example:

   ```
   tabadmin set gateway.trusted_hosts "bigbox.myco.com, bigbox,
   ftp.myco.com, www.myco.com"
   ```

6. Start the server so the changes can take effect.

Trusted Authentication

When you embed Tableau Server views into webpages, everyone who visits the page must be a licensed user on Tableau Server. When users visit the page they are prompted to log into Tableau Server before they can see the view. If you already have a way of authenticating users on the webpage or within your web application, you can avoid this prompt and save your users from having to log in twice by setting up trusted authentication.

Trusted authentication simply means that you have set up a trusted relationship between Tableau Server and one or more web servers. When Tableau Server receives requests from these trusted web servers it assumes that your web server has handled whatever authentication is necessary.

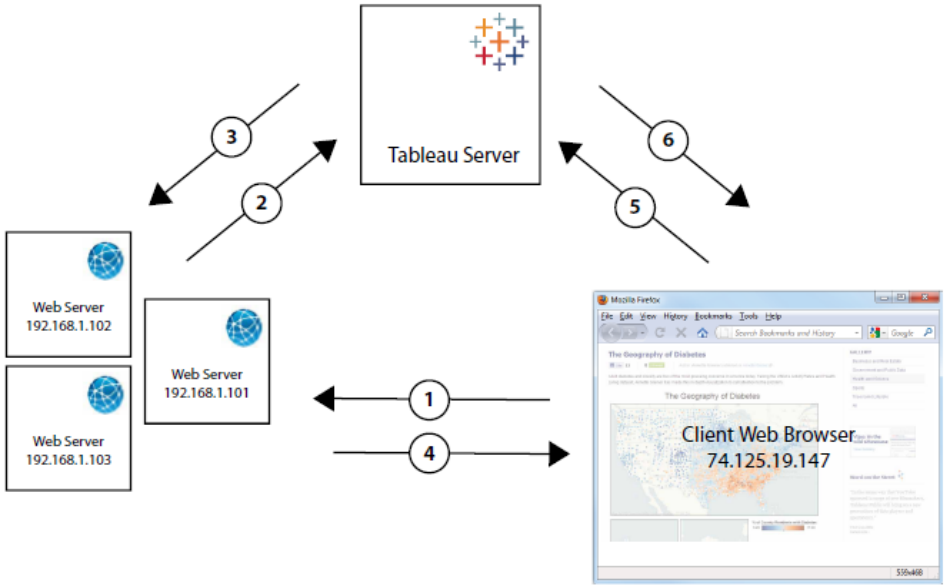
If your web server uses SSPI (Security Support Provider Interface), you do not need to set up trusted authentication. You can embed views and your users will have secure access to them as long as they are licensed Tableau Server users and members of your Active Directory. Using both Enable Automatic Login (an option configured during Setup that uses Microsoft SSPI) and trusted authentication is not supported. If you are not using SSPI with Active Direc-

tory and you want your users to have secure access to Tableau Server views without being prompted for credentials, you can set up trusted authentication.

> So that users can be authenticated when they click an embedded view, their browsers must be configured to allow third-party cookies.

How Trusted Authentication Works

The diagram below describes how trusted authentication works between the client's web browser, your web server(s) and Tableau Server.



1. **User visits the webpage**: When a user visits the webpage with the embedded Tableau Server view, it sends a GET request to your web server for the HTML for that page.

2. **Web server POSTS to Tableau Server**: The web server sends a POST request to the trusted Tableau Server (for example, `http://tabaserver/trusted`, not `http://tabserver`).

4. **Web server passes the URL to the browser**: The web server constructs the URL for the view using either the view's URL or its object tag (if the view's embedded), and inserts it into the HTML for the page. The ticket is included (e.g., http://tabserver/trusted/<ticket>/views/requestedviewname). The web server passes all the HTML for the page back to the client's web browser.

5. **Browser requests view from Tableau Server**: The client web browser sends a request to Tableau Server using a GET request that includes the URL with the ticket.

That POST request must have a `username` parameter. The `username` value must be the username for a licensed Tableau Server user. If the server is running multiple sites and the view is on a site other than the Default site, the POST request must also include a `target_site` parameter.

3 **Tableau Server creates a ticket**: Tableau Server checks the IP address of the web server (192.168.1.XXX in the above diagram) that sent the POST request. If it is set up as a trusted host then Tableau Server creates a ticket in the form of a unique nine-digit string. Tableau Server responds to the POST request with that ticket. If there is an error and the ticket cannot be created Tableau Server responds with a value of -1.

6 **Tableau Server redeems the ticket**: Tableau Server sees that the web browser requested a URL with a ticket in it and redeems the ticket. Tickets must be redeemed within three minutes after they are issued. Once the ticket is redeemed, Tableau Server logs the user in, removes the ticket from the URL, and sends back the final URL for the embedded view.

Add Trusted IP Addresses to Tableau Server

The first step in setting up trusted authentication is to configure Tableau Server to recognize and trust requests from one or more web servers:

1. Open a command prompt as an administrator and navigate to your Tableau Server bin directory (for example, C:\Program Files (x86)\Tableau\Tableau Server\8.0\bin).
2. Next, type the following command:

   ```
   tabadmin set wgserver.trusted_hosts "<Trusted IP Addresses>"
   ```

   In the command above, `<Trusted IP Addresses>` should be a comma-separated list of the IP addresses of your web server(s). For example:

   ```
   tabadmin set wgserver.trusted_hosts "192.168.1.101,
   192.168.1.102, 192.168.1.103"
   ```

Note:

The comma separated list should be within quotes with one space after each comma. Host names are not allowed.

3. If you have one or more proxy servers between the machine that is creating the trusted ticket (step 2, above) and Tableau Server, you also need to add them as trusted gateways. See Configure Tableau to Work with a Proxy Server for steps.
4. Finally, type the following command to restart the server:

   ```
   tabadmin restart
   ```

Next, you need to configure your web server to receive tickets from Tableau Server.

Get a Ticket from Tableau Server

After you've added trusted IP addresses to Tableau Server, you're ready to configure your web server to get tickets from Tableau Server via POST requests (step 3 in the diagram). The POST request must be sent to `http://<server name>/trusted`—for example `http://tabserv/trusted`, not `http://tabserv`.

For code examples that you can use to create the POST request in Java, Ruby, and PHP, see the following:

32-bit:`C:\Program Files\Tableau\Tableau Server\8.0\extras\embedding`
64-bit:`C:\Program Files (x86)\Tableau\Tableau Server\8-.0\extras\embedding`

Here's the data you can use in a POST request to Tableau Server:

- **`username=<username>`** (required): The username for a licensed Tableau Server user. If you are using Local Authentication the username can be a simple string (for example, `username=jsmith`). If you are using Active Directory with multiple domains you must include the domain name with the user name (for example, `username=MyCo\jsmith`).
- **`target_site=<site id>`** (required if view not on Default site): Specifies the site containing the view if Tableau Server is running multiple sites and the view is on a site other than the Default site (for example, `target_site=Sales`). The value you use for `<site id>` should be the site's Web Folder name.
- **`client_ip=<IP address>`** (optional): Used to specifiy the IP address of the computer whose web browser is accessing the view (for example, `client_ip=123.45.67.891`). It is not the IP address of the web server making the POST request of Tableau Server. If you decide to use this parameter, see Optional: Configure Client IP Matching for more information.

Tableau Server's response to the POST request will be a unique nine-digit string (the ticket). If Tableau Server isn't able to process the request, the return will be `-1`. See Ticket Value of -1 Returned from Tableau Server for tips on how to correct this. Also, in order for users to suc-

cessfully authenticate when they click an embedded view, their browsers must be configured to allow third-party cookies.

Next, you need to add code that allows the web server to construct an URL for the view that includes the view's location and the ticket.

Display the View with the Ticket

After you create the POST request, you need to write code that provides the web server with the view's location and the ticket from Tableau Server. It will use this information to display the view. How you specify it depends on whether the view is embedded, and if Tableau Server is running multiple sites.

Tableau Server View Examples

Here's an example of how to specify a view that users only access via Tableau Server (the view is not embedded):

```
http://tabserver/trusted/<ticket>/views/<workbook>/<view>
```

If Tableau Server is running multiple sites and the view is on a site other than the Default site, you need to add `t/<site name>` to the path. For example:

```
http://tabserver/trusted/<ticket>/t/Sales/views/<workbook>/<view>
```

Embedded View Examples

Here are some examples of how to specify embedded views. Because there are two approaches you can take with embed code, both ways are provided below. Regardless of which you use, there is some information unique to trusted authentication that you must provide.

Script Tag Examples

This example uses the ticket object parameter:

```
<script type="text/javascript" src="http-
://myserver/javascripts/api/viz_v1.js"></script>
<object class="tableauViz" width="800" height="600" style="d-
isplay:none;">
    <param name="name" value="MyCoSales/SalesScoreCard" />
    <param name="ticket" value="123456789" />
</object>
```

Here's what the above example looks like for a multi-site Tableau Server, where the view is published on the `Sales` site:

```
<script type="text/javascript" src="http-
://myserver/javascripts/api/viz_v1.js"></script>
```

```
<object class="tableauViz" width="800" height="600" style="d-
isplay:none;">
    <param name="site_root" value="/t/Sales" />
    <param name="name" value="MyCoSales/SalesScoreCard" />
    <param name="ticket" value="123456789" />
</object>
```

Instead of using `ticket`, you can use the path parameter to state the full path of the view explicitly. When `path` is used, you do not also need the `name` parameter, which is usually a required parameter in Tableau JavaScript embed code:

```
<script type="text/javascript" src="http-
://myserver/javascripts/api/viz_v1.js"></script>
<object class="tableauViz" width="900" height="700" style="d-
isplay:none;">
    <param name="path" value="tru-
sted/123456789/views/MyCoSales/SalesScoreCard" />
</object>
```

Here's the same example, but for a multi-site server. Note that `/t/<site name>` is used here:

```
<script type="text/javascript" src="http-
://myserver/javascripts/api/viz_v1.js"></script>
<object class="tableauViz" width="900" height="700" style="d-
isplay:none;">
     <param name="path" value="tru-
sted/123456789/t/Sales/views/MyCoSales/SalesScoreCard" />
</object>
```

Iframe Tag Example

```
<iframe src="http:-
//t-
abserver/trusted/123456789/views/workbookQ4/SalesQ4?:embed=yes"
width="800" height="600"></iframe>
```

Optional: Configure Client IP Matching

By default, Tableau Server does not consider the client web browser IP address when it creates or redeems tickets. To change this, you need to do two things: specify an IP address using the `client_ip` parameter in the POST request that obtains the ticket, and follow the steps below to configure Tableau Server to enforce client IP address matching.

1. Open a command window and change directories to the location of Tableau Server's bin directory. The default location is `C:\Program Files (x86)\Tableau\Tableau`

```
Server\8.0\bin
```

2. Open a command prompt as an administrator and type the following command:

   ```
   tabadmin set wgserver.extended_trusted_ip_checking true
   ```

3. Then type the following command:

   ```
   tabadmin configure
   ```

4. Finally, restart the server by typing the following:

   ```
   tabadmin restart
   ```

Troubleshoot Trusted Authentication

Below are some common issues and errors you might encounter when you're configuring trusted authentication. Trusted authentication information is written to `ProgramData\Tableau\Tableau Server\data\tabsvc\logs\vizqlserver\vizql-*.log`. To increase the logging level from `info` to `debug`, use the tabadmin setting vizqlserver.trustedticket.log_level.

For tips on testing trusted authentication, see the Tableau Knowledge Base.

Ticket Value of -1 Returned from Tableau Server

Tableau Server returns -1 for the ticket value if it cannot issue the ticket as part of the trusted authentication process. The exact reason for this message is written to the file `production*.log` in the following folder:

```
ProgramData\Tableau\Tableau Server\data\tabsvc\logs\wgserver
```

Here are some things to confirm:

- **All web server IP addresses are added to trusted hosts**

  The IP address for the machine sending the POST request must be in the list of trusted hosts on Tableau Server. See Add Trusted IP Addresses to Tableau Server to learn how to add IP addresses to this list.

- **Trusted hosts list is properly formatted**

  The list of trusted hosts on Tableau Server must be a comma-separated list with a space after each comma. For example, the list should be similar to the following: 192.168.1.101, 192.168.1.102, 192.168.1.103, and so on.

- **IP addresses are IPv4**

  The list of IP addresses must be in Internet Protocol version 4 (IPv4) format. An IPv4 address looks like this: 123.456.7.890. IPv6 addresses (for example, fe12::3c4a:5eab:6789:01c%34) are not supported.

- **Username in POST request is a valid Tableau Server user**

The username you send in the POST request must be a licensed Tableau Server user with a Viewer or Interactor license level. You can see a list of users and their license levels by logging into Tableau Server as an administrator and clicking the Licensing link on the left side of the page.

- **Username in POST request includes domain**

  If Tableau Server is configured to use Local Authentication, the username that you send in the POST can be a simple string. However, if the server is configured for Active Directory you must include the domain name with the user name (domain\username). For example, the username parameter might be: `username=dev\jsmith`

HTTP 401 - Not Authorized

If you receive a 401- Not Authorized error, you may have configured Tableau Server to use Active Directory with SSPI (see Enable Automatic Login). If your web server uses SSPI, you do not need to set up trusted authentication. You can embed views and your users will have access to them as long as they are licensed Tableau server users and members of your Active Directory.

Concurrent use of **Enable Automatic Login** and trusted authentication is not supported.

HTTP 404 - File Not Found

You may receive this error if your program code references a Tableau Server URL that does not exist. For example, your web server may construct an invalid URL that cannot be found when the webpage tries to retrieve it.

Invalid User (SharePoint or C#)

You may encounter this error if you've configured Tableau Server for trusted authentication.

The example code for the SharePoint .dll references the following GET request:

```
SPContext.Current.Web.CurrentUser.Name
```

The above request will return the display name of the current Windows Active Directory user. If you want to use the login ID, then you will need to change the code to:

```
SPContext.Current.Web.CurrentUser.LoginName
```

After you make the change, recompile the SharePoint .dll.

Attempting to Retrieve the Ticket from the Wrong IP Address

You may encounter this error if you've configured Tableau Server for trusted authentication.

The client web browser IP address is not considered by default when redeeming the ticket. If Tableau Server is configured to enforce client IP address matching, make sure that the client's web browser IP address that is sent in the POST to Tableau Server is the same as when the browser tries to retrieve the embedded view. For example, in the Trusted Authentication dia-

gram, if the POST request in step 3 sends the parameter client_ip=74.125.19.147, then the GET request in step 5 must come from that same IP address.

See Optional: Configure Client IP Matching to learn how to configure Tableau Server to enforce client IP address matching.

Cookie Restriction Error

When a user logs in to Tableau Server, a session cookie is stored in their local browser. The stored cookie is how Tableau Server maintains that the logged in user has been authenticated and can access the server. Because the cookie is set with the same domain or sub-domain as the browser's address bar, it is considered a first-party cookie. If a user's browser is configured to block first-party cookies, they will be unable to log in to Tableau Server.

When a user logs in to Tableau Server via an embedded view, or in an environment where trusted authentication has been configured, the same thing happens: a cookie is stored. In this case, however, the browser treats the cookie as a third-party cookie. This is because the cookie is set with a domain that's different from the one shown in the browser's address bar. If a user's web browser is set to block third-party cookies, authentication to Tableau Server will fail. To prevent this from occurring, web browsers must be configured to allow third-party cookies.

An error occurred communicating with the server (403)

If Tableau Server is configured for trusted authentication, you may receive this error after opening a new view in a browser and attempting to navigate back to views you'd opened earlier. Starting with version 8.0, Tableau Server provides protection against unauthorized reuse of VizQL sessions through the tabadmin set option vizqlserver.protect_session, which is set to `true` by default. Because Tableau Server is configured for trusted authentication, you may not also need to enable `vizqlserver.protect_session`. To disable it, use tabadmin set to change it to `false`.

Run As User

You can use a dedicated Active Directory (AD) user account for the Tableau Server service to run under, called a Run As User account. Some administrators choose to do this when published workbooks on Tableau Server connect to live data sources. The server's default Network Service account (NT AUTHORITY\NetworkService) doesn't have the correct permissions for connecting to data sources on other computers. A correctly configured AD account does.

For data sources that require NT authentication, the AD account can also automatically handle the authentication process, thus shielding users from prompts for credentials when the workbook connects to the live data source. Finally, a Run As User AD account that is dedicated to a specific resource is often less problematic to manage than an AD account associated with a person.

To configure Tableau Server to use a Run As User account, follow the procedures below. If you are running a distributed installation of Tableau Server, these steps should be performed

on workers as well as on the primary. Also note that the steps under Run As Account Settings to Confirm may vary from site to site.

Note:

If you are installing Tableau Server with your Run As User account in hand, before you run Setup, confirm that the Windows Secondary Login service has the correct values for Log On and Startup. See Verify Tableau Service Settings for more information.

Identify the Account

Your first step is to identify or create an Active Directory account for the Tableau Server service to run under. This will be the Tableau Server's Run As User account, and it should have the following:

- Permissions for connecting to the data source with at least read access.
- Credentials to allow Tableau Server to satisfy the NT authentication process with the data source. Microsoft data sources that perform NT authentication include Microsoft SQL Server and Microsoft Analytical Services (MSAS), but not Access or Excel.
- Permissions to query your Active Directory domain controller for users and groups. A user account created on the local machine that Tableau Server is running on probably won't have these permissions.

Confirm Domain Two-Way Trust

Confirm that there is a two-way trust between domains if any of the following are true:

- The machines hosting the Tableau Server and the data source are on separate domains.
- Tableau Server users are on a separate domain from Tableau Server or the data source.

Verify Tableau Service Settings

Confirm that Tableau services are assigned the correct Log On and Startup values. If you are running a distributed installation of Tableau Server, perform these steps on the workers as well as on the primary.

1. Log on as administrator to the computer running Tableau Server.
2. On the Tableau Server computer, select **Start > Control Panel > Administrative Tools > Computer Management > Services and Applications > Services**.
3. Open Services and Applications, then click **Services**. Confirm that the following services have the correct settings:

Service Name	Logon Value	Startup Value
FLEXnet Licensing Service	Local System	Manual
Secondary Login	Local System	Automatic
Tableau Server (tabsvc)	<domain>\<username> This is the Run As user account. See below.	Automatic
Tablicsrv	Local System	Automatic

Changing the Log On Value

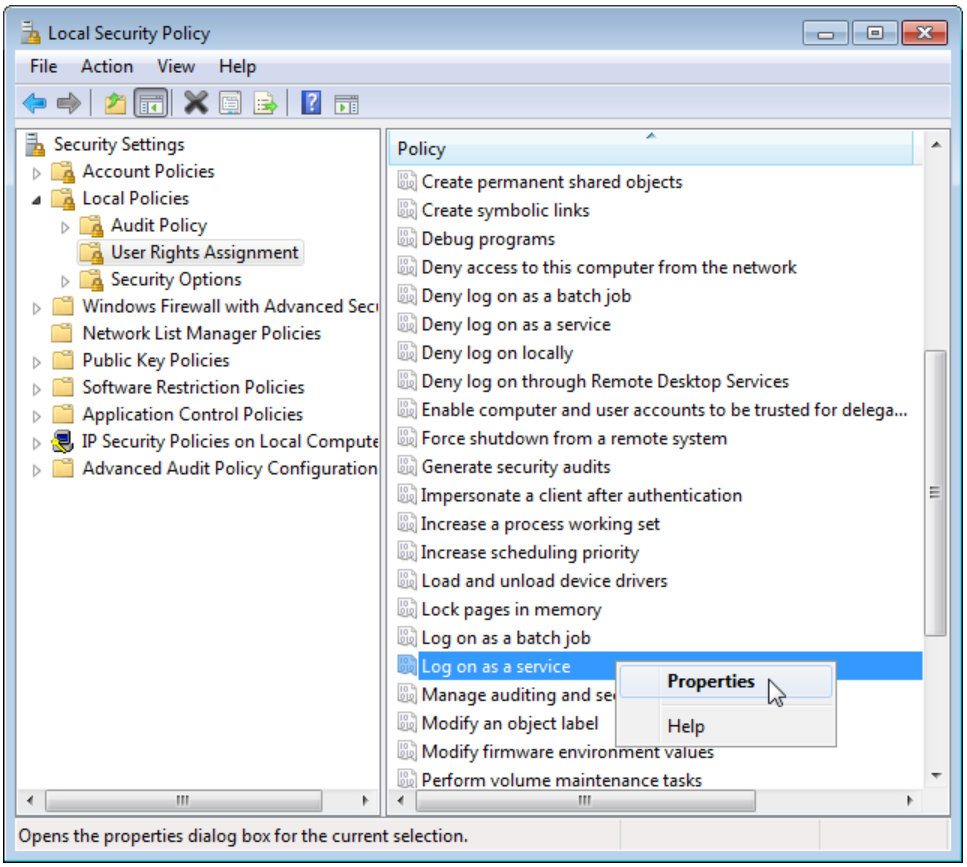
To change the **Log On** value for Tableau Server (tabsvc) to the Run As User account:

1. In the Services window, stop the Tableau Server service by right-clicking Tableau Server (tabsvc) and selecting **Stop**.
2. Select **Start > All Programs > Tableau Server > Configure Tableau Server**.
3. On the General tab, enter the domain, username, and password for Tableau Server's Run As User account.
4. Click **OK**, then restart Tableau Server (tabsvc).

Prepare the Local Security Policy

If your Run As User account isn't an administrator on the Tableau Server machine (primary and workers, if you're running a distributed installation), you must prepare the machine's local security policy so that the Tableau Server Run As User account can log onto the machine as a service and make configuration changes. To do this:

1. Select **Start > Control Panel > Administrative Tools > Local Security Policy**.
2. In the Local Security Settings window, open Local Policies, highlight User Rights Assignments, then right-click **Log on as a service** and select **Properties**.



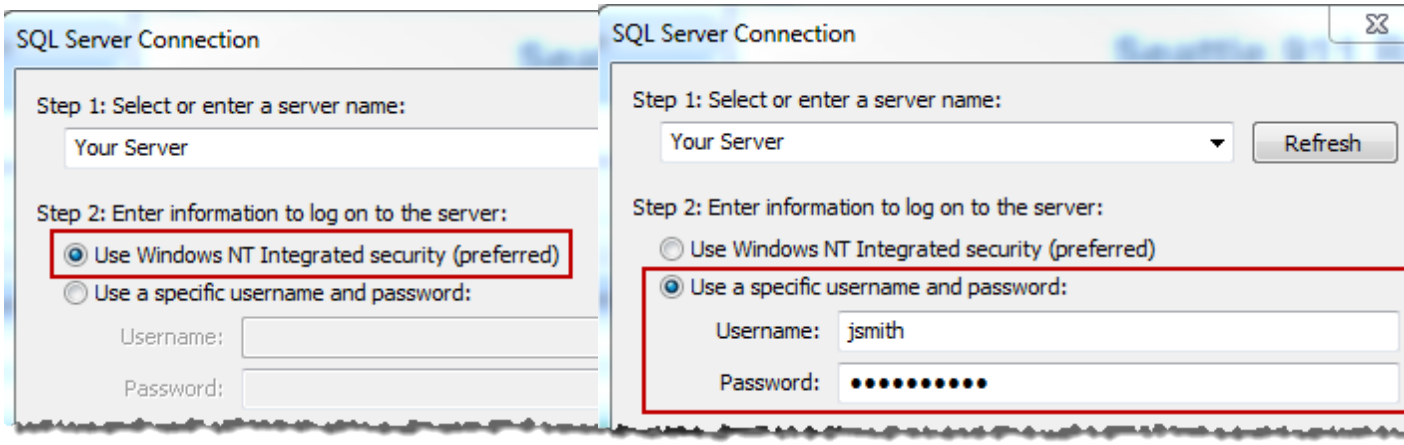
3. In the Log on as a service Properties window, click **Add User or Group**.
4. Type the `<domain>\<username>` for the Tableau Server Run As User account (for example: `MYCO\tableau_server`), and click **Check Names**.
5. When the account resolves correctly, it is underlined. Click **OK**.
6. Repeat these steps to add the Run As account to the **Log on Locally** policy.
7. Repeat these steps to remove the Run As account from the **Deny logon** policy.
8. Click **OK** to close the Local Security Settings windows.

Configure Data Source Connection Settings

To automatically authenticate your users when the workbook they're accessing connects to a live, NT-authenticated data source, configure your Tableau data connection with the **Use Windows NT Integrated security** option selected:

Windows NT Integrated Security	Username and Password
Authenticates with the server's Run As User account	Each Tableau Server user is prompted for database credentials

Windows NT Integrated Security **Username and Password**



Run As Account Settings to Confirm

The Run As User account needs permissions that allow it to read, execute, and sometimes modify files. Depending on the account you used as a starting point, it may already have the correct permissions. Any time you change the server's Run As account you should confirm that it meets the following requirements. If you're running a distributed installation, this applies to both the primary and workers.

Grant Read and Execute Permissions

The account the Tableau Server service runs under needs permission to read and execute files. Any time the server's Run As User account is changed, confirm or configure the following:

1. On the machine hosting Tableau Server (and Tableau Worker, if distributed), use Windows Explorer to right-click on **Local Disk (C:)** and select **Properties**.
2. In the Local Disk (C:) Properties Window, select the **Security** tab.
3. Click **Edit**, then **Add**.
4. In the Select Users, Computers, Service Accounts, or Groups dialog box, type the `<domain>\<username>` for the Tableau Server Run As User account. Do not use a group account.
5. Click **Check Names** to resolve the account, then **OK** to confirm.
6. With the Tableau Server Run As User account highlighted, confirm that it has **Read & execute** permissions. Selecting **Read & execute** automatically selects **List folder contents** and **Read**.
7. Click **OK** to exit.

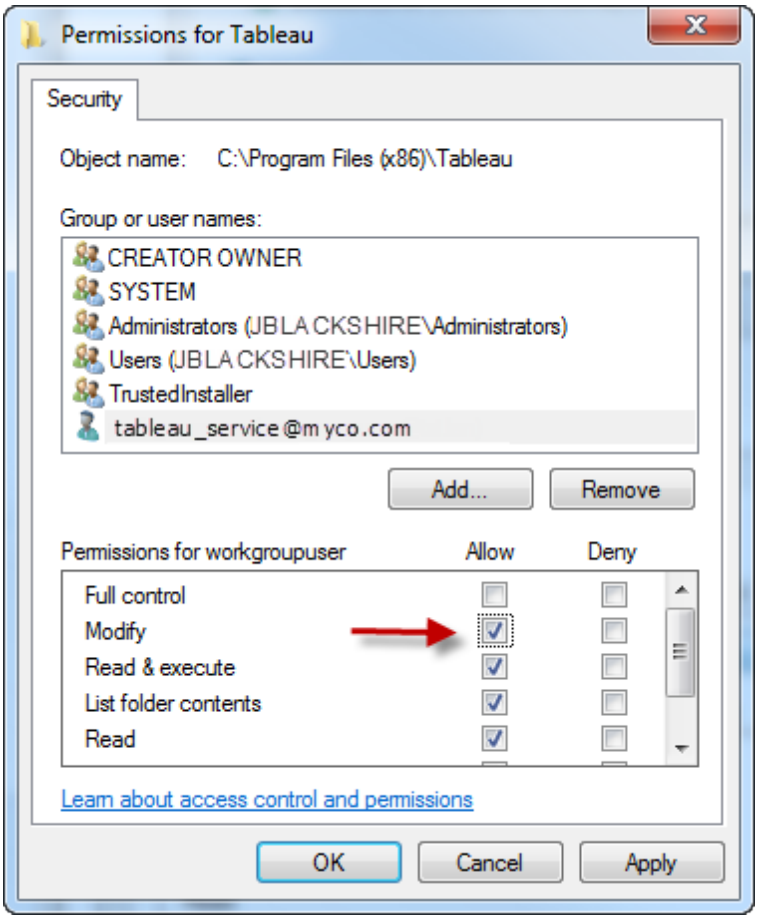
Grant Modify Permissions

The account also needs the ability to do things like create log files. Confirm or configure the following:

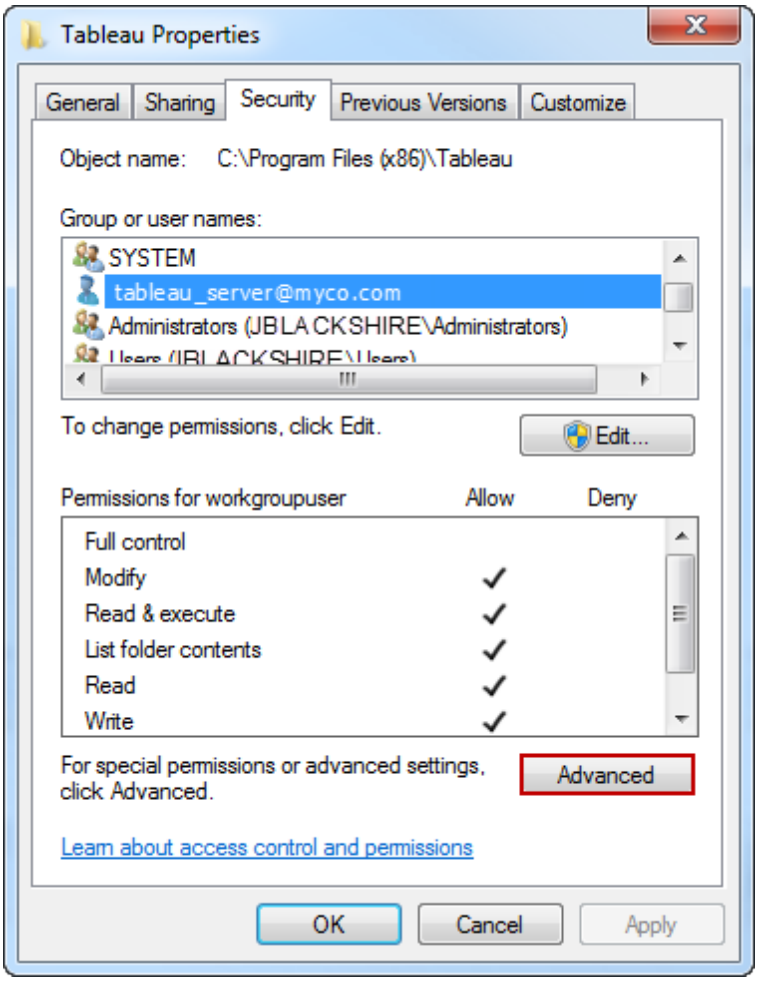
1. Navigate to the following folders:

   32-bit Windows Server 2003: `C:\Program Files\Tableau`
   64-bit Windows Server 2003: `C:\Program Files (x86)\Tableau\`
   Windows Server 2012, Windows Server 2008, Windows Vista, Windows 7, Windows 8: `C:\ProgramData\Tableau\`

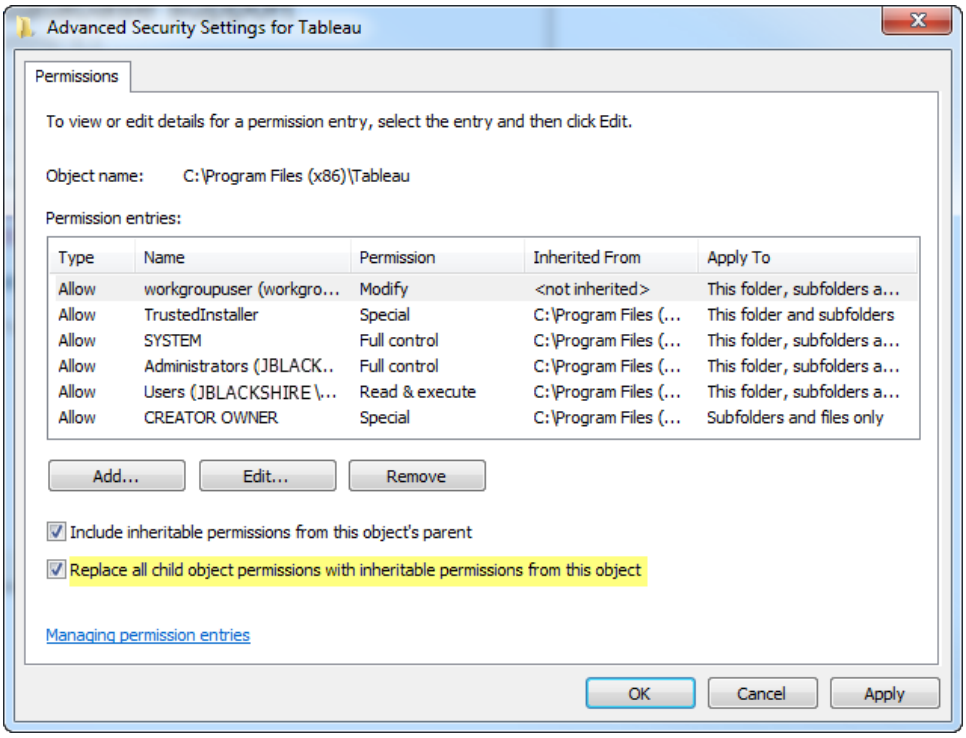
2. Right-click the folder, select **Properties**, and click the **Security** tab:
   - Click **Edit**, then **Add**.
   - Type the `<domain>\<username>` for the Tableau Server Run As User account.
   - Click **Check Names** to resolve the account, then **OK** to confirm.
   - With the Tableau Server Run As User account highlighted, confirm that it has **Modify** permissions. Selecting **Modify** automatically grants all permissions except for **Full Control** and **Special Permissions**:



3. For each folder, on the Tableau Properties Security tab, click **Advanced**:



4. In the Advanced Security Settings for Tableau window, click **Change Permissions**.
5. In the Advanced Security Settings for Tableau dialog box, highlight the Run As User account and select the **Replace all child object permissions with inheritable permissions from this object** check box:



6. Click **OK** to apply changes to all subfolders and files - this may take a few minutes.
7. Click **OK** to confirm changes, then click **OK** in the Tableau Properties dialog box.

Modify Registry Settings

The following step is optional and not seen in most environments. If the registry security is highly restrictive, grant the Tableau Server Run As User account read and write permissions to the following registry branches:

- HKEY_CURRENT_USER\Software\Tableau

and

- 32-bit machine: HKEY_LOCAL_MACHINE\Software\Tableau
- 64-bit machine: HKEY_LOCAL_MACHINE\Software\Wow6432Node\Tableau

SQL Server Impersonation

Impersonation is when one user account acts on behalf of another user account. You can configure Tableau and Microsoft SQL Server to perform database user impersonation, so that the SQL Server database account used by Tableau Server queries on behalf of SQL Server database users, who are also Tableau users.

The main benefit of this feature is it allows administrators to implement and control their data security policy in one place: their databases. When Tableau users access a view with a live connection to a SQL Server database, the view only displays what the users' database

permissions authorize them to see. An additional benefit is that the users don't have to respond to a database login prompt when they access the view. Also, workbook publishers don't have to rely on user-specific filters to restrict what's seen in views.

Use the topics below for more information on what you need to use this feature.

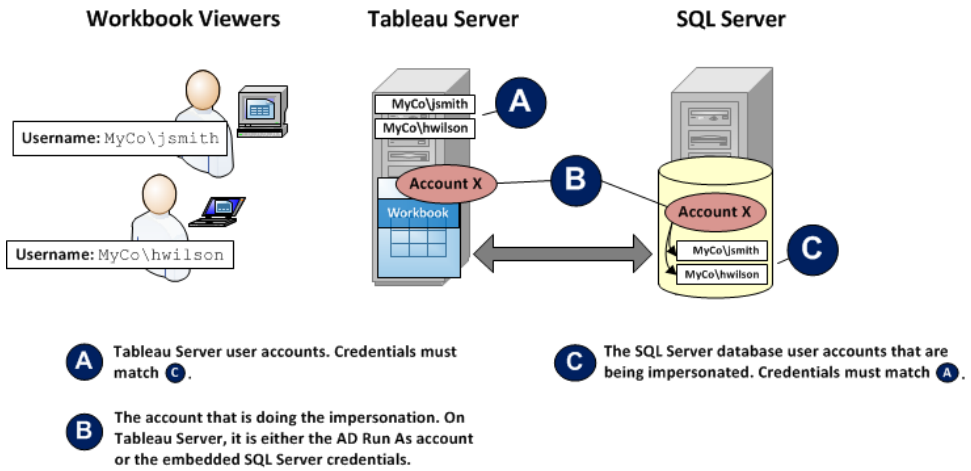
Impersonation Requirements

Here's what you need to use feature:

- **Live connections to SQL Server only:** Impersonation can only be used for views that have a live connection to a SQL Server database, version 2005 or newer.
- I**ndividual database accounts:** Each person who'll be accessing the view must have an explicit, individual account in the SQL Server database to which the view connects. Members of an Active Directory (AD) group cannot be impersonated. For example, if Jane Smith is a member of the AD group Sales, and her database administrator adds the Sales AD group to the SQL Server database, Jane cannot be impersonated.
- **Matching credentials and authentication type:** The credentials of each Tableau user's account and their Tableau user authentication type must match their credentials and authentication type in the SQL Server database. In other words, if Jane Smith's Tableau Server user account has a username of MyCo\jsmith and Tableau Server is using Active Directory for user authentication, her username on the SQL Server database must also be MyCo\jsmith and SQL Server must be using Windows Integrated Authentication.
- **SQL Server prerequisites:** In SQL Server you should have a data security table, a view that enforces data security, and you should require that your database users use the view.
- **SQL IMPERSONATE account:** You need a SQL Server database account that has IMPERSONATE permission for the above database users. This is either an account with the sysadmin role or one that has been granted IMPERSONATE permission for each individual user account (see the MSDN article on EXECUTE AS). This SQL Server account must also be one of two accounts on the Tableau side of things:
    - The Tableau Server Run As User account (see Impersonate with a Run As User Account).
    - The workbook publisher's account (see Impersonate with Embedded SQL Credentials).

How Impersonation Works

Here's an illustration of how database user impersonation works:



In the above illustration, Jane Smith (MyCo\jsmith) is a West Coast sales representative and Henry Wilson (MyCo\hwilson) covers the East. In the SQL Server database, the account permissions for Jane's account, MyCo\jsmith, only give her access to West Coast data. Henry's account, MyCo\hwilson, can only access data for the East Coast.

A view has been created that displays data for the entire country. It has a live connection to a SQL Server database. Both users log into Tableau Server and click the view. Tableau Server connects to SQL Server using a database account with IMPERSONATE permission for each user's database account. This account acts on behalf of each user's database account.

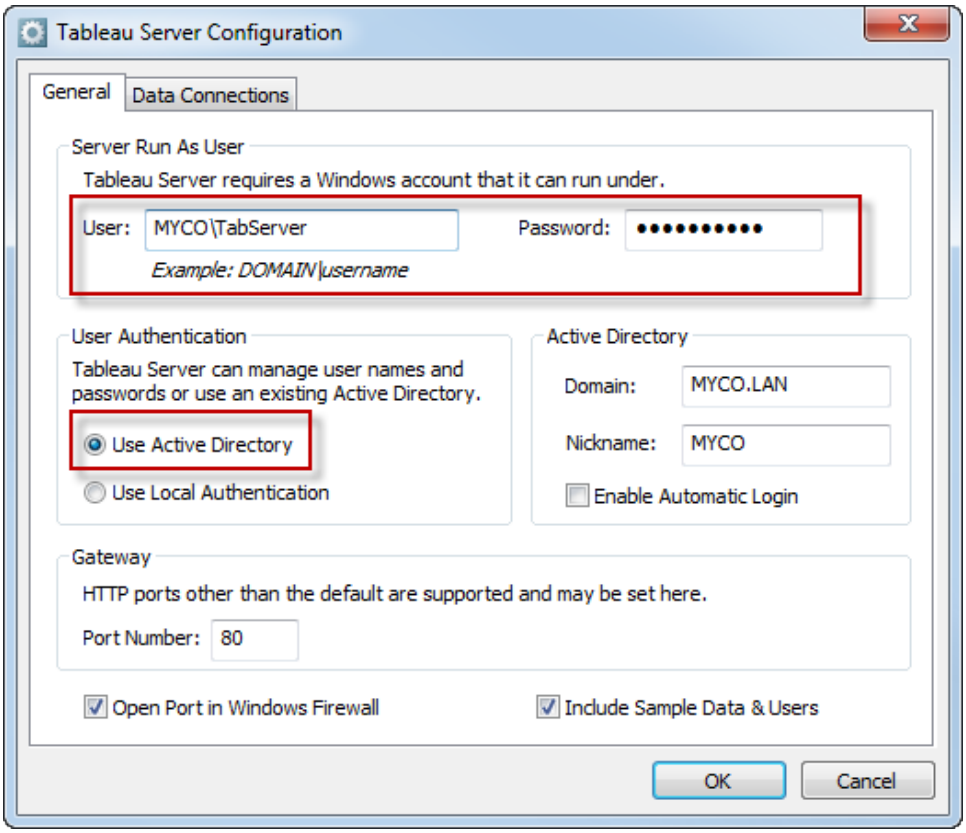
When the view displays, it is restricted by each user's individual database permissions: Jane sees only the West Coast sales data, Henry sees only the East Coast data.

Impersonate with a Run As User Account

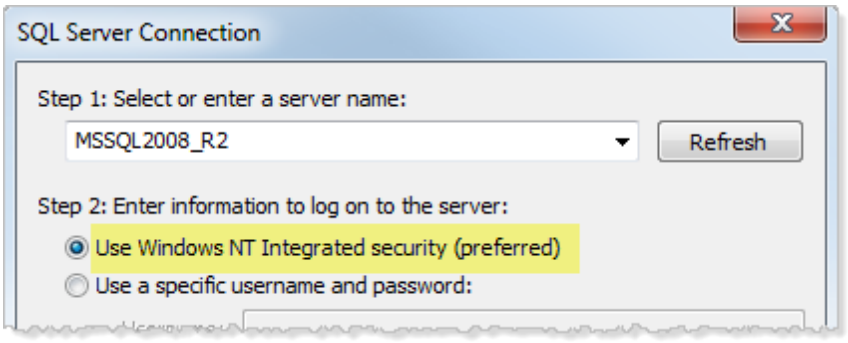
Impersonating via a Run As User account is the recommended way to peform impersonation. The Run As User account is an AD account the Tableau Server service can run under on the machine hosting Tableau Server (see Run As User). This same account must have IMPERSONATE permission for the database user accounts in SQL Server. From a data security standpoint, using the Tableau Server Run As account for impersonation gives the administrator the most control.

To set up impersonation with a Run As User account:

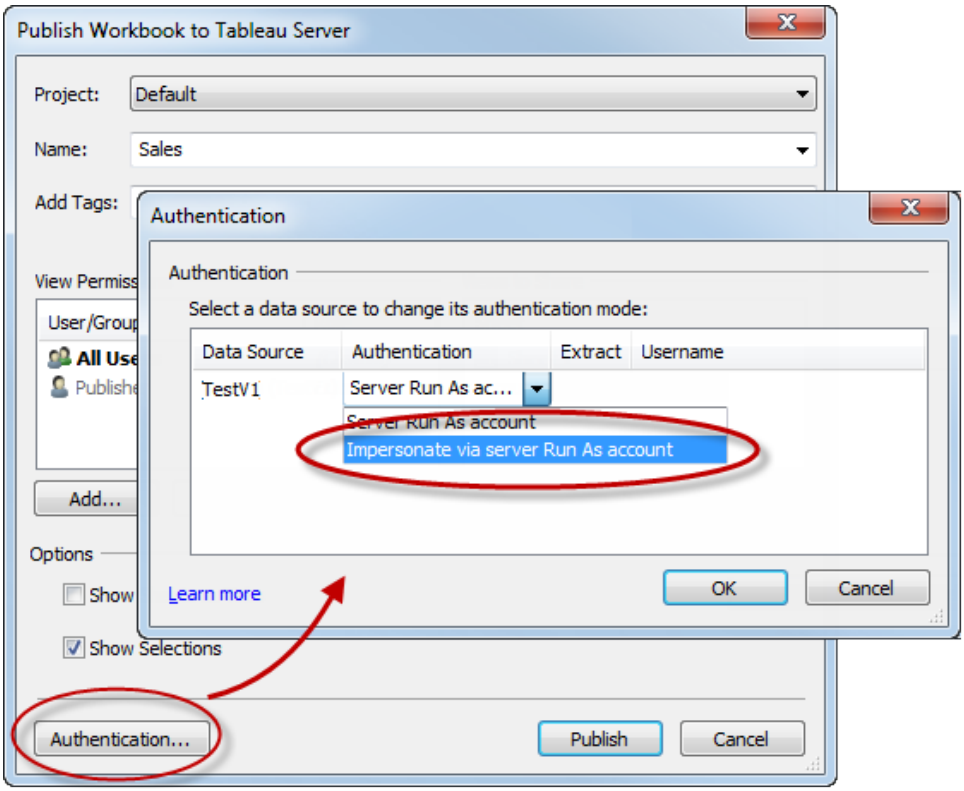
1. When you configure Tableau Server as part of Setup, under **Server Run As User**, enter the Run As User AD account that has IMPERSONATE permission for the user accounts. Under User Authentication, select **Use Active Directory**:



2. Click **OK** to finish configuration.
3. Create a workbook in Tableau Desktop. When you create the data connection, select **Use Windows NT Integrated security** for the workbook's live connection to a SQL Server database:



4. In Tableau Desktop, publish the workbook to Tableau Server (**Server > Publish Workbook**).
5. In the Publish dialog box, click Authentication, then in the Authentication dialog box, select **Impersonate via server Run As account** from the drop-down list:



6. Click **OK**.
7. Test the connection by logging into Tableau Server as a user. When you click a view, you should not be prompted for database credentials and you should only see the data the user is authorized to see.

Impersonate with Embedded SQL Credentials

You can also perform impersonation by having the person who publishes a view embed their SQL Server account credentials in the view. Tableau Server can be running under any type of account, but it will use these credentials, supplied by the publisher, to connect to the database.

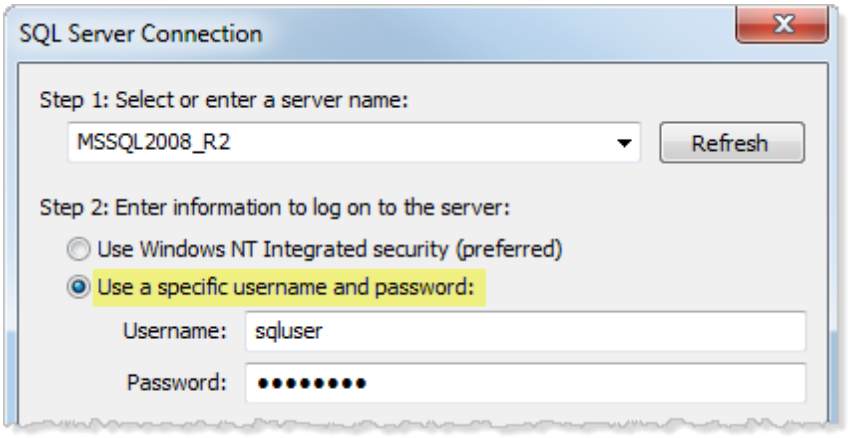
This may be the right choice for your site if the account that handles the impersonation cannot be an AD account and if you're comfortable giving workbook publishers an account with a potentially high permission level on SQL Server.

Note:

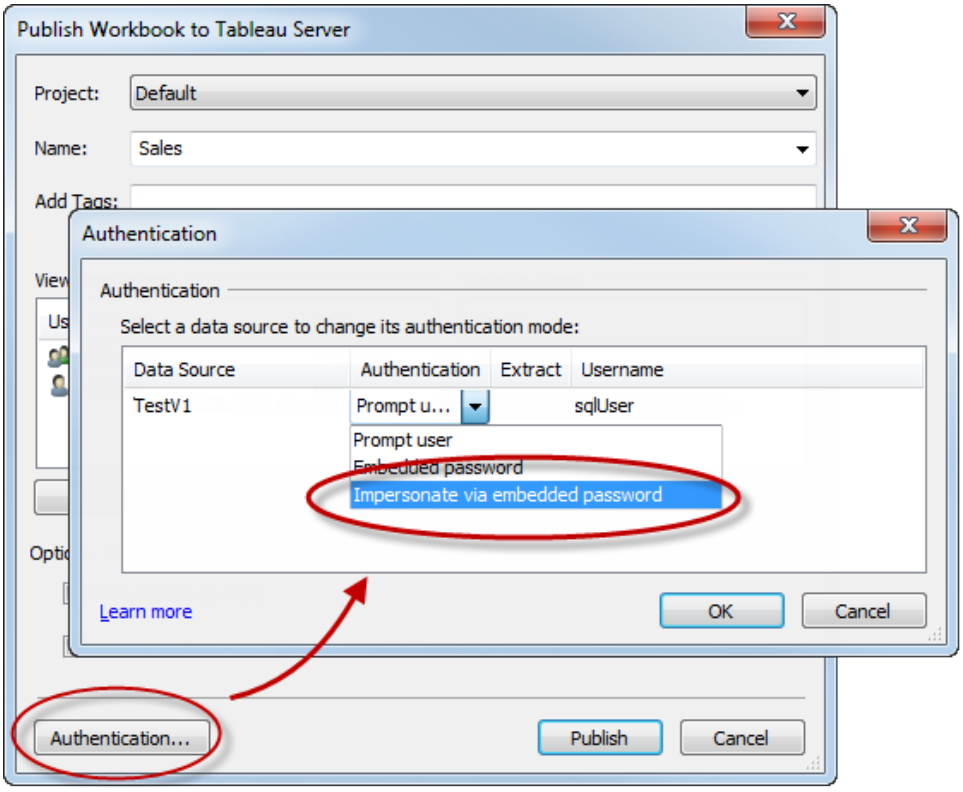
To use this approach, Embedded Credentials must be enabled on Tableau Server:

To impersonate with the workbook publisher's SQL account:

1. In Tableau Desktop, create a workbook. When you create the data connection, select Use a specific username and password for the workbook's live connection to a SQL Server database:



2. Publish the workbook to Tableau Server (**Server > Publish Workbook**).
3. In the Publish dialog box, click Authentication, then in the Authentication dialog box, select **Impersonate via embedded password** from the drop-down list:



4. Click **OK**.
5. Test the connection by logging into Tableau Server as a user. When you click a view,

you should not be prompted for database credentials and you should only see the data the user is authorized to see.

TCP/IP Ports

The following table lists the ports that Tableau Server uses by default, and which must be available for binding. If Windows Firewall is enabled, Tableau Server will open the ports it needs—you do not need to do it yourself (for distributed installations with a worker running Windows 7, refer to the Tableau Knowledge Base).

Port	Used by this server process...	TYPE OF INSTALLATION			Parameter
		All	Distributed	High Availability	
80	Application server.	X			gateway.public.port, worker0.gateway.port
443	SSL. When Tableau Server is configured for SSL, the application server redirects requests to this port.	X			--
3729	Tableau Server Setup.	X			--
3730-3731	Tableau worker servers in distributed and highly available environments (the primary Tableau Server does not listen on these ports).		X	X	--
8000 - 8059	Application server (base port 8000). Consecutive ports after 8000 are used, up to the number of processes. By default, Tableau Server installs with two application server processes (ports 8000 and 8001).	X			wgserver.port
8060	PostgreSQL database.	X			pgsql.port
8061	Firebird.	X			firebird.port
8062	Process that performs discovery in a distributed environment that's been			X	pgsql.initport

Port	Used by this server process...	TYPE OF INSTALLATION			Parameter
		All	Distributed	High Availability	
	configured for high availability.				
8080	Solr and Tomcat HTTP.	X			solr.port, tomcat.http.port[1]
9090	Process that performs replication in a distributed environment that's been configured for high availability.			X	rsync.port
9100 - 9199	VizQL server (base port 9100). Consecutive ports after 9100, up to the number of processes, are also used. By default, Tableau Server installs with two VizQL Server processes (ports 9100 and 9101).	X			vizqlserver.port
9700 - 9899	Data server (base port 9700). Consecutive ports after 9700, up to the number of processes, are also used. By default, Tableau Server installs with two data server processes (ports 9700 and 9701).	X			dataserver.port
27000 - 27009	Workers and primary server to communicate licensing information in distributed and highly available environments.		X	X	--
	One additional port is dynamically chosen for workers and the primary server to communicate licensing information in dis-		X	X	--

[1]These parameters must be set to the same value.

Port	Used by this server process...	TYPE OF INSTALLATION			Parameter
		All	Distributed	High Availability	
	tributed and highly available environments. Instead, you can specify a fixed port (27010 is recommended). See the Tableau Knowledge Base for details.				
27042	Data engine. Tableau Server installs with one data server process. There can only be one active data engine per deployment.	X			dataengine.port
27043	Data engine initialization in a distributed environment that's been configured for high availability.			X	

Edit the Default Ports

You can modify the default ports used by Tableau Server processes by using the command line administrative tool, tabadmin. For example, the default port for the application server process (`wgserver`) is 8000. You can use the tabadmin parameter `workerX.wgserver.port` to change it to a different port. Follow the steps below to change the Tableau Server port configuration. If you are enabling the server's JMX ports, see Enable the JMX Ports

1. Open a command prompt as an administrator and type the following:

   ```
   cd "C:\Program Files (x86)\Tableau\Tableau Server\8.0\bin"
   ```

2. Modify a port value by typing the following:

   ```
   tabadmin set <workerX>.<parameter> <new port value>
   ```

   In the above command, `<workerX>` refers to the machine whose port you want to change, `<parameter>` is one of the values in the table below (a server process' port, such as `wgserver.port`), and `<new port value>` is the new port number you want the server process to use. If Tableau Server is running on one machine, `<workerX>` is `worker0`. If you're running a cluster, `worker0` is the primary, `worker1` is your first worker server, `worker2` is your second, and so on. In this last case, you would

need to run the command (from a command prompt on the primary) once for each machine in the cluster.

Here's an example that sets the port on the primary or a standalone server to 8020 for the application server process (`wgserver`):

```
tabadmin set worker0.wgserver.port 8020
```

The following example sets the port for a 3-machine cluster (one primary and two workers) to 9200 for the VizQL server process.

```
tabadmin set worker0.vizqlserver.port 9200

tabadmin set worker1.vizqlserver.port 9200

tabadmin set worker2.vizqlserver.port 9200
```

You can use the following parameters to modify the corresponding ports—see TCP/IP Ports for a complete list of tabadmin parameters that can be set.

Port to Change	Parameter
80	gateway.public.port, worker0.gateway.port
8000	wgserver.port
8060	pgsql.port
8080	solr.port, tomcat.http.port[1]
9100	vizqlserver.port
9700	dataserver.port

3. After you make the necessary port configuration changes, restart Tableau Server by typing the following:

   ```
   tabadmin restart
   ```

While the server is restarting it will be unavailable to all users. Be sure to warn your users of the outage prior to this operation or schedule this maintenance during non-business hours.

Enable the JMX Ports

To help you work through a problem with Tableau Server, Tableau Support may ask you to enable the server's JMX ports. These ports can be useful for monitoring and troubleshooting, usually with a tool like JConsole.

To enable the JMX ports on Tableau Server:

---

[1]These parameters should be set to the same value.

1. Stop the server.
2. Enter the following command:

   ```
   tabadmin set service.jmx_enabled true
   ```

3. Enter the configure command:

   ```
   tabadmin configure
   ```

4. Start the server.

JMX Port List

Here's the list of JMX ports, all of which are disabled by default. When these ports are enabled, they are used for all types of installations: single-server, distributed, and highly available:

Port	Used by this server process...	Parameter
8300 - 8359	Application server JMX. Determined by the application server port(s) + 300.	--
8550	Backgrounder monitor JMX. Determined by the unused backgrounder port of 8250 + 300.	--
9095	Service monitor JMX.	svcmonitor.jmx.port
9400 - 9499	VizQL server JMX. Determined by the VizQL server port (s) + 300.	--
10000 - 10299	Data server JMX. Determined by the data server port(s) + 300.	--

How the JMX Ports are Determined

The JMX ports for the application server (8300 - 8359), backgrounder (8550), VizQL server (9400 - 9599), and the data server (10000 - 10299) are assigned using the formula “base port + 300” (see TCP/IP Ports for a list of the default base ports). In addition, if there are multiple instances of a process, each will have a JMX port. For example, if you configure Tableau Server to run four instances of the application server process, ports 8000 (default base port), 8001, 8002, and 8003 are used. Application server JMX ports 8300 (base port + 300), 8301, 8302, and 8303 are then bound to their respective process instances.

Even though they’re not directly used by Tableau Server, if a JMX port is being used by another application, Tableau Server processes won’t run. In addition, JMX ports cannot be edited directly using tabadmin. You change a JMX port by changing the base port for its process. In other words, if port 10000 isn’t available for the data server JMX process, you use tabadmin (as described in Edit the Default Ports) to change the data server base port from 9700 to 9800. This will move the data server JMX port to 11000.

The backgrounder process is unique in that it doesn’t use its base port of 8250, but that port number is used to determine its JMX port of 8550 (8250 + 300).

To reduce security risks, it's a good practice to configure your firewall to block outside traffic to the JMX ports.

Restore the Default Value for a Port

You can restore the default value for a port by following the procedure below:

1. Open a command prompt as an administrator and type the following:

   ```
   cd "C:\Program Files (x86)\Tableau\Tableau Server\8.0\bin"
   ```

2. Restore the default port value by typing the following:

   ```
   tabadmin set <workerX>.<parameter> --default
   ```

   If Tableau Server is running on one machine, `<workerX>` is `worker0`. If you're running a cluster, `worker0` is the primary, `worker1` is your first worker server, `worker2` is your second, and so on.

   Here's an example:

   ```
   tabadmin set worker0.wgserver.port --default
   ```

3. Restart Tableau Server by typing the following:

   ```
   tabadmin restart
   ```

tabcmd

The tabcmd utility is one of the two command line tools that installs with Tableau Server (the other is tabadmin). The commands provided through tabcmd can help you automate common tasks, such as publishing workbooks in batches and administering users and groups. The tabcmd utility installs in the Tableau Server bin folder (C:\Program Files\Tableau Server\8.0\bin), but you can install and run tabcmd on another machine as well. See the topics below for more information:

Install tabcmd

The tabcmd command line utility installs with Tableau Server by default and can be run from the server's bin folder (for example, C:\Program Files (x86)\Tableau\Tableau Server\8.0\bin). For administrative flexiblility, it can be installed on another machine as well.

To install tabcmd:

1. Navigate to the extras folder on Tableau Server:

   32-bit: C:\Program Files\Tableau\Tableau Server\8.0\extras\TabcmdInstaller.exe

   64-bit: C:\Program Files (x86)\Tableau\Tableau Server\8.0\extras\TabcmdInstaller.exe

2. Copy TabcmdInstaller.exe to the computer where you want to install it.
3. Double-click TabcmdInstaller.exe to run it.
4. Follow the prompts to install tabcmd. Because tabmd is a command line tool, and due to some limitations with the Windows operating system, Tableau suggests you install tabcmd to its own tabcmd folder at the root of the C:\ drive (C:\tabcmd).

Running the tabmd Setup program does not automatically add tabcmd to the Windows PATH variable, you will need to either explicitly call tabcmd using its full path or add its directory to the PATH variable.

How to Use tabcmd

The first step to using tabcmd is to open a command prompt as an administrator. You then navigate to Tableau Server's bin folder (for example, C:\Program Files (x86)\Tableau\Tableau Server\8.0\bin) or include that location in your commands.

To use tabcmd to perform tasks on Tableau Server, you must establish an authenticated server session. The session identifies the Tableau Server and the Tableau Server user who's running the session. You can start a session first, then specify your command next, or you can start a session and execute a command all at once. The following demonstrates starting a session with the Tableau Server named *tabserver.myco.com*:

```
tabcmd login -s http://tabserver.myco.com -u admin -p p@sswr0d!
```

Now, here is a command that deletes a workbook named *Sales_Workbook*:

```
tabcmd delete "Sales_Workbook"
```

Here's how to accomplish all of the above with one command—note that you don't need `login` here:

```
tabcmd delete "Sales_Workbook" -s http://tabserver.myco.com -u
admin -p p@ssw0rd!
```

A Tableau Server can run multiple sites. When a workbook is on the Default site of a multi-site server you don't need to specify Default, the above command is sufficient. However, if the command applies to something on a site other than Defalt, you need to specify the site ID for that site (see login). Here's the same command for a workbook that's on the West Coast Sales site (site ID `wsales`):

```
tabcmd delete "Sales_Workbook" -s http://tabserver.myco.com -t
wsales -u admin -p p@ssw0rd!
```

The options `-s`, `-t`, `-u`, and `-p` are among tabcmd's global variables, which means they can be used with any command.

When the command is successful, tabcmd returns a status code of zero. A full error message for non-zero status codes is printed to stderr. In addition, informative or progress messages

may be printed to stdout. A full log named tabcmd.log that includes debugging, progress, and error messages is written to:

- Windows Server 2012, Windows Server 2008 R2, Windows Vista, Windows 7, Windows 8: C:\Users\<username>\AppData\Roaming\Tableau
- Windows Server 2003: C:\Documents and Settings\<username>\Application Data\Tableau

tabcmd Global Options

Some options are common to all commands. The table below shows the options that are used by all commands. The `--server`, `--user`, and `--password` options are required at least once to begin a session. An authentication token is stored so subsequent commands can be run without including these options. This token remains valid for five minutes after the last command that used it.

Option (short)	Option (long)	Argument	Description
`-h`	`--help`		Displays the help for the command.
`-s`	`--server`	Tableau Server URL	Required at least once to begin session.
`-u`	`--user`	Tableau Server username	Required at least once to begin session.
`-p`	`--password`	Tableau Server password	Required at least once to begin session. You can alternatively use the `-P` option.
`-P`	`--password-file`	filename.txt	Allows the password to be stored in the given file rather than the command line for increased security.
`-t`	`--site`	Tableau Server site name	Use the specified Tableau Server site. If you do not specify a site, the Default site is assumed. Only applies to servers with multiple sites.
`-x`	`--proxy`	Host:Port	Uses the specified HTTP proxy.
	`--no-prompt`		When specified, the command will not prompt for a password. If no valid password is provided the command will fail.
	`--no-proxy`		When specified, an HTTP proxy will not be used.

Option (short)	Option (long)	Argument	Description
	`--no-cert-check`		When specified, the SSL certificate is not validated.
	`--[no-]cookie`		When specified, the session id is saved on login so subsequent commands will not need to log in. Use the `no-` prefix to not save the session id. By default the session is saved.
	`--timeout`	seconds	Waits the specified number of seconds for the server to complete processing the command. By default the process will timeout in 30 seconds.

tabcmd Commands

Here are the commands that can be used with the tabcmd command line tool:

addusers group-name
createproject project-name
createsite site-name
createsiteusers filename.csv
createusers filename.csv
delete workbook-name or datasource-name
deletegroup group-name
deletesite site-name
deleteusers filename.csv
editsite site-name
export
get url
listsites
login
logout
publish file-name.twb(x), filename.tds(x), or file-name.tde
refreshextracts workbook-name or data-source-name
removeusers group-name
runschedule schedule-name
set setting
syncgroup group-name
version

addusers *group-name*

Adds the users listed in the --users argument to the group with the given group-name.

Example

```
tabcmd addusers "Development" --users "users.csv"
```

Option (short)	Option (long)	Argument	Description
	`--users`	filename.csv	Add the users in the given file to the specified group. The file should be a simple list with one username per line. The users should already be created on Tableau Server. See also Import Users from a CSV File.
	`--[no-]complete`		When set to `complete` this option requires that all rows be valid for any change to succeed. If not specified, `--complete` is used.

creategroup *group-name*

Creates a group with the given group name. Use `addusers` (for local groups) and `sync-group` (for Active Directory groups) commands to add users after the group has been created.

Example

```
tabcmd creategroup "Development"
```

createproject *project-name*

Creates a project with the given project name.

Example

```
tabcmd createproject -n "Quarterly_Reports" -d "Workbooks showing
quarterly sales reports."
```

Option (short)	Option (long)	Argument	Description
`-n`	`--name`	name	Specify the name of the project that you want to create.
`-d`	`--descrip-tion`	description	Specify a description for the project.

createsite *site-name*

Creates a site with the given site name.

Examples

Create a site named West Coast Sales. A site ID of `WestCoastSales` will be automatically created, the site will have no storage quota limit, and site administrators will be able to add and remove users:

```
tabcmd createsite "West Coast Sales"
```

Create a site named `West Coast Sales` with a site ID of `wsales`:

```
tabcmd createsite "West Coast Sales" -r "wcoast"
```

Prevent site administrators from adding users to the site:

```
tabcmd createsite "West Coast Sales" --no-site-mode
```

Set a storage quota, in MB::

```
tabcmd createsite "West Coast Sales" --storage-quota 100
```

Opti-on (sh-ort)	Option (long)	Argument	Description
`-r`	`--url`	site ID	Used in URLs to specify the site. Different from the site name.
	`--user-quota`	number of users	Maximum number of users that can be added to the site.
	`--[no-]site-mode`		Allow or deny site administrators the ability to add users to or remove users from the site.
	`--stor-`	number of	In MB, the amount of workbooks, extracts, and data

Option (short)	Option (long)	Argument	Description
	`age-quota`	MB	sources that can be stored on the site.

createsiteusers *filename.csv*

This command allows site administrators to add users to a site. It creates users on the current site, using the given comma separated values (csv) file. The file may have the following columns, in the order shown below:

1. `Username`
2. `Password`
3. `Full Name`
4. `License Level` (`interactor/viewer/unlicensed`)
5. `Administrator` (`site/none`)
6. `Publisher` (`yes/true/1` or `no/false/0`)

The file can have fewer columns. For example it can be a simple list with one username per line. When the server is using Active Directory authentication, the `Password` column is ignored. Quotes may be used if a value contains commas. See Import Users from a CSV File for other details.

Example

```
tabcmd createsiteusers "users.csv" --license "Interactor" --publishers
```

Option (short)	Option (long)	Argument	Description
	`--nowait`		Do not wait for asynchronous jobs to complete.
	`--silent-progress`		Do not display progress messages for asynchronous jobs.
	`--license`	`Interactor`, `Viewer`, or `Unlicensed`	Sets the default license level for all users. This setting may be overridden by the value in the CSV file.
	`--admin-`	`Site` or	Assigns or removes the site admin

Option (short)	Option (long)	Argument	Description
	`type`	`None`	right to all users in the CSV file. This setting may be overridden by the value in the CSV file. The default is `None` for new users and unchanged for existing users. System administrators cannot be created or demoted using `createsiteusers` (use `createusers` instead).
	`--[no-]publishers`		Assigns or removes the Publish right to all users in the CSV file by default. This setting may be overridden by the value in the CSV file. The default is `no` for new users and unchanged for existing users.
	`--[no-]complete`		Require (or not require) that all rows be valid for any change to succeed. By default, `--complete` option is used.

createusers *filename.csv*

Creates the users listed in the given comma separated values (csv) file. This command can only be used by system administrators. The file may have the following columns, in the order shown below:

1. `Username`
2. `Password`
3. `Full Name`
4. `License Level` (`interactor/viewer/unlicensed`)
5. `Administrator` (`system/site/none`)
6. `Publisher` (`yes/true/1` or `no/false/0`)

The file can have fewer columns. For example it can be a simple list with one username per line. When the server is using Active Directory authentication, the `Password` column should be left blank. Quotes may be used if a value contains commas. See Import Users from a CSV File for other details.

Example

```
tabcmd createusers "users.csv" --license "Interactor" --
```

`publishers`

Option (short)	Option (long)	Argument	Description
	`--nowait`		Do not wait for asynchronous jobs to complete.
	`--silent-progress`		Do not display progress messages for asynchronous jobs.
	`--license`	`Interactor`, `Viewer`, or `Unlicensed`	Sets the default license level for all users. This setting may be overridden by the value in the CSV file.
	`--admin-type`	`System`, `Site`, or `None`	Assigns or removes the Admin right to all users in the CSV file by default. This setting may be overridden by the value in the CSV file. The default is `None` for new users and unchanged for existing users.
	`--[no-]publishers`		Assigns the Publish right to all users in the CSV file by default. This setting may be overridden by the value in the CSV file. The default is `no` for new users and unchanged for existing users.
	`--[no-]complete`		Requires that all rows be valid for any change to succeed. By default, `--complete` option is used.

delete *workbook-name* or *datasource-name*

Deletes the given workbook or data source from the server. This command takes the name of the workbook or data source as it is on the server, not the file name when it was published.

Example

```
tabcmd delete "Sales_Analysis"
```

Option (short)	Option (long)	Argument	Description
`-r`	`--project`	Project name	The name of the project containing the workbook or data source you want to delete. If not specified, the “Default”

Option (short)	Option (long)	Argument	Description
			project is assumed.
	`--workbook`	Workbook name	The name of the workbook you want to delete.
	`--data-source`	Data source name	The name of the data source you want to delete.

deletegroup *group-name*

Deletes the group with the given group-name from the server.

Example

```
tabcmd deletegroup "Development"
```

deletesite *site-name*

Deletes the site with the given site-name from the server.

Example

```
tabcmd deletesite "Development"
```

deleteusers *filename.csv*

Deletes the users listed in the given comma separated (csv) file. The file is a simple list of one username per line.

Example

```
tabcmd deleteusers "users.csv"
```

Option (short)	Option (long)	Argument	Description
	`--[no-]complete`		When set to `--complete` this option requires that all rows be valid for any change to succeed. If not specified, `--complete` is used.

editsite *site-name*

Allows you to change the name of a site or its web folder name. You can also use this command to allow or deny site administrators the ability to add and remove users. If site administrators have user management rights, you can specify how many users they can add to a site.

Examples

```
tabcmd editsite wc_sales --site-name "West Coast Sales"

tabcmd editsite wc_sales --site-id "wsales"

tabcmd editsite wsales --status ACTIVE

tabcmd editsite wsales --user-quota 50
```

Option (long)	Argument	Description
`--site-name`	Name to change the site to	The name of the site that's dislayed.
`--site-id`	The site ID to change the site to	Used in the URL to uniquely identify the site.
`--user-quota`	Number of users	Maximum number of users who can be members of the site.
`--[no-]site-mode`		Allow or prevent site administrators from adding users to the site.
`--status`	ACTIVE or SUSPENDED	Activate or suspend a site.
`--storage-quota`	Number of MB	In MB, the amount of workbooks, extracts, and data sources that can be stored on the site.

export

Exports a view or workbook from Tableau Server and saves it to a file. Note the following when you use this command:

- **Permissions**: To export, you must have the **Export Image** permission. By default, this permission is Allowed or Inherited for all roles, although permissions can be set per workbook or view.
- **The view, workbook, or data being exported**: You specify this using the

"workbook/view" string as it appears in the URL for the workbook or view, not using its “friendly name.” For example, to export the Tableau sample view *Investment Growth* from the *Finance* workbook, you would use the string Finance/InvestmentGrowth. Use -t <site_id> if the server is running multiple sites and the view or workbook is on a site other than Default.

To export a workbook, you still include a valid view in the string you use. Using the above example, to export the *Finance* workbook, you would use the string Finance/InvestmentGrowth. Finally, to export a workbook, it must have been published with **Show Sheets as Tabs** selected in the Tableau Desktop Publish dialog box.

- **The saved file’s format**: Your format options depend on what’s being exported. A workbook can only be exported as a PDF using the --fullpdf argument. A view can be exported as a PDF (--pdf), a PNG (--png), or you can export the view’s data as a CSV file (--csv).
- **The saved file’s name and location** (optional): If you don’t provide a name, it will be derived from the view or workbook name. If you don’t provide a location, the file will be saved to your current working directory. Otherwise, you can specify a full path or one that’s relative to your current working directory.

Clearing the Cache to Use Real-Time Data

You can optionally add the URL parameter ?:refresh=yes to force a fresh data query instead of pulling the results from the cache. If you are using tabcmd with your own scripting and the refresh URL parameter is being used a great deal, this can have a negative impact on performance. It's recommended that you only use refresh when real-time data is required—for example, on a single dashboard instead of on an entire workbook.

Examples

Views

```
tabcmd export "Q1Sales/Sales_Report" --csv -f "Weekly-Report"

tabcmd export -t Sales "Sales/Sales_Analysis" --pdf -f "C:\Tableau_Workbooks\Weekly-Reports"

tabcmd export "Finance/InvestmentGrowth" --png

tabcmd export "Finance/InvestmentGrowth?:refresh=yes" --png
```

Workbooks

```
tabcmd export "Q1Sales/Sales_Report" --fullpdf

tabcmd export -t Sales "Sales/Sales_Analysis" --fullpdf --page-size tabloid -f "C:\Tableau_Workbooks\Weekly-Reports"
```

Option (short)	Option (long)	Argument	Description
`-f`	`--filename`	Name to save the file as	Saves the file with the given filename.
	`--csv`		View only. Export the view's data in CSV format.
	`--pdf`		View only. Export as a PDF.
	`--png`		View only. Export as an image in PNG format.
	`--fullpdf`		Workbook only. Export as a PDF. The workbook must have been published with **Show Sheets as Tabs** enabled.
	`--page-layout`	`landscape, portrait`	Sets the page orientation of the exported PDF. If not specified, its Tableau Desktop setting will be used.
	`--pagesize`	`unspecified, letter, legal, note folio, tabloid, ledger, statement, executive, a3, a4, a5, b4, b5, quatro`	Sets the page size of the exported PDF. Default is `letter`.
	`--width`	Number of pixels	Sets the width. Default is 800 px.
	`--height`	Number of pixels	Sets the height.Default is 600 px.

get *url*

Using a URL string as one of its parameters, makes an HTTP "GET" request of Tableau Server. The result is returned as a file. Note the following when you use this command:

- **Permissions**: To get a file, you must have the **Download/Web Save As** permission. By default, this permission is Allowed or Inherited for all roles, although permissions can be set per workbook or view.
- **File extension**: The URL string of the file you want to GET must include a file extension—such as `"/views/Finance/InvestmentGrowth.pdf"`. The extension (for example, `.pdf`) determines what's returned. A view can be returned in PDF, PNG, CSV (data only), or XML (information only) format. A workbook can be returned as a TWB or TWBX. To figure out the correct extension to use, you can use a web browser to navigate to the item you're interested in on Tableau Server and add the file extension to the end of the URL.
- **The saved file's name and location** (optional): The name you use for `--filename` should include the file extension. If you don't provide a name and file extension, both will be derived from the URL string. If you don't provide a location, the file will be saved to your current working directory. Otherwise, you can specify a full path or one that's relative to your current working directory.
- **PNG size** (optional): If the saved file is a PNG, you can specify the size, in pixels, in the URL.

Clearing the Cache to Use Real-Time Data

You can optionally add the URL parameter `?:refresh=yes` to force a fresh data query instead of pulling the results from the cache. If you are using tabcmd with your own scripting and the `refresh` URL parameter is being used a great deal, this can have a negative impact on performance. It's recommended that you only use `refresh` when real-time data is required—for example, on a single dashboard instead of on an entire workbook.

Examples

Views

```
tabcmd get "/views/Sales_Analysis/Sales_Report.png" --filename
"Weekly-Report.png"

tabcmd get "/views/Finance/InvestmentGrowth.pdf" -f
"Q1Growth.pdf"

tabcmd get "/views/Finance/InvestmentGrowth.csv"

tabcmd get "/views/Finance/InvestmentGrowth.png?:size=640,480" -f
growth.png

tabcmd get "/views/Finance/InvestmentGrowth.png?:refresh=yes" -f
growth.png
```

Workbooks

```
tabcmd get "/workbooks/Sales_Analysis.twb" -f "C:\Tableau_
```

```
Workbooks\Weekly-Reports.twb"

tabcmd get "/workbooks/Sales.xml"
```

Other

```
tabcmd get "/users.xml" --filename "UserList.xml"
```

Option (short)	Option (long)	Argument	Description
`-f`	`--file-name`	Name to save the file as	Saves the file with the given filename.

listsites

Returns a list of sites to which the logged in user belongs.

Example

```
tabcmd listsites -u corman -pw P@ssword!
```

login

Logs a Tableau Server user in to the server. Use the `--server`, `--site`, `--username`, `--password` global options to create a session. If you want to log in using the same information you've already used to create a session just specify the `--password` option. The server and username stored in the cookie will be used.

If the server is using a port other than 80 (the default), you will need to specify it.

You only need the `--site` (`-t`) option if the server is running multiple sites and you are logging in to a site other than the Default site. If you do not provide a password you will be prompted for one. If the `--no-prompt` option is specified and no password is provided the command will fail.

Once you log in, the session will continue until it expires on the server or the `logout` command is run.

s**Example**

Logs you in to the Tableau Server running on your local machine:

```
tabcmd login -s http://localhost -u jsmith -p p@ssw0rd!
```

Logs you in to the Sales stie on sales-server:

```
tabcmd login -s http://sales-server -t Sales -u administrator -p
```

```
p@ssw0rd!

tabcmd login -s http://sales-server:8000 -t Sales -u admin-
istrator -p p@ssw0rd!
```

Logs you in to the Sales stie on sales-server using SSL:

```
tabcmd login -s https://sales-server -t Sales -u administrator -p
p@ssw0rd!
```

Establishes a forward proxy and port for localhost:

```
tabcmd login --proxy myfwdproxyserver:8888 -s http://localhost -u
jsmith -p p@ssW0rd!
```

Logs you in to the reverse proxy using SSL:

```
tabcmd login -s https://myreverseproxy -u jsmith -p p@ssW0rd!
```

Option (short)	Option (long)	Argument	Description
`-s`	`--server`	URL of the Tableau Server	If you are running the command from the Tableau Server computer, you can use http://localhost. Otherwise, specify the computer's URL, such as `http://bigbox.myco.com` or `http://bigbox`.
`-t`	`--site`	Site ID of the Tableau Server site	The site ID for the site. This is what's used in the URL to uniquely identify the site. For example, a site named West Coast Sales might have a site ID of WestCoastSales. Use this option if the server is running multiple sites and you are logging in to a site other than the Default site.
`-u`	`--username`	Tableau Server username	The username of the Tableau Server user.
`-p`	`--password`	Tableau Server password	Password for the Tableau Server user. If you do not provide a password you will be prompted for one.
`-x`	`--proxy`	Host:Port	Use to specify the HTTP proxy server and port for the tabcmd request.
	`--no-prompt`		Do not prompt for a password. If no password is specified, the `login`

Option (short)	Option (long)	Argument	Description
			command will fail.
	`--no-proxy`		Do not use an HTTP proxy server.
	`--[no-]cookie`		Saves the session ID on login. Subsequent commands will not require a login. Cookies are enabled (`--cookie`) by default.
	`--time-out SECONDS`	Number of seconds	The number of seconds the server should wait before processing the `login` command. Default: 30 seconds.

logout

Logs out of the server.

Example

```
tabcmd logout
```

publish *filename.twb(x), filename.tds(x), or filename.tde*

Publishes the given workbook (.twb(x)), data source (.tds(x)), or data extract (.tde) to Tableau Server. By default, all sheets in the workbook are published without database usernames or passwords.

Example

```
tabcmd publish "analysis.twbx" -n "Sales_Analysis" --db-user
"jsmith" --db-password "p@ssw0rd"
```

Option (short)	Option (long)	Argument	Description
`-n`	`--name`	Name of the workbook or data source on the server	If omitted, the workbook, data source, or data extract will be named after filename.

Option (short)	Option (long)	Argument	Description
`-o`	`--overwrite`		Overwrites the workbook, data source, or data extract if it already exists on the server.
`-r`	`--project`	Name of a project	Publishes the workbook, data source, or data extract into the specified project. Publishes to the "Default" project if not specified.
	`--db-username`		Use this option to publish a database username with the workbook, data source, or data extract.
	`--db-password`		Use this option to publish a database password with the workbook, data source, or data extract.
	`--save-db-password`		Stores the provided database password on the server.
	`--thumbnail-username`		If the workbook contains users filters, the thumbnails will be generated based on what the specified user can see. Cannot be specified when `--thumbnail-group` option is set.
	`--thumbnail-group`		If the workbook contains users filters the thumbnails will be generated based on what the specified group can see. Cannot be specified when `--thumbnail-username` option is set.
	`--tabbed`		When a workbook with tabbed views is published, each sheet becomes a tab that viewers can use to navigate through the workbook. Note that this setting will override any sheet-level security.
	`--append`		Append the extract file to the existing data source.

Option (short)	Option (long)	Argument	Description
	`--replace`		Use the extract file to replace the existing data source.
	`--disable-uploader`		Disable the incremental file uploader.
	`--disable-tde-compression`		Stop compressing the extract file before it's uploaded.
	`--restart`		Restart the file upload.

If the workbook contains user filters, one of the thumbnail options must be specified.

refreshextracts *workbook-name* or *datasource-name*

Performs a full or incremental refresh of extracts belonging to the specified workbook or data source. This command takes the name of the workbook or data source as it appears on the server, not the file name when it was published.

Examples

```
tabcmd refreshextracts --datasource sales_ds

tabcmd refreshextracts --workbook "My Workbook"

tabcmd refreshextracts --url SalesAnalysis
```

Option (short)	Option (long)	Argument	Description
	`--incremental`		Runs the incremental refresh operation.
	`--synchronous`		Runs the full refresh operation immediately in the foreground.
	`--workbook`	Name of a workbook	The name of the workbook containing extracts to refresh. If the workbook has spaces in its name, enclose it in quotes.
	`--data-source`	Name of a data source	The name of the data source containing extracts to refresh.
	`--project`	Name of a	Use with `--workbook` or `--data-`

Option (short)	Option (long)	Argument	Description
		project	`source` to identify a workbook or data source in a project other than *Default*. If not specified, the Default project is assumed.
	`--url`	URL name of a workbook	The name of the workbook as it appears in the URL. A workbook published as “Sales Analysis” has a URL name of “SalesAnalysis”.

removeusers *group-name*

Removes the users listed in the `--users` argument from the group with the given group-name.

Example

```
tabcmd removeusers "Development" --users "users.csv"
```

Option (short)	Option (long)	Argument	Description
	`--users`	filename.csv	Remove the users in the given file from the specified group. The file should be a simple list with one username per line.
	`--[no-]complete`		Requires that all rows be valid for any change to succeed. If not specified `--complete` is used.

runschedule *schedule-name*

Runs the specified schedule. This command takes the name of the schedule as it is on the server.

Example

```
tabcmd runschedule "5AM Sales Refresh"
```

set *setting*

Enables the specified setting on the server. Details about each setting can be seen on the Maintenance page on the server. Use an exclamation mark in front of the setting name to disable the setting. You can enable or disable the following settings:

- embedded_credentials
- public_users_list
- remember_passwords_forever

Example

```
tabcmd set embedded_credentials
```

syncgroup *group-name*

Synchronizes the group with the given group-name with Active Directory. This command can also be used to create a new group on the server that is based on an existing Active Directory group.

Example

```
tabcmd syncgroup "Development"
```

Option (short)	Option (long)	Argument	Description
	`--license`	`viewer interactor unlicensed`	Sets the license level for all users in the group.
	`--administrator`	`system site none`	Assigns or removes the Administrator right for all users in the group. The Administrator user type can be system, site, or none. Default is none (new users do not get the Administrator right) and existing users are unchanged.
	`--[no-]publisher`		Assigns or removes the Publish right for all users in the gorup. If unspecified, new users are not assigned this right and existing users are unchanged.
	`--[no-]com-`		Requires that all rows be valid for

Option (short)	Option (long)	Argument	Description
	`plete`		any change to succeed. If unspecified, `--complete` is used.
	`--silent-progress`		Suppresses progress messages.

version

Prints the version information for the current installation of the tabcmd utility.

Example

```
tabcmd version
```

tabadmin

You can perform certain administrative tasks and change Tableau Server configuration settings using the tabadmin command line tool. To access it, open a command prompt as an administrator and change directories using the command below:

32-bit: `cd "C:\Program Files\Tableau\Tableau Server\8.0\bin"`
64-bit: `cd "C:\Program Files (x86)\Tableau\Tableau Server\8.0\bin"`

To see a list of all available tabadmin commands, type the following:

```
tabadmin help commands
```

For more information, see the topics below:

tabadmin set

One of the most commonly used tabadmin commands is `tabadmin set`, which allows you to change the value of Tableau Server configuration options. The syntax is as follows:

```
tabadmin set option-name value
```

To use this command:

1. Stop the server.
2. Type `tabadmin set` followed by the name of the option and the value.

   For example, to change the default value of the tabadmin option backgrounder.querylimit from 7200 seconds (2 hours, the default) to 9000 seconds, type the following:

   ```
   tabadmin set backgrounder.querylimit 9000
   ```

   If you're setting an option whose value begins with a hyphen, use nested quotes:

   ```
   tabadmin set option-name "'-value'"

   tabadmin set option-name "'-value1 -value2'"
   ```

3. After you use `tabadmin set`, enter the `configure` command:

   ```
   tabadmin configure
   ```

4. Finally, start Tableau Server.

tabadmin set options

Use the table below to learn more about Tableau Server options you can configure using the tabadmin set command. See TCP/IP Ports for a complete list of ports.

Option	Default Value	Description
auditing.enabled	true	Allows access to the PostgreSQL (Tableau Server's own database) historical auditing tables. See Create Custom Administrative Views for details.
backgrounder.querylimit	7200	Longest allowable time for completing an extract refresh, in seconds (7200 seconds = 2 hours).
dataengine.port	27042	Port that the data engine runs on.
dataserver.port	9700	Port that the data server runs on.
gateway.public.host	Name of the machine	The canonical name of the server, used for external access to Tableau Server. If Tableau Server is configured to work with a proxy server, it is the canonical name of the proxy server (not Tableau Server).
gateway.public.port	80 (443 if SSL)	Applies to proxy server environments only. The external port the proxy server listens on.
gateway.timeout	1800	Longest amount of time, in seconds, that the

Option	Default Value	Description
		gateway will wait for certain events before failing a request (1800 seconds = 30 minutes).
gateway.trusted	IP address of proxy server machine	Applies to proxy server environments only. The IP address(es) of the proxy server.
gateway.trusted_hosts	Alternate name (s) of proxy server	Applies to proxy server environments only. Any alternate host name(s) for the proxy server.
java.heap.size	128m	Size of heap for Tomcat (repository and solr). This generally does not need to change except on advice from Tableau.
pgsql.port	8060	Port that PostgreSQL listens on.
rsync.timeout	600	Longest allowable time, in seconds, for completing file synchronization (600 seconds = 10 minutes). File synchronization occurs as part of configuring high availability, or moving the data engine and repository processes.
service.jmx_enabled	false	Setting to `true` enables JMX ports for optional monitoring and troubleshooting. See Enable the JMX Ports for details.
service.max_procs	# of processes	Maximum number of server processes.
subscriptions.enabled	false	Controls whether subscriptions are configurable system-wide. See Manage Subscriptions.
subscriptions.timeout	1800	Number of seconds after which the background process handling a subscription times out.
solr.port	8080	Port that solr listens on. This must be the same value as tomcat.http.port.
tomcat.http.port	8080	Port that Tomcat runs on.
tomcat.https.port	8443	SSL port for Tomcat (unused).
tomcat.server.port	8085	Port that tomcat listens on for shutdown messages.
vizqlserver.browser.render	true	Views under the threshold set by `vizqlserver.browser.render_threshold` or `vizqlserver.browser.render_`

Option	Default Value	Description
		`threshold_mobile` are rendered by the client web browser instead of by the server. See About Client-Side Rendering for details.
vizqlserver.browser.render_threshold	100	The default value (100) represents a high level of complexity for a view displayed on a PC. Complexity factors include number of marks, headers, reference lines, and annotations. Views that exceed this level of complexity are rendered by the server instead of in the PC's web browser.
vizqlserver.browser.render_threshold_mobile	20	The default value (20) represents a high level of complexity for a view displayed on a tablet. Complexity factors include number of marks, headers, reference lines, and annotations. Views that exceed this level of complexity are rendered by the server instead of in the tablet's web browser.
vizqlserver.port	9100	Base port for the VizQL servers.
vizqlserver.protect_session	true	When set to `true` (the default), prevents VizQL sessions from being reused after the original user logs off.
vizqlserver.querylimit	1800	Longest allowable time for updating a view, in seconds.
vizqlserver.session.expiry.minimum	5	Number of minutes of idle time after which a VizQL session is eligible to be discarded if the VizQL process starts to run out of memory.
vizqlserver.session.expiry.timeout	30	Number of minutes of idle time after which a VizQL session is discarded.
vizqlserver.showdownload	true	Controls the display of the Download button above views.
vizqlserver.showshare	true	Controls the display of the Share button above views.
vizqlserver.trustedticket.log_level	`info`	The logging level for trusted authentication, written to `ProgramData\Tableau\Tableau Server\data\tabsvc\logs\vizqlserver\vizql-*.log`. Set to `debug` for more information.
wgserver.audit_history_	183	Number of days after which historical events

Option	Default Value	Description
expiration_days		records are removed from the PostgreSQL database (Tableau Server's own database). See Create Custom Administrative Views for details.
wgserver.domain.fqdn	value of %USERDOMAIN%	The fully qualified domain name of the Active Directory server to use.
wgserver.password_autocomplete.enabled	false	Controls whether web browsers are allowed to automatically complete password fields.
wgserver.session.idle_limit	240	The number of minutes of idle time before a login to the web application times out.
wgserver.show_view_titles_not_names	true	You can only use this option if you upgraded to version 7.0 from 6.0 or earlier. A value of true keeps the earlier behavior and causes Tableau Server to display view titles as their identifiers (for ex., in search results); false causes the server to display view names as their identifiers.
wgserver.trusted_hosts		This option takes a comma separated list of trusted IP addresses (not host names) for the machine you want to accept trusted requests from. A common value is 127.0.0.1 if you want to put the webserver and Tableau Server on the same machine. This option is used when setting up a trusted relationship between the web server and Tableau Server when embedding views.
workerX.gateway.port	80 (443 if SSL)	External port that Apache listens on for workerX. worker0.gateway.port is Tableau Server's external port. In a distributed environment, worker0 is the primary Tableau Server.
workerX.vizqlserver.procs	# of processes	Number of VizQL servers.
workerX.vizqlserver.port	9100	Base port for the vizQL server on workerX.
workerX.wgserver.port	8000	Base port for the web application server on workerX.
workerX.wgserver.procs	# of processors	Number of web application server processes.

Restore a Setting to its Default Value

You can restore the default value for a Tableau Server configuration setting by doing the following:

1. Stop the server.
2. Still in the bin directory, restore the default value for a particular setting by typing the following:

   ```
   tabadmin set option-name --default
   ```

   For example, to set the tabadmin vizqlserver.session.expiry.timeout option back to its default value of 30 minutes, you would type the following:

   ```
   tabadmin set vizqlserver.session.expiry.timeout --default
   ```

   Alternatively, you can use the shorter `-d` command. For example:

   ```
   tabadmin set vizqlserver.querylimit -d
   ```

3. Next, run the configure command:

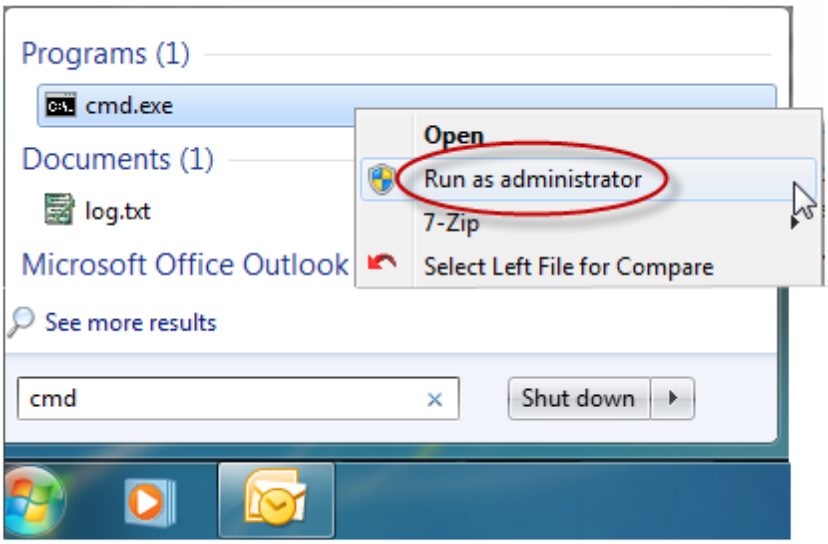
   ```
   tabadmin configure
   ```

4. Start the server.

tabadmin stop

To use tabadmin to stop Tableau Server:

1. Open a command prompt as an administrator:



2. Type the following:

64-bit: `cd "C:\Program Files (x86)\Tableau\Tableau Server\8.0\bin"`
32-bit: `cd "C:\Program Files\Tableau\Tableau Server\8.0\bin"`

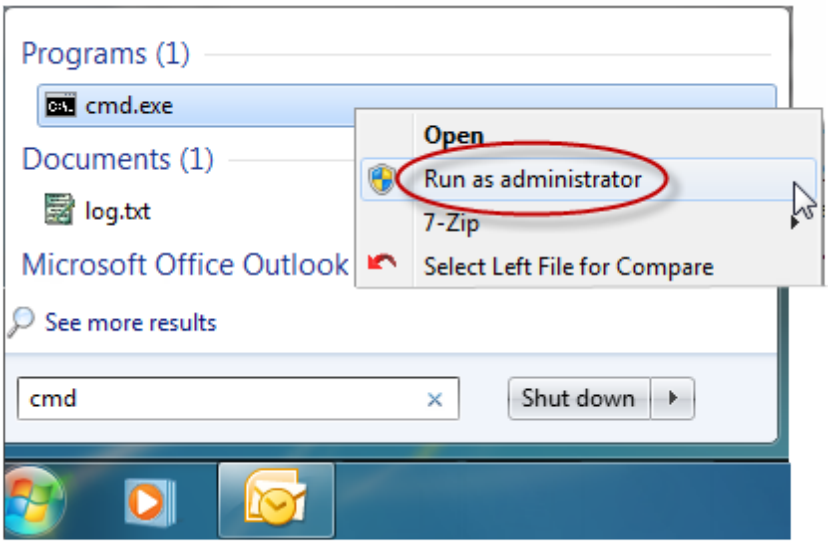
3. Type the following to stop the server:

```
tabadmin stop
```

tabadmin start

To use tabadmin to start Tableau Server:

1. Open a command prompt as an administrator:



2. Type the following:

64-bit: `cd "C:\Program Files (x86)\Tableau\Tableau Server\8.0\bin"`
32-bit: `cd "C:\Program Files\Tableau\Tableau Server\8.0\bin"`

3. Type the following to start the server:

```
tabadmin start
```

Database Maintenance

You can use the tabadmin command line tool to back up and restore your Tableau data. Tableau data includes Tableau Server's own PostgreSQL database, which stores workbook and user metadata, data extract (.tde) files, and server configuration data. To automate backing up this data, you can use the commands described in the topics below, along with the built-in Windows task scheduler.

Back Up the Tableau Data

It is important to back up your Tableau data so you can restore published views and other information in the case of a system failure. The data managed by Tableau Server consists of Tableau's own PostgreSQL database, which contains workbook and user metadata, data extract (.tde) files, and configuration data. When you create a backup, all these things are placed in a single file with a .tsbak extension. If you are running a distributed installation of Tableau Server this step is performed on the primary, even if the data engine, which handles the .tde files, is on a worker.So that it can be an effective backup for you, be sure to store the .tsbak on a machine that's separate from your Tableau Server machine.

You can create a .tsbak file using the procedure below. Tableau Uninstall, which is the first step to upgrading to a new version, also automatically creates a .tsbak file. This same .tsbak file is used to automatically migrate your data to your newer version.

> Running the `backup` command also removes Tableau Server log files older than seven days as well as some of the information displayed in certain Tableau Server Administrative Views.

1. Open a command prompt as an administrator and type the following:

   32-bit:`cd "C:\Program Files\Tableau\Tableau Server\8.0\bin"`

   64-bit:`cd "C:\Program Files (x86)\Tableau\Tableau Server\8.0\bin"`

2. Create a backup file by typing `tabadmin backup <filename> --stop-server`, where `<filename>` is the name or location and name of your backup file. For example:

   ```
   tabadmin backup tabserver --stop-server
   ```

   or

   ```
   tabadmin backup C:\backups\tableau\tabserver --stop-server
   ```

   The `--stop-server` option stops the server for the backup and then restarts it when it is done. If you are doing a simple backup you must either have this option or the `--unsafe` option to backup while the server is running. Backups that are part of a batch operation do not require these options.

   You can also optionally use `-d` to append the current date to the file name and `-t`, followed by a path, to specify a location for temporary files that are created during the backup process. For example:

   ```
   tabadmin backup tabserver --stop-server -d -t C:\-
   mytemp\tableau
   ```

Restore from a Backup

When you restore, the contents of the Tableau PostgreSQL database, data extracts, and configuration files are overwritten with the content in the backup file (.tsbak). If you are running a

distributed installation of Tableau Server, this step is performed on the primary.

To restore from a database backup file:

1. Stop the server by typing:

   ```
   tabadmin stop
   ```

2. Restore the database from a backup file by typing:

   ```
   tabadmin restore <filename>
   ```

   In the above line, replace `<filename>` with the name of the backup file you want to restore from.

   To restore only the data and no configuration settings, type the following instead:

   ```
   tabadmin restore --no-config <filename>
   ```

3. Restart the Tableau Server processes by typing:

   ```
   tabadmin start
   ```

Recover Extracts from a Backup

The file *uninstall-<version>.tsbak* (for example, *uninstall-7.0.tsbak*) is created as part of the uninstall process. After you upgrade to version 8.0, you can use this file to restore data extracts—for example, if you mistakenly deleted the dataengine folder during the upgrade. To use *uninstall-<version>.tsbak* to restore data extracts:

1. Stop the server.
2. From within your version 8.0 Tableau Server bin directory, type the following:

   Windows Server 2012, Windows Server 2008, Windows Vista, Windows 7, Windows 8: `tabadmin restore \ProgramData\Tableau\Tableau Server\uninstall-7.0.tsbak`

   64-bit Windows Server 2003: `tabadmin restore \Program Files (x86)\Tableau\Tableau Server\uninstall-7.0.tsbak`

   32-bit Windows Server 2003: `tabadmin restore \Program Files\Tableau\Tableau Server\uninstall-7.0.tsbak`

Troubleshooting

Use the following topics to troubleshoot issues you may be having with Tableau Server. For tips on troubleshooting trusted authentication, see Troubleshoot Trusted Authentication:

Logs and Temporary Files

The Tableau Service generates several logs and temporary files that can help you understand and track recent activity as well as debug any problems that may arise. If you need to save space on the hard drive, you can occasionally delete these files.

Use the next topics to learn where the log files are located, their purpose, how to archive them, and how to save space by running the clean up command.

Log File Locations

Tableau Server log files can be found in the following folders:

Tableau Service Logs

The following log files track activities related to the web application, database, and index:

```
C:\ProgramData\Tableau\Tableau Server\data\tabsvc
```

VizQL Logs

These log files track activities related to displaying views, such as querying the database and generating images:

```
C:\ProgramData\Tableau\Tableau
Server\data\tabsvc\vizqlserver\Logs
```

Temporary Files

Any file that starts with exe_ in the folder below is a Tableau Server file and can be deleted.

```
C:\ProgramData\Tableau\Tableau Server\temp
```

Archive Log Files

You can archive Tableau Server log files using the ziplogs command. This command creates a zip file containing all of the log files and is useful when you're working with Tableau Support. The ziplogs command does not remove the log files, rather it copies them into a zip file. If you are running a distributed installation of Tableau Server, this step is performed from the primary. Any worker logs will be included in the zip file.

1. Open a command prompt as administrator and navigate to the Tableau Server bin directory. For example:

   32-bit: `cd "C:\Program Files\Tableau\Tableau Server\8.0\bin"`

   64-bit: `cd "C:\Program Files (x86)\Tableau\Tableau Server\8.0\bin"`

2. Stop Tableau Server by typing:

   ```
   tabadmin stop
   ```

3. Create the zip file by typing `tabadmin ziplogs -l -n <filename>` where

`<filename>` is the name of the zipped file you want to create. Choose a unique name with no spaces. Tableau will not overwrite an existing file. For example:

```
tabadmin ziplogs -l -n my_logs
```

If you don't specify a file name, the file is named `logs.zip`. You can also use `-d mm/dd/yyyy` to only include logs generated since a certain date. For example:

```
tabadmin ziplogs -l -n -d 02/14/2013
```

The above command creates a zipped file named `logs.zip` that includes logs dated February 14, 2013 up to the present; earlier logs are excluded. The -n option captures information about the server environment, including which ports are in use. To see a list of all the ziplogs options, type `tabadmin ziplogs -h`.

4. Restart Tableau Server by typing:

   ```
   tabadmin restart
   ```

   You can find the zipped log file in the Tableau Server bin directory.

Remove Log Files

The `cleanup` command removes Tableau Server process logs and HTTP table entries older than seven days in order to save space. This will affect some of the information presented in the Tableau Server Administrative Views.

- At a command prompt type the following:

  ```
  tabadmin cleanup --restart
  ```

  > In general you should shut down the server prior to running this command. However, if the server is running you should include the `--restart` option to ensure a successful clean up and restore.

Handle an Unlicensed Server

Tableau offers two licensing models: user-based and core-based. User-based licensing requires each active user account to be covered by a license. User-based licenses have a defined capacity, or number of users that it allows. Each user has a unique username assigned to him on the server and is required to identify himself when connecting to the server. The software may be installed on a single machine or distributed across any number of machines in a distributed server environment.

Core-based licensing has no constraints on the number of user accounts in the system, but it does restrict the maximum number of processor cores that Tableau Server can use. You may install the server on one or more machines to create a cluster, with the restrictions that the total number of cores in all the machines do not exceed the number of cores you have licensed and that all of the cores on a particular machine are covered by the license.

Unlicensed User-Based Server

The most common reason for a server that has user-based licensing to be unlicensed is an expired product key or an expired maintenance contract. You can see your products keys and add new ones by selecting **Start > All Programs > Tableau Server > Manage Product Keys**.

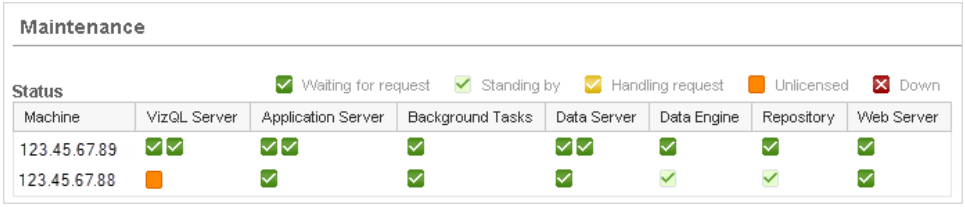
Unlicensed Core-Based Server

A core-based server can become unlicensed for a variety of reasons. A common problem is that the primary or a worker machine has more cores than the license allows. When the server is unlicensed you may not be able to start or administer the server. You can, however, manage your licenses using the tabadmin command line tool. Follow the steps below to see a list of your licenses and number of cores by machine.

1. Open a command prompt and type the following: `cd C:\Program Files (x86)\Tableau\Tableau Server\8.0\bin`
2. Type the following: `tabadmin licenses`.

Handle an Unlicensed VizQL Server Process

There are several status indicators on the Tableau Server Maintenance page that help you understand the state of Tableau Server processes. An orange-color status box, "Unlicensed", indicates that one of the VizQL server processes is unable to retrieve the Tableau Server license information.



There may be several reasons why the process is unable to access this information. For example, there may be network issues preventing a VizQL process, which is running on a worker machine, from communicating with the primary machine. Or, the process may be getting sent more requests than it can accept at that time and can't handle the licensing request. As a result, some of your users may be able to access views while others cannot.

To resolve the problem, stop, then start Tableau Server.

VizQL 'Out of Memory' Error

If a VizQL process reaches its limit of concurrent viewing sessions you may see an 'Out of Memory' error, which will also be written to the *vizqlserver*.txt* logs located here:

`C:\ProgramData\Tableau\Tableau Server\data\tabsvc\vizqlserver\Logs`

The VizQL process doesn't terminate when this error occurs, but it will not accept additional connections. You can handle this problem by doing the following:

- **Increasing the number of VizQL processes**: This may mean that you need to add one or more workers. See Install and Configure Worker Servers for how to do this.
- **Edit vizqlserver.session.expiry.timeout**: Use tabadmin to change the vizqlserver.session.expiry.timeout setting from its default (30 minutes) to a shorter time period such as 10 or 5 minutes. This will allow idle sessions to expire sooner, thus freeing memory for new sessions.

JavaScript API

With Tableau's JavaScript API you can integrate Tableau visualizations into your own web applications. The API lets you tightly control your users' interactions and combine functionality that otherwise couldn't be combined. For example, you can code a single control that filters a group of marks, selects some of those marks, and presents their data for download. See the topics below for details:

- **Requirements**
- **Concepts**
- **Tutorial**
- **API Reference**

Requirements

The requirements for Tableau's JavaScript API are as follows:

Tableau Server: To program with Tableau's JavaScript API you need Tableau Server, version 8.0 or higher. The API is powered by the file **tableau_v8.js**, which is located in the following folder: `Program Files (x86)\Tableau\Tableau Server\8.0\wgserver\public\javascripts\api`. Your web application shouldn't be on the same computer as Tableau Server, but it needs to be able to access it.

Supported browsers: Your end-users can experience the web application you create in most supported web browsers, specifically: Chrome, Firefox, Safari 3.2.1 and higher, and Internet Explorer 8.0 and higher. If you are using Internet Explorer 8.0, it must have compatability mode disabled.

Concepts

This topic is designed for people familiar with JavaScript and object-oriented programming concepts. You should also be familiar with Tableau visualizations from a user's point of view. If you are just getting started, a good place to begin is the Tutorial.

Programming Approach

Tableau's JavaScript API uses an object model. The entry point into the object model is to instantiate a new `Viz` object as follows:

```
var viz = new tableauSoftware.Viz(/* params omitted */);
```

Nearly every Tableau object allows you to navigate back to its parent `Viz` object by way of "parent" properties on the object.

Accessing the API

The API is about programmatically controlling views embedded from Tableau Server. To use it, you need access to a Tableau server running version 8.0 or higher, and a published workbook on that server. The API is provided through the file **tableau_v8.js** (minimized) or **tableau_v8.debug.js**, which install in the following folder on Tableau Server:

```
Program Files (x86)\Tableau\Tableau Server\8.0\wgserver\public\javascripts\api
```

If you don't have a Tableau Server on-premise, Tableau Public is a free Tableau Server implementation that you can also use to access the JavaScript API. The workbooks published to Tableau Public must be packaged .twbx workbooks (data extract and workbook bundled together). Worbooks published to Tableau Public can't use live database connections or rely on extract refresh schedules (Tableau can't connect to databases behind your firewall). The Tutorial uses a workbook published to Tableau Public for its example.

Bootstrapping

There is only one entry point into the API: instantiating a new `Viz` object, which creates the HTML necessary to embed a Tableau visualization. To instantiate a new `Viz` object, simply call the `Viz` constructor via `new`, passing the required `parentElement` and URL parameters and an optional set of options. The URL parameter is where you specify the name of the Tableau server:

```
var placeholderDiv = document.getElementById("tableauViz");
var url = "http://tabserver/views/workbookname/viewname";
var options = {
   hideTabs: true,
   width: "800px",
   height: "700px"
};
var viz = new tableauSoftware.Viz(plachoderDiv, url, options);
```

Trusted Authentication

If Tableau Server is using trusted authentication and the trusted ticket server is configured to return the ticket value itself, add a `ticket` option. For example:

```
var placeholderDiv = document.getElementById("tableauViz");
var url = "http://tabserver/views/workbookname/viewname";
var options = {
   hideTabs: true,
   width: "800px",
   height: "700px",
   ticket: "123456789"
};
var viz = new tableauSoftware.Viz(placeholderDiv, url,
options);
```

If the trusted ticket server is configured to return a full URL, specify the ticket in the URL by first adding `trusted` after the server name, followed by the ticket. For example:

```
var placeholderDiv = document.getElementById("tableauViz");
var url = "http://tabserver/trusted/123456789/views/wor-
kbookname/viewname";
var options = {
   hideTabs: true,
   width: "800px",
   height: "700px",
};
var viz = new tableauSoftware.Viz(placeholderDiv, url,
options);
```

Property Getters and Setters

Getters and setters are always functions that start with `get` or `set`. They can be called multiple times with little performance impact (in other words, they should simply return cached fields or do very simple calculations). Properties are always synchronous and return immediately with the value, rather than having a callback function.

Call Behavior - Asynchronous

By default, calls are asynchronous since many of them may require a roundtrip to the server. Methods use the following naming convention:

- Asynchronous calls are indicated by an `Async` suffix on the method name, for example, `Worksheet.applyFilterAsync()`.
- Asynchronous calls return a `Promise` object, allowing chaining.

The Tableau JavaScript API uses the CommonJS Promises/A standard. The premise behind Tableau's implementation is that asynchronous methods return an object that has a `then` method in the following form:

```
then(fulfilledHandler, errorHandler)
```

The `fulfilledHandler` is called when the promise is fulfilled (on success). The `errorHandler` is called when a promise fails. All arguments are optional and non-function values are ignored.

Chaining Promises

The promised result of an asynchronous method is passed as the parameter to the next `then()` method. For example:

```
var activeSheet;
viz.getWorkbook().activateSheetAsync("Sheet 1")
  .then(selectLemon).then(filterToLemonAndMint);


function selectLemon(sheet) {
  activeSheet = sheet;
  return sheet.selectMarksAsync("Product", "Lemon",
"replace");
}

function filterToLemonAndMint() {
  return activeSheet.applyFilterAsync("Product", ["Lemon",
"Mint"], "replace");
}
```

The result of `activateSheetAsync()` is a promise to eventually return the Sheet object that was activated, which is passed as the first parameter to the `selectLemon()` method. Notice that the `selectLemon()` method returns a `Promise` object (the return value of the `selectMarksAsync()` method), not the result after the marks have been selected. However, since it's a `Promise` object, the next `then()` method is not called until that promise is fulfilled.

If a link in the chain is added after the promise has been fulfilled, the callback will be immediately called with the value that was previously returned. As the programmer, this means you

don't need to determine if the response has already been received from the server. The asynchronous methods will always be called, whether it's now or later.

```
var promise = viz.getWorkbook().activateSheetAsync("Sheet 1");

// Pretend that activatSheeteAsync() has already returned from
the server.
promise.then(callback);


// callback will be called immediately using the Sheet object
// returned from activateSheetAsync()
```

Return Values of Then() Methods

Whatever is returned in a `then()` method will get passed as the first parameter to the next `then()` method. It can be a scalar value (`Number`, `Boolean`, `String`, etc.), an object, or another `Promise`. The infrastructure will automatically wrap non-`Promise` values into a `Promise` value so that they can be chained.

```
viz.getWorkbook().activateSheetAsync("Sheet 1")

.then(function (sheet) {
     return "First link";
})

.then(function (message) {
     if (message === "First link") { alert("Expected"); }
     // no return value here means nothing is passed to the
next link
})

.then(function () {
});
```

Breaking Out of a Chain

Technically, there's no way to break out of a chain since that would invalidate the guarantee that subsequent links in the chain will be called. If there is an exception thrown in part of the chain, the rest of the chain is run but the `errorHandler` is called instead of the `fulfilledHandler`.

If a link in the chain depends on the results of earlier links, then you should write an `if` statement to check your condition. Here's an example:

```
viz.getWorkbook().activateSheetAsync("Sheet 1")

.then(function (sheet) {
  // I'm returning a Promise
  return sheet.selectMarksAsync("Product", "NoProduct",
"replace");
})

.then(function () {
  return viz.getWorkbook().getActiveSheet().g-
etSelectedMarksAsync();
})

.then(function (marks) {
  // The getSelectedMarksAsync call succeeded, but no marks
were selected
  // because there are not any marks corresponding to "NoP-
roduct".
  if (marks.length === 0) {
    throw new Error("No marks selected");
  }

  var firstMarkValue = marks[0].getPairs().get("Prod-
uct").value;
  return sheet.applyFilterAsync("Product", firstMarkValue,
"replace");
})

.then(function (filterName) {
  // applyFilterAsync succeeded

}, function(err) {
  if (err.message === "No marks selected") {
    alert("This was caused by the first link above");
  }
})

.otherwise(function (err) {
  alert("We handled the error above, so it's not propagated to
this handler.");
});
```

If a callback is not provided (or is null or undefined), then the results are passed to the next link in the chain:

```
viz.getWorkbook().activateSheetAsync("Sheet 1")
.then()
.then(function (sheet) {
  // this is called
});
```

In this way, you can specify a single `otherwise` function to handle all errors in the chain. The `always` function works the same way, but it is called regardless of success or failure. The `then/otherwise/always` functions work similarly to a `try/catch/finally` block.

```
viz.getWorkbook().activateSheetAsync("Sheet 1")
.then(function () {
     return sheet.selectMarksAsync(...);
})
.then(function (marks) {
     // Do something with the marks.
})
.otherwise(function (err) {
     // I'm handling all errors in one place.
     console.log(err.message);
})
.always(function () {
    // Do some cleanup or logging
});
```

Collections

Many classes have collections of items, where each item has a key (typically an ID or a name). Examples include a collection of sheets keyed by name or the list of parameters on a sheet keyed by name. Collections are publicly immutable, exposing read-only functionality. Each Collection array is keyed with its elements' identifiers. For example, the result of `Workbook.getPublishedSheetsInfo()` is an array with the index corresponding to the position of the sheet in the workbook. It is also keyed by the sheet name so that you can access it like this:

```
var sheet = workbook.getPublishedSheetsInfo()[0];
```

```
var sameSheet = workbook.getPublishedSheetsInfo().get("Sheet
1");
```

Collection Interface

Name	Return Type	Description
get(key : string)	Collection item type	Gets the element in the collection associated with the key, or undefined if there is nothing associated with it.
has(key : string)	bool	Returns true if there is an element in the collection associated with the key; otherwise, false.

Events

The `Viz` class acts as the central event hub. This way you only have to go to one place for all events. It also means that events can be raised on an object that may not have been created yet. For example, the `marksselection` event can be raised for a particular sheet even though the `Sheet` object hasn't been created yet. Each event contains an anonymous object with information pertaining to that event, such as the sheet the event occurred on.

Listening to an event is done by calling `Viz.addEventListener(type, callback)` and passing in a function callback. Here's an example of listening to an event:

```
viz.addEventListener("marksSelection", function (marks) {
   changeMySelectionUI(marks);
});
```

Removing a listener is done by calling `Viz.removeEventListener(type, listener)` and passing in the same callback function that was passed into `Viz.addEventListener()`. For example:

```
function changeMySelectionUI(marks) {
   viz.removeEventListener("marksSelection", change-
MySelectionUI);
}
viz.addEventListener("marksSelection", changeMySelectionUI);
```

Events are multicast delegates, meaning that multiple listeners are supported. The order in which notifications are called is not specified. Every event callback takes a single object containing a pointer to the `Viz` that raised the event. Each event also adds additional fields to the event, as specified in the API Reference.

Filtering

When you program filtering you are mimicking the user behavior of clicking a filter in a view to narrow down the data that is displayed. Here's an example of filtering on a single value:

```
worksheet.applyFilterAsync("Container", "Jumbo Box",
   tableauSoftware.FilterUpdateType.REPLACE);
```

There is a difference between querying existing filter state and setting new or existing filters. Querying filters is done via `Worksheet.getFiltersAsync()` which returns a collection of `Filter` classes. Setting filters is done via `Worksheet.applyFilterAsync` (and its variants) and is a function call that doesn't require you to instantiate a `Filter` class.

When you specify fields in a filter, you should use the caption as shown in the user interface, not the database field name. For example, you should use **Container** (the caption) instead of **Product Container** (the actual field name). In some cases, Tableau Desktop renames fields after they've been dragged to a shelf. For example the **Date** field might be renamed to **YEAR (Date)** after being dragged to the rows shelf. In this case, you should use **YEAR(Date)** as the parameter. The exception is hierarchical filters, which use the full hierarchical name (for example, `[Product].[All Product].[Espresso]`). Captions can use the optional `[]` delimiters around names.

Here are samples for many types of filtering:

```
var worksheet;
viz.getWorkbook().activateSheetAsync("Sheet 4").then(function
(sheet) {
  worksheet = sheet;
})

// Single value
.then(function () {
  return worksheet.applyFilterAsync("Product Type", "Coffee",
    tableauSoftware.FilterUpdateType.REPLACE);
})

// Multiple values
.then(function () {
  return worksheet.applyFilterAsync(
    "Product Type", ["Coffee", "Tea"],
    tableauSoftware.FilterUpdateType.REPLACE);
})
```

```
// Multiple Values - adding and removing
.then(function () {
  return worksheet.applyFilterAsync("Product", ["Lemon",
"Mint"],
    tableauSoftware.FilterUpdateType.ADD);
})

.then(function () {
  return worksheet.applyFilterAsync("Product", ["Caffe Latte",
"Green Tea"],
    tableauSoftware.FilterUpdateType.REMOVE);
})

// All values
.then(function () {
  return worksheet.applyFilterAsync("Product Type", "",
    tableauSoftware.FilterUpdateType.ALL);
})

// Date Range
.then(function () {
  return; worksheet.applyRangeFilterAsync("Date", {
    min: new Date(Date.UTC(2010, 3, 1)),
    max: new Date(Date.UTC(2010, 12, 31))
  });
})

// Clearing a Filter
.then(function () {
  return worksheet.clearFilterAsync("Date");
})


// Relative Date
.then(function () {
  return worksheet.applyRelativeDateFilterAsync("Date", {
    anchorDate: new Date(Date.UTC(2011, 5, 1)),
    periodType: tableauSoftware.PeriodType.YEAR,
    rangeType: tableauSoftware.DateRangeType.LASTN,
    rangeN: 1
  });
})
```

```
// Quantitative Filters
// SUM(Sales) > 2000 and SUM(Sales) < 4000
.then(function () {
  return worksheet.applyRangeFilterAsync("SUM(Sales)", {
    min: 2000,
    max: 4000
  });
})

// SUM(Sales) > 1000
.then(function () {
  return worksheet.applyRangeFilterAsync("SUM(Sales)", {
    min: 1000
  });
})

// Hierarchical Filters - selecting all on a level
.then(function () {
  return worksheet.applyHierarchicalFilterAsync("[Product].
[Product Categories]", {
    levels: [0, 1]
  }, tableauSoftware.FilterUpdateType.ADD);
}, function (err) { /* ignore errors */ })

// Hierarchical Filters - adding one item
.then(function () {
  return worksheet.applyHierarchicalFilterAsync(
    "[Product].[Product Categories].[Product Name]",
    "Accessories.Bike Racks.Hitch Rack - 4-Bike",
    tableauSoftware.FilterUpdateType.REPLACE);
}, function (err) { /* ignore errors */ })

// Hierarchical Filters - adding multiple items
.then(function () {
  return worksheet.applyHierarchicalFilterAsync(
  "[Product].[Product Categories].[Product Name]",
  [
    "Accessories.Bike Racks.Hitch Rack - 4-Bike",
    "Accessories.Bike Stands.All-Purpose Bike Stand"
  ],
  tableauSoftware.FilterUpdateType.REPLACE);
```

```
}, function (err) { /* ignore errors */ })

.otherwise(function (err) {
  console.log(err);
});
```

Selecting Marks

Selecting marks is almost identical to filtering. For filtering,you use one of the `Worksheet.applyFilterAsync()` methods. For selecting marks, you use `Worksheet.selectMarksAsync()`. The parameters for mark selection are almost identical to those used for filtering.

```
worksheet.selectMarksAsync("Product", "Caffe Latte",
   tableauSoftware.SelectionUpdateType.REPLACE);
```

Here are samples of other types of selecting you can use:

```
var worksheet;
viz.getWorkbook().activateSheetAsync("Sheet 4").then(function
(sheet) {
  worksheet = sheet;
})

// Single dimensions work just like filtering

// Single dimension - single value
.then(function () {
  return worksheet.selectMarksAsync("Product", "Mint",
    tableauSoftware.SelectionUpdateType.REPLACE);
})

// Single dimension - Multiple values
.then(function () {
  return worksheet.selectMarksAsync(
    "Product", ["Chamomile", "Mint"],
    tableauSoftware.SelectionUpdateType.REPLACE);
})

// Single dimension - Multiple values (delta)
```

```
.then(function () {
  return worksheet.selectMarksAsync("Product", ["Lemon", "Earl
Grey"],
    tableauSoftware.SelectionUpdateType.ADD);
})
.then(function () {
  return worksheet.selectMarksAsync(
    "Product", ["Chamomile", "Earl Grey"],
    tableauSoftware.SelectionUpdateType.REMOVE);
})

// Quantitative selection
.then(function () {
  return worksheet.selectMarksAsync({
    "State": ["Florida", "Missouri"],
    "SUM(Sales)": { min: 3000, max: 4000 }
}, tableauSoftware.SelectionUpdateType.REPLACE);
})

// Hierarchical dimensions
.then(function () {
  return worksheet.selectMarksAsync(
    "[Product].[Product Categories].[Category]",
    "Bikes",
    tableauSoftware.SelectionUpdateType.REPLACE);
}, function (err) { /* ignore errors */ })

// Multiple dimensions - multiple values
// ((State = Washington OR Oregon) AND Product = Lemon)
// OR
// (State = Oregon AND Product = Chamomile)
.then(function () {
  return worksheet.selectMarksAsync({
    "State": ["Washington", "Oregon"],
    "Product": "Lemon"
}, tableauSoftware.SelectionUpdateType.REPLACE);
})
.then(function () {
  return worksheet.selectMarksAsync({
    "State": "Oregon",
    "Product": "Chamomile"
}, tableauSoftware.SelectionUpdateType.ADD);
```

```
})

// Round-tripping selection
.then(function () {
  return worksheet.selectMarksAsync(
    "Product",
    "Lemon",
tableauSoftware.SelectionUpdateType.REPLACE);
})
.then(function () {
  return worksheet.getSelectedMarksAsync();
}).then(function (marks) {
  // filter out only the Washington and Oregon marks
  var onlyWashingtonAndOregon = [];
  for (var i = 0, len = marks.length; i < len; i++) {
    var m = marks[i];
    var pair = m.getPairs().get("State");
    if (pair &&
      (pair.value === "Washington" || pair.value === "Ore-
gon")) {
      onlyWashingtonAndOregon.push(m);
    }
  }
  return worksheet.selectMarksAsync(
    onlyWashingtonAndOregon,
    tableauSoftware.SelectionUpdateType.REPLACE);
})

.otherwise(function (err) {
  console.log(err);
});
```

API Reference

Style and Conventions

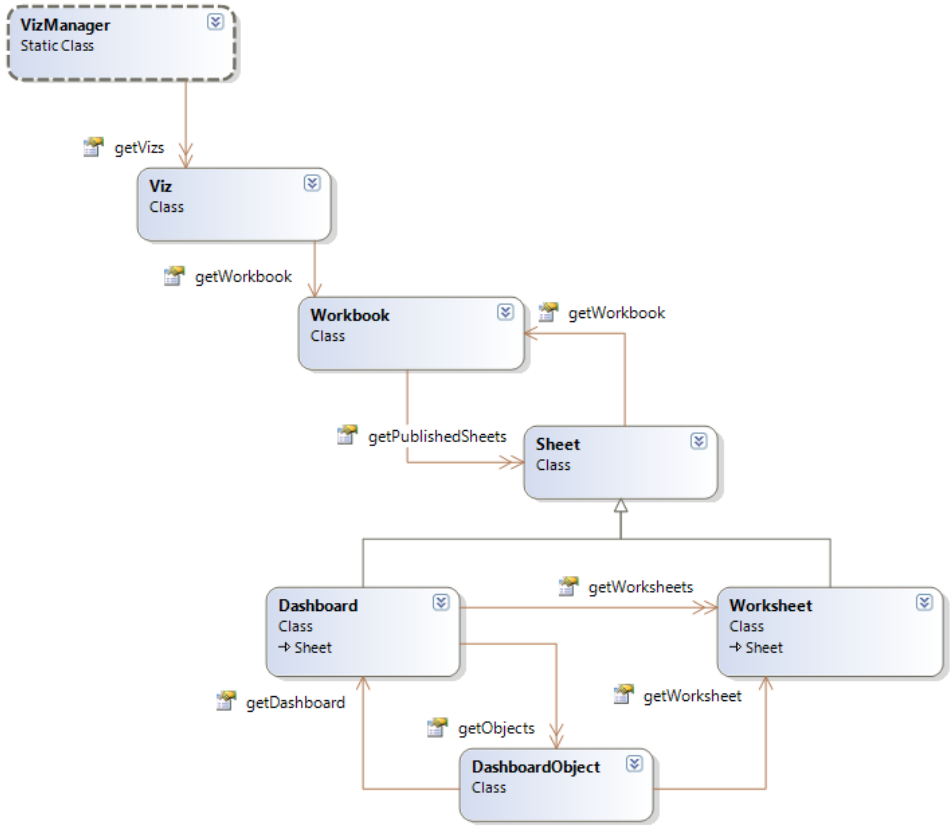
Tableau's JavaScript API follows these JavaScript standards:

- **Classes** are PascalCase (initial capital letter, with each subsequent word capitalized)
- **Namespaces, methods, parameters, and variables** are camelCase (initial lower-case letter, with each subsequent word capitalized)
- **Constants and enum values** are UPPERCASE_UNDERSCORE_DELIMITED

- **Protected variables or methods** start with an initial underscore '_' character, indicating that it should not be referenced by the programmer.

Top-Level Class Diagram

The following class diagram shows the relationships between the top-level classes, as well as the inheritance hierarchy for the `Sheet`, `Dashboard`, and `Worksheet` classes. Note that there's always a way to traverse back up the containment hierarchy with parent pointers, with the exception of `VizManager`, as it's a static class and always accessible.



Asynchronous and Error Classes

Promise Class

Represents a promise to return a value from an asynchronous method in the future. The Tableau JavaScript API implements the Promises/A specification.

Methods

Name	Return Type	Description
then(callback: Function, errback: Func-	Promise	Creates a link in the asynchronous callable chain. The callback function is called on success. The errback function is called when there is an error. Both parameters are

tion)		optional.
always(callback: Function)	Promise	Registers a callback that will be called when a promise is resolved or rejected. Shortcut for `then(callback, callback)`.
otherwise (errback: Function)	Promise	Registers a rejection handler. Shortcut for then(null, errback).

TableauException Class

The `TableauException` class is not a real class, but simply adds an id field to the standard JavaScript `Error` object when an exception is thrown from within the API. As the programmer, this allows you to uniquely identify the error without having to parse the error string. It also allows you to add localized messages.

Constructor

There is no public constructor. The only way to get a reference to a TableauException is within a `catch` block.

Fields

Name	Type	Description
tableauSoftwareErrorCode	ErrorCode	Represents an `ErrorCode` enum value.
message	string	This is already defined on the standard Error object, but the message will contain a description of the exception that the API code specifies.

ErrorCode Enum

Here's a list of the different exceptions that the API can throw:

Name	Return Type	Description
BROWSER_NOT_CAPABLE	browserNotCapable	The browser is not capable of supporting the Tableau JavaScript API.
DOWNLOAD_WORKBOOK_NOT_ALLOWED	downloadWorkbookNotAllowed	The permissions on a workbook or a view do not allow downloading the workbook.
FILTER_CANNOT_BE_PERFORMED	filterCannotBePerformed	An error occurred while attempting to perform a filter operation.
INDEX_OUT_OF_RANGE	indexOutOfRange	Attempted to switch to a sheet by index that does not exist in the workbook.
INTERNAL_ERROR	internalError	An error occurred within the Tableau JavaScript API. Contact Tableau Support.

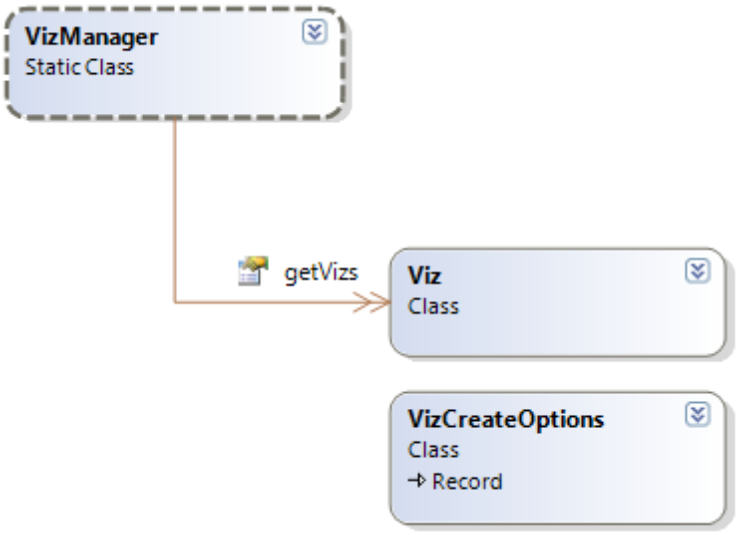
INVALID_CUSTOM_VIEW_NAME	invalidCustomViewName	An operation was attempted on a custom view that does not exist.
INVALID_DATE_PARAMETER	invalidDateParameter	An invalid date was specified in a method that required a date parameter.
INVALID_FILTER_FIELDNAME	invalidFilterFieldName	A filter operation was attempted on a field that does not exist in the data source.
INVALID_FILTER_FIELDNAME_OR_VALUE	invalidFilterFieldNameOrValue	Either a filter operation was attempted on a field that does not exist in the data source, or the value supplied in the filter operation is the wrong data type or format.
INVALID_FILTER_FIELDVALUE	invalidFilterFieldValue	A filter operation was attempted using a value that is the wrong data type or format.
INVALID_PARAMETER	invalidParameter	A parameter is not the correct data type or format. The name of the parameter is specified in the `Error.message` field.
INVALID_SELECTION_DATE	invalidSelectionDate	An invalid date value was specified in a `Sheet.selectMarksAsync()` call for a date field.
INVALID_SELECTION_FIELDNAME	invalidSelectionFieldName	A field was specified in a `Sheet.selectMarksAsync()` call that does not exist in the data source.
INVALID_SELECTION_VALUE	invalidSelectionValue	An invalid value was specified in a `Sheet.selectMarksAsync()` call.
INVALID_SIZE	invalidSize	A negative size was specified or the maxSize value is less than minSize in Sheet.changeSizeAsync().
INVALID_SIZE_BEHAVIOR_ON_WORKSHEET	invalidSizeBehaviorOnWorksheet	A behavior other than `SheetSizeBehavior.AUTOMATIC` was specified in `Sheet.changeSizeAsync()` when the sheet is a Worksheet instance.
INVALID_URL	invalidUrl	The URL specified in the `Viz` class constructor is not valid.
MISSING_	missingMaxSize	The `maxSize` field is missing in

MAX_SIZE		`Sheet.changeSizeAsync()` when specifying `SheetSizeBehavior.ATMOST`.
MISSING_MIN_SIZE	missingMinSize	The `minSize` field is missing in `Sheet.changeSizeAsync()` when specifying `SheetSizeBehavior.ATLEAST`.
MISSING_MINMAX_SIZE	missingMinMaxSize	Either or both of the `minSize` or `maxSize` fields is missing in `Sheet.changeSizeAsync()` when specifying `SheetSizeBehavior.RANGE`.
MISSING_RANGEN_FOR_RELATIVE_DATE_FILTERS	missingRangeNForRelativeDateFilters	The `rangeN` field is missing for a relative date filter of type `LASTN` or `NEXTN`.
NO_URL_FOR_HIDDEN_WORKSHEET	noUrlForHiddenWorksheet	An attempt was made to access `Sheet.getUrl()` on a hidden sheet. Hidden sheets do not have URLs.
NO_URL_OR_PARENT_ELEMENT_NOT_FOUND	noUrlOrParentElementNotFound	One or both of the `parentElement` or the URL parameters is not specified in the `Viz` constructor.
NOT_ACTIVE_SHEET	notActiveSheet	An operation was attempted on a sheet that is not active or embedded within the active dashboard.
NULL_OR_EMPTY_PARAMETER	nullOrEmptyParameter	A required parameter was not specified, null, or an empty string/array.
SERVER_ERROR	serverError	A general-purpose server error occurred. Details are contained in the `Error` object.
SHEET_NOT_IN_WORKBOOK	sheetNotInWorkbook	An operation was attempted on a sheet that does not exist in the workbook.
STALE_DATA_REFERENCE	staleDataReference	An operation is performed on a `CustomView` object that is no longer valid (it has been removed).
UNSUPPORTED_EVENT_	unsupportedEventName	An unknown event name was specified in the call to `Viz.addEventListener` or `Viz.r-`

NAME		emoveEventListener.
VIZ_ALREADY_IN_MANAGER	vizAlreadyInManager	A `Viz` object has already been created as a child of the `parentElement` specified in the `Viz` constructor.

Viz Classes

Class Diagram



VizManager Class

Manages all of the `Viz` instances on the page, but does not manage vizzes (views) earlier than version 8.0. This is a static class, meaning that all properties and methods are static and there is only a single instance of the class.

Properties

Name	Type	Description
getVizs()	Viz[]	Collection of version 8.0 views on the hosting page.

Viz Class

Wraps an `<iframe>` showing one or more sheets in a Tableau workbook. Contains all of the chrome surrounding the view.

Constructor

Signature	Description
Viz(parentElement: domNode, url: string, options: VizCreateOptions)	Creates a new Tableau `Viz` inside of the given HTML container, which is typically a `<div>` element. Each option as well as the options parameter is optional. If there is already a `Viz` associated with the `parentElement`, an exception is thrown. Before reusing the `parentElement` you must first call `dispose()`.

Properties

Name	Type	Description
getAreTabsHidden()	bool	Indicates whether the tabs are displayed in the UI. It does not actually hide individual tabs.
getIsToolbarHidden()	bool	Indicates whether the toolbar is displayed.
getIsHidden()	bool	Indicates whether the viz is displayed on the hosting page.
getParentElement()	domNode	Returns the node that was specified in the constructor.
getUrl()	string	The URL of the viz, as specified in the constructor
getWorkbook()	Workbook	One `Workbook` is supported per viz.
getAreAutomaticUpdatesPaused()	bool	Indicates whether automatic updates are currently paused.

Events

Events are added or removed via the following two calls.

Name	Return Type	Description
addEventListener(type: TableauEventName, listener: Function)	None	Adds an event listener to the specified event.
removeEventListener(type: TableauEventName, listener: Function)	None	Removes an event listener from the specified event.

Methods

Name	Type	Description
show() **hide()**	None	Shows or hides the `<iframe>` hosting the viz.
dispose()	None	Cleans up any resources associated with the viz, removes the viz from the VizManager, and removes any DOM elements from `parentElement`. Basically, it restores the page to what it was before instantiating a new `Viz` object.
pauseAutomaticUpdatesAsync() **resumeAutomaticUpdatesAsync()**	None	Pauses or resumes layout updates. This is useful if you are resizing the viz or performing multiple calls that

toggleAutomaticUpdatesAsync()		could affect the layout.
revertAllAsync()	Promise	Equivalent to clicking on the Revert All toolbar button, which restores the workbook to its starting state.
refreshData()	None	Equivalent to clicking on the Refresh Data toolbar button.
showDownloadWorkbookDialog()	None	Equivalent to clicking on the Download toolbar button, which downloads a copy of the original workbook.
showExportImageDialog()	None	Equivalent to clicking on the Export Image toolbar button, which creates a PNG file of the current viz.
showExportPDFDialog()	None	Equivalent to clicking on the Export PDF toolbar button, which shows a dialog allowing the user to select options for the export.
showExportDataDialog(worksheetInDashboard: Sheet or SheetInfo or string)	None	Shows the Export Data dialog, which is currently a popup window. The `worksheetInDashboard` parameter is optional. If not specified, the currently active `Worksheet` is used.
showExportCrossTabDialog(worksheetInDashboard: Sheet or SheetInfo or string)	None	Shows the Export CrossTab dialog. The `worksheetInDashboard` parameter is optional. If not specified, the currently active `Worksheet` is used.
showShareDialog()	None	Equivalent to clicking on the Share toolbar button, which displays a dialog allowing the user to share the viz by email or by embedding its HTML in a web page.
setFrameSize(width: int, height: int)	None	Sets the size of the iFrame, which causes the viz to expand or collapse to fit the iFrame if the viz's current sheet's size is set to `AUTOMATIC`.

VizCreateOptions Record

These are the options that are specified in the `Viz` constructor.

Fields

Name	Type	Description
hideTabs	bool	Indicates whether tabs are hidden or shown.
hideToolbar	bool	Indicates whether the toolbar is hidden or

		shown.
toolbarPosition	ToolbarPosition	Indicates where the toolbar should be placed if `hideToolbar` is false.
height	string	Can be any valid CSS size specifier. If not specified, defaults to 600px.
width	string	Can be any valid CSS size specifier. If not specified, defaults to 800px.
onFirstInteractive	callback(viz: Viz)	Callback when the viz first becomes interactive. This is only called once, but it's guaranteed to be called. If the viz is already interactive, it will be called immediately.

ToolbarPosition Enum

Enumeration

Name	Description
TOP	Positions the toolbar along the top of the viz.
BOTTOM	Positions the toolbar along the bottom of the viz.

Viz Event Classes

TableauEventName Enum

These are strings passed to Viz.addEventListener/removeEventListener. Note that the value of the enums are all lowercase strings with no underscores. For example, `CUSTOM_VIEW_LOAD` is `customviewload`. Either the fully-qualified enum (`tableauSoftware.TableauEventName.FILTER_CHANGE`) or the raw string (`filterchange`) is acceptable.

Name	Event Class Passed in the Callback	Description
CUSTOM_VIEW_LOAD	CustomViewEvent	Raised when a custom view has finished loading.
CUSTOM_VIEW_REMOVE	CustomViewEvent	Raised when the user removes a custom view.
CUSTOM_VIEW_SAVE	CustomViewEvent	Raised when the user saves a new or existing custom view.
CUSTOM_VIEW_SET_DEFAULT	CustomViewEvent	Raised when a custom view has been made the default view for this viz.
FILTER_CHANGE	FilterEvent	Raised when any filter has changed state. The viz may not be interactive yet.
MARKS_SELECTION	MarksEvent	Raised when marks are selected or deselected.
PARAMETER_VALUE_CHANGE	ParameterEvent	Raised when any parameter has changed state.
TAB_SWITCH	TabSwitchEvent	Raised after the tab switched, but the viz may not yet be interactive.

TableauEvent Class

Properties

Name	Type	Description
getViz()	Viz	Gets the Viz object associated with the event.
getEventName()	TableauEventName	Gets the name of the event which is a string, but is also one of the items in the TableauEventName enum.

CustomViewEvent Class

Properties

Name	Type	Description
getViz()	Viz	Gets the Viz object associated with the event.
getEventName()	TableauEventName	Gets the name of the event which is a string, but is also one of the items in the TableauEventName enum.

Methods

Name	Return Type	Description
getCustomViewAsync()	Promise<CustomView>	Gets the CustomView object associated with the event.

FilterEvent Class

Properties

Name	Type	Description
getViz()	Viz	Gets the `Viz` object associated with the event.
getWorksheet()	Worksheet	Gets the `Worksheet` object associated with the event.
getEventName()	TableauEventName	Gets the name of the event which is a string, but is also one of the items in the TableauEventName enum.
getFieldName()	string	Gets the name of the field.

Methods

Name	Return Type	Description
getFilterAsync()	Promise<Field>	Gets the Field object associated with the event.

MarksEvent Class

Properties

Name	Type	Description
getViz()	Viz	Gets the `Viz` object associated with the event.

getWorksheet()	Worksheet	Gets the `Worksheet` object associated with the event.
getEventName()	TableauEventName	Gets the name of the event which is a string, but is also one of the items in the `TableauEventName` enum.

Methods

Name	Return Type	Description
getMarksAsync()	Promise<Mark[]>	Gets the selected marks on the `Worksheet` that triggered the event.

ParameterEvent Class

Properties

Name	Type	Description
getViz()	Viz	Gets the `Viz` object associated with the event.
getEventName()	TableauEventName	Gets the name of the event which is a string, but is also one of the items in the `TableauEventName` enum.
getParameterName()	string	Gets the name of the parameter that changed.

Methods

Name	Return Type	Description
getParameterAsync()	Promise<Parameter>	Gets the `Parameter` object that triggered the event.

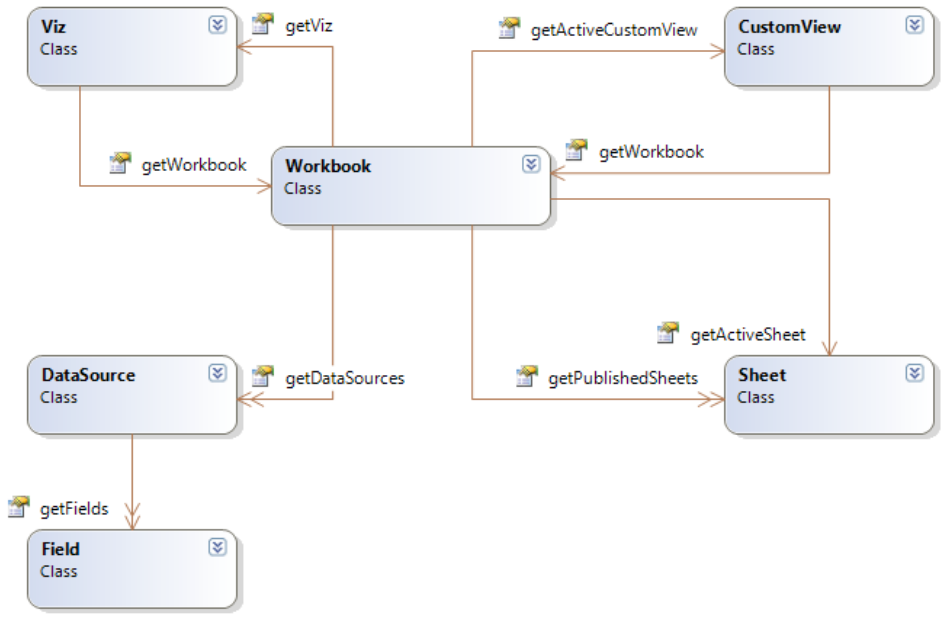
TabSwitchEvent Class

Properties

Name	Type	Description
getViz()	Viz	Gets the `Viz` object associated with the event.
getEventName()	TableauEventName	Gets the name of the event which is a string, but is also one of the items in the `TableauEventName` enum.
getOldSheetName()	string	Gets the name of the sheet that was active before the tab switch event occurred.
getNewSheetName()	string	Gets the name of the sheet that is currently active.

Workbook Classes

Class Diagram



Workbook Class

A Workbook has a collection of Sheets represented as tabs. It also has associated objects like `DataSources` and `CustomViews`.

Properties

Name	Type	Description
getViz()	Viz	Gets the `Viz` object that contains the workbook.
getActiveSheet()	Sheet	Gets the currently active sheet (the active tab)
getActiveCustomView()	CustomView	Gets the currently active custom view, or null if no custom view is active.
getPublishedSheetsInfo()	SheetInfo[]	Note that this is synchronous, meaning that all of the sheets are expected when loaded.
getName()	string	Gets the name of the workbook saved to the server. Note that this is not necessarily the file name.

Methods

Name	Return Type	Description
revertAllAsync()	Promise	Reverts the workbook to its last saved state.

getParametersAsync()	Promise<Parameter[]>	Fetches the parameters for this workbook.
changeParameterValueAsync(name: string, value: object)	Promise<Parameter>	Changes the value of the `Parameter` with the given name and returns the new `Parameter`.
getCustomViewsAsync()	Promise<CustomView[]>	Gets the collection of `CustomView` objects associated with the workbook.
showCustomViewAsync(customViewName: string)	Promise<CustomView>	Changes the viz to show the named saved state.
removeCustomViewAsync(customViewName: string)	Promise<CustomView>	Removes the named custom view.
rememberCustomViewAsync(customViewName: string)	Promise<CustomView>	Remembers the current state of the workbook by assigning a custom view name.
setActiveCustomViewAsDefaultAsync()	None	Sets the active custom view as the default.

DataSource Class

The `Workbook` contains one or more data sources. Each `Worksheet` will have a primary data source and can have multiple secondary data sources.

Properties

Name	Type	Description
getName()	string	The name of the `DataSource` as seen in the UI.
getIsPrimary()	bool	Indicates whether this `DataSource` is a primary or a secondary data source.

getFields()	Field []	Gets an array of `Fields` associated with the `DataSource`.

Field Class

A field contains information about what data source it belongs to, its role, and the ability to fetch the domain values.

Properties

Name	Type	Description
getName()	string	Gets the field name (i.e. caption).
getAggregation()	FieldAggregationType	Gets the type of aggregation, which is one of the following values: `SUM`, `AVG`, `MIN`, `MAX`, `STDEV`, `STDEVP`, `VAR`, `VARP`, `COUNT`, `COUNTD`, `MEDIAN`, `ATTR`, `NONE`, `YEAR`, `QTR`, `MONTH`, `DAY`, `HOUR`, `MINUTE`, `SECOND`, `WEEK`, `WEEKDAY`, `MONTHYEAR`, `MDY`, `END`, `TRUNC_YEAR`, `TRUNC_QTR`, `TRUNC_MONTH`, `TRUNC_WEEK`, `TRUNC_DAY`, `TRUNC_HOUR`, `TRUNC_MINUTE`, `TRUNC_SECOND`, `QUART1`, `QUART3`, `SKEWNESS`, `KURTOSIS`, `INOUT`, `USER`
getDataSource()	DataSource	Gets the data source to which this field belongs.
getRole()	FieldRoleType	One of the following values: `DIMENSION`, `MEASURE`, `UKNOWN`

CustomView Class

Represents a named snapshot of the workbook at a specific moment.

Properties

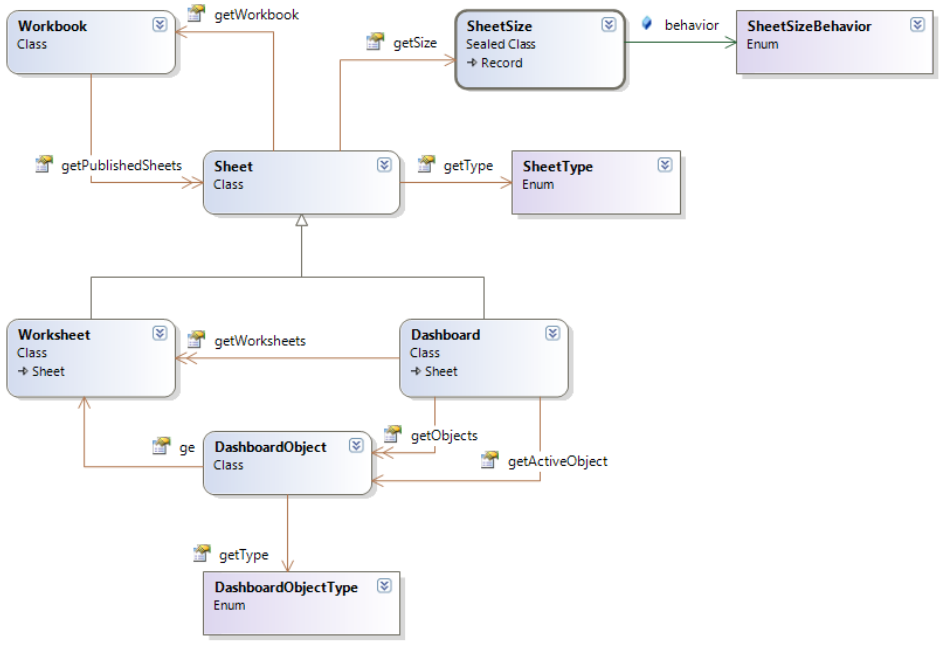
Name	Type	Description
getName() **setName()**	string	User-friendly name for the custom view
getAdvertised() **setAdvertised()**	bool	Indicates whether the custom view is public or private.
getDefault()	bool	Gets or sets whether this is the default custom view.
getOwnerName()	string	Gets the user that created the custom view.
getUrl()	string	Unique URL to load this view again.
getWorkbook()	Workbook	Gets the `Workbook` to which this `CustomView` belongs.

Methods

Name	Return Type	Description
saveAsync()	Promise<CustomView>	After `saveAsync()` is called, the result of the `getUrl` method is no longer blank.

Sheet Classes

Class Diagram



SheetInfo Class

Contains information about a `Worksheet` or a `Dashboard` and no methods. Returned as part of `Workbook.getPublishedSheetsInfo()`.

Constructor

There is no public constructor. You can only get instances of this class from `Workbook.getPublishedSheetsInfo()`.

Properties

Name	Type	Description
getName()	string	Gets the name of the sheet.
getIndex()	int	Gets the index of the sheet within the published tabs. Note that hidden tabs are still counted in the ordering, as long as they are published.
getIsActive()	bool	Gets a value indicating whether the sheet is the currently active sheet.Due to a technical limitation in version 8.0, this will always return false if the object is a `Worksheet` instance that is part of a `Dashboard`.
getIsHidden()	bool	Gets a value indicating whether the sheet is hidden in the UI. Note that if the entire tab control is hidden, it does not affect the state of this flag. This sheet may still report that it is visible even when the tabs control is hidden.
getSheetType()	SheetType	Gets the type of the sheet. `SheetType` is an enum with the following values: `WORKSHEET` and `DASH-`

		BOARD.
getSize()	SheetSize	Gets the size information that the author specified when publishing the workbook.
getUrl()	string	Gets the URL for this sheet.
getWorkbook()	Workbook	Gets the `Workbook` to which this `Sheet` belongs.

Sheet Class

Constructor

There is no public constructor. You can only get instances of this class from `Workbook.getActiveSheet()` or `Dashboard.getObjects()`.

Properties

Name	Type	Description
getName()	string	Gets the name of the sheet.
getIndex()	int	Gets the index of the sheet within the published tabs. Note that hidden tabs are still counted in the ordering, as long as they are published.
getIsActive()	bool	Gets a value indicating whether the sheet is the currently active sheet.
getIsHidden()	bool	Gets a value indicating whether the sheet is hidden in the UI. Note that if the entire tab control is hidden, it does not affect the state of this flag. This sheet may still report that it is visible even when the tabs control is hidden.
getSheetType()	SheetType	Gets the type of the sheet. `SheetType` is an enum with the following values: `WORKSHEET` and `DASHBOARD`.
getSize()	SheetSize	Gets the size information that the author specified when publishing the workbook.
getUrl()	string	Gets the URL for this sheet.
getWorkbook()	Workbook	Gets the `Workbook` to which this `Sheet` belongs.

Methods

Name	Return Type	Description
changeSizeAsync(size: SheetSize)	Promise<SheetSize>	Sets the size information on a sheet. Note that if the sheet is a `Worksheet`, only `SheetSizeBehavior.AUTOMATIC` is allowed since you can't actually set a `Worksheet` to a fixed size.

SheetSize Record

Describes the way a Sheet should be sized.

Fields

Name	Type	Description
behavior	SheetSizeBehavior	Contains an enumeration value of one of the following: `AUTOMATIC`, `EXACTLY`, `RANGE`, `ATLEAST`, and `ATMOST`.
maxSize	int	This is only defined when behavior is `EXACTLY`, `RANGE` or `ATMOST`.
minSize	int	This is only defined when behavior is `EXACTLY`, `RANGE`, or `ATLEAST`.

Worksheet Class

Represents a worksheet tab in the workbook or a dashboard object. These are two separate concepts: a worksheet and a worksheet instance. However, for simplicity in the API, they are both combined into the `Worksheet` class.

Constructor

There is no public constructor. You can only get instances of this class from `Workbook.getPublishedSheets()` or `Dashboard.getObjects()`.

Properties

Name	Type	Description
getParentDashboard()	Dashboard	Returns the `Dashboard` object to which this `Worksheet` belongs (if it's on a dashboard). Otherwise, it returns null.

Methods

Name	Return Type	Description
getDataSourcesAsync()	Promise<DataSource[]>	Gets the primary and all of the secondary data sources for this worksheet. Note that by convention the primary data source should always be the first element.

Filtering methods are described in Worksheet Class (Filtering). Marks selection methods are described in Worksheet Class (Selecting Marks).

Dashboard Class

Contains a collection of `DashboardObject` instances and a notion of the active object.

Constructor

There is no constructor. You get an instance of this from `Workbook.getPublishedSheets()`.

Properties

Name	Type	Description
getObjects()	DashboardObject[]	Gets the collection of objects.
get-	Worksheet[]	Gets the collection of worksheets contained in the

Worksheets ()	dashboard. Note that this is a helper method and is equivalent to looping through `getObjects()` and collecting all of the `Dashboard.Worksheet` pointers when `DashboardObject.getType() === tableau.Software.DashboardObjectType.WORKSHEET`.

DashboardObject Class

Represents one of a series of different areas of the dashboard.

Constructor

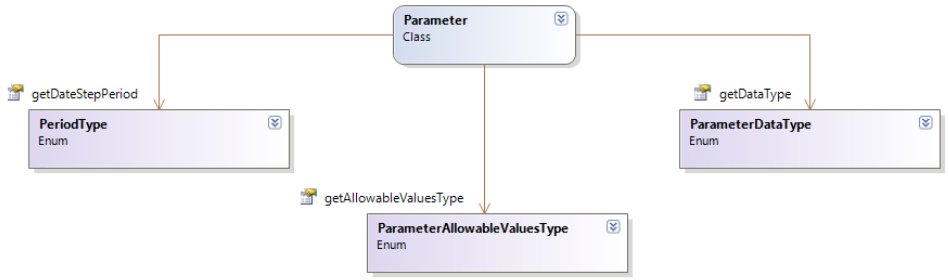
There is no constructor. You get an instance of this from `Dashboard.getObjects()`.

Properties

Name	Type	Description
getObjectType ()	DashboardObjectType	Gets what the object represents, which is an `enum` with the following values: `BLANK`, `WORKSHEET`, `QUICK_FILTER`, `PARAMETER_CONTROL`, `PAGE_FILTER`, `LEGEND`, `TITLE`, `TEXT`, `IMAGE`, `WEB_PAGE`.
getDashboard ()	Dashboard	Gets the `Dashboard` object that contains this object.
getWorksheet ()	Worksheet	If `getType()` returns `WORKSHEET`, this contains a pointer to the `Worksheet` object.
getPosition()	Point	Gets the coordinates relative to the top-left corner of the dashboard of the object.
getSize()	Size	Gets the size of the object.

Parameter Classes

Class Diagram



Parameter Class

Contains information about a workbook parameter. To actually set a parameter's value, call `workbook.changeParameterValueAsync()`.

Properties

Name	Type	Description
getName()	string	A unique identifier for the parameter, as specified by the user.
getCurrentValue()	DataValue	The current value of the parameter.
getDataType()	ParameterDataType	The data type of the parameter can be one of the following: `FLOAT`, `INTEGER`, `STRING`, `BOOLEAN`, `DATE`, `DATETIME`.
getAllowableValuesType()	ParameterAllowableValuesType	The type of allowable values that the parameter can accept. It can be one of the following enumeration items: `ALL`, `LIST`, `RANGE`.
getAllowableValues()	DataValue[]	If the parameter is restricted to a list of allowable values, this property contains the array of those values. Note that this is not a standard collection, but a JavaScript array.
getMinValue()	DataValue	If `getAllowableValuesType` is `RANGE`, this defines the minimum allowable value, inclusive. Otherwise it's undefined/null.
getMaxValue()	DataValue	If `getAllowableValuesType` is `RANGE`, this defines the maximum allowable value, inclusive. Otherwise it's undefined/null.
getStepSize()	Number	If `getAllowableValuesType` is `RANGE`, this defines the step size used in the parameter UI control slider. Otherwise

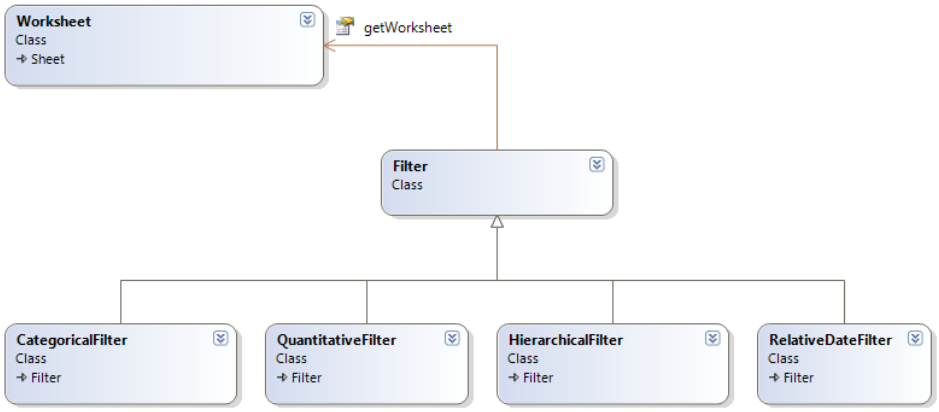
		it's undefined/null.
getDateStepPeriod()	PeriodType	If `getAllowableValuesType` is `RANGE` and `getDataType` is `DATE` or `DATETIME`, this defines the step date period used in the Parameter UI control slider. Otherwise it's undefined/null.

Filtering

There is a difference between querying existing filter state and setting new or existing filters. Querying filters is done via `Worksheet.getFiltersAsync()` which returns a collection of `Filter` classes. Setting filters is done via `Worksheet.applyFilterAsync` (and its variants) and is a function call that doesn't require you to instantiate a `Filter` class.

When you specify fields in a filter, you should use the caption as shown in the UI, not the database field name. For example, use **Container** (the caption) instead of **Product Container** (the actual field name). The exception is hierarchical filters, which use the full hierarchical name (for example, `[Product].[All Product].[Espresso]`). Captions can use the optional `[]` delimiters around names.

Class Diagram



Worksheet Class (Filtering)

These methods are on the `Worksheet` class, but are listed here for convenience.

Methods

Name	Return Type	Description
getFiltersAsync()	Promise<Filter[]>	Fetches the collection of filters used on the sheet.
applyFilterAsync(fieldName: string,	Promise<string>	Applies a simple categorical filter (non-date). See the fil-

values: object[] or object, updateType: FilterUpdateType, options?: FilterOptions)		tering examples for more details on these functions. Returns the `fieldName` that was filtered.
applyRangeFilterAsync(fieldName: string, range: RangeFilterOptions)	Promise<string>	Applies a quantitative filter.
applyRelativeDateFilterAsync(fieldName: string, options: RelativeDateFilterOptions)	Promise<string>	Applies a relative date filter.
applyHierarchicalFilterAsync(fieldName: string, values: object, options: HierarchicalFilterOptions)	Promise<string>	Applies a hierarchical filter. The values parameter is either a single value, an array of values, or an object { levels: ["1", "2"] }.
clearFilterAsync(fieldName: string)	Promise<string>	Clears the filter, no matter what kind of filter it is. Note that the filter is removed as long as no associated quick filter is showing for the field. If there is a quick filter showing, then the filter is kept, but it's reset to the "All" state (effectually canceling the filter). For relative date filters, however, an error is returned since there is no "All" state for a relative date filter. To clear a relative date filter with a quick filter showing, you can call `applyRelativeDateFilter()` instead using a range that makes sense for the specific field.

FilterOptions Record

Passed into the `applyFilter` methods to control advanced filtering options.

Fields

Name	Type	Description
isExcludeMode	bool	Determines whether the filter will apply in exclude mode or include mode. The default is include, which

	means that you use the fields as part of a filter. Exclude mode means that you include everything else except the specified fields.

RangeFilterOptions Record

Passed into the `applyRangeFilterAsync` method to control advanced filtering options.

Fields

Name	Type	Description
min	int	Minimum value for the range (inclusive). Optional. Leave blank if you want a <= filter.
max	int	Maximum value for the range (inclusive). Optional. Leave blank if you want a >= filter.
nullOption	NullOption	The null values to include.

RelativeDateFilterOptions Record

Passed into the `applyRelativeDateFilterAsync` method to control advanced filtering options.

Fields

Name	Type	Description
anchorDate	Date	The UTC date from which to filter.
periodType	PeriodType	Year, quarter, month, etc.
rangeType	DateRangeType	`LAST`, `LASTN`, `NEXT`, etc.
rangeN	int	The number used when the `rangeType` is `LASTN` or `NEXTN`.

Filter Class

An abstract base class for all of the various filter types.

Properties

Name	Type	Description
getWorksheet()	Worksheet	Gets the parent worksheet
getFilterType()	FilterType	Gets the type of the filter. See FilterType Enum for the values in the enum.
getFieldName()	string	Gets the name of the field being filtered. Note that this is the caption as shown in the UI and not the actual database field name.

Methods

Name	Return Type	Description
getFieldAsync()	Promise<Field>	Gets the field that is currently being filtered.

NullOption Enum

An enumeration that indicates what to do with null values for a given filter or mark selection call.

Enumeration

Name	Description
NULL_VALUES	Only include null values in the filter.
NON_NULL_VALUES	Only include non-null values in the filter.
ALL_VALUES	Include null and non-null values in the filter.

CategoricalFilter Class

Properties

Name	Type	Description
getIsExcludeMode()	bool	Gets a value indicating whether the filter is exclude or include (default).
getAppliedValues()	DataValue[]	Gets the collection of values that are currently set on the filter. Note that this is a native JavaScript array and not a keyed collection.

QuantitativeFilter Class

Properties

Name	Type	Description
getDomainMinValue()	DataValue	Gets the minimum value as specified in the domain.
getDomainMaxValue()	DataValue	Gets the maximum value as specified in the domain.
getMinValue()	DataValue	Gets the minimum value, inclusive, applied to the filter.
getMaxValue()	DataValue	Gets the maximum value, inclusive, applied to the filter.
getIncludeNullValues()	bool	Indicates whether null values are included in the filter.

RelativeDataFilter Class

Properties

Name	Type	Description
getPeriod()	PeriodType	The date period of the filter. See PeriodType Enum for the values in the enum.
getRange()	DateRangeType	The range of the date filter (years, months, etc.). See DateRangeType Enum for the values in the enum.
getRangeN()	int	When `getRange` returns `LASTN` or `NEXTN`, this is the N value (how many years, months, etc.).

DataValue Record

A DataValue contains both the raw and formatted value for a filter or parameter. Date values are always expressed in UTC dates.

Properties

Name	Type	Description
value	object	Contains the raw native value as a JavaScript type, which is one of String, Number, Boolean, or Date
formattedValue	string	The value formatted according to the locale and the formatting applied to the field or parameter.

FilterType Enum

An enumeration of the valid types of filters that can be applied.

Enumeration

Name	Description
CATEGORICAL	Categorical filters are used to filter to a set of values within the domain.
QUANTITATIVE	Quantitative filters are used to filter to a range of values from a continuous domain.
HIERARCHICAL	Hierarchical filters are used to filter to a set of values organized into a hierarchy within the domain.
RELATIVE_DATE	Relative date filters are used to filter a date/time domain to a range of values relative to a fixed point in time.

FilterUpdateType Enum

An enumeration of the valid types of filtering that can be done.

Enumeration

Name	Description
ALL	Adds all values to the filter. Equivalent to checking the (All) value in a quick filter.
REPLACE	Replaces the current filter values with new ones specified in the call.
ADD	Adds the filter values as specified in the call to the current filter values. Equivalent to checking a value in a quick filter.
REMOVE	Removes the filter values as specified in the call from the current filter values. Equivalent to unchecking a value in a quick filter.

PeriodType Enum

An enumeration of a date period used in filters and in parameters.

Enumeration

Name
YEARS
QUARTERS
MONTHS
WEEKS
DAYS
HOURS

MINUTES
SECONDS

DateRangeType Enum

An enumeration of the valid date ranges for a relative date filter.

Enumeration

Name	Description
LAST	Refers to the last day, week, month, etc. of the date period.
LASTN	Refers to the last N days, weeks, months, etc. of the date period.
NEXT	Refers to the next day, week, month, etc. of the date period.
NEXTN	Refers to the next N days, weeks, months, etc. of the date period.
CURRENT	Refers to the current day, week, month, etc. of the date period.
TODATE	Refers to everything up to and including the current day, week, month, etc. of the date period.

Marks Selection

Selecting marks is almost identical to filtering. For filtering, you use one of the `Worksheet.applyFilterAsync()` methods. For selecting marks, you use `Worksheet.selectMarksAsync()`. The parameters for marks selection are almost identical to those used for filtering.This provides a very simple, easy to learn, and consistent way to accomplish the two most common use cases for the API: filtering and selection.

Worksheet Class (Selecting Marks)

These methods are on the `Worksheet class`, but are shown here for convenience.

Methods

Name	Return Type	Description
clearSelectedMarksAsync()	void	Clears the selection for this worksheet.
getSelectedMarksAsync()	Promise<Mark[]>	Gets the collection of marks that are currently selected.
selectMarksAsync(fieldName: string, value: object or object[], updateType: SelectionUpdateType)	void	Selects the marks and returns them.

selectMarksAsync(fieldValuesMap: object, updateType: SelectionUpdateType)	void	Allows selection based on this syntax for the first parameter:
		`{`
		`"Field1": value,`
		`"Field2": [1, 2, 3]`
		`}`

selectMarksAsync(marks: Mark[], updateType: SelectionUpdateType)	void	

Mark Class

A mark represents a single data point on the viz. It is independent of the type of viz (bar, line, pie, etc.).

Constructor

Signature	Description
Pair(fieldName: string, value: object)	Creates a new `Pair` with the specified field name/value pairing

Properties

Name	Type	Description
getPairs()	Pair[]	Gets a collection of field name/value pairs associated with the mark.

Pair Class

A pair contains a field name and a value (both the raw and formatted values).

Constructor

Signature	Description
Mark(pairs: Pair[])	Creates a new `Mark` with the specified pairs.

Fields

Name	Type	Description
fieldName	string	The field name to which the value is applied.
value	object	Contains the raw native value for the field as a JavaScript type, which is one of String, Number, Boolean, or Date.
formattedValue	string	The value formatted according to the locale and the formatting applied to the field.

SelectionUpdateType Enum

An enumeration of the valid types of marks selection that can be done.

Enumeration

Name	Description
REPLACE	Replaces the current marks values with new ones specified in the call.
ADD	Adds the values as specified in the call to the current selection. Equivalent to control-clicking in desktop.
REMOVE	Removes the values as specified in the call from the current selection. Equivalent to control-clicking an already selected mark in desktop.

Other Classes

Size Record

Represents a width and height in pixels. This is implemented as a plain JavaScript object (it's not a class).

Fields

Name	Type	Description
width	int	Expressed in pixel units.
height	int	Expressed in pixel units.

Point Record

Represents an x/y coordinate in pixels. This is implemented as a plain JavaScript object (it's not a class).

Fields

Name	Type	Description
x	int	Expressed in pixel units.
y	int	Expressed in pixel units.

Contact

For sales inquries please contact sales@tableausoftware.com.

For customer support, please contact support@tableausoftware.com.

For other inquries, please contact info@tableausoftware.com.

Copyright