TOC

Composer 24 ..... **18**

Manage Composer 24 18

Configure Composer **19**

Composer System Metrics **20**

Discovering Metrics 20

Monitoring Microservices: Prometheus 21

Configure Prometheus 21

Monitoring Microservices: Statsd and Graphite 22

Setup and Configure Statsd and Graphite 22

Edit a Configuration File **24**

Configuration Property Files **27**

zoomdata.properties Properties **29**

zoomdata.jvm Options **37**

Query Engine Properties **38**

screenshot-service.properties Properties **40**

Connector Properties and Property Files **42**

Configure Memory Settings **45**

Configure a High Load Environment **47**

Install PostgreSQL 49

Configure the Metadata Store for a Distributed Environment 49

PostgreSQL Setup for CentOS Environments 50

PostgreSQL Setup for Ubuntu Environments 51

Install Consul Cluster 57

Install Microservices 57

Next Steps 57

Install and Configure the Configuration Microservice on Each Node 57

PostgreSQL Database Setup Notes 58

GitHub Repository Setup Notes 58

Install and Configure the Service Monitor (Optional) 60

Configure a High Availability Environment **63**

Setting Up Your Load Balancer 65

Add a New Node to Existing Composer Multi-Node Deployments **67**

Environment Prerequisites 67

Node Installation and Configuration 67

Install A Java 17 Distribution 67
Add an OS Package Repository 67
Install the zoomdata-consul Package 68
Configure the Zoomdata Consul Package 69
Install Composer Components 69
Validation 70
Manage the Composer Query Engine **71**
Configure the Query Engine 71
Pass Sensitive Data Using Linux Environment Variables **72**
Method 1: Use systemd Override Files (Less Secure) 72
Method 2: Use Parameters Passed to Microservices (More Secure) 73
Encrypt Configuration Properties **75**
Prerequisites 75
Encrypting a Property 75
Sample Script to Encrypt a Property 76
Change the Encryption Mode **78**
Create a Symmetric Key to Encrypt Data Source Passwords **79**
Generate a Keystore with a Symmetric Key 79

Request and Apply a New License Key **81**

Manage License Keys in the UI 81

Manage License Keys Using the Licensing API 82

Manage License Keys Using Configuration Properties or Environment Variables 82

Set Up Unified Logging Using Fluentd **83**

Configure Unified Logging Using Fluentd **84**

Enable Unified Logging in Composer Using Fluentd **86**

Manage Activity Logs **88**

Activity Logging **89**

Configure the Composer Activity Log 89

Enable or Disable the Logging of Specific Activities 90

Determine Whether an Activity Is Being Logged 90

Example of Composer Activity Logging Monitored by Fluentd 91

Linux 91

Windows 91

Template 91

Activities Log Reference Sheet **95**

Activity Logging Defaults 95

Common Log Fields 95
Activity Log Fields 96
account: Account Maintenance 96
authentication: User Authentication 97
group: Group Maintenance 97
raw_data_export: Text Search 97
raw_data_export_csv: Data Export to CSV 98
source: Data Source Maintenance 98
upload: Flat File Upload 99
user: User Account Maintenance 99
vis: Visual Maintenance 100
User Auditing 101
Enable and Configure User Auditing 102
Create a Database 102
Enable User Auditing 102
Restart Composer 104
User Auditing for Multi Tenancy Environments 105
Separation Between Composer Accounts 105

Separation Between Tenants Within the Same Composer Account ... 105

Enable User Audit Data for Composer Accounts ... 106

Tables for a Composer Account ... 106

Views by Composer Account ... 106

Row Level Security in the Database ... 107

Enable User Audit Data for Multi Tenant Accounts ... 108

Tenants Without Connection Creation Permissions ... 108

Tenants With Connection Creation Permissions ... 108

Views Per Tenant ... 108

Row Level Security ... 109

Consuming Audit Data ... 110

Consume Audit Data Using Composer ... 110

Consume Audit Data Using Third-Party Tools ... 110

User Audit Events ... 111

Enable Composer Component Access From Other Sites Using Cross-Origin Resource Sharing (CORS) ... 112

Manage the Upload API Source ... 113

Before You Begin ... 113

Use Another PostgreSQL Instance for Upload API ... 113

Translate the Composer Interface **114**

Server-Level Variables **116**

Configure Composer Logs **117**

Log Properties in Config Files 117

Values for log.file.level and log.console.level 117

Enable Logging for a Service to the Console 118

Consul Logging 118

Structured Logging 118

Components Support for Structured Logging 119

Enable Structured Logging 119

Manage Composer Microservices **120**

Scaling Composer Microservices **121**

Estimate User and Microservice Loads 121

Add or Remove Nodes in a High Availability Environment 122

Configure Throughput for Microservices 122

Load Monitor Microservices 122

Scale Microservices Up 123

Scale Microservices Down 123

Start Composer Microservices **125**

Stop Composer Microservices **126**

Enable Composer Microservices **127**

Disable Composer Microservices **128**

Restart Composer Microservices **129**

Manage Composer Microservices Using the Command Line Utility **130**

Prerequisites 130

Using the Command Line Utility Tool 130

Common Commands 131

Composer Monitoring Solution **132**

Monitoring Solution Components 132

Downloading and Installing the Solution Package 134

Authorize Composer Access **135**

Manage Composer Tenants **136**

About the Supplied Composer Tenant **137**

List and Review Tenants **138**

List Tenants 138

Review Tenants 138

Add and Remove Tenants **139**

Create a New Tenant 139

Remove Tenants 139

Modify Tenants **141**

Modify an Existing Tenant 141

Modify Tenant Administrators **142**

Add or Remove a Tenant Administrator 142

Enable or Disable Tenants **143**

Disable a Tenant 143

Enable a Tenant 143

Manage User Groups **145**

About Supplied Groups **146**

Administrators Group (System Admins) 146

Administrators Group (Tenant Admins) 147

Supervisors Group 147

Content Distributors Group 148

List and Review User Groups **149**

Add User Groups **150**

Add Groups ... 150

Modify User Groups ... **151**

Modify a Group ... 151

Add and Remove Members of a Group ... **152**

Add or Remove Members ... 152

Delete User Groups ... **154**

Delete Groups ... 154

Group Privilege Reference ... **155**

Manage Users ... **161**

Supplied Users and User Groups ... **163**

admin User ... 163

Administrators Group ... 163

Administrators Group (Tenant Admins) ... 164

Supervisors Group ... 164

Content Distributors Group ... 165

List and Review Users ... **166**

List and review all users in your environment (Supervisors Group Members) ... 166

List and Review all Users in Your Environment (System Administrator or a User with User Management Privileges) ... 166

Add Users **168**

When Logged In as Member of the Supervisors Group 168

When Logged In as an Administrator or a Group User with User Management Privileges 169

Add and Remove Supervisors **170**

Add a User to the Supervisors Group 170

Remove a User from the Supervisors Group 170

Modify Users **172**

Modify a User (Supervisors Group Members) 172

Modify a User (Administrator Group Members, Group Members with User Management Privileges) 172

Delete Users **174**

Delete a User (Supervisors Group Members) 174

Modify a User (Administrator Group Members, Group Members with User Management Privileges) 174

Specify General User Information **175**

Enable and Disable Users **178**

Disable a User 178

Enable a User 178

Enable and Disable the Supplied Supervisor User **179**

Disable the Supplied Supervisor User 179

Enable the Supplied Supervisor User Definition ... 179

Assign and Remove Users in Tenants ... 180

Assign a User to a Tenant ... 180

Remove a User from a Tenant ... 180

Set the Current Tenant for a User ... 182

Set the Current Tenant for a User ... 182

Specify Custom User Attributes ... 183

Supplied Context Variables ... 183

Add A Custom Attribute for a User ... 184

Remove a Custom Attribute from a User ... 184

Specify Multivalue (Array) Custom Attributes ... 185

Specify Numeric Values in Custom Attributes ... 185

Specify Time Values in Custom Attributes ... 185

Specify A User's Regional Settings ... 186

Set the User Locale for a User ... 186

Change Passwords ... 187

Change or Reset a Password ... 187

Change a Supervisor Password ... 188

Change a Password for Supervisor Group Members 188
Change a Supervisor Password in Earlier Versions of Composer 189
Import Users **190**
Import user definitions into Composer from LDAP or Active Directory 190
Supported Authentication Tools **192**
Enable or Disable a Security Service **194**
Enable a Security Service 194
Disable a Security Service 195
Configure Client Certificate Authentication **197**
Caveat 197
Configuration Steps 197
Troubleshooting 197
Configure Composer to Support SAML **199**
Prerequisites 199
Obtain Identity Provider Metadata File 199
Generate a Key File and Configuring SAML with SSL 199
Step-By-Step Configuration Instructions 200
Mapping to Custom User Attribute 202

Optional Configurations ... 203
Configure SAML Behind a Load Balancer ... 203
Customize the Composer Entity ID ... 203
Configure the Auto-Redirect to the IDP ... 204
Troubleshooting ... 204
Implement Single Sign-On (SSO) via SAML ... 205
Prepare to Integrate Composer into Your SAML-Enabled Network ... 205
Configure Kerberos Single Sign-On (SSO) Settings ... 207
How It Works ... 207
Prerequisites ... 207
How to Generate the Keytab File ... 208
Configure General Settings ... 208
Configure the Settings on the Client Side ... 209
Kerberos Authentication for Connectors ... 211
Use Lightweight Directory Access Protocol (LDAP) ... 213
Configure LDAP ... 214
Enable User Provisioning ... 215
Configure Mappings ... 215

Manage Mappings to Custom User Attributes ... 216
Manually Import Users from the LDAP Directory ... 216
Use the Secure LDAP Connection ... 217
Trusted Access ... **218**
Trusted Access Prerequisites ... 219
Trusted Access Recommendations ... 219
Register a Client ... 219
Generate a User's Access Token ... 219
Generate a User's Access Token for Existing Composer Users ... 219
Generate a User's Access Token for New Composer Users ... 220
Trusted Access API Endpoints ... **221**
Trusted Access Client Properties ... **222**
Composer Service Monitor ... **223**
Prerequisites ... 223
Install and Configure the Composer Service Monitor ... **224**
Access the Composer Service Monitor ... **226**
Service Monitor Views ... **227**
Wallboard View ... 227

Applications View 228
Journal View 229
Downloads View 230
Properties View 230
Use the Composer Configuration Microservice to Maintain Application Properties 231
Set Up the Configuration Microservice Metadata Store or Repository 233
PostgreSQL Database Setup Notes 233
GitHub Repository Setup Notes 233
Configure and Start the Configuration Microservice 235
Maintain Application Properties 237
Diagnose Problems 239
Prerequisites 239
Download the Diagnostics Bundle 240
Distributed Tracing for Composer 242
Install a Tracing Server 242
Install and Configure OTel Collector 242
Configure the Collector 243
Configure Composer Microservice Tracing 244

Install the Java Agent 245

Configure the Java Agent 245

Required Configuration Properties 245

Configure the Composer Front End 246

Required Configuration Properties 246

Extract the traceID 247

Composer 24

Manage Composer 24

Configure Composer

The Composer server uses multiple configuration files to ensure successful deployment of Composer in your operating environment. Each Composer component has configuration files including a `<component_name>.properties` file and a `<component_name>.jvm` file.

Use the Composer configuration files to make changes to the Composer server and how it functions. The configuration files need to be created after the Composer server has been installed to override Composer's default configuration. To modify the settings in these files, edit them in the `/etc/zoomdata` directory, as described in Edit A Configuration File.

A full list of the configuration file names can be found in Configuration Property Files.

Every `<component_name>.jvm` file should include an `Xmx` JVM option to control the maximum heap memory limit for the application. See Configure Memory Settings.

Composer System Metrics

Composer exposes metrics for discovering and monitoring Composer microservices. Discover system metrics using one of several available tools.

Discovering Metrics

Connect to the composer-webservice actuator endpoints at: `http://[host_ip]:[port]/composer/actuator`.

Available endpoints include:

```
"prometheus": {
       "href": "http://127.0.0.1:8080/composer/actuator/prometheus",
       "templated": false
},
       "metrics-requiredMetricName": (
              "href": "http://127.0.0.1:8080/composer/actuator/metrics/{requiredMetricName)",
              "templated": true
       },
       "metrics":{
              "href": "http://127.0.0.:8080/composer/actuator/metrics",
              "templated": false
       },
```

Connect to all other microservices actuator endpoints at: `http://[host_ip]:[port]/actuator`.

Available endpoints include:

```
{
       "_links": {
              "self": {
                     "http://127.0.0.1:8105/actuator",
                     "templated": false
              },
              "metrics-sorted": (
                     "href": "http://127.0.0.1:8105/actuator/metrics-sorted",
                     "templated": false
              },
              "health-path": (
                     "href": "http://127.0.0.1:8105/actuator/health/{*path}",
                     "templated": true
              },
```

For a list of all metrics exposed by Composer, connect to: `http://[host_ip]:[port]/actuator/metrics`.

Monitoring Microservices: Prometheus

Composer metrics data is formatted so you can access that data through a Prometheus server. Navigate to the Prometheus composer-web actuator endpoint to see your data: `http://[host_ip]:[port]/composer/actuator/prometheus`.

Example Prometheus actuator response:

```
# HELP jvm_classes_unloaded_classes_total The total number of classes unloaded since the Java virtual machine has started
execution
# TYPE jvm_classes_unloaded_classes_total counter jvm_classes_unloaded_classes_total{service_instance="Zoomdata:ip-127-0-0-
0.ec2.internal:8080",} 63.0
# HELP process_files_max_files The maximum file descriptor count
# TYPE process_files_max_files gauge process_files_max_files_{service_instance="Zoomdata:ip-127-0-0-0:ec2.internal:8080",}
10240.0
# HELP jvm_gc_memory_allocated_bytes_total Incremented for an increase in the size of the young generation memory pool
after one GC to be
# TYPE jvm_gc_memory_allocated_bytes_total counter jvm_gc_memory_allocated_bytes_total{service_instance="Zoomdata:ip-127-0-
0-0.ec2.internal:8080",} 4.1484812288E10
# HELP websocket_sessions_active
# TYPE websocket_sessions_active gauge websocket_sessions_active{service_instance="Zoomdata:ip-127-0-0-
0.ec2.internal.8080",} 2.0
# HELP zoomdata_sessions_count_last_minute
# TYPE zoomdata_sessions_count_last_minute gauge zoomdata_sessions_count_last_minute{service_instance=Zoomdata:ip-127-0-0-
0.ec2.internal:8080",} 0.0
```

Configure Prometheus

Install Prometheus. See https://prometheus.io/docs/prometheus/latest/getting_started/ for more information.

Build your `prometheus.yml` as shown in the example here to connect to the composer-webservice. Replace `[username]`, `[password]`, `[host_ip]` and `[port]` with your appropriate values.

```
scrape_configs:
       - job_name: 'composer-web'
              metrics_path: 'composer/actuator/prometheus'
              scrape_interval: 5s
              basic_auth:
                     username: [username]
                     password: [password]
              static_configs:
              - targets: ['[host_ip]:[port]']
```

```
- job_name: 'query-engine'
        metrics_path: 'actuator/prometheus'
        scrape_interval: 5s
        static_configs:
        - targets: ['[host_ip]:[port]']
```

Monitoring Microservices: Statsd and Graphite

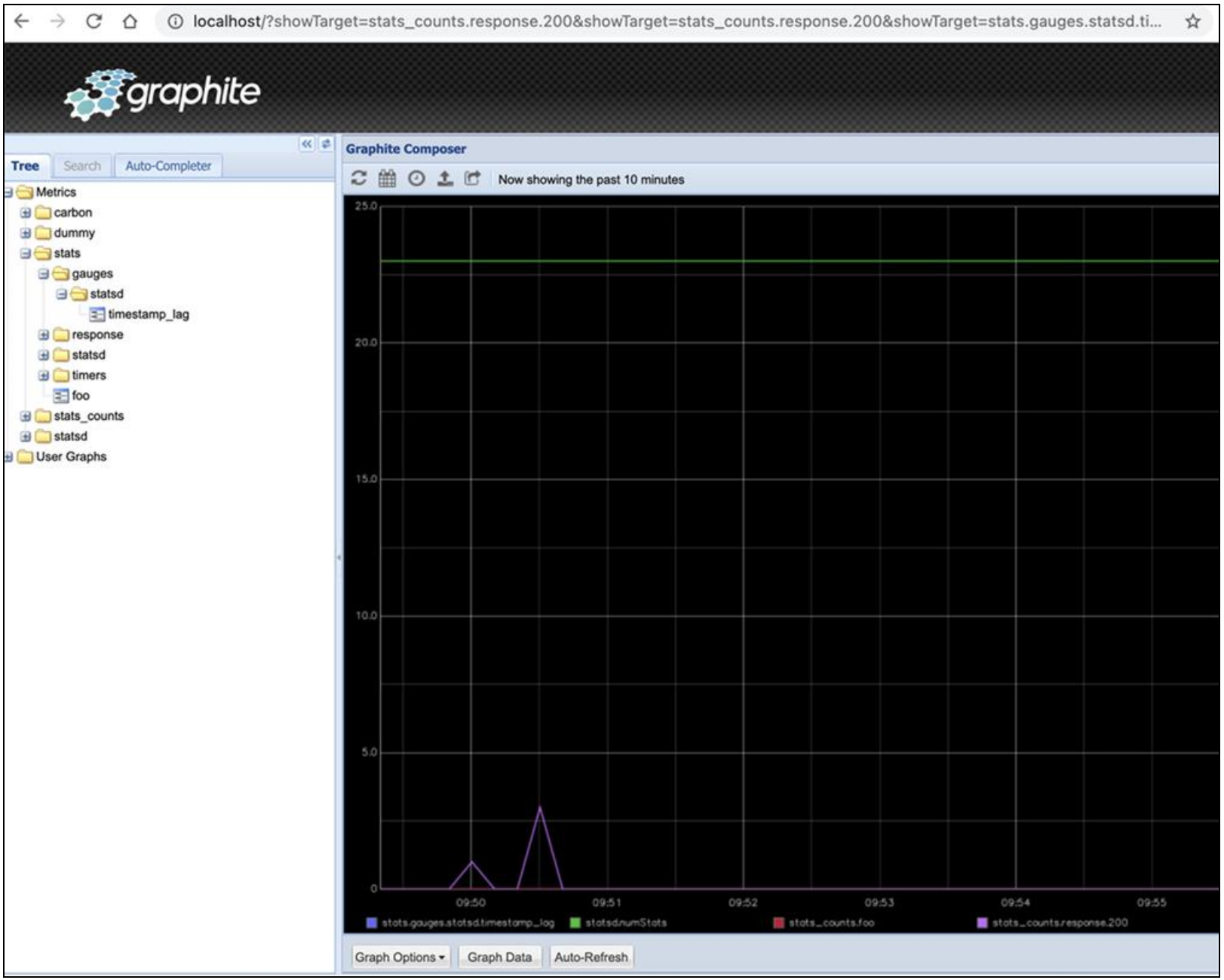
You can also collect Composer metrics using the network daemon statsd. This network daemon runs on the Node.js platform to listen for statistics sent over UDP or TCP and sends aggregate information to one or more backend services. Unlike Prometheus, statsd does not collect metric data from specific endpoints.

Use in conjunction with Graphite to store numeric time-series data and render data graphs on demand.

Setup and Configure Statsd and Graphite

Install statsd and Graphite to connect to and access system metrics.

Example of a Graphite dashboard with metrics collected by statsd:


localhost/?showTarget=stats_counts.response.200&showTarget=stats_counts.response.200&showTarget=stats.gauges.statsd.ti...
graphite
Tree
Search
Auto-Completer
Metrics
carbon
dummy
stats
gauges
statsd
timestamp_lag
response
statsd
timers
foo
stats_counts
statsd
User Graphs
Graphite Composer
Now showing the past 10 minutes
25.0
20.0
15.0
10.0
5.0
0
09:50
09:51
09:52
09:53
09:54
09:55
stats.gauges.statsd.timestamp_lag
statsd.numStats
stats_counts.foo
stats_counts.response.200
Graph Options
Graph Data
Auto-Refresh

Edit a Configuration File

You can edit the properties for your Composer configuration in the configuration files. For a complete list of configuration files, see Configuration Property Files.

> **Note:** insightsoftware discourages changing properties in the `/opt/zoomdata/conf` directory (Linux) or `<install-path>/conf` (Windows). Copy the files you want to change to the `/etc/zoomdata` directory (Linux) or `<install-path>/conf-modify` (Windows) and change them there. This will ensure that your changes are not overwritten when Composer is next upgraded.

Quickly determine what changes you've made to a properties file using `diff` in Linux. For example:

```
diff /opt/zoomdata/conf/edc-<connector-name>.properties /etc/zoomdata/<edc-<connector-name>.properties
```

or

```
diff /opt/zoomdata/conf/zoomdata.properties /etc/zoomdata/zoomdata.properties
```

For Windows environments, use your preferred diff utility to compare the differences between your original and updated property files.

Edit a Composer configuration file in Linux:

1. From your terminal, SSH to your Composer server.
2. Stop the appropriate Composer microservice (Composer, Screenshot, or the specific connector server).

   For the appropriate Linux command, see Stop Composer Microservices.
3. Use the following command to access and open the configuration file:

   ```
   vi /etc/zoomdata/<config-file-name>
   ```

   For example:

   ```
   vi /etc/zoomdata/zoomdata.properties
   ```

   Names of valid configuration files are listed in Configuration Property Files.

   If the configuration file does not exist, this command creates it.

> **Note:** If you are not logged in as a root user, then you need to enter `sudo vi /etc/zoomdata/zoomdata.properties` to create the desired file.

Make sure the file is readable by the `zoomdata` microservice account.

4. Add the new variable or property into the file on a new line or edit an existing one, as needed. See:

i. Zoomdata.properties Properties

ii. zoomdata.jvm Options

iii. Query Engine Properties

iv. screenshot-service.properties Properties

v. Connector Properties and Property Files

5. Save and exit the configuration file.

6. Restart Composer microservices.

   For the appropriate Linux command line, see Restart Composer Microservices.

Edit a Composer configuration file in Windows:

1. Open the configuration file using a text editor that can edit Windows property files.

   Names of valid configuration files are listed in Configuration Property Files.

2. Stop the appropriate Composer microservice (Composer, Screenshot, or the specific connector server).

   For the appropriate Windows commands, see Windows Bootstrap Reference.

3. Add the new variable or property into the file on a new line or edit an existing one, as needed. See:

i. Zoomdata.properties Properties

ii. zoomdata.jvm Options

iii. Query Engine Properties

iv. screenshot-service.properties Properties

v. Connector Properties and Property Files

4. Save and exit the configuration file.

5. Restart Composer microservices.

   For the appropriate Windows commands, see Windows Bootstrap Reference.

Wait a few minutes for the microservice to restart completely, then open a new browser window to log back into Composer. Confirm that your changes have taken effect.

Configuration Property Files

Composer uses the configuration files in the following table to ensure successful deployment in your operating environment. You can edit many of the properties and options in these files. You can edit them in `/etc/zoomdata` as described in Edit A Configuration File.

Note: insightsoftware discourages changing properties in the `/opt/zoomdata/conf` directory (Linux) or `<install-path>/conf` (Windows). Copy the files you want to change to the `/etc/zoomdata` directory (Linux) or `<install-path>/conf-modify` (Windows) and change them there. This will ensure that your changes are not overwritten when Composer is next upgraded.

Quickly determine what changes you've made to a properties file using `diff` in Linux. For example:

```
diff /opt/zoomdata/conf/edc-<connector-name>.properties /etc/zoomdata/<edc-<connector-name>.properties
```

or

```
diff /opt/zoomdata/conf/zoomdata.properties /etc/zoomdata/zoomdata.properties
```

For Windows environments, use your preferred diff utility to compare the differences between your original and updated property files.

File	Defines....
`admin-server.jvm`	JVM options related to the Composer Service Monitor.
`admin-server.properties`	Configuration properties related to the Composer Service Monitor.
`config-server.jvm`	JVM options related to the Composer configuration microservice.
`config-server.properties`	Configuration properties related to the Composer configuration microservice.
`consul.json`	Settings related to the Composer Service Discovery microservice. For a description of these settings, see https://www.consul.io/docs/agent/options.
`data-writer-postgresql.jvm`	JVM options related to the Data Writer microservice's Postgres relational database.
`data-writer-postgresql.properties`	Configuration properties related to the Data Writer microservice's Postgres relational database.
`edc-<connector>.jvm`	JVM options related to the specific Composer connector microservice. For example, `edc-impala.jvm` contains JVM options for the Composer Cloudera Impala connector microservice. See Connector Properties And Property Files.
`edc-<connector>.properties`	Configuration properties related to the specific Composer connector microservice. For example, `edc-`

File	Defines....
	`impala.properties` contains configuration properties for the Composer Cloudera Impala connector microservice. See Connector Properties And Property Files.
`query-engine.jvm`	JVM options related to the Composer query engine. See Manage The Composer Query Engine.
`query-engine.properties`	Configuration properties related to the Composer query engine. See Query Engine Properties.
`screenshot-service.jvm`	JVM options related to Composer Screenshot microservice processing.
`screenshot-service.properties`	Configuration properties related to Composer Screenshot microservice processing. See Screenshot-service.properties Properties.
`zoomdata.jvm`	JVM options (e.g. memory configuration) and Java system properties (e.g. timezone, temp directory, etc.) related to Composer. See Zoomdata.jvm Options.
`zoomdata.properties`	Configuration properties related to Composer. See Zoomdata.properties Properties.

zoomdata.properties Properties

The zoomdata.properties file can be edited in the `/etc/zoomdata` directory. Each property is described in the table below.

Some situations where the `zoomdata.properties` file needs to be updated include:

- Add An SSL Certificate
- Disable The SSL Certificate In ComposerSymphony Data Discovery
- Set Up The Screenshot Microservice
- Create A Symmetric Key To Encrypt Data Source Passwords
- About Scheduled Dashboard ReportsAbout Scheduled Data Discovery Dashboard Reports
- Set Up and Use the Data Gateway Service

For information on editing configuration files, see Edit A Configuration File.

Property	Default Value	Description
access.control.allow.origin	*	By default, CORS is set to --- in the Composer Server. You can set CORS to restrict access: `access.control.allow.origin=<user-defined>` For more information, see Enable Composer Component Access From Other Sites Using Cross-Origin Resource Sharing (CORS).
logs.dir	*<ZD_install_directory>* /logs	Path to Composer logs. The placeholder *<ZD_install_directory>* is replaced with the actual location where Composer is installed. Verify that this log directory has all the necessary permissions and that the owner of the directory is set to `zoomdata`. The `/home` directory cannot be used for logging. Example: `logs.dir=/opt/<composer>/logs)`
data-gateway.client-api.enabled	False	Set to true to enable use of the Data Gateway API and Data Gateway Service. See Set Up and Use the Data Gateway Service.

Property	Default Value	Description
saml.maxAuthAge	86400	Sets the timeout for SAML, in seconds. The default is 24 hours. Example: `saml.maxAuthAge=86400`
server.compression.enabled	true	Enables gzip compression for http requests. Example: `server.compression.enabled=true`
server.port	8080	The default server port, which is set to use http. Prior releases used `http.port`
server.servlet.context-path	/composer	Example: `server.servlet.context-path=/composer`
server.session-timeout	1800 seconds	Sets when your Composer session will timeout (in seconds). Example: `server.session-timeout=1800` If you alter this value, also alter the value of the zoomdata.server.ws.idle.timeout property to match it.
source.attribute.values.limit	1000	Sets the limit for the number of attribute values that can be displayed in the Filter list. Example: `source.attribute.values.limit=1000`
spring.servlet.multipart.max-file-size	500Mb	Example: `spring.servlet.multipart.max-file-size=500Mb`
spring.servlet.multipart.max-request-size	500Mb	Example: `spring.servlet.multipart.max-request-size=500Mb`
zoomdata.server.ws.idle.timeout	1800000 ms	Idle time that allows the WebSocket to be still valid. If you alter this value, also alter the value of the server.session-timeout property to match it.
http.response.header.content-security-policy	frame-ancestors *;	Add appropriate values to override the default values and support your business needs, such as `default-src 'self'`, `script-src 'self'`, and more. Secure your implementation by including values for these resources using appropriate source URLs. Example: `frame-ancestors *; default-src 'self'; script-src 'self' 'nonce-composerScript' https://*.storage.example.com; style-src 'self'`

Property	Default Value	Description
		`'unsafe-inline' https://*.storage.example.com; img-src 'self'; connect-src 'self'; font-src 'self'; base-uri 'self';`
Source Metadata Properties		
zoomdata.source.refresh.metadata.cache.timeout.minutes	10080 (one week)	Sets the default time-to-live for cache values, in minutes.
zoomdata.source.refresh.values.maxDistinctValues	100000	Sets the number of queried distinct values that can be stored in cache.
Encryption Properties		
security.encryption.algorithm		The encryption algorithm used for file encryption.
security.encryption.key.algorithm		The algorithm type of the encryption key used for file encryption.
Keystore Properties		
keystore.location	classpath:security/zoomkeystore.jks	Composer uses symmetric encryption. You can point to a new keystore to strengthen security. Example: `keystore.location=classpath:security/zoomkeystore.jks` See Create A Symmetric Key To Encrypt Data Source Passwords for further guidance.
keystore.password	zoomkey	Lets you set up a unique password for the keystore. Example: `keystore.password=zoomkey`
keystore.key.alias	zoomkey	Example: `keystore.key.alias=zoomkey`
keystore.key.password	zoomkey	Example:`keystore.key.password=zoomkey`
Server SSL Properties		
server.ssl.key-store	<Composer_install_directory>/conf/keystore	Sets the path for the keystore location. Example: `server.ssl.key-store=HOME/conf/keystore`
server.ssl.key-store-password	changeit	Stores the keystore password. Example: `server.ssl.key-store-password=<YourPassword>`
SAML Configuration Properties		
saml.artifactBindingDefault	true	Example: `saml.artifactBindingDefault=true`

Property	Default Value	Description
saml.useMultiValueList	true	Example: `saml.useMultiValueList=true`
saml.stringDelimiter	,	Example: `saml.stringDelimiter=;`
Kerberized PostgreSQL Properties		
spring.datasource.url	jdbc:postgresql://<IP_ address>:<port>/zoomdata	The URL of the `zoomdata` database in the PostgreSQL metadata store. Example: `spring.datasource.url=jdbc:postgresql://localhost:5432/zoomdata`
spring.datasource.username	zoomdata	The user name for the `zoomdata` database in the PostgreSQL metadata store. Example: `spring.datasource.username=zoomdata`
spring.datasource.password	---	The password associated with the user name for the `zoomdata` database in the PostgreSQL metadata store.
keyset.destination.params.jdbc_url	jdbc:postgresql://<IP_ address>:<port>/zoomdata-keyset	The URL of the `zoomdata-keyset` database in the PostgreSQL metadata store. Example: `keyset.destination.params.jdbc_url=jdbc:postgresql://10.2.1.4:5432/zoomdata-keyset`
keyset.destination.params.user_name	zoomdata	The user name for the `zoomdata-keyset` database in the PostgreSQL metadata store. Example: `keyset.destination.params.user_name=zoomdata`
keyset.destination.params.password	---	The password associated with the user name for the `zoomdata-keyset` database in the PostgreSQL metadata store.
upload.destination.params.jdbc_url	jdbc:postgresql://<IP_ address>:<port>/zoomdata-upload	The URL of the `zoomdata-upload` database in the PostgreSQL metadata store. Example: `upload.destination.params.jdbc_url=jdbc:postgresql://10.2.1.4:5432/zoomdata-upload`
upload.destination.params.user_name	zoomdata	The user name for the `zoomdata-upload` database in the PostgreSQL metadata store. Example: `upload.destination.params.user_name=zoomdata`
upload.destination.params.password	---	The password associated with the user name for the `zoomdata-upload` database in the PostgreSQL metadata store.

Property	Default Value	Description
Source Sampling Properties		
source.sampling.rows	1000	Example: `source.sampling.rows=1000`
source.attribute.values.limit	1000	Example: `source.attribute.values.limit=1000`
Logging Properties		
logging.unified.host	127.0.0.1	Sets the host IP address for Fluentd server message logging. For more information, see Set Up Unified Logging Using Fluentd. Example: `logging.unified.host=123.4.5.6`
logging.unified.level	OFF	Sets the log level for messages logged to the Fluentd server. The following options are available for this property: TRACE, DEBUG, INFO, WARN, ERROR, and OFF. If set to OFF, Fluentd unified logging is disabled. For more information, see Set Up Unified Logging Using Fluentd. Example: `logging.unified.level=INFO`
logging.unified.port	24224	Sets the port for Fluentd server message logging. For more information, see Set Up Unified Logging Using Fluentd. Example: `logging.unified.port=1234`
logging.unified.tag	zoomdata-server	Sets the microservice tag name for messages logged to the Fluentd server. This is important because the tag identifies the microservice to which the log messages apply. Valid values are `query-engine`, `zoomdata-server`, `stream-writer`, `upload-service`, and `edc-<connector-name>` (where `<connector-name>` is one of the names listed in Connector Properties And Property Files). For more information, see Set Up Unified Logging Using Fluentd. Example: `logging.unified.tag = zoomdata-server`
syslog.host	127.0.0.1	Sets the host IP address for Syslog server message logging. Example: `syslog.host=127.0.0.1`
syslog.log.level	OFF	Sets the syslog log level for messages logged to the Syslog server. The following options are available for this property: TRACE, DEBUG, INFO, WARN, ERROR, and OFF. Example: `syslog.log.level=DEBUG`
syslog.port	1514	Sets the port for Syslog server message logging.

Property	Default Value	Description
		Example: `syslog.port=1514`
syslog.suffix	local	Specifies a suffix that is appended at the end of the Syslog server log entry that Composer generates. Example: `syslog.suffix=local`
Password Policy		
auth.password.policy.specialCharacters	!@#$%^&*()-_=+,.:;<>	
auth.password.policy.minCharacters	9	
auth.password.policy.maxCharacters	255	
auth.password.policy.minLowercaseCharacters	1	
auth.password.policy.minUppercaseCharacters	1	
auth.password.policy.minNumericCharacters	1	
auth.password.policy.minSpecialCharacters	1	
auth.password.policy.helpMessage	Password must contain at least 9 characters including 1 lowercase, 1 uppercase, 1 number and 1 special (!@#$%^&*()-_=+,.:;<>).	Text is not enclosed in quotation marks.
Data Export Properties		
zoomdata.export.data.max.cols	1000 columns	Use this property to define the maximum number of columns that can be exported for two-dimensional visuals (such as a pivot table). Composer enforces this limit for visual data, but does not enforce it for raw data. The distributed default for this setting is 1000 columns. Valid values can range from 0 through 2147483647 columns.
zoomdata.export.data.max.rows	100000 rows	Use this property to define the maximum number of rows that can be exported for visuals. Composer enforces this limit for visual data. However, for raw data, Composer produces an error if the number of rows requested for export exceeds this setting. The distributed default for this setting is 100000 rows. Valid values can range from 0 through 2147483647 rows.
zoomdata.export.visualdata.max.rows	100000 rows	Use this property to define the maximum number of rows that can be exported for visuals. Composer enforces this limit for visual data. However,

Property	Default Value	Description
		for raw data, Composer produces an error if the number of rows requested for export exceeds this setting. The distributed default for this setting is 100000 rows. Valid values can range from 0 through 2147483647 rows.
Screenshot Microservice Client & Dashboard Scheduling Properties		
screenshot.service.name	screenshot-service	
screenshot.service.url	`http://localhost:8083/`	
screenshot.service.http.client.connect.timeout.milliseconds	10000 milliseconds	Specifies the number of milliseconds that can elapse before Composer stops trying to connect to the screenshot microservice client.
screenshot.service.http.client.read.timeout.milliseconds	60000 milliseconds	Specifies the number of milliseconds that can elapse before Composer stops trying to read from the screenshot microservice client. **Note:** If you increase the time set by the `dashboard.scheduling.screenshot.timeout` property, make sure that you increase the value of this property as well. The total time set by `screenshot.service.http.client.read.timeout.milliseconds` should always be greater than or equal to the time set by the `dashboard.scheduling.screenshot.timeout` property. Bear in mind that this property is specified in milliseconds, but the `dashboard.scheduling.screenshot.timeout` property is specified in seconds.
screenshot.service.http.client.write.timeout.milliseconds	60000 milliseconds	Specifies the number of milliseconds that can elapse before Composer stops trying to write to the screenshot microservice client.
dashboard.scheduling.screenshot.png.height	720 pixels	Identifies the height (in pixels) of the screenshot PNG file that will be sent.
dashboard.scheduling.screenshot.png.width	1280 pixels	Identifies the width (in pixels) of the screenshot PNG file that will be sent.
dashboard.scheduling.screenshot.timeout	60 seconds	Specifies the timeout (in seconds) to take a screenshot for a dashboard email report.

Property	Default Value	Description
		Note: The time specified by this property must be less than or equal to the time set by the `screenshot.service.http.client.read.timeout.milliseconds` property. If you increase the value of this property, make sure that you increase the value of the `screenshot.service.http.client.read.timeout.milliseconds property` accordingly. Bear in mind that this property is specified in seconds, but the `screenshot.service.http.client.read.timeout.milliseconds` property is specified in milliseconds.
mail.from		Specifies the email address identifying where the email comes from.
mail.login		Specifies the email login to use to access the mail server.
mail.password		Specifies the password associated with the email login identified in the `mail.login` property.
In addition, JavaMail API properties (Composer supports both IMAP and SMTP protocols) should be added to the `zoomdata.properties` file to identify the mail server and other mail properties required to use that server to send the scheduled dashboard (for example, `mail.smtp.auth`, `mail.smtp.host`, `mail.smtp.port`, `mail.imap.host`, and `mail.imap.port`). Complete descriptions of IMAP and SMTP protocol JavaMail properties can be found at these links: ■ IMAP: https://javaee.github.io/javamail/docs/api/com/sun/mail/imap/package-summary.html ■ SMTP: https://javaee.github.io/javamail/docs/api/com/sun/mail/smtp/package-summary.html		
Field Settings		
zoomdata.detect.type.attribute.max.length	200 characters	Use this property to set the maximum character length of attribute fields. If this limit is exceeded, the field will be recognized as a Text field.

zoomdata.jvm Options

The `zoomdata.jvm` file contains JVM options (e.g. memory configuration) and Java system properties (e.g. timezone, temp directory, etc.) related to Composer. The table below describes the options you can adjust.

Situations where the `zoomdata.jvm` file needs to be edited include:

- Configure Memory Settings
- Connect To A Kerberized CDH Cluster

For information on editing configuration files, see Edit A Configuration File.

Option	Default Value	Description
DEBUG_ENABLED	0/false	Toggle switch to enable or disable the Java debug capability. To enable, enter '1' or 'true'. Example: `DEBUG_ENABLED=false`
DEBUG_PORT	9393	The default port for the Java debug capability. Example: `DEBUG_PORT=9393`
JAVA_OPTS	-Xss256k -Xms2048m -Xmx8192m	Java-related options for JVM. Refer to Oracle's article on Java HotSpot VM Options for information.
KERBEROS_CONFIG	/etc/krb5.conf	Default location for the Kerberos configuration details. However, the path to the file may be different in your environment. Refer to Oracle's article on File Formats for information.
KERBEROS_PRINCIPAL	hdfs@HADOOP.COM	Kerberos principal name.
KERBEROS_KEYTAB	/etc/zoomdata/zoomdata.keytab	Kerberos keytab location.
PROXY_HOST	user-defined	For cloud-based connectors (including Google Analytics and Salesforce) being used in a proxy configuration, this property specifies the server host to be returned for calls, and identifies the proxy host server that will provide internet access.
PROXY_PORT	user-defined	For cloud-based connectors (including Google Analytics and Salesforce) being used in a proxy configuration, this property specifies the server port to be returned for calls, and identifies the proxy port server that will provide internet access.

Query Engine Properties

Most of the query engine components can be edited in the `query-engine.properties` file. You can manage the query engine using the following properties. For information on editing configuration files, see Edit A Configuration File.

Property	Default Value	Description
Service Logging		
access.log.file.size	10	Access log file size (in MB).
qe.error.log.file.size	5	Error log file size (in MB).
log.file.size	20	General log file size (in MB).
websocket.log.file.size	10	Websocket log file size (in MB).
syslog.host	127.0.0.1	Sets the host IP address for Syslog server message logging.
syslog.log.level	OFF	Sets the syslog log level for messages logged to the Syslog server. The following options are available for this property: TRACE, DEBUG, INFO, WARN, ERROR. and OFF.
trace.requests	false	Enables detailed tracing of HTTP request.
tracing.sampler.probability	1	The distributed tracing request rate percentage. Valid values are between 0 to 1.0, where 0 disables tracing and 1.0 indicates tracing 100% of requests.
General Microservice Configuration for Jetty Web Server		
server.jetty.max-threads	200	Number of threads to serve the HTTP & WebSocket clients.
qe.server.ws.idle.timeout	86400000	Idle time (in milliseconds) that allows the WebSocket to be still valid.
qe.server.ws.input.buffer.size	16384	Buffer size (in bytes) for the WebSocket message.
qe.server.ws.max.message.size	1048576	Max size (in bytes) of a message that could be sent over the query engine WebSocket.
server.port	5580	REST/WebSocket microservice port.
server.ssl.enabled	false	Defines whether the REST/WS API should use SSL.
server.ssl.key-store		Path to the file with the keystore.
Graceful Shutdown		
application.graceful.shutdown.enabled	true	Indicates whether or not graceful shutdown processing should occur. Valid values are `true` (perform graceful shutdown processing) or `false` (do not perform graceful shutdown processing).
application.graceful.shutdown.event-propagation-timeout-sec	5	Specifies how long (in seconds) a query engine instance will wait to allow clients to receive the information that it is out of service.
application.graceful.shutdown.force-kill-	30	The maximum number of seconds that a query engine instance will wait for the number of its

Property	Default Value	Description
timeout-sec		active tasks to reach zero. When this time has elapsed, all remaining active WebSockets serving in-flight queries are closed and then the query engine instance will stop.
Topology Configuration		
calculations.detect.array.fields	true	Enables and disables the query engine validation of multivalue fields in a derived field. By disabling (set the value to `false`) this functionality, any custom metrics that you may have created in earlier versions of Composer that are aggregations of multivalue fields will produce valid values.
qe.max.allowed.time.groups	10000	The maximum amount of time in which groups can be generated by the **Include Blanks** function. For information on even time intervals, see Even Time Intervals .
qe.max.rows.during.execution	5000000	The maximum number of rows that can be processed during a single query. The value specified must be at least slightly greater than the value for the `qe.zengine.edc.rows.limit` property. If you increase the value of `qe.zengine.edc.rows.limit`, consider increasing the value of this property. When a query exceeds the limit set by this property, a `Resource limit is reached during query execution` message appears.
qe.zengine.edc.rows.limit	1000000	The maximum number of records that can be fused from a single data source.
sharpening.enabled	true	Enables or disables Data Sharpening. Set this property to `true` to enable Data Sharpening; set it to `false` to disable it.
sharpening.samples.max.count	100	The maximum number of Data Sharpening requests that can be generated by the sharpening process.
sharpening.samples.skip.first	1	Defines how many first samples are skipped before visualizing the result of sharpening.
zoomdata.validation.histogram.max.size	1000	Histogram max bucket count.

screenshot-service.properties Properties

The following table lists properties you might adjust for the Composer Screenshot microservice.

For information on editing configuration files, see Edit A Configuration File.

Property Name	Default Value	Description
pool.queue.size	10	Sets the queue size for Screenshot microservice requests. This property and the pool.thread.size property specify the upper limits for Screenshot microservice processing. When the number of screenshot requests exceeds the limits set by these two properties, you will receive HTTP 429 "Too Many Requests" errors. You can exceed these limits in a number of ways, including: ■ starting up Composer with lots of dashboards and visuals ■ deleting all your screenshots at once ■ rapidly creating a lot of large dashboards You can increase the values of this property and the pool.thread.size property when you encounter too many failed screenshots. However, do so with caution.
pool.thread.size	20	Sets the thread count for Screenshot microservice requests. This property and the pool.queue.size property specify the upper limits for Screenshot microservice processing. When the number of screenshot requests exceeds the limits set by these two properties, you will receive HTTP 429 "Too Many Requests" errors. You can exceed these limits in a number of ways, including: ■ starting up Composer with lots of dashboards and visuals ■ deleting all your screenshots at once ■ rapidly creating a lot of large dashboards If your use of Composer includes scheduling dashboard reports, update this setting so it is greater than the number of concurrent reports. You can also increase the values of this property

Property Name	Default Value	Description
		and the pool.queue.size property when you encounter too many failed screenshots. However, do so with caution.
syslog.host	127.0.0.1	Sets the host IP address for Syslog server message logging. Example: `syslog.host=127.0.0.1`
syslog.log.level	OFF	Sets the syslog log level for messages logged to the Syslog server. The following options are available for this property: TRACE, DEBUG, INFO, WARN, and ERROR. Example: `syslog.log.level=DEBUG`
syslog.port	1514	Sets the port for Syslog server message logging. Example: `syslog.port=1514`
syslog.suffix	local	Specifies a suffix that is appended at the end of the Syslog server log entry that Composer generates. Example: `syslog.suffix=local`

Connector Properties and Property Files

Composer's architecture enables the deployment of Composer's data connectors as standalone components running in their own process space. Each connector has its own dedicated connector server and a corresponding property files. Some properties are common to all connectors and some properties are unique to a specific connector. The properties for each connector are documented in the property files.

The kinds of configuration properties found in these files include:

- logging properties
- actuator properties (Spring Boot management endpoint configuration)
- data source connection pool properties
- data source-specific properties
- Kerberos configuration properties (for some connectors)
- service discovery properties
- other connector-specific properties.

Note: insightsoftware discourages changing properties in the `/opt/zoomdata/conf` directory (Linux) or `<install-path>/conf` (Windows). Copy the files you want to change to the `/etc/zoomdata` directory (Linux) or `<install-path>/conf-modify` (Windows) and change them there. This will ensure that your changes are not overwritten when Composer is next upgraded.

Quickly determine what changes you've made to a properties file using `diff` in Linux. For example:

```
diff /opt/zoomdata/conf/edc-<connector-name>.properties /etc/zoomdata/<edc-<connector-name>.properties
```

or

```
diff /opt/zoomdata/conf/zoomdata.properties /etc/zoomdata/zoomdata.properties
```

For Windows environments, use your preferred diff utility to compare the differences between your original and updated property files.

The connector property files have names in the following format: `edc-<connector>.properties`. The following table lists these property files and identifies the associated connector.

Connector	Property File Name
Amazon Redshift	`edc-redshift.properties`
Amazon S3	`edc-s3.properties`
Apache Drill	`edc-drill.properties`
Apache Phoenix 4.7	`edc-phoenix-4.7.properties`
Apache Phoenix Query Server 4.7	`edc-phoenix-4.7-queryserver.properties`
Apache Solr	`edc-apache-solr.properties`
BigQuery	`edc-bigquery.properties`
Cloudera Impala	`edc-impala.properties`
Cloudera Search	`edc-cloudera-search.properties`
Couchbase	`edc-couchbase.properties`
Dremio	`edc-dremio.properties`
Elasticsearch 7.0	`edc-elasticsearch-7.0.properties`
Elasticsearch 8.0	`edc-elasticsearch-8.0.properties`
HDFS	`edc-hdfs.properties`
Hive	`edc-hive.properties`
Jira	`edc-jira.properties`
MemSQL	`edc-memsql.properties`
Microsoft SQL Server	`edc-mssql.properties`
MongoDB	`edc-mongo.properties`
MySQL	`edc-mysql.properties`
Oracle	`edc-oracle.properties`
PostgreSQL	`edc-postgresql.properties`
Python	`edc-python.properties`
Real-Time Sales	`edc-rts.properties`
Salesforce	`edc-salesforce.properties`
SAP Hana	`edc-saphana.properties`
SAP S/4HANA	`edc-saphanacloud.properties`

Connector	Property File Name
SAP IQ	`edc-sapiq.properties`
Snowflake	`edc-snowflake.properties`
Spark SQL	`edc-sparksql.properties`
Teradata	`edc-teradata.properties`
TIBCO DV	`edc-tibcodv.properties`
Trino	`edc-trino.properties`
Vertica	`edc-vertica.properties`

For more information about editing property files, see Edit A Configuration File.

Configure Memory Settings

Unified service memory configuration recommendations can help you avoid out of memory issues, unpredictable service stability, and memory allocation. We strongly recommend you use the new default for Java: Xms=Xmx. These updated default memory allocations are designed configure your services more effectively for a production load after installation or upgrade.

Note: Composer v23.2 requires slightly higher memory expectations than previous releases. Upgrading will not overwrite your memory configuration if you have already overwritten the default configuration settings.

If you're upgrading your environment:

- Review your data source connector usage; stop unused connectors.
- Review your memory settings based on the chart below.

	7.x / 23.1 Xms/Xmx	23.2 and later Xms/Xmx
Admin Server	512M/1024M	Deprecated
Config Server	512M/1024M	Deprecated
Data Writer	128M/256M	512M/512M
EDC *	128M/512M	512M/512M
EDC hdfs/s3	512M/3584M	1500M/1500M
Query Engine	1G/6G	4G/4G
Screenshot Service	128M/1G	750M/750M
Zoomdata Web	1G/2G	4G/4G

Change memory allocation settings for Composer microservices

1. From your terminal, SSH to your Composer server.
2. To modify the memory settings for Composer microservices, you need to edit or create the corresponding `.jvm` files in `/etc/zoomdata/`. Perform the following steps:

```
vi/etc/zoomdata/<component_name>.jvm
```

For complete information on editing configuration files, see Edit A Configuration File.

3. Add or update the following line(s) in the corresponding `.jvm` files.

   Important: Do not allocate more than 85% of your total system memory to all microservices.

   The following example configures the Composer server to use 20 GB of RAM in the `zoomdata.jvm` configuration file. You can adjust the number to fit your system's needs. Replace the **20** with the necessary memory allocation for your operating environment.

   ```
   -Xms20g
   -Xmx20g
   ```

4. Save and exit the `.jvm` file.

5. Restart the microservice for which you have modified the settings:

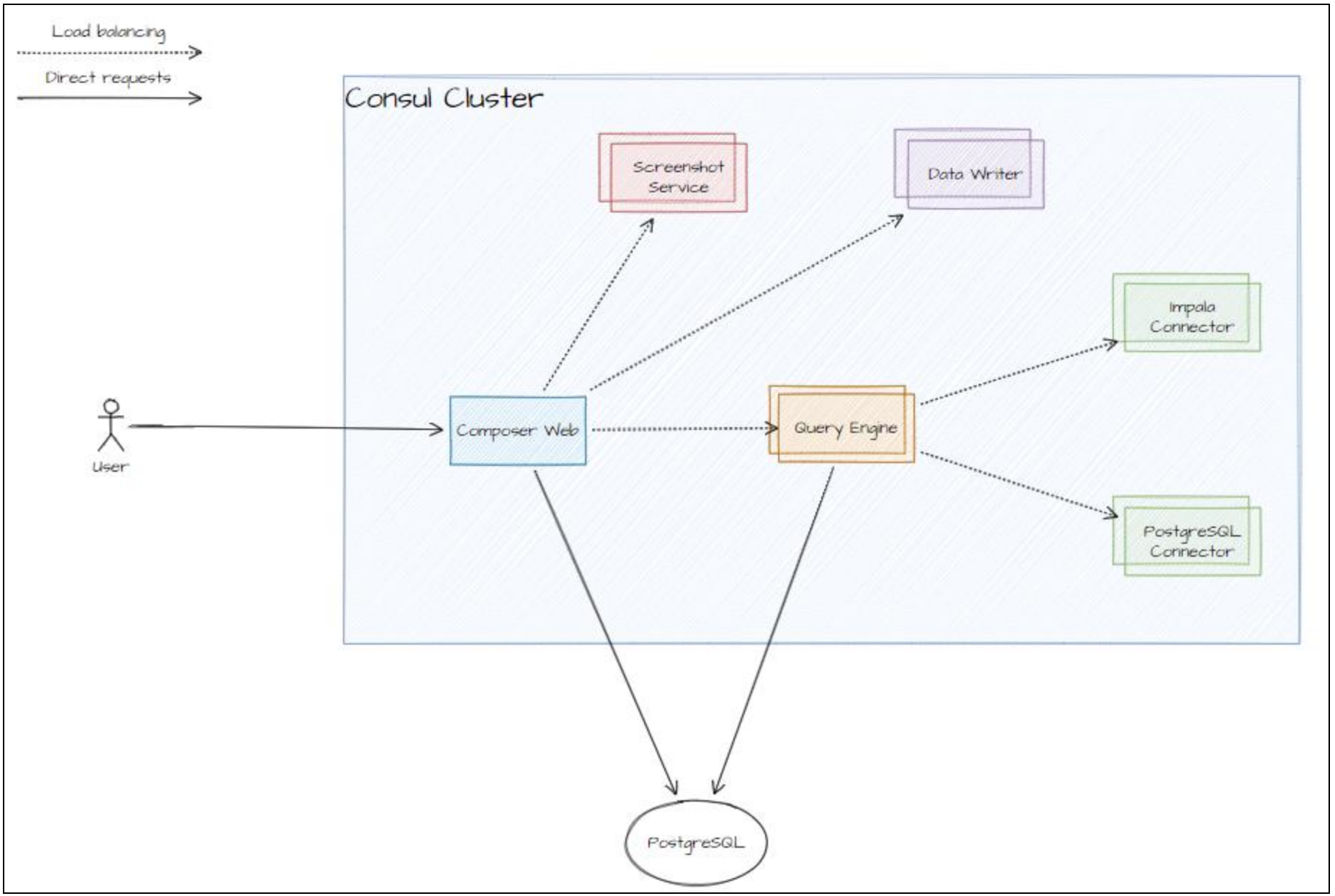
   For CentOS and Ubuntu 18, 20, or 22:

   ```
   sudo systemctl restart <service-name>
   ```

Wait a few minutes for the microservice to restart completely, then open a new browser window to log back into Composer.

Configure a High Load Environment

A high load environment should be able to handle hundreds of concurrent users. One of the simplest solutions is to dedicate powerful machines to our microservices, but it’s cost-inefficient and hardly scalable. Composer has a microservice architecture and supports client-side load balancing for all microservices except COMPOSER-WEB. It enables you to deploy several instances of the same microservice on smaller machines and get a performance gain. You are also able to scale each microservice independently according to the type of load your system receives.


Load balancing
Direct requests
Consul Cluster
Screenshot Service
Data Writer
Impala Connector
User
Composer Web
Query Engine
PostgreSQL Connector
PostgreSQL

Important: To configure Composer in a distributed environment, you need a multi-node license before you start setting up your distributed environment. Contact Technical Support for assistance.

Important: Composer is designed to handle high loads and with an ability to scale. Bottlenecks may not be caused by microservices, but instead in the metastore (PostgreSQL) or other data stores from which you are retrieving data for visualizations.

Install PostgreSQL

When setting up your distributed environment, you need to determine if a metadata store is already installed in your environment. If you have already installed Composer and are now setting up a distributed environment, your metadata store may be installed with your existing Composer instance or instances.

If this is the first time you're installing Composer, and you want to set up a distributed environment, you must set up a new metadata store.

1. Set Up The PostgreSQL Metadata Store
2. Change Metadata Store Authentication To MD5
3. Create The Metadata Store User & Stores
4. Configure The Metadata Store For SSL
5. Configure The Metadata Store For A Distributed Environment

Configure the Metadata Store for a Distributed Environment

Configure the Metadata Store for a Distributed Environment

All Composer installations require a Postgres metadata store. If you install Composer on a single server using the supplied bootstrap installation script, this metadata store is installed for you. If you install Composer manually, you must also manually set up the metadata store.

The metadata store should be centrally installed, accessible by all Composer nodes. In a high availability environment, it can be clustered.

When setting up a distributed environment, you need to determine whether the metadata store is installed. If you have already installed Composer and are now trying to set up a distributed environment, there is a good chance that the metadata store was installed with your existing Composer instance or instances.

If this is the first time you have installed Composer, and you want to set up a distributed environment, a new metadata store is required.

- If the metadata store has already been installed, identify its host name and port. Then follow the steps for upgrading a distributed environment. See Upgrade A Composer Distributed Environment.

- If the metadata store has not already been installed, manually install it now. Complete the following sub-steps. Then identify its host name and port.

A. Set up the Metadata Store - Necessary for all installations.

The instructions to set up PostgreSQL as Composer's metadata store differ depending on the Linux operating system used by the target server. Select a topic below:

i. PostgreSQL Setup for CentOS Environments

ii. PostgreSQL Setup for Ubuntu Environments

PostgreSQL Setup for CentOS Environments

Note: New installations of Composer (v24.3 and later) use PostgreSQL 16. If you are upgrading your environment to v24.3 or later, you can retain your existing PostgreSQL version.

a. Add the PostgreSQL Yum repository to CentOS by running this command calling the appropriate PostgreSQL version:

```
sudo yum -y install https://download.postgresql.org/pub/repos/yum/reporpms/EL-7-x86_64/pgdg-redhat-repo-latest.noarch.rpm
```

b. Install the PostgreSQL client and server packages by running these commands:

```
sudo yum -y install epel-release yum-utils
sudo yum-config-manager --enable pgdg12
sudo yum install postgresql12-server postgresql12
```

c. After installation, initialize the PostgreSQL database:

```
sudo /usr/pgsql-12/bin/postgresql12-setup initdb
```

d. Start and enable the PostgreSQLmicroservice:

```
sudo systemctl enable --now postgresql-12
```

e. Confirm that the service started without errors:

```
sudo systemctl status postgresql-12
```

If necessary, start it:

```
sudo systemctl start postgresql-12
```

f. If you have a running firewall and remote clients should be able to connect to the PostgreSQL metadata store, modify the firewall to allow the PostgreSQL service:

```
sudo firewall-cmd --add-service=postgresql --permanent
sudo firewall-cmd --reload
```

g. If the PostgreSQL database is operating in a cluster, repeat steps 3-6 for each instance of the database.

h. Set up the PostgreSQL Admin user and password:

```
sudo su - postgres
~]$ psql -c "alter user postgres with password 'StrongPassword'"
ALTER ROLE
```

PostgreSQL Setup for Ubuntu Environments

Note: New installations of Composer (v24.3 and later) use PostgreSQL 16. If you are upgrading your environment to v24.3 or later, you can retain your existing PostgreSQL version.

a. If this is a new server instance, update your current system packages:

```
sudo apt update
sudo apt -y install vim bash-completion wget
sudo apt -y upgrade
```

A reboot is necessary after an upgrade.

```
sudo reboot
```

b. Import the GPG key and add the appropriate PostgreSQL version repository to your Ubuntu machine. Run the following commands:

```
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add
echo "deb http://apt.postgresql.org/pub/repos/apt/ `lsb_release -cs`-pgdg main" |sudo tee
/etc/apt/sources.list.d/pgdg.list
```

The added repository contains many different packages and third-party add-ons, including: `postgresql-client`, `postgresql`, `libpq-dev`, `postgresql-server-dev`, and `pgadmin packages`.

c. Update the package list and install the PostgreSQL server and client packages:

```
sudo apt update
sudo apt -y install postgresql-12 postgresql-client-12
```

The PostgreSQL microservice is started and will start with every system reboot.

d. If you have a running firewall and remote clients should be able to connect to the PostgreSQL metadata store, modify the firewall to allow the PostgreSQL service port:

```
sudo ufw allow 5432/tcp
```

e. Test the PostgreSQL connection.

i. During installation, a user named `postgres` is created automatically with full superadmin access to your entire PostgreSQL instance. Before you switch to this account, your logged in system user should have sudo privileges:

```
sudo su - postgres
```

ii. Replace the `postgres` password with a strong password:

```
psql -c "alter user postgres with password 'StrongAdminP@ssw0rd'"
```

iii. Start PostgreSQL using this command.

```
$ psql
```

iv. Get connection details as shown below.

```
postgres=# \conninfo
You are connected to database "postgres" as user "postgres" via socket in "/var/run/postgresql" at port "5432".
```

v. Create a test database called `mytestdb` to see if everything is working.

```
postgres=# CREATE DATABASE mytestdb;
CREATE DATABASE
postgres=# CREATE USER mytestuser WITH ENCRYPTED PASSWORD 'MyStr0ngP@SS';
CREATE ROLE
postgres=# GRANT ALL PRIVILEGES ON DATABASE mytestdb to mytestuser;
GRANT
```

You can list the created databases by running:

```
postgres=# \l
```

vi. Connect to your test database.

```
postgres-# \c mytestdb
You are now connected to database "mytestdb" as user "postgres".
```

B. **Change Metadata Store Authentication to MD5** -- Not necessary when installing a high availability environment in the cloud.

If you installed Composer's metadata store on a server running CentOS or RedHat, complete the configuration steps below. If the server is running Ubuntu, ignore these instructions.

Note: New installations of Composer (v24.3 and later) use PostgreSQL 16. If you are upgrading your environment to v24.3 or later, you can retain your existing PostgreSQL version.

Change authentication for your metadata store to MD5

a. Edit the `pg_hba.conf` file for the appropriate version of PosgreSQL.

```
sudo vi /var/lib/pgsql/12/data/pg_hba.conf
```

b. Change METHOD to MD5.

```
# IPv4 local connections:
host all all 127.0.0.1/32  md5 # IPv6 local connections: host all all ::1/128       md5
```

c. Restart PostgreSQL. In CentOS environments, run:

```
sudo systemctl restart postgresql-12
```

C. **Configure the Metadata Store for SSL** - Not necessary if installing a high availability environment in the cloud.

If you have specified SSL connections for the metadata store JDBC connections in `zoomdata.properties` file, the root CA certificate that is used for the PostgreSQL database must be added to the `/opt/zoomdata/.postgresql` directory. This directory does not exist by default and will need to be created. Complete the following steps.

a. Change to the `/opt/zoomdata` directory as a superuser:

```
sudo cd /opt/zoomdata
```

b. Create a `.postgresql` subdirectory.

```
sudo mkdir .postgresql
```

c. Copy the root CA certificate for the PostgreSQL database into the new directory:

```
sudo cp root.ca /opt/zoomdata/.postgresql/root.ca
```

D. **Configure the Metadata Store for a Distributed Environment** - Not necessary if installing a high availability environment in the cloud.

The Composer PostgreSQL data store must be configured so it is available to all Composer instances in a distributed environment. For more information about PostgreSQL high availability clustering, see PostgreSQL documentation on high availability environments.

> **Note:** New installations of Composer (v24.3 and later) use PostgreSQL 16. If you are upgrading your environment to v24.3 or later, you can retain your existing PostgreSQL version.

Configure the PostgreSQL data store so it is available to all instances

a. Edit the `postgresql.conf` file using the appropriate version and paths:

```
vi /var/lib/pgsql/12/data/postgresql.conf
```

b. Set the following property in `postgresql.conf` and save the file.

```
listen_address='*'
```

c. Edit the `pg_hba.conf file`:

```
vi /var/lib/pgsql/12/data/pg_hba.conf
```

d. Add the following to the `pg_hba.conf file`:

```
# TYPE DATABASE USER ADDRESS METHOD
# IPv4 local connections
host all all 0.0.0.0 /0 md5
```

e. Save the `pg_hba.conf` file.

f. Restart the PostgreSQL service:

```
sudo service postgresql-12 restart
```

E. **Create user and Database** - Not necessary if installing a high availability environment in the cloud.

The PostgreSQL data store must be configured so it is available to all instances in a distributed environment. For more information about PostgreSQL high availability clustering, see https://www.postgresql.org/docs/12/high-availability.html.

Create users and databases in the PostgreSQL data store so they are available to all Composer instances

```
CREATE USER zoomdata WITH PASSWORD 'StrongZoomdataPassword';
CREATE DATABASE "zoomdata" WITH OWNER zoomdata;
CREATE DATABASE "zoomdata-upload" WITH OWNER zoomdata;
CREATE DATABASE "zoomdata-keyset" WITH OWNER zoomdata;
CREATE DATABASE "zoomdata-auth" WITH OWNER zoomdata;
CREATE DATABASE "zoomdata-qe" WITH OWNER zoomdata;
CREATE DATABASE "zoomdata-user-auditing" WITH OWNER zoomdata;
```

Install Consul Cluster

Composer requires consul 1.2.2 to discover microservices. We recommend using consul reference deployment with 3 dedicated consul servers, and consul agents on each node that contains a Composer microservice. See the deployment guidelines for more information on installing consul server nodes.

Install Microservices

Composer consists of several microservices. In a distributed environment, we recommend installing each microservice on a separate machine with a consul agent. The consul agent should be connected to the consul cluster you installed in a previous step. See the deployment guidelines for more information on installing consul server nodes.

We recommend including at least two instances of each microservice except COMPOSER-WEB. Enable intra-service communication by making any necessary networking (ports) and firewall changes. To install a microservice on a machine, see Add A New Node To Existing Composer Multi-Node Deployments.

Microservice	Required
COMPOSER-WEB	Yes.
QUERY ENGINE	Yes.
CONNECTOR	Required only for target data sources.
SCREENSHOT SERVICE	Required only if you use dashboard reporting.
DATA WRITER	Required only if you use upload sources.
CONFIG SERVER	No, this is a platform microservice.
SERVICE MONITOR	No, this is a platform microservice.

Next Steps

- Scaling Composer Microservices
- Install And Configure The Configuration Microservice On Each Node
- Install And Configure The Service Monitor (Optional)

Install and Configure the Configuration Microservice on Each Node

Install and Configure the Configuration Microservice On Each Node

Complete the following steps.

1. **Set Up the Configuration Microservice Metadata Store or Repository**

   Before you can install and start Composer's configuration microservice, you must set up a PostgreSQL metastore or a GitHub repository to store Composer property metadata. A separate PostgreSQL database or a separate GitHub repository must be configured.

PostgreSQL Database Setup Notes

If you elect to persist property metadata to a PostgreSQL metastore:

a. Configure a separate database and make it accessible to the connection user account:

```
CREATE DATABASE <composer-config> WITH OWNER <db_username>;
```

where `<composer-config>` is the name of the PostgreSQL database and `<db_username>` is the connection user account name.

b. Add the following properties to the Composer`config-server.properties` file, located in the `/etc/zoomdata` directory:

```
# metadata storage settings
spring.datasource.url=jdbc:postgresql://localhost:5432/<composer-config>
spring.datasource.username=<db_username>
spring.datasource.password=<db_password>
```

Substitute the connection user account name and password you set up in Step 1 for `<db_username>` and `<db_password>`. Substitute the name of the PostgreSQL database for `<composer-config>`.

c. Save the properties file. You will restart the configuration microservice when you configure it. See Configure And Start The Configuration Microservice.

GitHub Repository Setup Notes

If you elect to persist property metadata to a GitHub repository:

a. Add the following properties to the Composer`config-server.properties` file, located in the `/etc/zoomdata` directory:

```
# metadata storage settings
spring.cloud.config.server.git.uri=<repo_uri>
spring.cloud.config.server.git.skipSslValidation=true
```

```
spring.cloud.config.server.git.username=<repo_username>
spring.cloud.config.server.git.password=<repo_password>
```

Substitute the repository user account name and password for `<repo_username>` and `<repo_password>`. Substitute the URI of the repository for `<repo_uri>` (for example, `https://example.com/my/repo`).

Additional and advanced configuration information can be found in Spring.io's documentation.

b. Save the properties file. You will restart the configuration microservice when you configure it. See Configure And Start The Configuration Microservice.

2. **Install, Configure, and Start the Configuration Microservice**

Install, configure, and start the Composer configuration microservice

a. Verify that you have set up a PostgreSQL metastore or a GitHub repository to store the property metadata. See Set Up The Configuration Microservice Metadata Store Or Repository.

b. Open the SSH client associated with your Composer instance.

c. Configure all installed and enabled Composer microservices to use the Composer configuration microservice. Run the following script:

```
for i in $(systemctl list-unit-files | grep zoomdata | grep enabled | awk '{print $1}'|sed -e
's/\.service/.properties/g' -e 's/zoomdata\-//g');
do
echo "config-server.enabled=true">>/etc/zoomdata/$i;
done
```

d. Each Composer microservice supports two configuration properties related to the configuration microservice:

i. `config-server.enabled`: Enables or disables integration with the configuration microservice. Valid values are `true` (enable integration) and `false` (disable integration). The default is `true`.

ii. `config-client.retry.max-attempts`: Sets the maximum number of attempts that should be made to connect to the configuration microservice. The default is 20. The Composer microservice will fail if the number of attempts to connect to the configuration microservice exceeds this value. This property is

useful in situations where the configuration microservice starts with a delay. If your Composer microservice fails while waiting for the configuration microservice and you want to give it more time, increase this value.

Update these properties, as appropriate, for each microservice.

e. Install, enable and start the Composer configuration microservice. Enter the following commands:

```
sudo yum install zoomdata-config-server \
&& systemctl enable zoomdata-config-server \
&& systemctl start zoomdata-config-server
```

Note: After starting the configuration microservice with a valid database configuration, the microservice should connect to the database and create a properties table.

f. Restart all the other Composer microservices. Enter the following command:

```
sudo systemctl restart $(systemctl list-unit-files | grep zoomdata | grep enabled | awk '{print $1}')
```

See also Restart Composer Microservices.

Install and Configure the Service Monitor (Optional)

Install and Configure the Service Monitor (Optional)

The Composer Service Monitor microservice is not installed as part of a default Composer installation. The microservice name is `zoomdata-admin-server`.

Note: If you are installing Composer in a Windows environment, you can install the Service Monitor as part of running the initial bootstrap script. See Install Composer In A Windows Environment and Windows Bootstrap Reference.

Install, configure, and start the Service Monitor

1. Open your SSH client.
2. Use the following command to install the Composer Service Monitor in a CentOS environment:

```
sudo yum install zoomdata-admin-server -y
```

Use the following command to install the Composer Service Monitor in an Ubuntu environment:

```
sudo apt-get install zoomdata-admin-server
```

3. After the Service Monitor is installed, you must specify a user name and password in its properties file. The properties file is called `admin-server.properties` and can be found in the `/etc/zoomdata/` directory (Linux) or the `<install-path>/conf-modify/` directory (Windows). If the properties file is not there, create it. The properties that must be defined are:

i. `monitor.user.name=<username>`

ii. `monitor.user.password=<password>`

Edit the properties file with a text editor and substitute a Service Monitor user name for `<username>` and its associated password for `<password>`. The user name and password can be any user name and password you want.

When you have finished, save the file.

4. Add the following properties to the `zoomdata.properties` file, located in the `/etc/zoomdata` directory (Linux) or the `<install-path>/conf-modify/` (Windows). These properties ensure that the Service Monitor has access to the Composer server actuator endpoints.

i. `actuator.user.name=<Composer-admin-username>`

ii. `actuator.user.password=<Composer-pswd>`

iii. `actuator.logging.external-file=<log-file-path>`

Edit the properties file and substitute the valid user name and password of a Composer administrator for `<Composer-admin-username>` and `<Composer-pswd>`. If the default Composer log file path is not used for your installation, substitute your custom log file path for `<log-file-path>`. The default log file path is `/opt/zoomdata/logs/zoomdata.log` for Linux and `<install-path>/logs/zoomdata.log` for Windows.

Note: Setting these properties exposes valid Composer credentials as plain text in both the properties file and as tags in the Composer Consul. Anyone in your network with the ability to communicate directly with the Consul API or view the Consul UI will be able to see these values.

When finished, save the file.

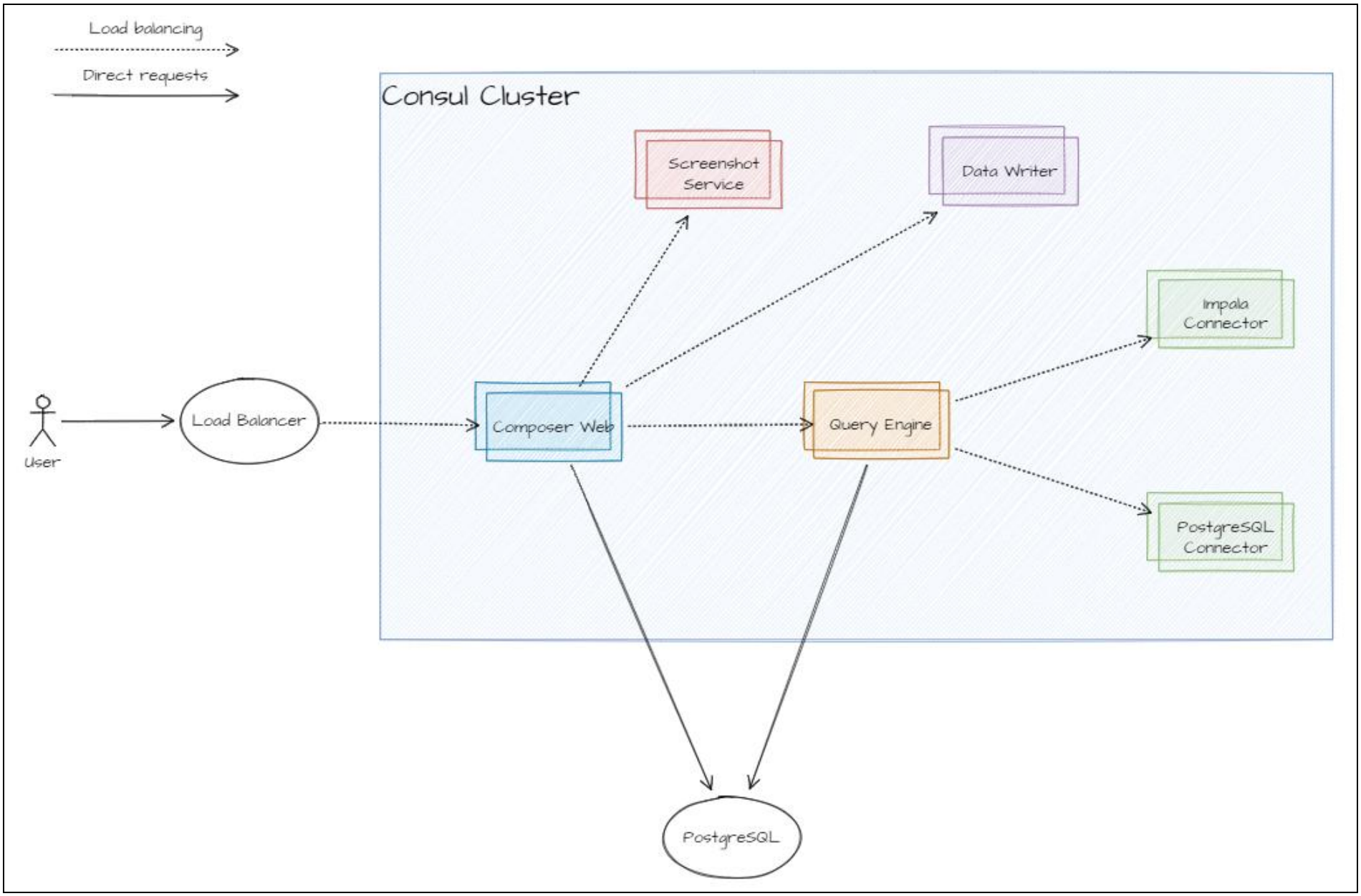
5. Start the microservice. For example, use the following command to start the Composer Service Monitor using `systemd` in a CentOS or Ubuntu environment:

```
sudo systemctl start zoomdata-admin-server
```

See also Start Composer Microservices.

Configure a High Availability Environment

A high availability environment is designed to ensure concurrent users having access to the same microservices at the same time, including the COMPOSER-WEB service. To have a high availability environment, you need to Configure a High Load Environment along with setting up at least two instances of each microservice, including COMPOSER-WEB. These services must be accessible through a Load Balancer.



Note how this is different from how you would Configure a High Load Environment, because you are adding multiple instances of each microservice.

Setting Up Your Load Balancer

The following steps provide an example of setting up HAProxy as a load balancer. Regardless of what kind of load balancer you use, ensure that port 443 is open on it.

Set up an HAProxy load balancer

1. On your machine, run the following command to install HAProxy:

```
sudo yum install haproxy
```

2. Navigate to the HAProxy folder.

```
cd /etc/haproxy
```

3. Create a certificate or copy an existing certificate to the `/etc/haproxy` folder. If you need to create a certificate, run the following commands:

```
sudo openssl genrsa -out ca.key 1024
sudo openssl req -new -key ca.key -out ca.csr
sudo openssl x509 -req -days 365 -in ca.csr -signkey ca.key -out ca.crt
sudo vi cert.pem #Create and save an empty file
sudo chmod a+w cert.pem
sudo cat ca.key ca.crt > cert.pem
```

4. In the same folder, replace the contents of the `haproxy.cfg` file with the contents of the Composer haproxy configuration file. In the file, replace the `<node1-ip>` and `<node2-ip>` with the IP addresses of your servers. If you have more than two servers, add additional lines for each server.

5. Save your changes and exit the file.

6. Start the HAProxy microservice

```
sudo service haproxy restart
```

7. Use the following command to configure the HAProxy microservice to start automatically in CentOS environments:

```
sudo systemctl enable haproxy
```

8. Ensure that port 443 is open on your load balancer. If not, run the following command:

```
sudo iptables -I INPUT 1 -p tcp --dport 443 -j ACCEPT
sudo service iptables save
```

For assistance with configuring SAML for use with HAProxy, contact insightsoftware Technical Support.

Add a New Node to Existing Composer Multi-Node Deployments

Environment Prerequisites

- You must have previously configured your Composer multi-node deployment. `zoomdata-consul` and `postgresql`, configured as clusters external to the nodes, must be configured to accept incoming connections.
  - See https://www.consul.io/commands/join (`zoomdata-consul join <ip-address>`).
  - See https://www.postgresql.org/docs/12/high-availability.html.
- You must have Linux machines where additonal Composer components will be installed.

Node Installation and Configuration

Each process described here must be performed on each additional node of your Composer multi-node deployment.

Install A Java 17 Distribution

Supported options include:

- Oracle: download here
- OpenJDK: download here
- AWS Corretto: download here

Add an OS Package Repository

You'll need to include an OS package repository of your version of Composer.

CentOS/RHEL `/etc/yum.repos.d/zoomdata.repo`

> **Note:** `<COMPOSER-VERSION` must be replaced by the same trunk version (such as 22.4 or 23.2) as your other Composer cluster nodes are equipped with. `RHEL-RELEASE` must be replaced by the supported version of CentOS/RHEL.

```
[zoomdata-stable]
baseurl=https://composer-repo.logianalytics.com/<COMPOSER-VERSION>/yum/redhat/<RHEL-RELEASE>/x86_64/stable
```

```
       gpgcheck=1
       gpgkey=https://composer-repo.logianalytics.com/ZOOMDATA-GPG-KEY.pub
       name=Zoomdata stable RPMs
       enabled=1

[zoomdata-tools-stable]
baseurl=https://composer-repo.logianalytics.com/tools/yum/redhat/<RHEL-RELEASE>/x86_64/stable
       gpgcheck=1
       gpgkey=https://composer-repo.logianalytics.com/ZOOMDATA-GPG-KEY.pub
       name=Zoomdata tools stable RPMs
       enabled=1
```

Ubuntu `/etc/apt/sources.list.d/zoomdata.list`

Note: `<COMPOSER-VERSION` must be replaced by the same trunk version (such as 6.9 or 7.10) as your other Composer cluster nodes are equipped with. `UBUNTU-CODENAME` must be replaced by one of the supported versions of Ubuntu.

Important: Call `apt update` after adding the additional repository list.

```
deb https://composer-repo.logianalytics.com/<COMPOSER-VERSION>/apt/ubuntu <UBUNTU-CODENAME> stable
deb https://composer-repo.logianalytics.com/tools/apt/ubuntu <UBUNTU-CODENAME> stable
```

Install the `zoomdata-consul` Package

This allows the cluster node to communicate in Service Discovery with other cluster nodes.

CentOS/RHEL

```
yum install -y zoomdata-consul
```

Ubuntu

```
apt install -y zoomdata-consul
```

Configure the Zoomdata Consul Package

Configure the Zoomdata Consul package to be a part of the existing cluster.

1. Stop the `zoomdata-consul` service if it is running.
2. Edit the `/etc/zoomdata/consul.json` file and add:

```
{
        "bind_addr": "0.0.0.0",
        "bootstrap": false,
        "bootstrap_expect": 3,
        "client_addr": "0.0.0.0",
        "data_dir": "/opt/zoomdata/data/consul",
        "server": true
        "retry_join": ["<IP1>", "<IP2>", "<IP3>"]
}
```

3. Start the `zoomdata-consul` service. Check if it joined the cluster: `/opt/zoomdata/bin/zoomdata-consul members`

Install Composer Components

Install the subset of Composer components required by the role of each node you're adding.

Note: COMPOSER-WEB (`zoomdata`) and Composer Query Engine (`zoomdata-query-engine`) require extra configuration to use the shared metadata storage you configured.

For example, to install an additional Composer Query Engine and MSSQL connector:

- RHEL/CentOS

```
yum install -y zoomdata-query-engine zoomdata-edc-mssql && \
systemctl enable zoomdata\* && \
systemctl start zoomdata-query-engine zoomdata-edc-mssql
```

- Ubuntu

```
apt install -y zoomdata-query-engine zoomdata-edc-mssql && \
systemctl enable zoomdata\* && \
systemctl start zoomdata-query-engine zoomdata-edc-mssql
```

Validation

1. Ensure `zoomdata-consul` service, and other zoomdata services are running and not in a failed state.

   ```
   systemctl status zoomdata\*
   ```

2. Check that the Consul cluster formed successfully. It should show all nodes, including the node or nodes you just added.

   ```
   /opt/zoomdata/bin/zoomdata-consul members
   ```

3. Check the logs to troubleshoot any issues you may have.

   ```
   /opt/zoomdata/logs
   ```

Manage the Composer Query Engine

The query engine is a stand-alone microservice within the Composer environment that processes visual queries. If you install or upgrade Composer using the supplied installation script, the query engine microservice is started automatically. If you install or upgrade Composer manually, you must manually enable and start the query engine microservice.

Configure the Query Engine

In your environment, the query engine is called `zoomdata-query-engine`. To configure and manage the microservices that the query engine executes and runs, you need to make changes to the `query-engine.properties` file. It contains properties that are specific to how the query engine operates within your environment.

The query engine also has a `query-engine.env` file and a `query-engine.jvm` file. You can edit these files to configure the following:

- To define the environment variables that are visible for the microservice, edit the `query-engine.env` file.
- To configure the JVM options that are used to start up Composer microservices, edit the `query-engine.jvm` file.

The default memory configurations give the query engine is 4 GB of heap memory. During microservice startup, the query engine consumes at least 4 GB of memory (plus some off heap memory). You can alter this using the following parameters in the query-engine.jvm file, i.e. by increasing to 6Gb.

```
-Xms6g
-Xmx6g
```

For information on editing configuration files, see Edit A Configuration File. For information about query engine properties, see Query Engine Properties.

Pass Sensitive Data Using Linux Environment Variables

You can pass sensitive property values (such as database passwords or user names) to Composer using Linux environment variables, rather than hard coding the values in Composer property files. Using the SpringBoot Java application's ability to consume application properties as environment variables, you can pass sensitive data to Composer without showing the data as plain text in the Composer property files.

Note: The information provided here requires a strong knowledge of Linux internals.

You can do this using either of two methods:

- Method 1: Use Systemd Override Files (Less Secure)
- Method 2: Use Parameters Passed To Microservices (More Secure)

You can also encrypt sensitive property values. See Encrypt Configuration Properties.

Method 1: Use `systemd` Override Files (Less Secure)

You can use `systemd` with default unit files override. This is less secure than Method 2: Use Parameters Passed To Microservices (More Secure) because the passwords are still exposed in the `env-pass.conf` files.

Complete the following steps:

1. Stop the affected microservices. See Stop Composer Microservices.

```
sudo systemctl stop <service>
```

2. Add a `systemd` override file, called `env-pass.conf`, for each affected microservice. For example, the `systemd` override file for the `zoomdata` microservice would be `/etc/systemd/system/zoomdata.service.d/env-pass.conf` and the override file for the `zoomdata-query-engine` microservice would be `/etc/systemd/system/zoomdata-query-engine.service.d/env-pass.conf`. A complete list of microservice names is provided in ComposerSymphony Data Discovery Microservice Name Reference.

3. Edit the `env-pass.conf` file for each microservice and add lines specifying environment variable values for every microservice configuration property containing sensitive information. For example, the following environment variables specify the data store password, the upload destination password, and the keyset destination password in the `env-pass.conf` file for the `zoomdata` microservice:

```
[Service]
Environment="SPRING_DATASOURCE_PASSWORD=<data store password>"
Environment="UPLOAD_DESTINATION_PARAMS_PASSWORD=<upload destination password>"
Environment="KEYSET_DESTINATION_PARAMS_PASSWORD=<keyset destination password>"
```

In the `env-pass.conf` file for the `zoomdata-query-engine` microservice, you might add the following environment variable to specify the query engine database password:

```
[Service]
Environment="SPRING_QE_DATASOURCE_PASSWORD=<query engine database password>"
```

The environment variable names are the same as the property names in the corresponding microservice property files, but in all capital letters and substituting underscores for the periods in the property names. Review the property files for valid property names. For example, the environment variable name for the `spring.datasource.password` property is `SPRING_DATASOURCE_PASSWORD`.

4. Apply the changes to `systemd`:

```
sudo systemctl daemon-reload
```

5. Start the affected microservices. See Start Composer Microservices.

Method 2: Use Parameters Passed to Microservices (More Secure)

The most secure method is to pass sensitive parameters to Composer microservices as environment variables when you start the microservices. After the microservice is started, the settings for the sensitive parameters are no longer visible.

Important: If you use this method, Linux service management will be unable to automatically start or restart the affected Composer microservices when your system or server restarts. To resolve this, develop a wrapper script for `systemd` that will automatically export the required environment variables and restart the microservices.

Complete the following steps:

1. Stop the affected Composer microservice. See Stop Composer Microservices.

```
sudo systemctl stop <service>
```

2. Disable the affected microservice. See Disable Composer Microservices.

```
sudo systemctl disable <service>
```

3. Start the affected microservice, passing the sensitive parameters in environment variables as command line arguments. Make sure you use appropriate escape quotes if your password includes quotes.

   The environment variable names are the same as the property names in the corresponding microservice property files, but in all capital letters and substituting underscores for the periods in the property names. Review the property files for valid property names. For example, the environment variable name for the `spring.datasource.password` property is `SPRING_DATASOURCE_PASSWORD`.

   In the following start command, the `zoomdata` microservice is started and the data store password, the upload destination password, and the keyset destination password are all passed as command line arguments:

```
sudo -u zoomdata /opt/zoomdata/bin/zoomdata start -v
 -E 'SPRING_DATASOURCE_PASSWORD=<data store password>'
 -E 'UPLOAD_DESTINATION_PARAMS_PASSWORD=<upload destination password>'
 -E 'KEYSET_DESTINATION_PARAMS_PASSWORD=<keyset destination password>'
```

   The following start command starts the `zoomdata-query-engine` microservice and passes the query engine database password as a command line argument.

```
/opt/zoomdata/bin/zoomdata-query-engine start -v -E 'SPRING_QE_DATASOURCE_PASSWORD=<password>'
```

   See Start Composer Microservices.

Encrypt Configuration Properties

You can encrypt sensitive property values in Composer's property files, if needed. This can be accomplished using the Spring Cloud CLI.

- Prerequisites
- Encrypting A Property
- Sample Script To Encrypt A Property

See also Change The Encryption Mode.

You can also pass sensitive property values to Composer using Linux environment variables, rather than hard coding the values in Composer property files. See Pass Sensitive Data Using Linux Environment Variables.

Prerequisites

Before you can encrypt property values, your Composer environment must meet the following requirements.

- You must have the full-strength Java Cryptography Extension (JCE) installed in your Java virtual machine (it's not there by default). You can download the JCE Unlimited Strength Jurisdiction Policy Files from Oracle at the following link: https://www.oracle.com/java/technologies/javase-jce8-downloads.html.

Follow the installation instructions in the README.txt file (essentially replacing the two policy files in the JRE `lib/security` directory with the ones that you downloaded).

- Install or upgrade to Spring Cloud CLI version 2.0.0. This specific version is required. Review the installation and upgrade guidance here: https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#cli. To install the Spring Cloud CLI, follow the instructions here: https://cloud.spring.io/spring-cloud-static/spring-cloud-cli/2.0.0.RELEASE/single/spring-cloud-cli.html.

Encrypting a Property

Encrypt an individual configuration property

1. Ensure your environment meets the prerequisites listed in Prerequisites.
2. Use the following Spring Cloud CLI 2.0.0 command to encrypt a property value with a specified encryption key:

```
spring encrypt <property-value> --key <encryption-key>
```

Important: Version 2.0.0 of the Spring Cloud CLI must be installed. Other versions are not supported.

For example, the following command encrypts the value of the PASSWORD variable using the encryption key specified by the ENCRYPT_KEY variable.

```
spring encrypt $PASSWORD --key $ENCRYPT_KEY
```

The output of this command is the encrypted property. For example:

```
711448026e2c6a977b2be1b22f13649cc938366397fbd345113d2a50e27c348f)
```

3. Add the following to the properties file in which the encrypted property value will be stored:

i. The encrypted property you obtained in Step 2, prepended with `{cipher}`. The `{cipher}` prefix allows Spring Cloud to recognize encrypted properties. For example:

```
encrypted.property={cipher}711448026e2c6a977b2be1b22f13649cc938366397fbd345113d2a50e27c348f
```

ii. A new property (once per property file) that identifies the encryption key used. For example:

```
ENCRYPT.KEY=<encryption-key>
```

4. Save the properties file and restart its associated Composer microservice. See Restart Composer Microservices.

Alternatively, you could modify and use the sample script, provided next, to encrypt a property value in a properties file.

Sample Script to Encrypt a Property

You can modify and use the following sample script to encrypt a property value in a properties file.

```
#!/bin/bash

#define the property file location, property key, and encryption key
FILE=/etc/zoomdata/zoomdata.properties
```

```
PROPERTY_KEY=spring.datasource.password
ENCRYPT_KEY=zoomdata

#read the property value
PROPERTY_VALUE=$(< "$FILE" grep -w "$PROPERTY_KEY"|cut -d '=' -f2)

#encrypt the value
#NOTE: this step will fail with 'Unable to initialize due to invalid secret key'
#if the JCE prerequisite is not met
ENCRYPTED_VALUE="{cipher}"$(spring encrypt $PROPERTY_VALUE --key $ENCRYPT_KEY)

#update the file
sed -i "s/$PROPERTY_KEY=$PROPERTY_VALUE/$PROPERTY_KEY=$ENCRYPTED_VALUE/g" $FILE

#add the encryption key if not already present in the file
grep -q -F "encrypt.key=$ENCRYPT_KEY" $FILE || echo "encrypt.key=$ENCRYPT_KEY" >> $FILE
```

Change the Encryption Mode

You can change the encryption mode used by Composer (for example to change from AES to AES/CBC/PKCS5Padding encryption). The encryption mode you select is used to encrypt connection parameters, secure user attributes, and Trusted Access tokens.

Important: We recommend that you change the encryption mode used by Composer with the assistance of Composer Technical Support.

If you are upgrading to a newer version of Composer and you also want to change your encryption mode, perform the upgrade first and then complete the steps described here.

Note: You must have system administration privileges to change the encryption mode.

You must have the full-strength Java Cryptography Extension (JCE) installed in your Java virtual machine (it's not there by default). You can download the JCE Unlimited Strength Jurisdiction Policy Files from Oracle at the following link: https://www.oracle.com/java/technologies/javase-jce8-downloads.html.

See also Encrypt Configuration Properties.

Change the encryption mode

1. Start the Composer microservice. This will populate the Composer database using the original encryption (for example AES). See Start Composer Microservices.
2. Stop the Composer microservice. See Stop Composer Microservices.
3. Back up the Composer database. See Back Up The Metadata Store.
4. Modify the following encryption properties in the `zoomdata.properties` file: `security.encryption.algorithm` and `security.encryption.key.algorithm`. For example:

   ```
   security.encryption.algorithm=AES/CBC/PKCS5Padding
   security.encryption.key.algorithm=AES
   ```

   See Zoomdata.properties Properties.
5. Start the Composer microservice. Composer will start using the new properties and the new encryption method. See Start Composer Microservices.

Create a Symmetric Key to Encrypt Data Source Passwords

Composer provides a suite of prebuilt connectors that connect the Composer Server directly to your data source. If a data store requires a connection password to access the data, the credential information is saved in Composer's storage repository - PostgreSQL. Composer uses symmetric encryption to store the credential information so that it can access the data store, as needed, while providing a level of security for the saved information.

Composer administrators can generate their own KeyStore using a symmetric key algorithm. This capability provides an additional level of security in the connection to and access of the data sources.

A symmetric key can be generated using Oracle's keytool program, which is a key and certificate management tool. This tool manages a keystore (database) of cryptographic keys, X.509 certificate chains, and trusted certificates. Refer to Oracle documentation for additional details about this keytool program.

Use the latest Java SDK to install the keytool program (as older versions of the SDK may require different installation steps).

Note: Remember that this user-generated keystore should be provided to Composer after a new installation, prior to any connections being stored in Composer. If a new user-generated key is provided after some connections are already stored, the passwords for these connections have to be resupplied to Composer after the new key is provided.

Generate a Keystore with a Symmetric Key

1. Install the keytool program. Use the latest Java SDK to install the keytool program.
2. Enter the following command line to generate your symmetric key.

```
keytool -genseckey -alias <YourKeyAlias> -keyalg AES -keysize 256 -storetype jceks -keystore <YourKeyStoreName>.jks
```

3. Create a keystore password and press **Enter** to continue.
4. Create a key password and press **Enter** to continue.
5. Store the keystore file in a location where the Composer Server can access. For example:

```
/etc/zoomdata/<YourKeyStoreName>.jks
```

Next, you need to edit the `zoomdata.properties` file to add in the parameters needed for Composer to integrate your symmetric key. If you have already logged into Composer, be sure to log out first and close the browser.

6. Edit (or create) the Composer configuration file (`zoomdata.properties`):

```
vi /etc/zoomdata/zoomdata.properties
```

> **Note:** If the configuration file does not exist, this command creates it.

7. Incorporate instructions for accessing your newly generated keystore file into the **.properties** file as provided below:

```
keystore.location=file:/etc/zoomdata/<YourKeyStoreName>.jks
keystore.password=<YourKeyStorePassword>
keystore.key.alias=<YourKeyAlias>
keystore.key.password=<YourKeyPassword>
```

8. Restart Composer Server. This ensures that the new keystore file is enabled and active within Composer.

   For the appropriate Linux commands, see Restart Composer Microservices.

The symmetric key should now be active in Composer. If you see any error messages after the restart, submit a request for assistance.

Request and Apply a New License Key

Composer notifies you when your license key is about to expire.

Note: Use the DevNet License Manager to obtain license keys. These are provided to the person in your organization we term the License POC (Point of Contact). If you don't know who your License POC is, please reach out to your Account Manager.

Note: When a Composer standard or trial license expires, only system administrators can log in (to update the license). Regular users are shown a message that the license has expired.

Manage your license keys in one of three ways:

- Manage License Keys In The UI
- Manage License Keys Using The Licensing API
- Manage License Keys Using Configuration Properties Or Environment Variables

Manage License Keys in the UI

Apply a new license key:

1. In the environment for you which you need a new license key, log in system admin or a member of the Supervisors group.
2. Select **License** from the menu. The Manage License page appears.

   This screen shows your current Composer instance ID and its current expiration date.

   Note: If you select **Download Current Key**, your current license key is downloaded in a `license.key` file.
3. Select **Download Instance Token**. A `.tok` file is downloaded to your computer.
4. Log into DevNet and visit the License Management page to create a license file. You do this by assigning one of your license keys to a specific instance of Composer , by uploading the `.tok` file from the previous step. DevNet generates a license.key file, and downloads the file to your machine.
5. Browse to find the license key you just downloaded. Then, select **Submit**. Composer applies the new license to your instance.

Manage License Keys Using the Licensing API

Endpoint	Method	Description
`/api/license`	GET	Returns license information for the instance.
	POST	Updates license information for the instance

API documentation is provided with your Composer installation at this link: *`https://<composer-URL>/composer/swagger-ui.html`*.

If a problem occurs, contact your sales or technical support representative.

Manage License Keys Using Configuration Properties or Environment Variables

Add your license keys and installation id to `zoomdata.properties`. Format the variables as shown below:

```
zoomdata.license.key=<value>
zoomdata.installation.id=<value>
```

Important: License information provided via configuration properties or environment variables have a higher priority than keys stored in the database previously provided via API or user interface. If you attempt to change the keys using the API in an environment where licensing information from configuration properties or environment variables are in use, Composer returns a failure message. Update the license key in the configuration files instead.

Note: For more information about extending licensing to a Kubernetes deployment, see Apply Licenses.

Set Up Unified Logging Using Fluentd

In Composer, you can use Fluentd as a logging layer to which you can direct the logs for various components of Composer. This allows you to customize the log output to meet the needs of your environment.

Composer leverages Fluentd's unified logging layer to collect logs via a central API. Fluentd can be configured to aggregate logs to various data sources or outputs. For example, if you are directing all log files from your `zoomdata-websocket.log` and `zoomdata-errors.log` to a Fluentd server, you can add one of Fluentd's plug-ins to write the log files to Elasticsearch to analyze web client errors for your environment.

Unified logging does not replace Composer's default logging architecture. It only augments that experience with an option for those wanting additional logging control.

By default, unified logging is disabled and needs to be enabled for each microservice you want captured in Fluentd.

If written to a data store compatible with Composer's connectors, Composer can then be used to analyze and visualize your logs. The selected data store must be supported by Composer's connectors if you plan to view the log files in Composer dashboards. A list of supported data stores is provided in Data Connector Reference. The selected data store should be available and configured in conjunction with Fluentd to receive logs.

At a high level, the steps to set up Fluentd are as follows:

1. Set up a database to store the Fluentd log files. Typically, customers store log files in a log capture service such as Elasticsearch or in a database server such as PostgreSQL.

2. Configure Fluentd logging for your Composer installation. See Configure Unified Logging Using Fluentd. This includes configuring the Fluentd `td-agent.conf` file to identify the Composer microservice log data you want logged.

3. Enable unified logging and configure the host and port information for your Fluentd server for each microservice log file from which you are extracting data for Fluentd logging. See Enable Unified Logging In Composer Using Fluentd.

Configure Unified Logging Using Fluentd

For information and steps on installing Fluentd, refer to Fluentd's installation documentation. If Fluentd is not installed on the same server as the Composer, you will need to obtain the host and port numbers for the Fluentd server.

This section provides describes how

to configure Fluentd to log Composer log data.

Configure Fluentd for logging Composer log data:

1. Run the following command on the server where Composer is installed to install `td-agent` on that server.

```
curl -L https://toolbelt.treasuredata.com/sh/install-redhat-td-agent3.sh
```

2. Browse to the `/etc` folder on the Composer server to verify that `td-agent` is installed in the `/etc/td-agent` folder.

3. Install the Fluentd plugin for the data store you are using to store your Fluentd logs. Refer to your Fluentd documentation for more information.

   For example, if you are using an Elasticsearch database to store Fluentd logs. run the following command to install the Fluentd Elasticsearch plugin on the Composer server.

```
sudo td-agent-gem install fluent-plugin-elasticsearch
```

4. Modify the `td-agent.conf` file in the `/etc/td-agent` folder using the following template.

   The following is an example of `td-agent.conf` using an Elasticsearch database for Fluentd logs.

```
<source>
   @type forward
   port <fluentd_port>
   bind 0.0.0.0
</source>
<match *.**>
   @type copy
   <store>
      @type elasticsearch
      host <elasticsearch_host>
      port <elasticsearch_port>
```

```
      user <elasticsearch-user>
      password <elasticsearch-password>
      logstash_format true
      logstash_prefix composer-unified-log
      logstash_dateformat %Y%m%d
      include_tag_key true
      index_name composer-unified-log
      type_name composer-unified-log
      tag_key @log_name
      flush_interval 1s
   </store>
   <store>
      @type stdout
   </store>
</match>
```

5. Restart `td-agent`.

```
systemctl restart td-agent
```

6. Complete the steps described inEnable Unified Logging In Composer Using Fluentd to enable Fluentd logging by Composer.

Enable Unified Logging in Composer Using Fluentd

Identify the Composer microservices for which you want activity logged to Fluentd. A complete list of the Composer microservices is given in ComposerSymphony Data Discovery Microservice Name Reference, but not all microservices support unified logging. You can enable unified logging for these microservices:

- Web microservices (`zoomdata.properties`)
- Query engine (`query-engine.properties`)
- Data source connectors (see Connector Properties And Property Files for a list of property files)
- Data Writer framework (`data-writer-postgresql.properties`)

Enable unified logging using Fluentd for a Composer microservice

1. On your Composer server, edit the `.properties` file for the microservice. For example, `etc/zoomdata/query-engine.properties`.
2. Define the following parameters in the `.properties` file:

Property Syntax	Description
`logging.unified.level = <log-level>`	Specify one of the following log levels for `<log-level>`: ERROR, WARN, INFO, DEBUG, TRACE, or OFF. If set to OFF, Fluentd unified logging is disabled.
`logging.unified.tag = <service-tag>`	Specify the unique tag for the Composer microservice, used in grouping logs. This tag must be unique throughout Composer. Supply the Composer microservice name for `<service-tag>`. For example, `logging.unified.tag = query-engine`. This is important because the tag identifies which microservice the log message applies to. Valid values are `query-engine`, `zoomdata-server`, `data-writer-postgresql`, and `edc-<connector-name>` (where `<connector-name>` is one of the names listed in Connector Properties And Property Files).
`logging.unified.label = <service-label>`	Specify a more verbose label for the Composer microservice.
`logging.unified.host = <yourFluentdServerIPaddress>`	Specify the IP address of the Fluentd logging server.
`logging.unified.port = <yourFluentdServerPort>`	Specify the port of the Fluentd logging server.

3. Save your changes and exit the `.properties` file.
4. Restart the Composer server.

On the Fluentd server, you can then direct your logs to a data source of your choice. For information and steps, see Fluentd's documentation on Output.

Manage Activity Logs

The Composer server records user- and server-based activities in log files. These files can be used by Composer administrators to troubleshoot issues that may occur with the Composer server, for example, activities related to setting up data sources or creating visuals and dashboards. When Composer is installed in your network environment, a log file is created.

By default, activity logging is enabled, but some activities are not logged by default. For details, see the Activities Log Reference Sheet.

Important: Activity logging in `zoomdata-activity.log` is deprecated and will be removed in a future release. Information previously captured in `zoomdata-activity.log` can be found in other log files such as `access.log`, `service.log`, and by using Composer's User Auditing feature.

You can enable or disable activity logging by calling REST API endpoints. See Activity Logging. A list of the types of activities that are logged can be found in the Activities Log Reference Sheet.

For more information about other Composer log files, see ComposerSymphony Log Files Reference.

Activity Logging

The Composer server records user and server-based activities in log files. These files can be used by Composer administrators to troubleshoot issues that may occur with the Composer server (for example, activities related to setting up data sources or creating visuals and dashboards).

Composer logs activities in `zoomdata-activity.log`. By default, logging to this file is enabled. Some activity types are not automatically logged. To determine which activities are logged by default, see the Activities Log Reference Sheet.

Important: Activity logging in `zoomdata-activity.log` is deprecated and will be removed in a future release. Information previously captured in `zoomdata-activity.log` can be found in other log files such as `access.log`, `service.log`, and by using Composer's User Auditing feature.

This page covers the following topics:

- Configure The Composer Activity Log
- Enable Or Disable The Logging Of Specific Activities
- Determine Whether An Activity Is Being Logged
- Example Of Composer Activity Logging Monitored By Fluentd

If you want to use unified logging, see Set Up Unified Logging Using Fluentd. Fluentd unified logging is not enabled by default.

Configure the Composer Activity Log

The Composer activity log defaults can be configured through properties set in the `zoomdata.properties` file. See Configure Composer for more information about modifying property files.

Property	Description	Default
log.file.count	Number of log files to keep	1
activity.log.file.size	Maximum log file size in Mb	10
activity.logs.dir	File path to store log files. Verify that this log directory has all the necessary permissions and that the owner of the directory is set to `zoomdata`. The `/home` directory cannot be used for logging.	Linux: /opt/zoomdata/logs/ Windows: <install-path>/logs/

Enable or Disable the Logging of Specific Activities

Enabling or disabling the logging of specific activities is performed via a series of REST API calls or by directly updating the appropriate activity property in the `zoomdata.properties` file. For a list of the activities that can be logged and the file in which they can be logged, see the Activities Log Reference Sheet.

To enable or disable logging for a specific activity using the REST API, run the following cURL command:

```
curl -u supervisor: <password> -XPUT 'http://<host>:<port>/composer/api/system/activity/type/<activityType> ' -H "Content-Type: application/vnd.composer.v3+json" --data '<option>'
```

- `<host>:<port>` - Specify the host IP address and port of the Composer instance.
- `<activityType>` - Specify the name of selected activity (for example, `AUTHENTICATION` or `USER`).
- `<option>` - Specify either `true` or `false`. Set the value to `true` to enable the selected activity output or `false` to disable it.

To enable or disable logging for a specific activity by changing the appropriate activity property in the properties file:

1. Locate the `zoomdata.properties` file. For information about accessing Composer configuration (properties) files, see Configure Composer.
2. Add or edit the appropriate activity property in the file using the following format:

   ```
   activity.<activity-name>=<option>
   ```

   Activity names (`<activity-name>`) are listed in the Activities Log Reference Sheet.

   For `<option>`, specify either `true` or `false`. Set the value to `true` to enable the selected activity output or `false` to disable it.
3. Save the properties file.
4. Restart Composer microservices. See Restart Composer Microservices.

Determine Whether an Activity Is Being Logged

To determine whether a specific activity type is being logged in the log file, run the following cURL commsand:

```
curl -u supervisor: <password> -XGET 'http://<host>:<port>/composer/api/system/activity/type/<activityType>'
```

- `<host>:<port>` - Specify the address of the instance on which Composer is installed.
- `<activityType>` - Specify the type of selected activity (for example, `AUTHENTICATION` or `USER`). For a list of the activities that can be logged, see the Activities Log Reference Sheet.

Example of Composer Activity Logging Monitored by Fluentd

This example shows how to set up Fluentd to monitor Composer activity log files. It does not show how to connect your Fluentd service to an external data store or how to pipe the log information to the external data store.

Before you use this example, make sure Fluentd is installed. Next, create a log file and grant the `td-agent` service permissions to write to it. Finally, modify the `td-agent.conf` file.

Linux

Create the file `/opt/zoomdata/logs/zoomdata-activity.log.pos` and run the following command to grant the `td-agent` service permissions to write to it:

```
chmod 777 /opt/zoomdata/logs/zoomdata-activity.log.pos
```

Finally, modify the `td-agent.conf` file in the `/etc/td-agent` folder using the template below.

Windows

Create the file `<install-path>/logs/zoomdata-activity.log.pos`. Grant the `td-agent` service permissions to read, write and execute permission to the owner, group and public.

Finally, modify the `td-agent.conf` file in the `/etc/td-agent` folder using the template below. Replace `pattern /zoomdata/service/health` with `<install-path>/service/health`.

Template

- Substitute Composer fully qualified log file name for ***<logfile>*** and ***<logfilen>*** (for example, `opt/zoomdata/logs/zoomdata-activity.log`) for Linux, or `<install-path>/logs/zoomdata-activity.log` for Windows. Composer log files names are provided in ComposerSymphony Log Files Reference.
- For each log file you select, specify rules in `<rule>` sections for the information you want to extract from the Composer log files.

```
<source>
  @type forward
  @id input1
  port 24224
</source>
<source>
  @type tail
  path <logfile>[,<logfilen>]...
  pos_file zoomdata-activity.log.pos
  tag all.activity
  format json
  refresh_interval 1
</source>
<match all.activity>
  @type rewrite_tag_filter
  <rule>
    key     activityType
    pattern ^ACCOUNT$
    tag     account_logs
  </rule>
  <rule>
    key     activityType
    pattern ^AUTHENTICATION$
    tag     authentication_logs
  </rule>
  <rule>
    key     activityType
    pattern ^SOURCE$
    tag     source_logs
  </rule>
  <rule>
    key     activityType
    pattern ^RAW_DATA_EXPORT$
    tag     raw_data_export_logs
  </rule>
  <rule>
    key     activityType
    pattern ^RAW_DATA_EXPORT_CSV$
    tag     raw_data_export_csv_logs
  </rule>
  <rule>
    key     activityType
    pattern ^UPLOAD$
    tag     upload_logs
```

```
  </rule>
  <rule>
    key     activityType
    pattern ^USER$
    tag     user_logs
  </rule>
  <rule>
    key     activityType
    pattern ^VIS$
    tag     vis_logs
  </rule>
  <rule>
    key     activityType
    pattern ^GROUP$
    tag     group_logs
  </rule>
</match>

### Uncomment the section below to shift Zoomdata eventDates (UTC by default) to another timezone
### Adjust the quoted value in `Time.zone_offset("EDT")` below to the desired timezone value
### Additional details: https://docs.ruby-lang.org/en/2.4.0/Time.html#method-c-zone_offset
#<filter *_logs>
#  @type record_transformer
#  enable_ruby true
#  <record>
#     eventDate ${(Time.parse(record["eventDate"]).utc + Time.zone_offset("EDT")).strftime("%Y-%m-%d %H:%M:%S.%L")}
#  </record>
#</filter>

#type topology timeline data as json
<filter topology_performance_logs>
  @type record_transformer
  enable_ruby true
    <record>
      timeline ${record["timeline"].to_json}
    </record>
</filter>

#expand topology timeline json to new keys (attributes)
<filter topology_performance_logs>
  @type parser
  key_name timeline
  reserve_data true
  <parse>
```

```
    @type json
  </parse>
</filter>

#derive source information from the request payload
<filter vis_command_logs>
  @type record_transformer
  enable_ruby true
  <record>
    source_id ${if record["status"] == "STARTED" then JSON.parse(record["request"])["source"]["id"] end}
    source_name ${if record["status"] == "STARTED" then JSON.parse(record["request"])["source"]["name"] end}
    source_type ${if record["status"] == "STARTED" then JSON.parse(record["request"])["source"]["subStorageType"] end}
    source_schema ${if record["status"] == "STARTED" then JSON.parse(record["request"])["source"]["storageConfiguration"]
["schema"] end}
    source_collection ${if record["status"] == "STARTED" then JSON.parse(record["request"])["source"]
["storageConfiguration"]["collection"] end}
    connection_id ${if record["status"] == "STARTED" then JSON.parse(record["request"])["source"]["storageConfiguration"]
["connectionId"] end}
  </record>
</filter>

#exclude health check requests
#<filter request_logs>
#  @type grep
#  <exclude>
#    key uri
#    pattern /zoomdata/service/health
#  </exclude>
#</filter>

#send fluent log entries to stdout
<match fluent.**>
  @type stdout
</match>
```

Activities Log Reference Sheet

This reference sheet lists the types of activities and corresponding fields that are logged in the Composer activity log. By default, activity logging is enabled with a few exceptions (described later in this topic).

Activity Logging Defaults

The following table lists the activities that are logged by default and those that are not. Activities that are not logged by default must have logging manually enabled. See Activity Logging.

Important: Activity logging in `zoomdata-activity.log` is deprecated and will be removed in a future release. Information previously captured in `zoomdata-activity.log` can be found in other log files such as `access.log`, `service.log`, and by using Composer's User Auditing feature.

Activity Log File	Activities Logged by Default	Activity Not Logged by Default
`zoomdata-activity.log`	account authentication group SOURCE USER VIS	raw_data_export raw_data_export_csv upload

If logging or specific activity logging is disabled in a previous version, the disabled status is retained after an upgrade.

Common Log Fields

For each activity, the following common fields are logged as part of each activity record:

Field Name	Description
user	contains information about the tenant user and the actions performed
accountID	contains the user's current tenant ID
userGroups	list of the groups to which the user belong
IP	contains IP address from which the event has been performed

Additional fields are logged as part of each activity.

Activity Log Fields

The following activity types can be logged by Composer. For each activity, specific fields are logged, in addition to the common fields.

- account: Account Maintenance
- authentication: User Authentication
- Group: Group Maintenance
- Raw_data_export: Text Search
- Raw_data_export_csv: Data Export To CSV
- Source: Data Source Maintenance
- Upload: Flat File Upload
- User: User Account Maintenance
- Vis: Visual Maintenance

`account`: Account Maintenance

This activity is logged when a Composer tenant account is created, updated, or deleted. Logging for this activity is enabled by default and captured in the `zoomdata-activity.log` file.

The following fields are recorded for this activity:

Field Name	Description
activityType	account
status	CREATED / UPDATED / DELETED
accountId	Generated tenant account ID
name	Name of the tenant account from which the event has been triggered
createdBy	User who created this tenant account.

Field Name	Description
createdDate	Tenant account creation date
lastModifiedBy	User who modified the tenant account
disabled	TRUE / FALSE

`authentication`: User Authentication

This activity is logged while user authentication at login. Logging for this activity is enabled by default and captured in the `zoomdata-activity.log` file.

The following fields are recorded for this activity:

Field	Value
activityType	authentication
status	SUCCEEDED / FAILED
authenticationType	BASE, SAML, LDAP, x.509

`group`: Group Maintenance

This activity is logged when a user group is created, updated, or updated. Logging for this activity is enabled by default and captured in the `zoomdata-activity.log` file.

The following fields are recorded for this activity:

Field	Value
groupId	Group ID
label	Name of the group
description	Group description added by the user
group	Privileges that have been assigned for the group

`raw_data_export`: Text Search

This activity is logged when a text search (applicable only for Elasticsearch, Apache Solr, and Cloudera Search sources) is performed using the `/api/stream/search` endpoint. Logging for this activity is disabled by default. After you have manually enabled logging for this activity, its log records are captured in the `zoomdata-activity.log` file.

The following fields are recorded for this activity:

Field	Value
activityType	raw_data_export
status	SUCCEEDED
exportType	REST (view details) FILE (export file)
count	Number of exported rows
storageType	Data source
query	Queries sent to the data source
cid	Created ID
actionStartedOn	Export start time
duration	Request processing time

`raw_data_export_csv`: Data Export to CSV

This activity is logged when a user exports the raw data to a CSV file. Logging for this activity is disabled by default. After you have manually enabled logging for this activity, its log records are captured in the `zoomdata-activity.log` file.

The following fields are recorded for this activity:

Field	Value
activityType	raw_data_export_csv
status	SUCCEEDED
count	Number of exported rows
location	DB or FILE
file	location=FILE - the full path to the file in local file system
cid	Created ID
actionStartedOn	Export start time
duration	Request processing time

`source`: Data Source Maintenance

This activity is logged when a source is created, updated, or deleted. Logging for this activity is enabled by default and captured in the `zoomdata-activity.log` file.

The following fields are recorded for this activity:

Field	Value
activityType	source
status	SAVED, UPDATED, DELETED
sourceId	Generated source ID
sourceName	Name of the source
streamType	Demo_record, CSV, API
storageType	Data source type
sourceAsString	Contains microservice info about the data source
sourceDescription	Source description added by the user

`upload`: Flat File Upload

This activity is logged when a user uploads the flat file while creating a new data source. Logging for this activity is disabled by default. After you have manually enabled logging for this activity, its log records are captured in the `zoomdata-activity.log` file.

The following fields are recorded for this activity:

Field	Value
activityType	upload
status	SUCCEEDED
source	Data source ID to which the file will be uploaded
fileName	Name of the uploaded file
contentType	Type of the uploaded file
filesize	Size of the uploaded file

`user`: User Account Maintenance

This activity is logged when user account is created, updated, assigned to or removed from a group, or deleted. Logging for this activity is enabled by default and captured in the `zoomdata-activity.log` file.

The following fields are recorded for this activity:

Field	Value
activityType	user

Field	Value
status	CREATED / UPDATED / DELETED
userID	
userName	
userFullname	
email	
subjectUserGroups	Groups which a user belongs to
userOrigin	NATIVE / SAML / LDAP
accounts	Tenant a ccounts, which a user is assigned to

`vis`: Visual Maintenance

This activity is logged when a visual is created or updated. Logging for this activity is enabled by default and captured in the `zoomdata-activity.log` file.

The following fields are recorded for this activity:

Field	Value
activityType	vis
status	CREATED / UPDATED
visualizationID	
visualizationName	

User Auditing

User auditing allows you as a Composer administrator or client application administrator to track your users' access and actions to data considered sensitive. This includes tracking:

- What period of time any user interacted with data from dashboards, visuals, sources, and custom SQL queries.
- How the data was accessed: directly in the Composer user interface, via an API, or as embedded objects in your applications.
- Actions performed with the data. See User Audit Events.
- For all sensitive data your applications need to track data access for, such as regulated industry information, PII, HIPAA, GDPR, and more.

Write your audit logs to the PostgreSQL database installed with Composer, or an alternative PostgreSQL database of your choice. This enables your administrators to access and manage the data collected to suit your organization's needs.

The following topics cover enabling, configuring, and consuming user auditing and user audit data:

- Enable and Configure User Auditing
- Consuming Audit Data
- User Audit Events
- User Auditing for Multi Tenancy Environments
- Enable User Audit Data for Composer Accounts
- Enable User Audit Data for Multi Tenant Accounts

Enable and Configure User Auditing

Write your audit logs to the PostgreSQL database installed with Composer, or an alternative PostgreSQL database of your choice. Create the database, update the `zoomdata.properties` file, and restart Composer to begin capturing audit events.

Note: If you expect a large number of user audit events, accumulating a significant amount of information over time, consider using a separate PostgreSQL database.

Create a Database

Create the database for user auditing. There are two ways to do so:

Note: PostgreSQL databases are the only supported database type.

1. Bootstrap creation. When you install or upgrade Composer using a `bootstrap-zoomdata` script, a user audit data table is created on the local PostrgreSQL instance. The user audit database is named `zoomdata-user-auditing`, and installed at the same location as the Logi metadata for your Composer instance.

2. Manual creation. To use a separate database or PostgreSQL instance from your Composer metadata database, edit the `zoomdata.properties` file to reflect the target parameters to use. Set the default value of `destination.params.password` to the same database password used in your default PostgreSQL database. Composer creates the collections automatically in that database as needed.

Enable User Auditing

After you have created or linked your database, copy these properties into the `zoomdata.properties` file.

```
user-auditing.enabled=true
user-auditing.destination.name=PostgreSQL
user-auditing.destination.type=postgresql
user-auditing.destination.schema=public
user-auditing.destination.collection=audit_records
user-auditing.destination.collection-per-account=false
user-auditing.destination.params.user_name=${db.username:zoomdata}
user-auditing.destination.params.password=${db.password:}
user-auditing.destination.params.jdbc_url=jdbc:postgresql://localhost:5432/zoomdata-user-auditing
user-auditing.tenant.attribute=
```

Required and optional properties include:

Property	Default Value	Description
enabled	false	Use to enable or disable user auditing. Default is false, to disable user auditing. To enable, set to true. Required.
destination.name	PostgreSQL	The name of the type of database. Only PostgreSQL is supported.
destination.type	postgresql	The database type. Only postgresql is supported.
destination.schema	public	
destination.collection		The name of the collection of the user audit data. Use to separate data by accounts to prevent access to audit data by unauthorized users. For example, `audit_records<Account ID>`.
destination.collection-per-account	fals	Default is false, to disable To enable, set to true and include a field for Account ID in destination.collection.
destination.params.user_name	zoomdata	The user name sent to the database to write audit data. Required.
destination.params.password		The password sent to the database to write audit data. Required. Can be the same as your default PostgreSQL database.
destination.params.jdbc_url	jdbc:postgresql://localhost:5432/zoomdata-user-auditing	The path to the audit data database. This can be the same database as the user audit schema as shown, or a separate database or partition, as needed. Required.
tenant.attribute		Include any new or existing user attribute to support data capture by tenant customer users. Use in addition to account information that is always captured and retained in the database.

Restart Composer

Restart Composer to begin capturing audit events.

If you used the bootstrap script to install Composer and audit setup, the audit database table is created when the first event is logged. Test your setup by performing any of the event-triggering actions. See User Audit Events.

If you've set up your database manually, set up the Data Writer microservice to support the user audit processes.

User Auditing for Multi Tenancy Environments

Composer supports two different levels of separation for user audit data. By default, user audit data is separated between accounts. If needed, you can further separate user audit data within a Composer account by adding and using a custom attribute to separate the data.

Use Composer to write audit data to a single table in your database or multiple tables to suit your organization's needs.

Important: Database tables that contain user audit data are created automatically when you first trigger an audit event. Trigger table creation with user auditing enabled and properly configured before you create a data source for the tables or apply access controls at the database level.

Separation Between Composer Accounts

Audit data for Composer users, including administrators, are not visible to other accounts. You define this level of separation in the database in one of three ways:

- Tables: Configure Composerto collect and write user audit data to different tables, then give users database access only to their own audit data table.
- Views: Disable user auditing by account, then give users access to their own audit data, using specific views and database access rights.
- Row level security: Disable user auditing by account, then configure row level security, limiting access for each user to their own data.

See Enable User Audit Data for Composer Accounts.

Separation Between Tenants Within the Same Composer Account

Configure user auditing for tenants to restrict access to user audit data to their own user tenant data. Composer system administrators can see aggregated data for all tenants in the Composer account.

You can define this level of separation in the database in one of several ways, depending on tenant connection creation privileges.

- If tenant users don't have permission to create new connections, add a user attribute value, then make the data available by applying the appropriate row level filter to that value.
- If tenant users do have permission to create new connections, you must set up database-level access control mechanisms.
  - Views per tenant: Define a tenant attribute, then define what user audit data users can access, using specific views and database access rights.
  - Database level row level security: Define a tenant attribute, then configure row level security for the audit table data, allowing users access to appropriate user audit data through individual user database accounts.

See Enable User Audit Data for Multi Tenant Accounts.

Enable User Audit Data for Composer Accounts

Audit data for Composer users, including administrators, are not visible to other Composer accounts. You define this level of separation in the database in one of three ways:

- Tables: Configure Composer to collect and write user audit data to different tables, then give users database access only to their own audit data table.
- Views: Disable user auditing by account, then give users access to their own audit data, using specific views and database access rights.
- Row level security: Disable user auditing by account, then configure row level security, limiting access for each user to their own data.

Important: Database tables that contain user audit data are created automatically when you first trigger an audit event. Trigger table creation with user auditing enabled and properly configured before you create a data source for the tables or apply access controls at the database level.

Tables for a Composer Account

1. Use separate collections for accounts by defining `user-auditing.destination.collection-per-account=true` in the `zoomdata.properties` file.
2. Generate an audit event for all accounts to trigger audit data table creation.
3. Create user database accounts for each user. Define user access rights for each user to query only their own account's audit data table.
4. Create connections to the audit database using each account.

Views by Composer Account

1. Disable use of separate collections for accounts by defining `user-auditing.destination.collection-per-account=false` in the `zoomdata.properties` file.
2. Generate an audit event for any account to trigger audit data table creation.
3. Create a view for each account that filters the audit data table, using a condition such as `accountID='<Account ID>'`.

4. Create user database accounts for each user. Define user access rights for each user to query only their own account's audit data view.
5. Create connections to the audit database using each account.

Row Level Security in the Database

1. Enable user auditing by defining `user-auditing.destination.collection-per-account=false` in the `zoomdata.properties` file.
2. Generate an audit event for any account to trigger audit data table creation.
3. Create user database accounts for each user, and configure row level security for the audit data table to limit users to their own data. For example, use a filter such as `accountID='<Account ID>`.
4. Create connections to the audit database using each account.

Enable User Audit Data for Multi Tenant Accounts

Configure user auditing for tenants to restrict access to user audit data to their own user tenant data. Composer system administrators can see aggregated data for all tenants in the Composer account.

You can define this level of separation in the database in one of several ways, depending on tenant connection creation privileges.

Important: Database tables that contain user audit data are created automatically when you first trigger an audit event. Trigger table creation with user auditing enabled and properly configured before you create a data source for the tables or apply access controls at the database level.

Tenants Without Connection Creation Permissions

In environments where tenants don't have permission to create connections, you can set up and use user attribute value, such as `${User.tenant}` define row level filters to limit access to user audit data.

1. Set up an attribute for each user, such as `${User.tenant}`, to identify the tenant for each user.
2. Add the attribute name to the `zoomdata.properties` file. In this example, add `tenant: user-auditing.tenant.attribute=tenant`.
3. Generate an audit event for any account to trigger audit data table creation.
4. As a system administrator, create connection and data sources for the audit data in Composer.
5. Add a row level filter to the data source. Compare the column tenant with the user attribute you created. In this example, `tenant=${User.tenant}`

Tenants With Connection Creation Permissions

If your tenants do have permission to create connections, there are two ways to restrict user audit data access: by setting up views at the tenant level, or applying row level security. Use the features of your database to control access to the data, once attributes are defined and applied.

Views Per Tenant

1. Set up an attribute for each user, such as `${User.tenant}`, to identify the tenant for each user.
2. Add the attribute name to the `zoomdata.properties` file. In this example, add `tenant: user-auditing.tenant.attribute=tenant`.
3. Generate an audit event for any account to trigger audit data table creation.

4. Create a view for each tenant that filters the audit data table, using a condition such as `tenant='<Tenant>'`.
5. Create user database accounts for each tenant. Define access rights for account user to query only their own tenant audit data table.
6. Create connections to the audit database using each user database account.

Row Level Security

1. Set up an attribute for each user, such as `${User.tenant}`, to identify the tenant for each user.
2. Add the attribute name to the `zoomdata.properties` file. In this example, add `tenant: user-auditing.tenant.attribute=tenant`.
3. Generate an audit event for any account to trigger audit data table creation.
4. Create a separate database user for each tenant, and configure row level security for each account user to query only their own tenant audit data table, using a condition such as `tenant='<Tenant>'`
5. Create connections to the audit database using each user database account.

Consuming Audit Data

Users with appropriate access can review your audit data in one of three ways:

- Create visuals and a dedicated dashboard using Composer.
- Direct database queries using any third-party tool.
- Add the user auditing database as a source and connection, then export the information using the Composer API.

Consume Audit Data Using Composer

Use Composer to create a User Auditing dashboard.

1. Create a connection to your user auditing database.
2. Create a source from this connection.
3. Create a dashboard for User Auditing with visuals as needed.

Note: If you use a table visualization for user auditing data, add cross-visual filters such as List Filters and Tree Maps for a contextual look at your data.

Consume Audit Data Using Third-Party Tools

You can use any third-party tools you prefer to access the information for user auditing. Access to the tool or to the database using that tool should be restricted to appropriate users.

User Audit Events

Specific user auditing events captured include:

Event	Description
COLLECTION_PREVIEW	User sees a collection data preview while creating or editing a source.
FACECT_VALUES_ACCESS	User sees values of a text search facet.
FIELD_STATS_ACCESS	User sees a field minimum and maximum values while trying to apply a filter or while reviewing an applied filter. User opens a visual or dashboard that includes a Time Bar.
FIELD_VALUES_ACCESS	User sees a list of field values while trying to apply a filter or while reviewing an applied filter.
KEY_SET_PREVIEW	User creates a keyset from a visual result set or data point, showing distinct values for a field. One keyset event per field.
KEY_SET_REVIEW	User reviews a keyset before applying it to a field, or after application, or prior to deletion.
RAW_DATA_EXPORT	User exports raw data for a visual or Details view, in .csv format.
VISUAL_DATA_EXPORT	User exports displayed data for a visual, in .csv format.
VISUAL_DATA_ACCESS	User creates, opens, changes a dashboard or loads data as new or a refresh.
TEXT_SEARCH	User performs a text search.

Enable Composer Component Access From Other Sites Using Cross-Origin Resource Sharing (CORS)

Cross-origin resource sharing (CORS) is a standard introduced in HTML 5 that allows web applications to use HTTP headers to specify which origins are permitted to request resources on the server. Cross-origin resource sharing provides a web application access to selected resources running at a different location. Modern web browsers will consult the Access-Control-Allow-Origin header on how to relax the same origin policy for a given page. By default, this setting was disabled starting version 1.5.0SR1 due to security vulnerability concerns. However, CORS can be enabled for certain or all domains by editing the `zoomdata.properties` file (located in `/etc/zoomdata`). For more information about CORS, see https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS.

Modify CORS request permissions

1. From your terminal, open a command line session.
2. Connect to your Composer server via a command prompt.
3. Use the following command to access and open the `zoomdata.properties` file:

   ```
   vi /etc/zoomdata/zoomdata.properties
   ```

4. Add the following variables to the file on new lines:

   ```
   http.response.header.content-security-policy.frame-ancestors=<source1> <source2>
   access.control.allow.origin=<origin1>,<origin2>
   ```

   Replace `<source>` with the actual sources that may embed resources. The default is an asterisk (`*`), or all sources. For more information about the Content-Security-Policy (CSP) frame-ancestors directive, see https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy/frame-ancestors.

   Replace `<origin1>` and `<origin2>` with your specific URI including `http` or `https` as appropriate. Use `*` as a wildcard instead of a specific origin, thereby allowing any origin to access the resource, keeping in mind that this is a potential security vulnerability and may not work with all browsers. Refer to your Tomcat documentation to verify the appropriate syntax to use.

5. Save the document using standard `vi` commands.
6. Restart Composer microservices after making this change and also make sure to clear your browser cache. See Restart Composer Microservices.

Manage the Upload API Source

Composer supports live streaming sources. This source uses PostgreSQL as a storage medium for the incoming data flow. By default, Composer uses PostgreSQL to store the data for Upload API. This PostgreSQL instance is separate from other PostgreSQL instances you may have in existence. You can elect to use another PostgreSQL instance that you may have by changing the configuration properties.

Before You Begin

If you want to use an existing PostgreSQL instance in conjunction with the Upload API microservice, you need to find the above information in your `edc-postgresql.properties` file.

Use Another PostgreSQL Instance for Upload API

1. Use the following command to access and open the properties file:

```
vi /etc/zoomdata/zoomdata.properties
```

2. Add the following parameters to the file:

```
#  that are configure for Upload API
upload.destination.params.user_name=yourusername
upload.destination.params.password=yourpassword
upload.destination.params.jdbc_url=jdbc:postgresql://yourlocalhost:yourport//zd_upload
upload.batch-size=1000
```

3. Save and exit the configuration file.

4. Restart the Composer server.

Translate the Composer Interface

You can translate the Composer interface into other languages. Composer supports standard i18n languages. See https://perldoc.perl.org/I18N::LangTags::List#LIST-OF-LANGUAGES.

> **Note:** To translate the interface, you must have a specific advanced setting switched on. Please contact technical support for information about this setting.

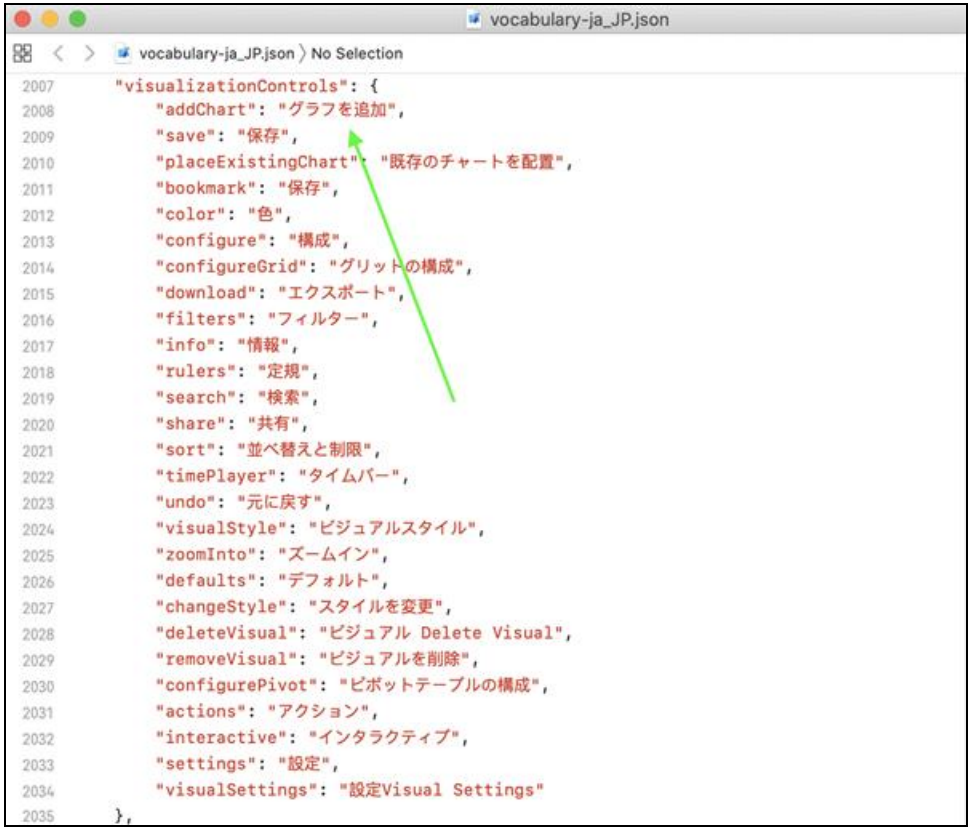
Translation files in the format `vocabulary-<xx>_<YY>.json` are used in the translation. The English translation file is provided and is named `vocabulary-en_US.json`.

- Linux Environments: Located in the `/opt/zoomdata/client/i18n` folder.
- Windows Environments: Located in the `<install-path>/client/i18n` folder.

Translate the Composer interface:

1. Locate and copy the provided English translation file `vocabulary-en_US.json` in the appropriate `/client/i18n` folder.
2. Rename the copy of the file. Be sure to name it in the format `vocabulary-<xx>_<YY>.json`, where `<xx>` is a two-character code representing the language you want to use and `<YY>` is the dialect. See https://perldoc.perl.org/I18N::LangTags::List#LIST-OF-LANGUAGES.
3. Translate the JSON values to the right of the JSON keys (after the colon) in the file. All values should be specified in quotes. The JSON keys should not be translated because the keys are used by the Composer code.

   Here is an example of part of a translated translation file:



This file is CSS-friendly. For example, `<b>` and `</b>` around a JSON value or part of a value indicate that the value should be bold.

Finally, placeholders (variables) are used throughout the file and should not be changed or removed, although they can be relocated within a JSON value, as needed for your translation. Placeholders appear surrounded by `%` symbols (for example, `%name%`) in the translation file. Composer code inserts a value for these placeholders, based on where the JSON key is used in the product. For example, a dashboard name might be inserted for the placeholder `%name%` for one JSON key, but a visual name might be inserted for a different JSON key that also uses the `%name%` placeholder.

4. Save your translated file.

5. Contact your insightsoftware technical support representative for additional steps.

Note: If you import an exported source that has an associated translation file, you must re-upload the translation for that source.

Server-Level Variables

Server-level variables can be viewed by Composer administrators or members of the Supervisors group in the Server-Level Variables work area.

Important: Server-level variables are set during installation and must not be changed without understanding how the change affects or compromises your environment. If you must change a variable here, contact Technical Support. Do not add or delete server-level variables.

Select the **Advanced** menu option to access the Server-Level Variables work area.

Server-level variables are defined as key-value pairs. You can enable or disable the listed variables below if needed. Select **Save** to save and apply any changes you make in this work area.

Caution: Changing toggles or editing content other than as instructed in this work area may prevent your users from using various components of Composer.

Key (Server-Level Variable)	Value	Description
allow-dashboard-sharing-within-tenant	false (default)	Enable or disable dashboard sharing options for sharing dashboards with groups and everyone within your environment. See Share a Dashboard with Users.
scheduled-report-file-drop	false (default)	Enable or disable users' ability to deliver a scheduled report to an SFTP location. When set to `true`, users can select an SFTP file location you have defined to accept the scheduled report. When set to `false`, users do not see an SFTP option for scheduled reports. Define the settings for your environment in `zoomdata.properties`. See Scheduled Dashboard Report PropertiesScheduled Data Discovery Dashboard Report Properties.

Configure Composer Logs

The microservices used in Composer write logs to their corresponding service's log files by default. You can configure Composer to write these logs to the console only, or to write logs both to the console and the service's log files. Structured logging is also supported. See Structured Logging.

Every service config file includes two properties you can use to define where the logs are written:

- `log.console.level` - edit to define the types of logs to write to the console
- `log.file.level` - edit to define the types of logs to write to the files

Log Properties in Config Files

If you install using the bootstrap script, logging to files is enabled by default. You can reroute the logging to the console as needed to customize your installation.

Note: If the value of log properties is set to `OFF`, Composer access logs are not written. For all other cases (`ALL`, `ERROR`, `WARN`, `DEBUG`, and `INFO`) access logs are written to the selected destination: file, console or both.

- Bootstrap installation: The `*.jvm` file specifies `log.console.level=OFF` and `log.file.level=ALL` to support compatibility with earlier versions of Composer.

Values for log.file.level and log.console.level

The following table lists possible values for `log.file.level` and `log.console.level`.

Value	Description
`ALL`	All available log information.
`ERROR`	Application error messages that may affect processes.
`WARN`	Unexpected application issues that may not affect processes.
`INFO`	Expected activities.
`DEBUG`	Triggers capture of `WARN`, `INFO`, and `ERROR`information.
`OFF`	Disables logging.

Enable Logging for a Service to the Console

Enable logging to the console and append the desired corresponding value of the properties to the appropriate service configuration file.

- Use `log.console.level=ALL` to route duplicates of all logs to the console.
- Use `log.console.level=ERROR` to route only duplicates of errors to the console.
- Alternatively, use `log.console.level=ALL` and `log.console.level=ERROR` to have all logs in files and errors duplicated to the console.

Adjust the values to meet your needs, then restart the service after you've edited the config.

Note: Composer services have two config files, `*properties` and `*.jvm`. If you add log configuration properties to both files, the priority of `*.jvm` is higher than `*. properties`.

Consul Logging

By default, Consul writes logs to the console, but you can duplicate logs to files if needed.

Configure Consul Logs to Duplicate to Files:

1. Edit the Consul config file, `consul.json`. For Linux installations, this is located in `/opt/zoomdata/conf/consul.json`.
2. Add the following properties:

```
"log_file": "/opt/zoomdata/logs/zoomdata-consul.log",
"log_rotate_bytes": 104857600,
"log_rotate_max_files" : 5
```

Adjust the path to your log file and other log configuration properties as needed for your environment. In Linux environments, the log is located in `/opt/zoomdata/logs`, and in `<install-path>/logs` for Windows environments. See Consul documentation for more information.

Structured Logging

Composer supports structured logging. You can enable for installations done via bootstrap, and is enabled by default for Kubernetes environments.

Components Support for Structured Logging

- zoomdata-web
- query engine
- admin server
- screenshot service
- data writer
- data connectors

Enable Structured Logging

In bootstrap environments, set the property `log.structured.enabled` to `true` in the config files of your microservices to output logs in JSON format to the `STDOUT` destination. Logs output to the `FILE` logs destination are sent in plain text. See Configuration Property Files.

In Kubernetes environments, structured logging is enabled by default. To disable, use one of the following approaches:

- Set the environment variable `LOG_STRUCTURED_ENABLED` to `false` to disable logging for specific services.
- Set `structuredLogsEnabled` in `values.yaml` to `false` to disable logging for all services.
- For Consul, change the `log_json` value to `false` in the `consul.server.extraConfig` section of `values.yaml`.

Manage Composer Microservices

You can manage Composer microservices in CentOS environments or in Ubuntu 18, 20, or 22 environments. You can also manage the microservices using the Composer CLI.

See the following links:

- Start Composer Microservices
- Stop Composer Microservices
- Enable Composer Microservices
- Disable Composer Microservices
- Restart Composer Microservices
- Scaling Composer Microservices
- Manage Composer Microservices Using The Command Line Utility
- Downloading And Installing The Solution Package

Scaling Composer Microservices

Composer microservices allow you to scale your Composer installation, giving you expanded performance gains. As you roll out your microservices changes, you'll need to:

- Estimate User and Microservice Loads
- Add or Remove Nodes in a High Availability Environment
- Configure Throughput for Microservices
- Load Monitor Microservices
- Scale Microservices Up
- Scale Microservices Down

All microservices, except composer web, support horizontal scaling. Composer web supports vertical scaling only.

Estimate User and Microservice Loads

The load used by each microservice depends on the type of interactions your users have while using Composer. Some actions load only the composer web component, while others may load multiple components.

Some examples of actions that load microservices are included in the table below.

Action	Microservices Loaded
Open the home page	composer web
Open a dashboard	composer web, connector, query engine
Open the visual Gallery	composer web
Create a source	composer web, connector, query engine
Create an uploaded source	composer web, connector, data writer, query engine
Upload new data via API	composer web, data writer
Execute scheduled dashboard report	composer web, connector, screenshot service, query engine
Live mode	composer web, connector, query engine

Note: Horizontal scaling up of a microservice doesn't provide doubled performance gains due to sharing of PostreSQL resources and overlapping of microservices.

Start your scale planning based on an approximate number of services and expected number of users. See Server Size Guidelines for more information on services and user estimation.

Add or Remove Nodes in a High Availability Environment

To scale a microservice up or down, you may need to Add Nodes to an Existing High Availability Installation or Remove Nodes from a High Availability Environment. Before you add or remove nodes, you should understand the actual load on your microservices to make the decision of scaling up or down. See Composer System Metrics.

Configure Throughput for Microservices

All microservices have a configuration property, `server.jetty.max-threads`, you can use to limit throughput. Once configured, you can more accurately monitor the load on each microservice.

Set your estimated concurrent users for each service at +20%, based on your calculations made using Server Size Guidelines. To manage configuration properties of `server.jetty.max-threads`, see Configure and Start the Configuration Microservice.

For example, if you have three instances of the query engine microservice and you expect 150 concurrent users, each instance is expected to handle 50 concurrent users. To accommodate this demand, set `server.jetty.max-threads` to (`60 = 150 / 3 & 120%`) for each instance of query engine.

Note: Composer v7.1 and earlier use the property `jetty.threadPool.maxThreads` to limit throughput for the composer web and query engine microservices.

Load Monitor Microservices

With throughput defined for your microservices, you can measure their use to decide what services to scale up or scale down.

Composer microservices exposes several different metrics to help you understand the current load on each service. The main outputs to monitor include:

- `jetty_threads_busy` shows a current count of concurrent users.
- `jetty_threads_config_max` shows the maximum allowed concurrent users. This should match the value you set in `server.jetty.max-threads`.

These show a moment in time, which can vary depending on use spikes that may not actually require scaling up or down. Smooth out the data by looking at a specific length of time, such as five minutes, and monitor the average load across all instances of the same microservice.

Average over five minutes:

```
avg by instance_type (jetty_threads_busy) over 5m / avg by instance_type (jetty_threads_config_max)
```

In Prometheus:

```
sum by (job)(sum_over_time(jetty_threads_busy[5m]))
/ sum by (job)(count_over_time(jetty_threads_busy[5m]))
/ avg by (job) (jetty_threads_config_max)
```

Scale Microservices Up

The recommended load threshold for all microservices is 90%. After setting up your thresholds, monitor them manually, or use a metric monitoring tool such as Prometheus to define alerts at that threshold.

```
sum by (job)(sum_over_time(jetty_threads_busy[5m]))
/ sum by (job)(count_over_time(jetty_threads_busy[5m]))
/ avg by (job) (jetty_threads_config_max)
> 90
```

For more information on scaling up your environment, see Add Nodes to an Existing High Availability Installation.

Note: Composer-Web does not support horizontal scaling, only vertical scaling. Increase machine resources to accommodate your needs. Configure `server.jetty.max-threads` or `jetty.threadPool.maxThreads` for Composer v7.1 and earlier after increasing machine resources.

Scale Microservices Down

When monitoring microservices for scaling up, you define and set alerts around defined load thresholds for types of services.

In contrast, when you monitor services to scale them down, monitor free resources, the amount of free threads dedicated to the system. The recommended threshold is dedicate two times the resources of a single instance `(2 * resources)`.

Average over five minutes:

```
sum by instance_type (jetty_threads_config_max) - avg by instance_type (jetty_threads_busy) over 5m
```

In Prometheus:

```
(sum by (job) (jetty_threads_config_max) - (sum by (job)(sum_over_time(jetty_threads_busy[5m])) / sum by (job)(count_over_
time(jetty_threads_busy[5m])))) / avg by (job) (jetty_threads_config_max) > 2
```

Start Composer Microservices

This topic describes how to start Composer microservices in CentOS environments or in Ubuntu 18, 20, or 22 environments.

The list of Composer microservices can be found in ComposerSymphony Data Discovery Microservice Name Reference. The order in which microservices should be started is described in ComposerSymphony Microservice Startup Order.

You can also start Composer microservices using the CLI. See Manage Composer Microservices Using The Command Line Utility.

To start all Composer microservices, enter the following command:

```
sudo systemctl start $(systemctl list-unit-files | grep zoomdata | grep edc | awk '{print $1}')
```

To start a specific Composer microservice, enter the following command:

```
sudo systemctl start <service>
```

Replace `<service>` with the name of the Composer microservice. See Composer Microservice Name Reference.

Stop Composer Microservices

This topic describes how to stop Composer microservices in CentOS environments or in Ubuntu 18, 20, or 22 environments.

The list of Composer microservices can be found in ComposerSymphony Data Discovery Microservice Name Reference.

You can also stop Composer microservices using the CLI. See Manage Composer Microservices Using The Command Line Utility.

To stop all Composer microservices, enter the following command:

```
sudo systemctl stop $(systemctl list-unit-files | grep zoomdata | awk '{print $1}')
```

To stop a specific Composer microservice, enter the following command:

```
sudo systemctl stop <service>
```

Replace `<service>` with the name of the Composer microservice. See Composer Microservice Name Reference.

Enable Composer Microservices

This topic describes how to enable Composer microservices in CentOS environments or in Ubuntu 18, 20, or 22 environments.

The list of Composer microservices can be found in ComposerSymphony Data Discovery Microservice Name Reference.

You can also enable Composer microservices using the CLI. See Manage Composer Microservices Using The Command Line Utility.

To enable all Composer microservices, enter the following command:

```
sudo systemctl enable $(systemctl list-unit-files | grep zoomdata | grep edc | awk '{print $1}')
```

To enable a specific Composer microservice, enter the following command:

```
sudo systemctl enable <service>
```

Replace `<service>` with the name of the Composer microservice. See ComposerSymphony Data Discovery Microservice Name Reference.

Disable Composer Microservices

This topic describes how to disable Composer microservices in CentOS environments or in Ubuntu 18, 20, or 22 environments.

The list of Composer microservices can be found in ComposerSymphony Data Discovery Microservice Name Reference.

You can also disable Composers microservices using the CLI. See Manage Composer Microservices Using The Command Line Utility.

To disable all Composer microservices, enter the following command:

```
sudo systemctl disable $(systemctl list-unit-files | grep zoomdata | awk '{print $1}')
```

To disable a specific Composer microservice, enter the following command:

```
sudo systemctl disable <service>
```

Replace `<service>` with the name of the Composer microservice. See ComposerSymphony Data Discovery Microservice Name Reference.

Restart Composer Microservices

This topic describes how to restart Composer microservices in CentOS Stream 9 environments or in Ubuntu 18, 20, or 22 environments.

The list of Composer microservices can be found in ComposerSymphony Data Discovery Microservice Name Reference.

You can also restart Composer microservices using the CLI. See Manage Composer Microservices Using The Command Line Utility.

To restart all Composer microservices, enter the following command:

```
sudo systemctl restart $(systemctl list-unit-files | grep zoomdata | grep edc | awk '{print $1}')
```

To restart a specific Composer microservice, enter the following command:

```
sudo systemctl restart <service>
```

Replace `<service>` with the name of the Composer microservice. See Composer Microservice Name Reference.

Manage Composer Microservices Using the Command Line Utility

You can use the `zdmanage` CLI command to manage the various microservices that are deployed in the Composer environment. This command line utility is automatically installed when installing your software using the automated installation script. This utility script wraps underlying UNIX commands to help you perform common management operations such as stopping and starting Composer microservices.

Prerequisites

To use the command line utility tool, you need root access.

Using the Command Line Utility Tool

The `zdmanage` command line utility is located in the following directory:

```
/opt/zoomdata/bin/zdmanage
```

The `zdmanage` CLI command syntax is as follows:

```
zdmanage services <command> <service_name>
```

The following commands (`<command>`) are supported:

Command	Action
`list`	Displays all the Composer microservices installed in your deployment.
`status`	Provides a status of all Composer microservices in your environment.
`start`	Starts Composer microservices. The order in which microservices should be restarted is described in ComposerSymphony Microservice Startup Order.
`stop`	Stops Composer microservices.
`restart`	Restarts Composer microservices.
`enable`	Enables Composer microservices automatically when the OS starts up.
`disable`	Shuts down Composer microservices automatically when the OS starts up.
`configure`	Opens the specified Composer property file.

For `<service_name>`, specify the microservice name or `all` (to apply the command to all Composer microservices). A complete list of the Composer microservices can be found in ComposerSymphony Data Discovery Microservice Name Reference.

Common Commands

Show all Composer microservices installed:

```
sudo /opt/zoomdata/bin/zdmanage services list
```

Provide a status of all installed Composer microservices:

```
sudo /opt/zoomdata/bin/zdmanage services status all
```

Show the status of the Composer web microservice:

```
sudo /opt/zoomdata/bin/zdmanage services status zoomdata
```

Stop a specific Composer microservice:

```
sudo /opt/zoomdata/bin/zdmanage services stop [zoomdata-service_name]
```

Composer Monitoring Solution

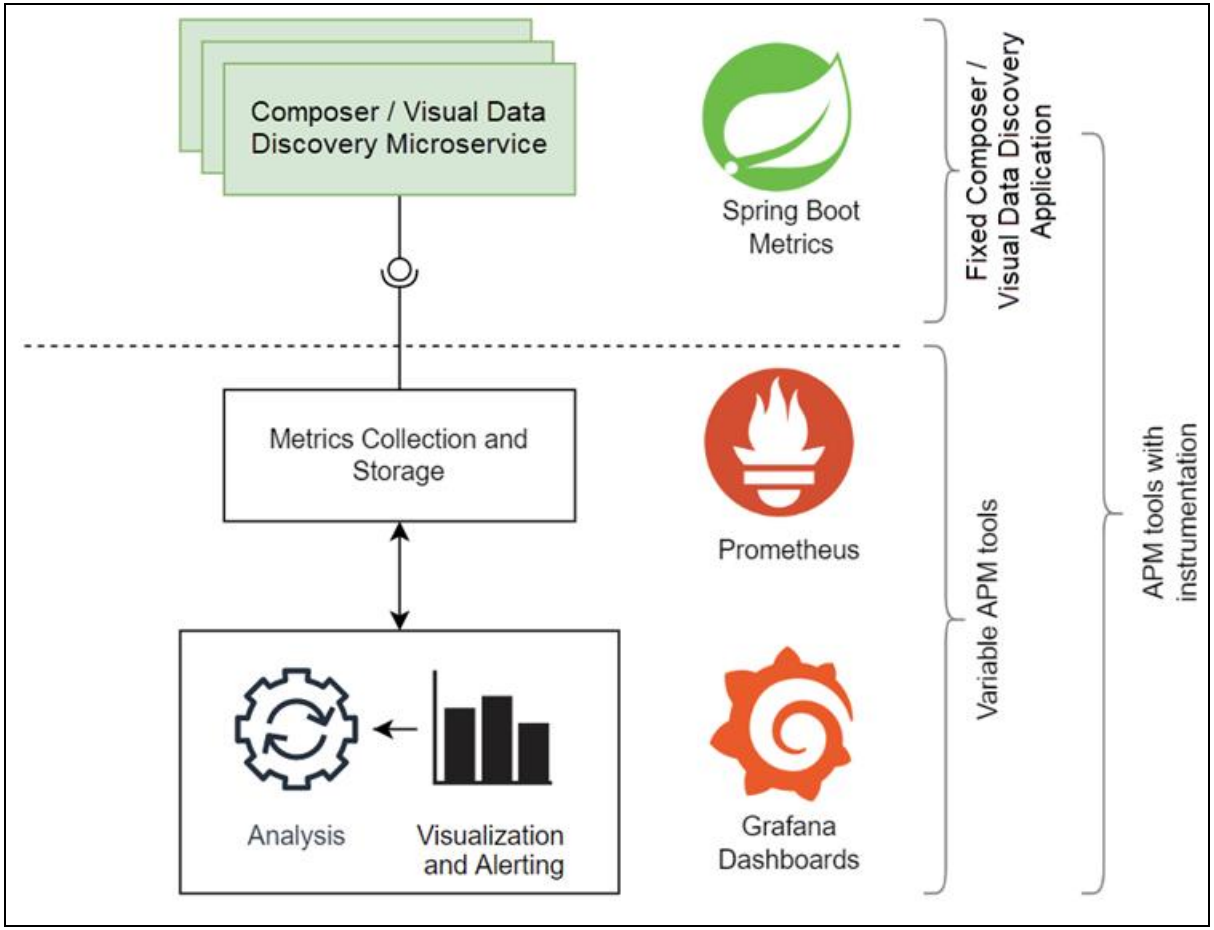
Composer system metrics are published by all Composer microservices. While these measurements provide some visibility of the operating state of the services, the metric data becomes more useful once it can be stored and viewed. In order to provide Composer deployments with observability into the health and performance of the running system, you can use a monitoring solution. This topic describes an example monitoring solution that we make available for download with each release of Composer.

Monitoring Solution Components

The monitoring solution consists of several components that are built up as layers.

- Generic operating and Composer specific metrics form the fixed foundation layer and are published in various formats.
- Subsequent layers provide storage and visualization capabilities to support further analysis.

Note: Subsequent components added to the metrics foundation can be implemented as provided in the monitoring solution package, or you can substitute them with preferred components that were already deployed.



The example solution is built using the following industry-standard, open-source tools:

1. Prometheus service collects and stores metrics in a time series database.
2. Grafana supports further analysis by providing visualization of stored metrics on pre-built application-specific dashboards.

We provide these components in a tarball containing docker-compose files. We include the configuration files required for monitoring Composer services, along with ready to use dashboards.

Downloading and Installing the Solution Package

1. Contact your ComposerCustomer Support representative to download the Solution Package: Customer Support.
2. When you get the tarball, extract the tarball to the directory you create.

```
mkdir composer-monitoring
tar -xvf composer-monitoring-X.Y.Z.tar.gz -C composer-monitoring
```

3. Open the READ.me file and follow the directions included.

Authorize Composer Access

Your users can access Composer, after you've defined their user accounts, added users to groups, and optionally, added users to tenants to grant access to content creation, content management, content access, and use.

Composer supports several approaches to authenticating users, including SAML and LDAP. Choose the best approach given your existing constraints and objectives. A complete list of authentication tools supported by Composer is provided in Supported Authentication Tools.

Note: SAML and LDAP groups that are automatically created in Composer must be manually assigned group data source access and privileges.

Once authenticated, users have authorization to perform Composer functions and access Composer resources as defined by their group membership.

Use the following features to define and provide product access and authorization in Composer.

- Composer tenants: Use to separate Composer product resources as necessary. Assign users to multiple tenants to allow access to each tenant's resources. You can further set up different groups, data connections, data sources, dashboards, and visuals for each tenant. See Manage Composer Tenants.
- User accounts: Define access for individual users in Composer. Assign users to one or more groups to give them access to Composer data sources and product features. Users can belong to multiple groups in multiple tenants.
- Groups: Use to assign privileges to groups of users. Groups are most useful when a number of Composer users require the same access restrictions. Users can be assigned to multiple groups. See Manage User Groups.

Manage Composer Tenants

As a member of the Supervisors group or a system admin user, you can manage Composer tenants.

Use tenants to separate Composer product resources as necessary. Assign users to multiple tenants to allow access to each tenant's resources. You can further set up different groups, data connections, data sources, dashboards, and visuals for each tenant.

Note: The default **supervisor** user is no longer installed; add users to the **Supervisors** group instead.

See the following topics:

- About the Supplied Composer Tenant
- List and Review Tenants
- Add and Remove Tenants
- Modify Tenants
- Modify Tenant Administrators
- Enable or Disable Tenants
- Switch Tenants

About the Supplied Composer Tenant

One unique Composer tenant is created during the installation process: *Visual Data Discovery*.

- The *Visual Data Discovery* tenant is a dedicated and permanent tenant that cannot be deleted. The supplied admin user and other system admin users can maintain other tenants and perform system administrator-level functions.
- Members of the Supervisors group have full access to Composer Supervisors group actions, permissions, and supervisory functions that can be performed in the *Visual Data Discovery* tenant.
- Other users can be assigned as admins of any tenant. Simply add them to the **Administrators** group for the desired tenants.

Important: When you install v23.4 of Composer in your environment, a default tenant , *Visual Data Discovery*, is created to manage users and content within your instance. If you're upgrading to v23.4, your existing *company* account remains unchanged. The *superaccount* work area is renamed **Visual Data Discovery**.

List and Review Tenants

List Tenants

1. Log in as the supplied admin user, a system administrator, or a member of the Supervisors group.
2. Select the UI menu () then select **Multi-Tenancy**.

   The Multi-Tenancy work area appears. It lists the defined tenants in your environment.
3. When you have reviewed the list of tenants in your environment, select **Add Tenant** to add a new tenant, or navigate away from this work area by selecting an option from the menu.

Review Tenants

1. Log in as the supplied admin user, a system administrator, or a member of the Supervisors group.
2. Select the UI menu () then select **Multi-Tenancy**.

   The Multi-Tenancy work area appears. It lists the defined tenants in your environment.
3. To review a tenant, select its name in the list.

   The Edit Tenant work area appears. If you want to review a different tenant, select the tenant name and pick a different tenant to edit in the dialog.

   If you want to see if a user is assigned to a tenant, see List and Review Users.
4. When you have reviewed the tenants in your environment, navigate away from this work area by selecting an option from the menu.

Add and Remove Tenants

If you want to use tenants to manage access to resources and data, add them to your environment quickly and easily.

Create a New Tenant

1. Log in as the supplied admin user, a system administrator, or a member of the Supervisors group.
2. Select **Tenants** on the UI menu. The Multi-Tenancy work area appears. This lists the defined tenants in your environment, including the default tenant, *Visual Data Discovery*.
3. To add a new tenant, select **Add Tenant** .

   The Create New Tenant work area opens.
4. Enter a name for the new tenant in the **Tenant name** box. The name must be at least four characters long.
5. Every tenant must have at least one administrator. Select one of the following options:

i. Select **Assign Existing User As Admin** to select an existing user as the administrator for the account. After selecting this option, select **Select Users**, then select a user from the list that opens. After you select one or more users, select **Apply**.

ii. Select **Create A New Admin User** to create a new user account to use as the administrator for the account. Supply the user name and password for the new user in the **User name**, **Password**, and **Confirm Password** boxes. The new user is assigned to the new account as a tenant administrator assigned to that tenant's Administrator's group.

6. Select **Create Tenant**.

   The new tenant is created and is automatically enabled. The administrator you assigned to is included as a tenant user (see List and Review Users). Only the Administrators group and no other user groups are part of the tenant until administrator users create user groups. See Manage User Groups. To disable the tenant, see Enable or Disable Tenants.

Remove Tenants

1. Log in as an admin user (System Administrator) or a member of the Supervisors group.
2. Select the UI menu then select **Multi-Tenancy**.

The Manage Tenants page appears. It lists the tenants that have been defined.

3. In the list of tenants, locate the tenant you want to delete and select the  in the **Delete** column.

   A warning dialog appears that prompts you to confirm that you want to delete the tenant.

4. Select **Delete** on the warning dialog to remove the tenant. All users, data source configurations, data source connections, and custom dashboards that are not associated with other tenants are removed.

Modify Tenants

Modify an Existing Tenant

1. Log in as the supplied admin user, a system administrator, or a member of the Supervisors group.
2. Select the UI menu () then select **Multi-Tenancy**.

   The Multi-Tenancy work area appears. It lists the defined tenants in your environment.
3. In the list of tenants, select the name of the tenant you want to modify.

   The Edit Tenant work area opens. It contains a single tab called **General**.

   You can switch to a different tenant on the Edit Tenant page. Simply select the current tenant name in the title to select a tenant name to switch to.
4. General information about the tenant is shown on the **General** tab. You change the tenant name in the **Tenant Name** box.

   You can also disable or enable a tenant here. See Enable or Disable Tenants.
5. After all changes have been made, select **Save** to save your changes.

Modify Tenant Administrators

When you create a tenant, you must assign an existing user as an administrator for the tenant or add a new user as an administrator of the tenant. The user will be automatically added to the Administrators group for that tenant.

Thereafter, you can only modify the administrators of the tenant when you are logged in as an administrator of the tenant.

Add or Remove a Tenant Administrator

1. Log in as an administrator for the tenant.

   If the system administrator user name you logged in with is also associated with other Composer tenants, verify that the correct tenant is selected. See Switch Tenants.

2. Select the **Users and Groups** option from the menu. The Users and Groups work area appears. It contains two options: **Users** and **Groups**.

3. Select **Users** to see a list of all the users who are assigned to this tenant.

4. Select the name of the user you want to add or remove as an administrator of the tenant from the user list. User details open to the right of the list.

5. Scroll to the **Groups** section of the **Info** tab in the selected user.

6. To make the user an administrator of the tenant, add the user to the **Administrators** group. Select **Add Groups** to bring up the Add Group(s) dialog.

   Select (check) the **Administrators** group in the dialog to add the user to this group and select **Apply**.

   To remove the user as an administrator of the tenant, clear (uncheck) the **Administrators** group in the dialog and select **Apply**.

7. Select **Save** to save the user definition.

Enable or Disable Tenants

Your tenants can be enabled and disabled. When you create tenant for the first time, it is automatically enabled.

Disable a Tenant

Note: The default tenant , *Visual Data Discovery,* can't be disabled.

Disable a tenant

1. Log in as the supplied admin user, a system administrator, or a member of the Supervisors group.
2. Select the UI menu ( ) then select **Multi-Tenancy**.

   The Multi-Tenancy work area appears. It lists the defined tenants in your environment.
3. Select the name of the tenant you want to disable from the list.

   The Edit Tenant work area appears. If you want to edit a different tenant, select the tenant name and pick a different tenant to edit in the dialog.
4. Select (check) the **Disable Tenant** checkbox for the chosen tenant.
5. Select **Save** to save your changes to the tenant.

After a tenant is disabled, users assigned to it can no longer switch to that tenant. If this is the only tenant a user is assigned to, they are shown a message to contact their system administrator the next time they log into Composer.

Enable a Tenant

Enable a tenant

1. Log in as the supplied admin user, a system administrator, or a member of the Supervisors group.
2. Select the UI menu ( ) then select **Multi-Tenancy**.

   The Multi-Tenancy work area appears. It lists the defined tenants in your environment.
3. Select the name of the tenant you want to enable from the list.

The Edit Tenant work area appears. If you want to edit a different tenant, select the tenant name and pick a different tenant to edit in the dialog.

4. Clear (uncheck) the **Disable Tenant** checkbox for the chosen tenant.
5. Select **Save** to save your changes to the tenant.

After a tenant is enabled, users that assigned to it can now switch to it while they are logged into other tenants to which they are assigned.

Manage User Groups

Use groups to assign privileges to groups of users. Groups are most useful when a number of users require the same access restrictions. Users can be assigned to multiple groups.

Note: SAML and LDAP groups that are automatically created in Composer must be manually assigned privileges.

Composer system administrator, tenant administrator, or a user who has been assigned to a group with group management privileges can manage groups. They can:

- Add, edit, or remove groups
- Assign and remove users in a group
- Authorize users in the group to perform specific functions

If your user account is not assigned the **Administer Groups** privilege or is not an administrator, you cannot assign groups to a user. In addition, only administrators can assign users to the **Administrators** group.

For more info about managing users, see Manage Users.

See the following topics:

- About Supplied Groups
- List and Review User Groups
- Add User Groups
- Modify User Groups
- Delete User Groups
- Add and Remove Members of a Group
- Group Privilege Reference

About Supplied Groups

Several default groups are supplied with Composer. Add users to these groups to allow them to perform specific tasks related to the tenant(s) they may belong to in your environment . You can't delete, rename, or edit the privileges of these default groups. Add users as needed to each group, or add more groups to accommodate your organization's needs.

- **Administrators** - Group members include the default admin user, who is a system administrator assigned to the Visual Data Discovery tenant (formerly the *superaccount*). If your environment includes tenants, each tenant includes an administrators group: all tenant admins belong to that group.

  Note: Composer v23.4 and later does not include a supervisor user. Add users to the Supervisors group.

- **Supervisors** - Add users as group members to allow them to perform specific functions without giving them access to reserved administrator tasks. Only the supplied admin user or other system administrator can add users to this group.

- **Content Distributors** - Add users as group members to allow them to create, maintain, and distribute content to any tenant.

  Important: If you use the Content Distributors group, add them to another user group you define to give them access to the features you designate beyond Content Distributors.

Important: The default tenant installed with Composer is the **Visual Data Discovery** tenant. If you do not add multiple tenants, all users belong to the **Visual Data Discovery** tenant by default. If your environment includes multiple tenants, users can be members of different groups in each tenant depending on your organization's needs.

Administrators Group (System Admins)

Administrators group members include the default admin user, who is a system administrator associated with the *Visual Data Discovery* tenant (formerly the *superaccount*).

Assign users to the Administrators group to give them administrative privileges to perform actions such as:

- Create and manage users and groups.
- Manage connectors.

- Create and manage custom charts.
- Access actions.

Note: The Administrators group is an integral part of Composer management. Users in the group can be system administrators, and tenant admins for one or more tenants.

Note: Management of the supplied **Administrators** group can only be performed by a member of that group or by a user in a group with *all* the following privileges: **Administer Users**, **Administer Groups**, and **Administer Dashboards**.

See Add Users, About Supplied Groups, Group Privilege Reference, and The Composer UI MenuSymphony Main Menu.

Administrators Group (Tenant Admins)

Administrators group members in tenants can:

- Create and manage users, groups, and content in their tenants.
- Define custom charts in their tenants.
- Perform Console and Actions related tasks.
- Composer users can be added as system admins and tenant admins in your environment.

Supervisors Group

The Supervisors group is designed to give you a group of users who can perform specific functions without giving them access to all tasks members of the Administrators group can perform in.

Assign users to the Supervisors group to allow those users to:

- Manage tenants, including creating, removing, enabling, and disabling tenants. Except when initially creating a tenant, supervisors cannot change or assign administrators to the tenant.
- Manage the look and feel of your software environment.

- Manage product licenses.
- Manage connectors.
- Enable security privileges, and more.

If you'd like a user to be a full system administrator, add them to this group and the Content Distributors group as well.

Content Distributors Group

If you'd like a user to be a full system administrator, add them to this group and the Supervisors group as well.

The Content Distributors group is part of the Visual Data Discovery tenant. Members can create, maintain, and distribute content to all tenants in your environment. Members of the Content Distributors group can:

- Import and export sources directly, or as part of importing and exporting dashboards.
- Import and export local and visual gallery visuals when importing and exporting dashboards.
- Import and export visual gallery visuals.
- Import and export dashboards.
- Import and export connections.

List and Review User Groups

You can list and review groups in a tenant when you are logged in as a system administrator or as a user who has been assigned to a group with group management privileges.

1. Log in as an administrator or a user who has been assigned to a group with group management privileges.

   If the user name you log in with is also associated with other tenants, verify that the correct tenant is selected. See Switch Tenants.

2. Select **Users and Groups** on the UI menu (). The Users and Groups work area appears. It consists of two sections: **Users** and **Groups**.

3. Select **Groups** to see a list of all the groups defined for this tenant.

   In the list of groups, locate the name of the group view. A work area for this group opens with three tabs: **General**,**Members**, and **Privileges**.

Add User Groups

Add Groups

System administrators and users who are assigned to a group with group management privileges can add group definitions to a tenant.

1. Log in as a system administrator or a user who has been assigned to a group with group management privileges.

   If the user name you log in with is also associated with other tenants, verify that the correct tenant is selected. See Switch Tenants.

2. Select **Users and Groups** on the UI menu (). The Users and Groups work area appears. It consists of two sections: **Users** and **Groups**.

3. Select **Groups** to see a list of all the groups defined for this tenant.

4. Select **New Group** to open a New Group work area, with three tabs: **General**,**Members**, and **Privileges**.

   Note: Add a group name on the **General** tab, then save the new group to access the other tabs.

5. Specify a group name on the **General** tab in the **Group Name** box. Optionally provide a description of the group in the **Description** box.

6. Select **Save** to save the new group. The group is now defined, but has no members and only the default assigned privileges.

7. Select the **Members** tab and assign users to the group. See Add and Remove Members of a Group for more information.

8. Select the **Privileges** tab and select privileges for the group. Any you add here grant permissions for all members of the group to perform specific Composer. See Group Privilege Reference.

9. When you're done adding members and setting privileges, select **Save** to save the group.

Modify User Groups

System administrators and users who are assigned to a group with group management privileges can modify groups in a tenant.

> **Note:** Management of the supplied **Administrators** group can only be performed by a member of that group or by a user in a group with *all* the following privileges: **Administer Users**, **Administer Groups**, and **Administer Dashboards**.

Modify a Group

1. Log in as a administrator or a user who has been assigned to a group with group management privileges.

   If the user name you log in with is also associated with other tenants, verify that the correct tenant is selected. See Switch Tenants.

2. Select **Users and Groups** on the UI menu (). The Users and Groups work area appears. It consists of two sections: **Users** and **Groups**.

3. Select **Groups** to see a list of all the groups defined for this tenant.

4. In the list of groups, locate the name of the group you want to modify. The group editor work area opens.

5. Select the **General** tab to change the group name in the **Group Name** box. Optionally update the description of the group in the **Description** box.

6. Select the **Members** tab and assign and remove users in the group. See Add and Remove Members of a Group for more information.

7. Select the **Privileges** tab and update the privileges for the group. Privileges allow the administrator to grant permission to perform specific functions to all members of a group. See Group Privilege Reference for more information.

8. After members and privileges have been updated for the group, select **Save** to save the group.

Add and Remove Members of a Group

You can add, remove, or delete users from a group when you are logged in as a system administrator or as a user who has been assigned to a group with group management privileges.

Add or Remove Members

Note: Management of the supplied **Administrators** group can only be performed by a member of that group or by a user in a group with *all* the following privileges: **Administer Users**, **Administer Groups**, and **Administer Dashboards**.

1. Log in as an administrator or a user who has been assigned to a group with group management privileges.

   If the user name you log in with is also associated with other tenants, verify you are working in the correct tenant. See Switch Tenants.

2. Select **Users and Groups** on the menu (). The Users and Groups work area appears. It consists of two sections: **Users** and **Groups**.

3. Select **Groups** to see a list of all the groups you can edit.

4. Select the group to which you want to add or remove members. The group editor work area opens.

5. Select the **Members** tab.

6. Select **Add Members**. An Add Member(s) work area appears.

7. Select (check) the names of the users you want to add to the group. To remove members, clear (uncheck) the checkboxes for the user names you want to remove from the group.

   If all users should be added or removed in the group, select the **Select All** option.

   You can sort the user list by name in ascending or descending order to help you locate the user names you need.

   Use the search bar to easily locate a specific user in longer lists.

8. After making your changes, select **Apply**. The selected user(s) are added or removed in the editor, but the group must still be saved.

   Note: You can remove users from the group on this screen by selecting the remove ( )icon next to a user name, and select **Delete** on the

resulting confirmation dialog.

9. Select **Save** to save the group and the membership changes. The selected user(s) are added or removed to the group. A save confirmation message displays.

Delete User Groups

Delete Groups

System administrators and users who are assigned to a group with group management privileges can delete groups in a tenant.

1. Log in as a administrator or a user who has been assigned to a group with group management privileges.

   If the user name you log in with is also associated with other tenants, verify that the correct tenant is selected. See Switch Tenants.

2. Select **Users and Groups** on the UI menu (). The Users and Groups work area appears. It consists of two sections: **Users** and **Groups**.

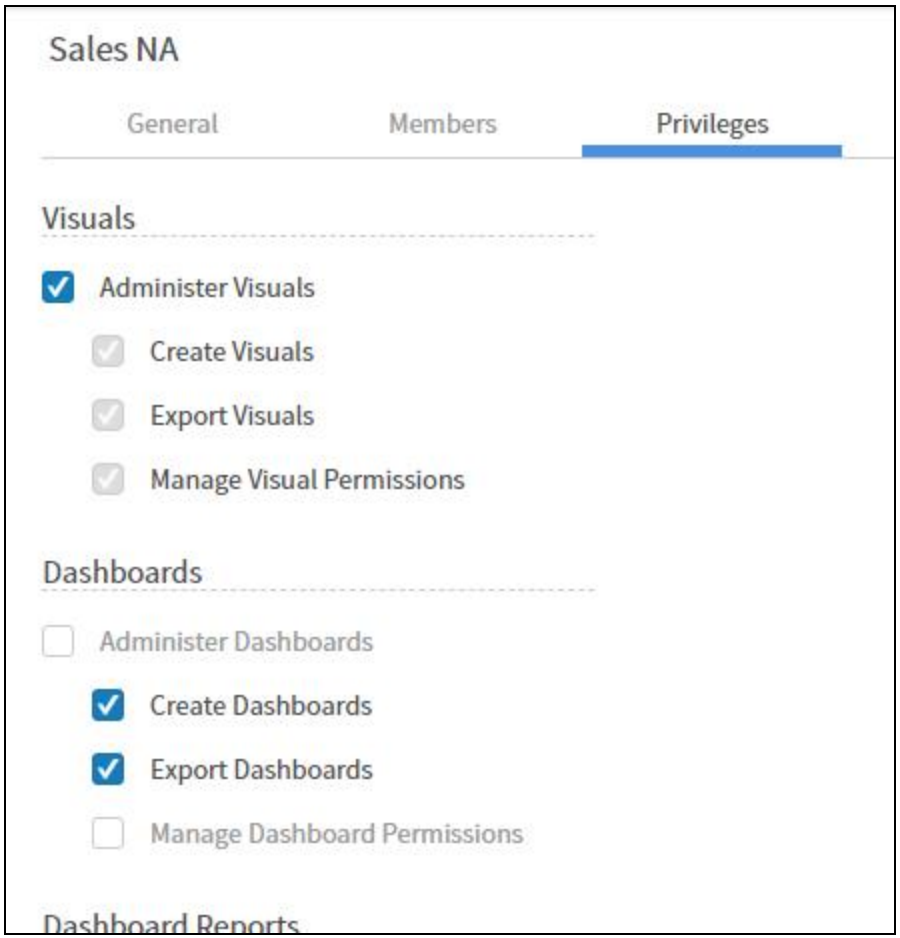
3. Select **Groups** to see a list of all the groups defined for this tenant.

4. In the list of groups, locate the name of the group you want to delete and select its associated remove () icon. A warning dialog appears that prompts you to confirm that you want to delete the group.

5. Select **Delete** on the warning dialog to remove the group.

Group Privilege Reference

Group privileges are specified on the **Privileges** tab for a group. Privileges allow a member of the Administrators group to grant permission to perform specific functions to all members of a group.



UI Privilege Name	Assign this privilege to allow group members to...	Enabled by default in these groups:
Administer Visuals ROLE_ADMINISTER_VISUALS	Create, read, update, and delete visuals from dashboards or from the Visual Gallery in the account.	■ Administrators group

UI Privilege Name	Assign this privilege to allow group members to...	Enabled by default in these groups:
	When the **Administer Visuals Privilege** is granted, the privileges **Create Visuals**, **Export Visuals**, and **Managed Visual Permissions** are automatically granted. In addition, users with the **Administer Visuals** privilege are automatically granted read, write, and delete permissions to all visuals in the account. However, if they do not also have Data Access permission for the data source associated with a visual, they cannot see any data on the visual.	
Create Visuals ROLE_CREATE_VISUALS	Create visuals in Composer. Users must also have Read permission for the data source selected for the visual. When the **Administer Visuals** privilege is granted, this privilege is also granted.	■ Administrators group
Export Visuals ROLE_EXPORT_VISUALS	Import and export visuals in Composer. Users must also have Read permission for the data source selected for the visual. When the **Administer Visuals** privilege is granted, this privilege is also granted.	■ Administrators group
Manage Visual Permissions ROLE_PERMISSION_VISUALS	Assign permissions to a visual. When the **Administer Visuals** privilege is granted, this privilege is also granted. If this privilege is *not* granted, the  icon and the Permissions column in the Visual Gallery do not appear in the UI. See About Visual Permissions.	■ Administrators group
Administer Dashboards ROLE_ADMINISTER_DASHBOARDS	Add, modify, or remove dashboards in the account, including dashboards created by other users. When this privilege is granted, the **Create Dashboards**, **Export Dashboards**, and **Manage Dashboard Permissions** privileges are automatically granted.	■ Administrators group
Create Dashboards ROLE_CREATE_DASHBOARDS	Create dashboards. If this privilege is not granted, the **Add Dashboard** button is not available in the dashboard library. In addition, if this privilege is not granted, you cannot import a dashboard or make of copy a dashboard using the Save As dialog. When the **Administer Dashboards** privilege is granted, this privilege is	■ All groups, except Supervisors group

UI Privilege Name	Assign this privilege to allow group members to...	Enabled by default in these groups:
	automatically granted.	
Export Dashboards ROLE_EXPORT_DASHBOARDS	Export dashboard configuration JSON files. If this privilege is not granted, users in the group can still export dashboards as screenshots (PNG) or PDF files, but they cannot export the dashboard configuration. When the **Administer Dashboards** privilege is granted, this privilege is automatically granted.	■ Administrators group ■ Content Distributors group
Manage Dashboard Permissions ROLE_PERMISSION_DASHBOARDS	Assign permissions to a dashboard. If your user has the **Administer Dashboards** privilege, they automatically have this privilege as well. If this privilege is not granted, the permissions () icon and the Permissions column on the Library page do not appear in the UI. See About Dashboard Permissions.	■ Administrators group
Administer Scheduled Reports ROLE_ADMINISTER_DASHBOARD_REPORTS	Create, edit, and delete all scheduled dashboard reports. Grant Read access for dashboards to users who receive reports.	■ Administrators group
Create Scheduled Reports ROLE_CREATE_DASHBOARD_REPORTS	Create, edit, and delete only your own scheduled dashboard reports.	■ Administrators group
Administer Tags ROLE_ADMINISTER_TAGS	Create, assign, remove, and delete all content tags.	■ Administrators group
Create Tags ROLE_CREATE_TAGS	Create, assign, and remove tags. Delete your own content tags.	■ Administrators group
Administer Sources ROLE_ADMINISTER_SOURCES	Create, import, export, modify, review, and remove data source configurations in the account. When this privilege is granted, the **Create New Data Sources**, **Manage Source Permissions** and **Edit Calculations** privilege are also granted. In addition, users with the **Edit Calculations** privilege are automatically granted	■ Administrators group

UI Privilege Name	Assign this privilege to allow group members to...	Enabled by default in these groups:
	read, write, and delete permissions to all sources in the account.	
Create New Data Sources ROLE_CREATE_SOURCES	Create new data source configurations. When the **Administer Sources** privilege is granted, this privilege is also granted.	■ Administrators group
Manage Source Permissions ROLE_PERMISSION_SOURCES	Assign permissions to a data source configuration and manage data source row and column security filters. If your user has the **Administer Sources** privilege, they automatically have this privilege as well. If this privilege is *not* granted, the  icon and the Permissions column on the Sources page do not appear in the UI. See About Source Permissions.	■ Administrators group
Edit Calculations ROLE_EDIT_FORMULAS	Add or edit custom metrics and derived fields. Users with Read permission for a source and **Edit Calculations** can create and edit custom metrics and derived fields for the source. Users with Write permission for a source can create and edit custom metrics and derived fields for the source.	■ All groups, except Supervisors group
Administer Alerts ROLE_ADMINISTER_ALERTS	Add, modify, or remove alert definitions in the account, including alerts created by other users. When this privilege is granted, the **Create Alerts** privilege is automatically granted.	■ Administrators group
Create Alerts ROLE_CREATE_ALERTS	Create alert definitions in Composer. When the **Administer Alerts** privilege is granted, this privilege is also granted.	■ Administrators group
Manage Connections ROLE_MANAGE_CONNECTIONS	Add, modify, or remove the data store connection definitions used by connectors and the query engine to connect to your data stores.	■ Administrators group
Manage File Uploads ROLE_MANAGE_UPLOADS	Remove unused files uploaded for a data source if they are not used by any other sources. When the **Manage Connections** privilege is granted, this privilege is also granted.	■ Administrators group
Manage Action Templates	Add, modify, or remove the action templates required to integrate Composer	■ Administrators group

UI Privilege Name	Assign this privilege to allow group members to...	Enabled by default in these groups:
ROLE_MANAGE_ACTION_TEMPLATES	visual and dashboard data into your third-party applications. Action templates define your third-party application to Composer.	
Invoke Actions ROLE_INVOKE_ACTIONS	Invoke an action template from a visual.	■ Administrators group
Administer Themes ROLE_ADMINISTER_THEMES	Create, read, update, delete, list, activate, and otherwise manage themes for the UI.	■ Administrators group
Administer Users ROLE_ADMINISTER_USERS	Administer other Composer user definitions. When this privilege is granted, group users can: ■ Add, disable, and remove user definitions ■ Reset user passwords ■ Define user custom attributes and regional settings This privilege does not allow group members to update groups or the groups to which a user is assigned. If your user ID is not assigned the **Administer Groups** privilege or is not an administrator, you cannot assign groups to a user. In addition, only administrators can assign users to the **Administrators** group.	■ Administrators group ■ Supervisors group
Administer Groups ROLE_ADMINISTER_GROUPS	Administer other Composer group definitions. When this privilege is granted, group users can: ■ Add or remove group definitions ■ Assign and remove users in a group definition ■ Authorize users in the group to perform specific Composer functions This privilege does not allow group members to add or otherwise maintain user definitions.	■ Administrators group

UI Privilege Name	Assign this privilege to allow group members to...	Enabled by default in these groups:
Save Filters ROLE_SAVE_FILTERS	Save (and share) filters created in visuals and dashboards. When this privilege is not granted, the **Save Filter** privilege does not appear on Filter dialogs in the UI.	■ All groups, except Supervisors group
Manage Custom Charts ROLE_MANAGE_VISUALIZATION_TYPES	When you have this privilege, you can use the CLI to create custom charts, and manage custom charts using the Composer UI.	■ Administrators group
Generate Embed Code ROLE_GENERATE_EMBED_CODE	Generate an embeddable dashboard or visual gallery snippet for a dashboard in the dashboard library or the visual gallery. See Embed Components Into Your Application.	■ Administrators group
Administer Initial Visuals ROLE_ADMINISTER_INITIAL_VISUALS	Users with this privilege have permission to update the list of available visualizations for a source. See Available Visual Types.	■ Administrators group
Administer Calendars ROLE_ADMINISTER_CALENDARS	Users with this privilege can create, update, and delete alternative fiscal calendars. See Fiscal Calendars.	■ Administrators group
ROLE_DISTRIBUTE_CONTENT	Users who belong to the Administrators group or Content Distributors group can make content available to your users and tenant users.	■ Administrators group ■ Content Distributors group

Manage Users

Composer provides the account management controls necessary to create users and manage user access of Composer. Authorization for users to use product features and functions is controlled by the groups to which the users are assigned.

Note: In Composer v23.4 and later, user management is performed by Composer system administrators, users with the **Administer Users** group privilege, and members of the Supervisors group. All such users can add and remove users, assign users to tenants when creating the user, and change or force a change of a user's password. Group assignment is managed by administrators (or users with the **Administer Groups** group privilege).

User definitions can be manually created or imported via the LDAP security protocol. In addition, if using SAML single sign-on protocol, users and groups may be automatically provisioned in Composer (account level synchronization). For more information, see Supported Authentication Tools.

See the following topics:

- Supplied Users And User Groups
- List And Review Users
- Add Users
- Add and Remove Supervisors
- Modify Users
- Delete Users
- Specify General User Information
- Enable And Disable Users
- Assign And Remove Users In Tenants
- Specify Custom User Attributes
- Specify A User's Regional Settings

- Change Passwords
- Change a Supervisor Password
- Import Users

Supplied Users and User Groups

Composer supplies one user: **admin**. The admin is the default system administrator, who can define other users and tenants, enable security features, and define user groups and privileges. To add more system administrators, create or add users to the Administrators group, Supervisors group, and Content Distributors group. When they belong to all three groups, they can perform all of the same functions as the default system admin.

Note: If you are installing or upgrading to v23.4 or later, the default admin user (system administrator) can perform all functions the former supervisor user previously performed, and provide authentication for embedded use cases.

After you've install and deployed Composer in your operating environment, access the application as the admin user from a web browser (see System Requirements for details).

admin User

The supplied **admin** user is defined as a system administrator and is a member of the Administrators group, the Supervisors group, and the Content Distributors group for the supplied tenant. The **admin** user cannot be deleted.

The first time you log into the UI as the **admin** user, you are prompted to change the **admin** user password. Thereafter, you can change the password for the **admin** user only if you are logged into the UI as an administrator. See Change Passwords.

Other users can be defined as administrators or can be assigned to groups with some administrator privileges. See Add Users, About Supplied Groups, and Group Privilege Reference.

Administrators Group

Administrators group members include the default admin user, who is a system administrator associated with the *Visual Data Discovery* tenant (formerly the *superaccount*).

Assign users to the Administrators group to give them administrative privileges to perform actions such as:

- Create and manage users and groups.
- Manage connectors.
- Create and manage custom charts.
- Access actions.

Note: The Administrators group is an integral part of Composer management. Users in the group can be system administrators, and tenant admins for one or more tenants.

Note: Management of the supplied **Administrators** group can only be performed by a member of that group or by a user in a group with *all* the following privileges: **Administer Users**, **Administer Groups**, and **Administer Dashboards**.

See Add Users, About Supplied Groups, Group Privilege Reference, and The Composer UI MenuSymphony Main Menu.

Administrators Group (Tenant Admins)

Administrators group members in tenants can:

- Create and manage users, groups, and content in their tenants.
- Define custom charts in their tenants.
- Perform Console and Actions related tasks.
- Composer users can be added as system admins and tenant admins in your environment.

Supervisors Group

The Supervisors group is designed to give you a group of users who can perform specific functions without giving them access to all tasks members of the Administrators group can perform in.

Assign users to the Supervisors group to allow those users to:

- Manage tenants, including creating, removing, enabling, and disabling tenants. Except when initially creating a tenant, supervisors cannot change or assign administrators to the tenant.
- Manage the look and feel of your software environment.
- Manage product licenses.

- Manage connectors.
- Enable security privileges, and more.

If you'd like a user to be a full system admin, add them to this group and the Content Distributors group as well. See About Supplied Groups.

Content Distributors Group

The Content Distributors group is part of the Visual Data Discovery tenant. Members can create, maintain, and distribute content and objects such as sources and connections to all tenants in your environment.

If you'd like a user to be a full system admin, add them to this group and the Supervisors group as well. See About Supplied Groups.

See About Supplied Groups.

List and Review Users

A system admin user and users who are assigned to a group with user management privileges can review users in Composer and tenants.

List and review all users in your environment (Supervisors Group Members)

1. Log in as a member of the Supervisors group, and select **System Users** from the menu. The Manage Users work area opens, listing all users in your environment, from all tenants.
2. Select a user name on the left side of the page. A work area opens that includes three tabs of user details you can edit as needed.

   You can:

i. Modify the user information on the **Info** tab. See Specify General User Information.

ii. Review and assign the user to specific Composer tenants and select a primary tenant for this user on the **Tenants(s)** tab. See Assign And Remove Users In Tenants.

iii. Specify regional settings for the user on the **Regional Settings** tab. See Specify A User's Regional Settings.

3. Select any of the tabs in the work area to review the user settings.

List and Review all Users in Your Environment (System Administrator or a User with User Management Privileges)

1. Log in as a user who has been assigned to a group with user management privileges.

   If the user name you log in with is also associated with other Composer tenants, verify you are using the *Visual Data Discovery* tenant to see all users, or a specific tenant to see only the users in that tenant. See Switch Tenants.

2. Select **Users and Groups** on the UI menu (). A work area opens you can use to manage users. Select the **Users** tab if both the **Users** and **Groups** tabs are visible.
3. Select a user name on the left side of the page. A work area opens that includes three tabs of user details you can edit as needed.

   You can:

i. Modify the user information on the **Info** tab. See Specify General User Information.

ii. Review and assign the user to specific Composer tenant and select a primary tenant for this user on the **Tenants(s)** tab. See Assign and Remove Users in Tenants.

iii. Specify regional settings for the user on the **Regional Settings** tab. See Specify A User's Regional Settings.

4. Select any of the tabs in the work area to review the user settings.

Add Users

The system admin users or a user who has been assigned to a group with user management privileges can add users to your environment. If you use tenants to manage access to resources and data, the supplied admin user and users belonging to the Supervisor group can add users to one or more tenants during user creation as well.

When Logged In as Member of the Supervisors Group

Members of the Supervisors group can add new users to your environment, with the following caveats:

- The new user is not assigned to any groups. Only the supplied admin user or a user who has been assigned to a group with user management privileges can assign groups to users. If you use tenants to manage access to resources and data, a tenant admin or a user who has been assigned to a group with user management privileges can assign groups to users in the tenant.
- The **Require password change** switch on the **Info** tab is always *on* (set to **Yes**) to require users to change their password when they first log into Composer. Change to **No** if needed.
- If you use tenants to manage access to resources and data, new users are created in your environment with no tenant membership. Members of the Supervisors group or any system admin user can add new users to tenants as required.

Add a user (Supervisor group members)

1. Log in as a system admin user or a member of the Supervisors group. Select **System Users** to open the Manage Users work area and list all users in your environment.
2. Select **New User**. A New User work area opens with three tabs: **Info**, **Tenant(s)**, and **Regional Settings**.
3. On the **Info** tab, enter the user login name, full name, email, and password values. Adjust the **Require password change** switch (**Yes** by default) as needed. See Specify General User Information.
4. Select **Save** to save the new user. Until you save the user, you can't access the **Tenants(s)** and **Regional Settings** tabs for this user.
5. Select the user you just created from the user list. Assign the user to one or more tenants on the Tenant(s) tab, and optionally select a **Current Tenant** if they are assigned to multiple tenants. See Assign and Remove Users in Tenants.
6. On the **Regional Settings** tab, select a regional language for the user definition. See Specify A User's Regional Settings.
7. Select **Save** to save the user.

When Logged In as an Administrator or a Group User with User Management Privileges

Add a user (admin user or user with user management group privileges)

1. Log in as an administrator or a user who has been assigned to a group with user management privileges.

   If the user name you log in with is also associated with other Composer tenants, verify that the correct tenant is selected. See Switch Tenants.

2. Select **Users and Groups** on the UI menu (). The Users and Groups page appears. It consists of two sections: **Users** and **Groups**.

3. Select **Users** to see a list of all the user definitions that have been defined for the account.

4. Select **New User**. A New User work area opens with three tabs: **Info**, **Custom Attributes**, and **Regional Settings**.

5. On the **Info** tab, enter the user login name, full name, email, and password values.

   Optionally, adjust the **Require password change** switch (**Yes** by default) as needed. See Specify General User Information.

   Optionally, add the user to available groups.

6. Select **Save** to save the new user. Until you save the user, you can't access the **Custom Attributes** and **Regional Settings** tabs for this user.

7. On the **Custom Attributes** tab, specify custom attributes for the user. See Specify Custom User Attributes.

8. On the **Regional Settings** tab, select a regional language for the user definition. See Specify A User's Regional Settings.

9. Select **Save** to save the user.

Add and Remove Supervisors

> **Note:** The default **supervisor** user is no longer installed with Composer v23.4 and later: add users to the **Supervisors** group instead. If you upgrade to Composer from an earlier version, the Supervisor user becomes a member of the Supervisors group in the *Visual Data Discovery* (formerly *superaccount*) tenant.

After upgrading to Composer v23.4 and later, the supplied admin user (System Administrator) or a member of the Administrators group can add or remove users in the Supervisors group.

Add any number of users to this group to allow them to perform specific functions without giving them access to all tasks members of the Administrators group can perform in the *Visual Data Discovery* tenant.

Add a User to the Supervisors Group

1. Log in as the supplied admin user or a member of the Administrators group in *Visual Data Discovery*.
2. Select **Users and Groups** on the UI menu (). A work area opens you can use to manage users. Select the **Users** tab.
3. Select the user from the list of users you want to add to the Supervisors group. The account details for that user appear on the right side of the page.
4. On the **Info** tab, select **Add Group(s)**. The Select Account(s) dialog appears.

   > **Note:** If the user is already a member of the Supervisors group, this option is not shown.

5. Select (check) the group or groups you want to add this user to. When you add a user to the Supervisors group, they have full access to the Composer supervisor UI and its supervisory functions. See About The Supplied Composer Tenant.
6. Select **Apply** when finished.
7. Select **Save** to save the user.

The user is now a member of the Supervisors group.

Remove a User from the Supervisors Group

You can remove a user from the Supervisors group by removing them from the Supervisors or by simply deleting their user account. See Delete Users.

1. Log in as the supplied admin user or a member of the Administrators group in *Visual Data Discovery*.
2. Select **Users and Groups** on the UI menu (). A work area opens you can use to manage users. Select the **Groups** tab if both the **Users** and **Groups** tabs are visible.
3. Select the Supervisors group from the list of groups. The details for that group appear on the right side of the page.
4. Select the **Members** tab. This tab lists all members of the Supervisors group.
5. Remove the user from the group by selecting the remove icon () next to the user's name. Confirm your deletion in the confirmation modal.
6. Select **Save** when finished. The list of users on the **Members** tab adjusts to show your changes.

Modify Users

Composer administrators, tenant administrators, or a user who has been assigned to a group with user management privileges can modify user accounts.

Modify a User (Supervisors Group Members)

1. Log in as the supplied admin user or a user in the Supervisors group. Select **System Users** to open the Manage Users work area and list all users in your environment. If you are logged in as a tenant admin, verify you're in or switch to the appropriate tenant.

2. On the left side of the Manage Users work area, select the name of the user you want to modify. The user information appears on the right side of the work area.

3. On the **Info** tab, modify the values for the user login name, full name, email, and password, as needed. See Specify General User Information and Change Passwords.

   > **Note:** You cannot change the groups to which a user is assigned. Groups can only be assigned by tenant administrators or tenant users who have been assigned to groups with user management privileges.

4. On the **Tenant(s)** tab, adjust the tenants to which the user is assigned or change the user's current tenant, as needed. See Assign and Remove Users in Tenants.

5. On the **Regional Settings** tab, change the regional language for the user as needed. See Specify A User's Regional Settings.

6. Select **Save** to save the user.

Modify a User (Administrator Group Members, Group Members with User Management Privileges)

1. Log in as an administrator or a user who has been assigned to a group with user management privileges. If you are logged in as a tenant admin, verify you're in or switch to the appropriate tenant.

   If the user name you log in with is also associated with other tenants, verify that the correct tenant is selected. See Switch Tenants.

2. Select **Users and Groups** on the UI menu (). The Users and Groups page appears. It consists of two sections: **Users** and **Groups**.

3. Select **Users** to see a list of all the users defined for the tenant.

4. On the left side of the work area, select the name of the user you want to modify. The user information appears on the right side of the work area.

5. On the **Info** tab, supply values for the user login name, full name, email, password, and assigned groups, as needed. See Specify General User Information. You can also use this tab to disable a user definition in the account. See Enable and Disable Users.

6. On the **Custom Attributes** tab, specify or change custom attributes for the user as needed. See Specify Custom User Attributes.

7. On the **Regional Settings** tab, change the regional language for the user as needed. See Specify A User's Regional Settings.

8. Select **Save** to save the user.

Delete Users

Composer administrators or a user who has been assigned to a group with user management privileges can delete users in Composer or individual Composer tenants.

Delete a User (Supervisors Group Members)

1. Log in as the supplied admin user (System Administrator) or a member of the Supervisors group. Select **System Users** to open the Manage Users work area and list all users in your environment.
2. On the left side of the Manage Users work area, select the name of the user you want to delete and select its delete () icon. A warning dialog opens.
3. Select **Delete** to remove the user.

   The user is removed from the Composer instance and from all Composer tenants to which it is assigned. See Assign and Remove Users in Tenants.

Modify a User (Administrator Group Members, Group Members with User Management Privileges)

1. Log in as an administrator or a user who has been assigned to a group with user management privileges.

   If the user name you log in with is also associated with other Composer tenants, verify that the correct tenant is selected. See Switch Tenants.
2. Select **Users and Groups** on the UI menu (). The Users and Groups page appears. It consists of two sections: **Users** and **Groups**.
3. Select **Users** to see a list of all the users defined for the tenant.
4. Select the name of the user you want to delete and select its delete () icon. A warning dialog opens.
5. Select **Delete** to remove the user.

   The user is removed from the selected Composer tenant, but will still exist in the instance and in any other Composer tenants to which it is assigned. See Assign and Remove Users in Tenants.

Specify General User Information

When you add or modify a user, you can specify the following general information:

- The login name for the user
- The user's full name
- The user's email address
- The user's password
- Groups to which the user is assigned
- Whether the user is enabled or disabled

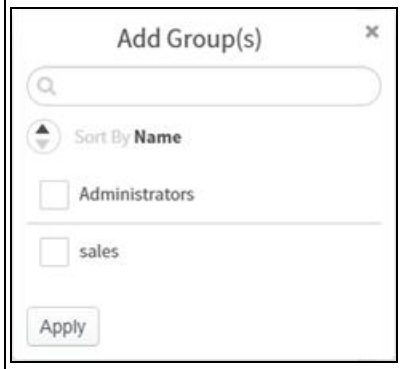
This general information is managed on the **Info** tab of the user editor. To access the **Info** tab, see Add Users or Modify Users. For information on the user settings that can be changed on other tabs, see:

- Assign and Remove Users in Tenants
- Specify Custom User Attributes
- Specify A User's Regional Settings
- Enable and Disable Users

In v23.4 of Composer and later, tasks formerly performed by the Supervisor user are now performed by the default system admin and members of the Supervisors group. Any user who belongs to the Supervisors group or to a group with user management privileges can make changes to the general information of a user's **Info** tab.

Note: The default **supervisor** user is no longer installed; add users to the **Supervisors** group instead.

The following table describes all the information you can change on the **Info** tab. The **Required?** column indicates whether the information is required in a user's definition.

Tab Field	Required?	Default	Description
Login Name	Yes	---	The login name for the user. This is also the name for the user definition. The login name must be unique in all accounts in the Composer instance. It must include at least one alphanumeric character. The login name is used to log into Composer. See Log Into the ComposerSymphony UI.
Full Name	No	---	The full name of the user. Specify up to 40 characters of the user's given name.
Email	No	---	The email address of the user. Specify up to 254 characters of the user's email.
Change Password	Yes	---	Select the arrow in this box to expand the Info tab and see the **Password** and **Confirm Password** boxes.
Password	Yes	---	The password associated with this user name. The password must contain at least nine characters and must include one lowercase, one uppercase, one numeric, and one special character. Special characters include these characters: !@#$%^&**()-_=+,.:;<>
Confirm Password	Yes	---	The password associated with this user name. Use this text box to retype the password you specified in the **Password** text box when you initially create the user definition and when you change the password for the user.
Require password change	No	Yes	Use the **Require password change** switch to indicate whether the user should be prompted to change their password when they log on for the first time (or the first time after their password was set or changed). When you create a new user, the **Require password change** switch on the **Info** tab is always on (set to **Yes**), but can be changed to **No**.
Add Groups	No	---	Select this button to assign the user to one or more groups in the account. The Add Group(s) dialog appears.  If your user ID is not assigned the **Administer Groups** privilege or is not an administrator, you cannot assign groups to a user. In addition, only administrators can assign users to the **Administrators** group.

Tab Field	Required?	Default	Description
			Use the search box at the top of the dialog to locate a group name in the list. You can also sort the group names in ascending or descending order using the **Sort By Name** arrows. After selecting (checking) one or more groups for the user definition, select **Apply** to assign the user to the groups and close the Add Group(s) dialog. In v23.4 of Composer and later, users who belong to the Supervisors group do not have this field on the Info tab. Update their group membership in the group directly. Navigate to the menu option **Users and Groups**, then select the Group tab to add or remove the user from a specific group in a specific tenant.
Disable User	No	not checked	This option appears only for administrators or users who are assigned to a group with user management privileges. Use the **Disable User** checkbox to disable the user. See Enable and Disable Users.

Enable and Disable Users

You can enable or disable a user when you add a new user or modify an existing user. New users are enabled by default.

When you disable a user, the user is not removed from Composer, but the user no longer has access to the tenant in which they were disabled. If a user belongs to more than one tenant, they can still log into , but only to the tenants in which their user account is enabled. If their user account is disabled in all tenants, they can no longer log into Composer.

Enable or disable a user on the **Info** tab of the user.

Enable and disable users in a tenant if you are logged in as an administrator for the tenant or as a user who is assigned to a group in the tenant with user management privileges.

Disable a User

1. Access the **Info** tab for a user as a member of a group with user management privileges. See Add Users or Modify Users.
2. Select (check) the **Disable User** checkbox at the bottom of the tab.
3. Select **Save** to save the user.

Enable a User

1. Access the **Info** tab for a user as a member of a group with user management privileges. See Add Users or Modify Users.
2. Clear (uncheck) the **Disable User** checkbox at the bottom of the tab.
3. Select **Save** to save the user.

Enable and Disable the Supplied Supervisor User

Note: The default **supervisor** user is no longer installed; add users to the **Supervisors** group instead.

You can enable or disable the supplied Composer supervisor user in Composer v23.3 and earlier.

When you disable the supplied supervisor user, the user is not removed from Composer, but the supervisor no longer has access to Composer.

Enabling or disabling the supplied supervisor user definition can only be done by another supervisor and occurs on the Manage Users page of the supervisor UI.

Disable the Supplied Supervisor User

1. Log into Composer as a supervisor who is not the supplied supervisor user (make sure you have selected **superaccount**).
2. On the left side of the page, select the **supervisor** user. The supervisor user information appears on the right side of the page.
3. Select (check) the **Disable User** checkbox at the bottom of the **Info** tab.
4. Select **Save** to save the supervisor definition.

Enable the Supplied Supervisor User Definition

1. Log in as a supervisor who is not the supplied supervisor user (make sure you have selected **superaccount** or **Visual Data Discovery**). The Manage Users page appears, listing all the user definitions in the instance.
2. On the left side of the page, select the **supervisor** user. The supervisor user information appears on the right side of the page.
3. Clear (uncheck) the **Disable User** checkbox at the bottom of the **Info** tab.
4. Select **Save** to save the supervisor.

Assign and Remove Users in Tenants

You can assign or remove users in Composer tenants as a system admin user or a user in the Supervisors group.

Assign a User to a Tenant

1. Log in as a system administrator or a member of the Supervisors group.
2. On the left side of the Manage Users work area, select the name of the user you want to modify. The user information appears on the right side of the work area.
3. Select the **Tenant(s)** tab. This tab lists all the tenants to which the user is assigned and the number of groups to which the user is assigned in each tenant.
4. Select **Add Tenant(s)**. The Select Tenants(s) dialog appears.
5. Select (check) the tenants to assign the user to those tenants. If you clear (uncheck) the checkbox associated with a tenant here, user is removed from the tenant.
6. Select **Apply** when finished. The list of tenants on the **Tenant(s)** tab updates to reflect your changes.
7. Optionally, select a tenant for the user to use the next time they log in from available tenants in the **Current Tenant** field. See Set the Current Tenant for a User.
8. Select **Save** to save the user.

Remove a User from a Tenant

1. Log in as a system administrator or a member of the Supervisors group.
2. On the left side of the Manage Users work area, select the name of the user you want to modify. The user information appears on the right side of the work area.
3. Select the **Tenant(s)** tab. This tab lists all the tenants to which the user is assigned and the number of groups to which the user is assigned in each tenant.
4. You can remove the user from tenants in one of two ways:

i. Select the remove () icon next to the tenant you want to remove the user from. Select **Delete** when prompted to remove the user from this tenant.

ii. Select **Add Tenant(s)**. The Select Tenants(s) dialog appears. Clear (uncheck) the checkbox associated with a tenant here, user is removed from the tenant. Select **Apply**: the list of tenants on the **Tenant(s)** tab updates to reflect your changes.

5. Optionally, select a tenant for the user to use the next time they log in from available tenants in the **Current Tenant** field. See Set the Current Tenant for a User.

6. Select **Save** to save the user.

Set the Current Tenant for a User

If a user has access to multiple Composer tenants, you can specify the tenant they should use the next time they log in. After logging in, they can switch tenants, as needed.

After a user switches tenants, the most recent Composer tenant they worked in is remembered and used the next time they log in. If a user is assigned to only one tenant, that tenant is always the current tenant.

Note: The default **supervisor** user is no longer installed; add users to the **Supervisors** group instead.

Set the Current Tenant for a User

1. Log in as a system admin user or a user in the Supervisors group. Select **System Users** to open the Manage Users work area and list all users in your environment.
2. On the left side of the Manage Users work area, select the name of the user whose tenants you want to modify. The user information appears on the right side of the work area.
3. Select the **Tenant(s)** tab. This tab lists all the tenants to which the user is assigned and the number of groups to which the user is assigned in each tenant.
4. Select a tenant for the user to use the next time they log in from available tenants in the **Current Tenant** field.
5. Select **Save** to save the user.

Specify Custom User Attributes

You can define custom attributes for a user's definition if you are logged in as a Composer system administrator, tenant administrator, or a user who is assigned to a group with user management privileges.

Use custom attributes to store values that can be used as parameters, or variables, in connection definitions, data source row security filters, and in custom SQL.

A primary use of custom attributes is for user credential pass-through. Add credentials to a user's custom attributes so users with access to a particular data source connected to your instance can use these saved credentials to maintain access privileges for that source within Composer Symphony.

Note: User attributes set for users via the UI or using LDAP user definitions are encrypted when stored in metadata. To specify the encryption modes, see Change the Encryption Mode.

Custom attributes are defined using the **Custom Attributes** tab when you edit the user.

When you reference custom attributes in connecton definitions, action URLs, attribute definitions, or a data source's custom SQL, they take the form of `${User.<custom-attribute-name>}`.

Custom attributes can also be specified dynamically in the LDAP or SAML configurations for your instance. See Use Lightweight Directory Access Protocol (LDAP) with Composer Symphony (link) and Configure Composer Symphony to Support SAML (link).

This section covers the following topics:

- Supplied Context Variables
- Add A Custom Attribute for a User
- Remove a Custom Attribute from a User
- Specify Multivalue (Array) Custom Attributes
- Specify Numeric Values in Custom Attributes
- Specify Time Values in Custom Attributes

Supplied Context Variables

The following context variables are provided with Composer.

- `${User.composerUserName}`: include to insert the name of the user who is currently logged in.
- `${User.accountId}`: include to insert the user account ID of the user who is currently logged in.
- `${User.credentials}`: include to pass the session ID or trusted access token (in embedded environments) of the user.

You do not need to create custom user attributes for the user name or account ID. Use these supplied context variables instead.

Add A Custom Attribute for a User

Add a custom attribute for a user

1. Access the Custom Attributes tab for the user. See Add Users or Modify Users.
2. Select Add Custom Attribute. A blank line is added to the Custom Attributes tab.
3. Supply values for the attribute, as described in the following table:

Tab Field	Description
Key	Specify the name of the custom attribute in the Composer displays. The name cannot include braces.
Value	Specify one or more values for the custom attribute. See below for more options.
Usage	Shows how the attribute appears in the text entry fields of the application.
Secure	Select this checkbox if you want to encrypt the custom attribute values.
Delete	Select the delete button to remove the attribute.

4. When you have specified all values, select **Save** to save the user.

Remove a Custom Attribute from a User

Remove a custom attribute from a user

1. Access the Custom Attributes tab for the user. See Add Users or Modify Users. The defined custom attributes are listed.
2. Select the delete button corresponding to the attribute you want to remove.
3. When are done removing your specific attributes, select **Save** to save the user.

Specify Multivalue (Array) Custom Attributes

For multivalue user attributes (arrays), the elements must be comma-separated and contain no redundant white spaces between elements. These multivalue custom attributes should only be used as arrays in INCLUDE and EXCLUDE filter operations. With all other filter operations, the custom attribute values are used as-is.

For example, an INCLUDE or EXCLUDE row security filter that uses a multivalue custom attribute containing the array (`["apples", "oranges", "bananas"]`) will interpret the values as three separate values, splitting the values on commas.

However, an EQUAL or NOT EQUAL row security filter using the same multivalue custom attribute would interpret the array values as a single value (`"apples,oranges,bananas"`).

Null values in multivalue custom attributes are processed as a special text marker - `[Null]`.

Specify Numeric Values in Custom Attributes

Numeric value in custom attributes are supported as integers or floating-point values represented as text.

Specify Time Values in Custom Attributes

For date-time values, the following formats are supported:

- ISO-8601 format with optional milliseconds (but no timezone): `yyyy-MM-ddTHH:mm:ss[.SSS]`
- Composer's default API time format: `yyyy-MM-dd HH:mm:ss.SSS`, milliseconds required.

Unix time stamp format (seconds/milliseconds) is not supported for date-time values.

The BETWEEN filter operator expects an array with two elements.

When you use a time-value custom attribute as a filter variable for a filter using BETWEEN, two explicit elements must be specified (for example `start_date` and `end_date`). The dates can be specified as static date-time values, dynamic time patterns, or single-value user attributes.

Specifications such as `[${User.date_array}]`, with the `date_array` attribute containing two elements are not supported.

Specify A User's Regional Settings

Set regional settings for a user to determine the language of each user and the format of numeric fields in a data source. For example, a user based in the United States sees number fields formatted for Italian currency differently than a user based in Italy. See Number Formatting for Data Sources for more information.

These settings can be changed by an administrator or a user who has been assigned to a group with user management privileges.

The user locale can be set using the **Regional Settings** tab, as well.

To access the **Regional Settings** tab, see Add Users orModify Users. For information on the user settings that can be changed on other tabs for a user, see:

- Assign and Remove Users in Tenants
- Specify General User Information
- Specify Custom User Attributes
- Enable And Disable Users

Set the User Locale for a User

1. Navigate to the **Regional Settings** tab for a user. See Add Users orModify Users.
2. Select the location for the user in the drop-down list of the **Regional Presets** box.
3. When all values have been specified, select **Save** to save the user.

Change Passwords

Note: Only users who are members of the Administrators group or the Supervisors group, or belong to a group that includes the **Administer Users** privilege can change an existing user's password or require them to change their password the next time they log in.

Change or Reset a Password

1. Log in as user who has been assigned to a group with user management privileges.
2. Select **Users and Groups** from the UI menu. A work area opens you can use to add and manage users. Select the **Users** tab if both the **Users** and **Groups** tabs are visible.
3. Select the user from the list of users whose password you want to reset. The account details for that user appear on the right side of the page.
4. On the **Info** tab, select **Change Password**.
5. Type the new password in the **Password** and **Confirm Password** boxes.

   Note: Optionally, change the **Require password change** switch from the default **No** to **Yes**. If you change this to **Yes**, the user is prompted (and forced) to change their password when they next attempt to log in.
6. Select **Save** to save your changes.
7. The next time the user logs in, they can use the new password you set. They are prompted to change this password if **Require password change** was set to **Yes**.

Change a Supervisor Password

Note: The default **supervisor** user is no longer installed with Composer v23.4 and later: add users to the **Supervisors** group instead. If you upgrade to Composer from an earlier version, the Supervisor user becomes a member of the Supervisors group in the *Visual Data Discovery* (formerly *superaccount*) tenant.

After upgrading to Composer v23.4 and later, members of the Supervisors group can change their own passwords, or force a password change at next log in. See Change Passwords. A system administrator or users who are members of a group with user management privileges can change a password or force a password change for a member of the Supervisors group.

Change a Password for Supervisor Group Members

Note: Only a system administrator or users who are members of a group with user management privileges can change a password or force a password change for a member of the Supervisors group.

Change or reset password for Supervisors group members

1. Log in as user who has been assigned to a group with user management privileges.
2. Select **Users and Groups** from the UI menu. A work area opens you can use to add and manage users. Select the **Users** tab if both the **Users** and **Groups** tabs are visible.
3. Select the user from the list of users whose password you want to reset. The account details for that user appear on the right side of the page.
4. On the **Info** tab, select **Change Password**.
5. Type the new password in the **Password** and **Confirm Password** boxes.

   Note: Optionally, change the **Require password change** switch from the default **No** to **Yes**. If you change this to **Yes**, the user is prompted (and forced) to change their password when they next attempt to log in.
6. Select **Save** to save your changes.
7. The next time the user logs in, they can use the new password you set. They are prompted to change this password if **Require password change** was set to **Yes**.

Change a Supervisor Password in Earlier Versions of Composer

Note: Only a Composer supervisor user can change or reset their own supervisor password. They cannot change the passwords for other supervisors.

To change or reset the supervisor password:

1. Log in as the supervisor user. Supervisor users are users assigned to the *superaccount*. The Manage Users page appears.
2. In the list on the left of the page, select the supervisor user name. The account details for the supervisor user appear on the right side of the page.
3. On the **Info** tab, select **Change Password**. The Info tab expands to show boxes that allow you to change the password.
4. Type the new password in the **Password** and **Confirm Password** boxes.
5. Optionally, change the **Require password change** switch. When this switch is set to **Yes**, the supervisor is prompted (and forced) to change their password on the next log in. When this switch is set to **No**, the supervisor is not prompted to change their password. The default setting for this switch is **No**.
6. Select **Save** to save the new password. As soon as you do this, the supervisor user can log into Composer using the new password.

Import Users

You can import users from Lightweight Directory Access Protocol (LDAP) or Active Directory if either protocol has been set up. Authorized users created and stored in these directories can be imported into your environment. You can import all users or select specific ones to add.

Composer can connect to an organization's Active Directory (AD) and OpenLDAP directory services using configured LDAP settings. See Use Lightweight Directory Access Protocol (LDAP).

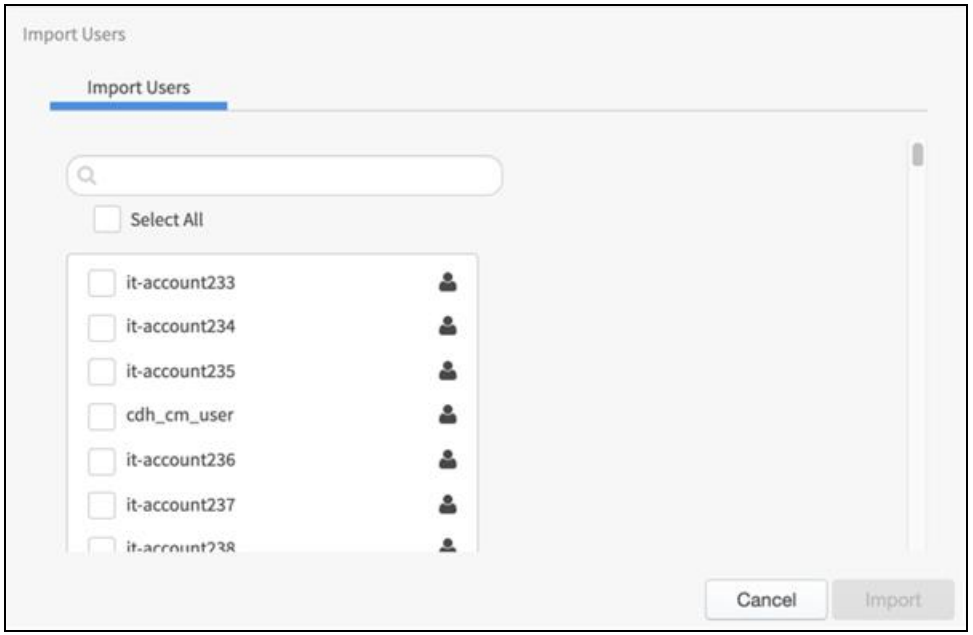
Note: Composer supports the auto-provisioning of users via the SAML single sign-on protocol. When auto-provisioning is set up, the user is automatically added when they first log in. For instructions on automatic provisioning of users and groups via SAML, see Configure Composer To Support SAML.

Note: When a user is imported from Active Directory or if user provisioning is enabled and a new Active Directory user is added, the corresponding user definition is automatically added. However, when a user is removed from Active Directory, the corresponding user is not automatically removed. Authentication does not occur for the removed user, but you will need to manually remove the user. See Delete Users.

Import user definitions into Composer from LDAP or Active Directory

1. Log into Composer as a system admin, or member of the Supervisors group and verify that LDAP or Active Directory have been set up. See Use Lightweight Directory Access Protocol (LDAP). Save the settings, then log out.
2. Log in as an administrator or a user who has been assigned to a group with user management privileges.

   If the user name you log in with is also associated with other tenants, verify that the correct tenant is selected. See Switch Tenants.
3. Select the **Users and Groups** option from the menu. The Users and Groups page appears. It consists of two sections: **Users** and **Groups**.
4. Select the **Users** section to see a list of all the user definitions that have been defined for the account.
5. Select **Import Users**. Upon successful access to your organization's secure directory, a new tab appears listing all the available users that can be imported. You have the option to either select all users or individual users.
6. On the Import Users tab, select the users to be imported.



7. Select **Import**. When the import completes, you will see the users added to the user list.
8. On the **Info** tab of the user editor, select groups for each user. See Specify General User Information.

   Do ***not*** change the user login name as this may cause conflicts when reconnecting to the LDAP or Active Directory are attempted.

   If you have enabled the auto-creation of groups, Composer creates the groups that user is assigned to if they do not already exist.
9. Select **Save** to save the user.

Supported Authentication Tools

Composer supports several approaches to authenticating users. Your organization must choose the best approach given your existing constraints and objectives.

- Composer provides basic login access to the Composer application. See Authorize Composer Access.
- X.509 client certificate authentication can be used to provide single sign-on capabilities, although it does not support auto-provisioning of user accounts. See Configure Client Certificate Authentication.
- SAML (Security Assertion Markup Language) can be used to provide single sign-on capabilities. See Configure Composer To Support SAML.
- Kerberos can be used to provide single sign-on capabilities. See Configure Kerberos Single Sign-On (SSO) Settings.
- Trusted Access can be used to allow for machine-to-machine authorization of Composer resources when embedded in your application. It allows users to log in once to the parent application and yet have their security information propagated to Composer, creating a seamless and secure user experience. See Trusted Access.

  Note: insightsoftware recommends using Trusted Access for all embed-related workflows.

- LDAP (Lightweight Directory Access Protocol) can be used to enable directory-based access to Composer. Composer can connect to an organization's Active Directory (AD) and OpenLDAP directory services using configured LDAP settings. See Use Lightweight Directory Access Protocol (LDAP).

Composer system administrators can enable or disable Composer's authentication services as required. The available services are listed on the Security Services tab.

View and edit security options

1. Log in as a system admin or a member of the Supervisors group.

   Note: The default **supervisor** user is no longer installed; add users to the **Supervisors** group instead.

2. Select **Security** from the UI menu. .

   The Security page appears. It consists of several sections: **Security Services**, **SAML Settings**, **LDAP Settings**, and **Kerberos Settings**. The Security Services tab is selected. Other tabs shown are accessible only when the corresponding service is enabled on the Security Services tab.

Settings for x.509 and Kerberos SSO authentication are handled using the `zoomdata.properties` file.

Enabling or disabling any of these security services requires a restart of the Composer service. Basically, any time you switch a security feature, the Composer service needs to be restarted before the change takes effect. The following switch status may appear for each of the authentication services: **Started**, **Stopped**, **Will start or stop on next restart**. See Enable Or Disable A Security Service.

When working with security authentication services, bear in mind that you cannot use them all at the same time. If you switch a particular security service on, others will become disabled. If you want to use a security service that is disabled, you must switch the running services off and then start the service you want.

Security services compatibility

Security Service	Can Be Used With
x.509	LDAP, Trusted Access
SAML SSO	Trusted Access
Kerberos	LDAP, Trusted Access
Trusted Access	SAML, LDAP, Kerberos, x.509
LDAP	Kerberos, x.509, Trusted Access

Enable or Disable a Security Service

The Composer supervisor can enable or disable a security authentication service.

Enable a Security Service

Enable a security authentication service

1. Log in as a system admin or a member of the Supervisors group.
2. Select **Security** from the UI menu.
3. The Security page appears. It consists of four tabs: **Security Services**, **SAML Settings**, **LDAP Settings**, and **Kerberos Settings**.
4. On the Security Services tab, turn on the appropriate switch (slide it to the right) for the authentication service you want to enable.

5. Select **Save**.

   After the settings have been saved, the status of the authentication service changes to **Will start on next restart**.
6. Close the Composer UI.
7. Access the Linux prompt and log into your Composer server (via Secure Shell or SSH).
8. Restart the Composer server service. See Restart Composer Microservices.

Wait a few minutes for the Composer service to fully restart, then open a new browser window and log back in as the supervisor. Verify that your changes took effect. After restarting Composer, the authentication service status changes to **Started**.

> **Note:** If you receive an error message stating that the connection to Composer could not be made, you may need to give it a little more time for the service to fully restart. If the problem persists, contact Technical Support.

Disable a Security Service

Disable a security authentication service

1. Log in as a system admin or a member of the Supervisors group.
2. Select **Security** from the UI menu.
3. The Security page appears. It consists of four tabs: **Security Services**, **SAML Settings**, **LDAP Settings**, and **Kerberos Settings**.
4. On the Security Services tab, turn off the appropriate switch (slide it to the left) for the authentication service you want to disable.

5. Select **Save**.

   After the settings have been saved, the status of the authentication service changes to **Will stop on next restart**. The tab that allows you to configure the settings for the service will still be available and the service will keep running until the Composer service is restarted.

6. Close the Composer UI.

7. Access the Linux prompt and log into your Composer server (via Secure Shell or SSH).

8. Restart the Composer server service. See Restart Composer Microservices.

   Wait a few minutes for the Composer service to fully restart, then open a new browser window and log back in as the supervisor. Verify that your changes took effect. After restarting Composer, the authentication service status changes to **Stopped** and access to the corresponding service tabs are disabled.

   > **Note:** If you receive an error message stating that the connection to Composer could not be made, you may need to give it a little more time for the service to fully restart. If the problem persists, contact Technical Support.

Configure Client Certificate Authentication

Composer supports X.509 client certificate authentication. However, note that auto-provisioning of user accounts is not available for client certificate authentication.

To use the X.509 authorization you need to:

- Enable the X.509 option in the Security Services section
- Configure the required properties in the `zoomdata.properties` file

Caveat

Composer does not support auto-provisioning of user accounts for client certificate authentication.

Configuration Steps

For guidance on accessing and editing a Composer property file, refer to the topic Configure Composer.

Add the following settings to your `zoomdata.properties` file:

```
server.port= 8443
server.ssl.enabled= true
server.ssl.client-auth= want
server.ssl.key-store= .../server.jks
server.ssl.key-store-password= <Your_password>
server.ssl.key-store-type= <use_either_jks_or_pkcs12>
server.ssl.trust-store= .../truststore.jks
server.ssl.trust-store-password=<Your_password>
server.ssl.trust-store-type= <use_either_jks_or pkcs12>
```

For each user, create an user account in Composer with the username set to the 'CN' in the user's certificate.

Troubleshooting

Challenges you may run into:

- User is never prompted to select a certificate:
  - Make sure you have added at least one CA to the trust-store file.
  - Verify `server.ssl.client-auth` is set to `want`.
- Selecting login brings me back to the login page:
  - Make sure the username matches the CN of the certificate being used.
  - Make sure the client certificate is signed by a CA in the trust-store.

For further troubleshooting assistance, contact Technical Support.

Configure Composer to Support SAML

This topic walks you through the steps to configure SAML settings in Composer for SAML Just-in-time (JIT) provisioning. When enabled and configured, the SAML directory can be used as a source for importing and autoprovisioning users in Composer.

1. Obtain the XML metadata file from your organization's SAML Identity Provider (IDP).
2. Create a Key File using the *Key Tool Generator* program.
3. Log into Composer as an administrator or member of the Supervisors group and configure the SAML settings in the Security tab.
4. Configure the service provider details in your IDP account that you want accessible by Composer.
5. Upload Identity Provider Metadata and add key files to Composer.
6. Configure the SAML mappings in Composer.

In addition, there are optional configurations you can set up:

- Customize The Composer Entity ID
- Configure The Auto-Redirect To The IDP
- Use The Network Time Protocol To Synchronize Time (avoids potential failure by the identity provider to authenticate SAML users)
- Implement Single Sign-On (SSO) Via SAML

Prerequisites

Obtain Identity Provider Metadata File

To enable Composer to support your organization's identity provider, first obtain the metadata content from your IDP and upload into Composer. Your Security Administrator responsible for managing IDPs can provide this file. Save this metadata file to your local hard drive for uploading into Composer. Refer to the configuration instructions provided below.

ACTION: Obtain the XML metadata file from your organization's IDP.

Generate a Key File and Configuring SAML with SSL

The key file helps you manage a keystore of cryptographic keys and trusted certificates. Generate the key file using the keytool program available from Oracle. After generating a key file, you are able to import your SSL certificate into the keystore. Private keys that are used to digitally sign SAML messages along with any SSL/TLS certificates need to be imported into this keystore.

After installing the keytool program, take the following steps:

1. Run the following command to generate your key:

```
keytool -genkey -alias <localhost> -keyalg RSA -keystore <mykeystore> .jks -keysize 2048 -validity 7000
```

i. Replace `<localhost>` with a unique alias (or name) for your key. This name is used (in the **Key** box) when you set up SAML in Composer.

ii. Replace `<mykeystore>` with a unique name for your keystore file. This is the file that is uploaded into Composer.

iii. The validity flag sets the expiration for the certificate for this key. This value can be changed accordingly based on how long your key is valid before expiring.

2. At the prompt, create the keystore password (`<yourKeyStorePassword>`) and press **Enter**.

3. For the next prompt, create a key password (`<yourKeyPassword)>`) and press **Enter**.

4. Enter the following command to import your SSL certificate (valid x.509) into the keystore:

```
keytool -import -trustcacerts -alias <yourAliasName> -file <yourcertr.cer> -keystore <mykeystore>.jks
```

i. Replace `<yourAliasName>` with a unique name for your certificate.

ii. Replace `<yourcertr.cer>` with the name of your SSL certificate.

iii. Replace `<mykeystore>` with the keystore filename that you created in Step 1 above.

The key file is saved to your local hard drive. You will upload the key file into Composer in the configuration instructions provided below.

Step-By-Step Configuration Instructions

Access Composer and take the following steps to configure SAML Settings:

1. Log in as an administrator or a member of the Supervisors group.

2. Select **Tools > Security**. On the **Security Services** tab, make sure that the SAML SSO service is running. Otherwise, enable it, save your changes, and restart Composer for the changes to take effect. For more information, see Enable Or Disable A Security Service.

3. Select the **SAML Settings** tab.
4. In the General Settings section enable SAML.
5. Select **Upload Identity Provider Metadata** to upload the identity provider's (IDP) metadata file into Composer.
6. Specify the base URL to your Composer Server that users will be logging into via SSO. It must be in the following format:

```
<scheme:>//<server>:<port>/composer
```

   If the base URL is changed after saving these SAML settings, update this value and then reboot Composer before downloading the metadata file.

7. Select **Add Key File** to upload the key file (that you generated from the Keytool program).
8. Enter the following information in the screen boxes:

i. **Key** (alias): the unique alias you created for the Key (for example, we used `localhost`)

ii. **Key Pass** (word): your `(yourKeyPassword)`

iii. **Key Store Pass** (word): your `(yourKeyStorePassword)`

9. Return to your IDP account and configure the service provider information to allow Composer access to the following attributes:

i. Required attribute in Composer: Username

ii. Optional Attributes: Groups, Account, Active account, Email, Full Name

   After you have configured the SAML attributes in your IDP account, you can update the Composer SAML mappings.

10. In the **Default Account** list, select the default account to which new users should be added if no account mapping attribute is specified.
11. In the **Account Mapping** box, specify the SAML attribute that contains the comma separated list of Composer accounts associated with the user to be imported into Composer.
12. In the **Active Account Mapping** box, specify the SAML attribute that contains the name of the default Composer account for the user.

13. In the **Login Name Mapping** box, specify the name of the attribute that contains user logins (as established in your IDP account). This value is required so that users can be imported into and given access to Composer. The imported values will be used as login names for Composer users.

14. In the **Full Name Mapping** box, specify the name of the SAML attribute containing users' full names.

15. In the **Email Mapping** box, specify the name of the SAML attribute containing users' emails.

16. In the **Group Mapping** box, specify the name of the SAML attribute containing the multivalue list of group names identifying user memberships.

17. If you want Composer to automatically create groups for users if they don't exist in your environment yet, turn the **Auto Create Groups** on (slide the switch to the right).

    By default, groups created in Composer via SAML do not have any permissions or access to data sources. The Composer Administrator must manually assign privileges to the group.

    > **Note:** If using ADFS, make sure to add this attribute specified in the **Username Mapping** box as a claim rule name in ADFS. Otherwise, this attribute is not sent by the identity provider and causes the SAML login to fail. For more information, refer to Issue #4 in our commonly reported SAML issues.

    The Group, Email, Account, Active Account, and Full Name Mapping values are optional. Use these values if you are looking to automate the setup of user and group attributes in Composer. For information about the mapping options, see Implement Single Sign-On (SSO) Via SAML.

18. Select **Save**.

19. After you have set up all the necessary information on the Composer SAML Settings page and saved the configuration, the last step is to have Composer generate the metadata file that is imported into your organization's IDP.

    Download the metadata file by selecting the corresponding button. The metadata file is an XML file that you upload to your IDP. Successfully enabling SSO in Composer results in a change to the login screen.

20. If you've already configured the SAML configuration and have made changes to the keystore, restart the Composer server for these changes to take effect.

You still have the option to log into Composer without using single sign-on. Selecting the **Show Composer Authentication** option lets you log in using your Composer credentials. The first time that users log into Composer via SAML, Composer automatically creates the user profile in the **Users and Groups** administrative page. In addition, if the user is a member of one or more groups, the Group(s) are also created (as long as the Group Mapping was provided during setup).

Mapping to Custom User Attribute

You can store additional attribute mappings that may be available in your IDP's SAML Assertions file (for example, Address, City, State and Zip Code). To add a custom user attribute, select the corresponding button. Specify the custom user attribute and SAML attribute in the corresponding boxes.

- SAML Attribute- the name of the user attribute in the SAML provider
- Custom User Attribute - the name of the attribute Composer displays
- Usage - how the name appears to the user
- Secure - if you want to encrypt specific custom attribute mappings, select this checkbox

If you want to encrypt specific custom attribute mappings, select the **Secure** checkbox for the required attributes pairs.

Optional Configurations

See also Implement Single Sign-On (SSO) Via SAML.

Configure SAML Behind a Load Balancer

You need to configure the settings to work with Composer using SAML if there is unencrypted communication between the proxy and back-end servers and the load balancer is configured to use SSL.

The default configuration parameters are as follows:

```
saml.lb.enabled=false
saml.lb.scheme=https
saml.lb.serverName=<www.myserver.com>
```

For example, for the following front-end URL `https://<myserver.com>/composer`, configure the settings in the `zoomdata.properties` file as described below:

```
saml.lb.enabled=true
saml.lb.serverName=<myserver>.com
```

Customize the Composer Entity ID

If your identity provider already contains a service provider using the entity ID 'zoomdata', you can create a unique entity ID for Composer. Also, this is applicable if you have two or more separate Composer instances and your IDP requires unique instance identifiers. To do this, you create an alias in the `zoomdata.properties` file using:

```
saml.entityId=<aliasName>
```

Make sure that the base URL you specified while configuring SAML settings is the same as was used for generating the metadata file with the corresponding **entityId**.

For help on accessing and editing the `zoomdata.properties` file, see Configuration Property Files.

Configure the Auto-Redirect to the IDP

SAML can be configured to automatically redirect to the identity provider without prompting you with the Composer login. To configure this, edit the `zoomdata.properties` file and add the following parameter:

```
login.page=/saml/login/**
```

You can still log into Composer using credentials by navigating your browser to `<your_URL>/composer/login`.

For guidance on accessing and editing the `zoomdata.properties` file, see Configuration Property Files.

Troubleshooting

For basic SAML troubleshooting, see Basic SAML Troubleshooting. If additional assistance is needed, reach out to our Technical Support team.

Implement Single Sign-On (SSO) via SAML

Composer supports single sign-on (SSO) using the Security Assertion Markup Language (SAML), a secure, XML-based communication standard for authenticating identities between organizations. SAML eliminates the need for a user to create and maintain multiple authentication credentials (that is, passwords) for different websites. Instead, by leveraging SAML, a user authenticates one time using a secure site (known as the 'Identity Provider' or 'IDP') that then authorizes access to different applications and services that is linked to the user.

Key points to implementing SAML SSO in an organization's operating environment:

- Service Providers must subscribe to an IDP service (or implement one internally) and complete a set up process. Since there are many IDPs options, service providers may subscribe to more than one service for the convenience of their users.
- Users need to complete a registration process to be added to your organization's secured directory including the selection of authentication methods offered by your organization.
- New applications and programs (such as Composer) must be integrated into your organization's existing security protocols.
- Authentication approval from the IDP is limited to a single use and there is a time limit for access.

Prepare to Integrate Composer into Your SAML-Enabled Network

If your organization already has SAML SSO integrated into the operating environment, Composer can be added to your list of secured applications and programs. Composer supports the SAML 2.0 security protocol. Composer provides the following security functionality using SAML: (1) user authentication, (2) group mappings, and (3) account level synchronization of users and groups in Composer. Your organization's Security Administrator or IT Manager responsible for network security may need to be involved if the Composer Administrator does not have account access to your IDP.

Note: Composer can only support one IDP account. If your organization uses multiple IDP accounts, select one that will connect with Composer.

Prior to set up, Composer recommends checking to ensure that Network Time Protocol service is used to synchronize your network with accurate time servers. NTP helps to avoid potential failure by the identity provider to authenticate SAML users.

For more information, see Using the Network Time Protocol.

Composer's SAML Settings provide mappings for the Group, Email, Account, Active account, and Full Name attributes that allow the Composer Administrator to import these settings directly into Composer's Users and Groups administrative function.

Composer also supports an SSL connection to SAML. In order to setup using secured SAML, a keystore needs to be generated and saved in the Composer SAML configuration page. The SSL Certificate needs to be uploaded into the keystore file so that Composer can validate the SSL connection. See Configure Composer to Support SAML for the setup instructions.

The organization's IDP account needs to be imported into Composer as a Service Provider. This entails importing the IDP's metadata file when configuring SAML in Composer. After completing all configuration steps, you need to generate Composer's metadata file so that it can be added to your IDP's account. Again, if your organization has a dedicated security administrator, contact them to assist in this setup procedure.

Note: Composer supplies two default users you can use to log into Composer: admin and supervisor. You must log in as the supervisor to access the SAML configuration page. See Supplied Users and User Groups.

Keep in mind the following SAML requirements that Composer supports:

- IDP account should support SAML 2.0: Your organization's IDP needs to support SAML 2.0 in order to successfully add Composer.
- Default Account section: users can be auto-provisioned to a specific account.
- Importing users and groups from the IDP into Composer: there are two scenarios to consider for importing users and groups:
  - If the user or group profile does not already exist in Composer, they are created the first time that a user logs into Composer. In this case, the profile contains no access privileges and the Composer Administrator needs to set up these profiles.
  - If the user or group profile already exist in Composer, the names must be ***an exact match*** in order for the IDP profile information to populate the corresponding Composer accounts. For example, if the username "johndoe" is stored in the IDP, the exact same username should be in Composer.

After you have successfully configured and enabled SAML, users and groups imported in this manner can be managed from Composer's Users and Groups function. For guidance to import and setup these accounts, see Manage Users .

See Configure Composer to Support SAML for instructions to setup SAML in Composer.

Configure Kerberos Single Sign-On (SSO) Settings

Kerberos is an enterprise authentication protocol that uses the concept of tickets and three-way authentication to enable users and computers to identify themselves and secure access to resources.

Composer supports Kerberos as a Single Sign On (SSO) mechanism. Using Kerberos SSO, users can seamlessly log into Composer and administrators can completely externalize and centrally manage users or group memberships using their existing Kerberos infrastructure. You can learn more about Kerberos here.

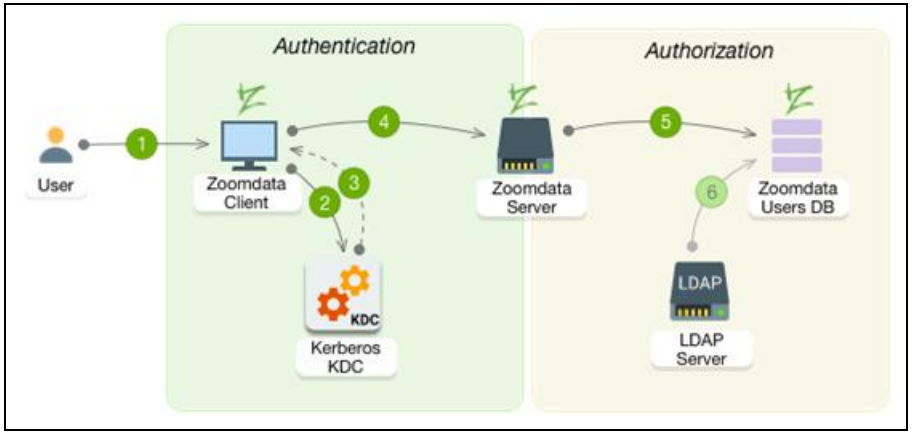
How It Works

When using Kerberos with Composer, the workflow is as follows:

1. A user logs in to his company domain (for example, logging into his windows workstation) and is authenticated with Kerberos. In the case of Windows environments, this is likely Active Directory.

2. In a browser window, a user visits the Composer application and Composer then leverages the user's Kerberos identity to automatically log them into Composer. If this is a users first visit to Composer, then Composer will auto-provision them as a new user in the Composer environment.

3. Kerberos authentication is often paired with LDAP to look up a user's authorization or group membership. Composer will look up the user's group membership. For more information, see the topic on Use Lightweight Directory Access Protocol (LDAP) .



Prerequisites

Before you start configuring settings for Composer to use Kerberos, you must:

- Create a Kerberos Principal. For more information on how to create a Kerberos Principal, refer to Kerberos documentation.
- Create the keytab file on the instance where KDC runs and upload it to Composer Server.

Note: If you have Kerberos enabled, you must use a static context path for the Composer login URL. Multiple or dynamic context paths are not supported at this time.

How to Generate the Keytab File

To generate the `.keytab` file for existing AD users, run the following command on AD server:

```
ktpass
-out <file_name>.keytab
-princ <HTTP/zoomdata_host> @<realm>
-mapUser <user_name>
-pass <user_password>
-crypto RC4-HMAC
-pType KRB5_NT_PRINCIPAL
```

To generate the `.keytab` file for existing MIT LDAP user, run the following command on LDAP server side:

```
$kadmin
kadmin: xst -e "rc4-hmac" -k <file_name>.keytab <HTTP/zoomdata_host> @<realm>
```

Configure General Settings

To configure the Kerberos settings for Composer Server, complete the following steps:

1. Enable the Kerberos SSO service on the Security Services tab. Keep in mind, that if you have SAML or x509 authentication enabled, you will have to disable them first to use Kerberos.
2. Restart the Composer server by running the following command:

   ```
   sudo service zoomdata restart
   ```

3. Log in as a system admin or a member of the Supervisors group and select **Security**. Navigate to the **Kerberos Settings** tab.

4. Slide the **Enable Kerberos** switch on (to the right).
5. Specify the **Kerberos Service Principal**.
6. Select **Upload Kerberos Keytab File** and upload the `.keytab` file, that you have generated before.
7. Select the **Include Kerberos realm/domain name in auto provisioned Composer username** if you want to have the user name in the following format: `username@realm`.
8. Save your settings.

Configure the Settings on the Client Side

Perform the steps listed below on the client instance that will connect to kerberized Composer.

1. Install the Kerberos command line tools:

```
sudo yum install krb5-workstation
sudo yum install krb5-libs
```

2. Navigate to the `krb5.conf` file and specify the host on which Composer server runs:

```
[libdefaults]
default_realm = ZOOMDATA.LOCAL
[realms]
ZOOMDATA.LOCAL = {
kdc = <composer_host>.local
admin_server = <composer_host>.local
}
[logging]
default = FILE:/var/log/krb5.log
kdc = FILE:/var/log/krb5.log
[domain_realm]
.zoomdata.local = ZOOMDATA.LOCAL
zoomdata.local = ZOOMDATA.LOCAL
```

3. List all the Kerberos tickets available on the current instance:

```
klist
```

4. Remove all the Kerberos tickets (if there are any):

```
kdestroy -A
```

5. Obtain a Kerberos ticket for a user (the realm is not required if there is a default one specified in krb5.conf):

```
kinit user@ZOOMDATA.LOCAL
```

6. Configure your browser to support Kerberos SSO to Composer.

Note: To authenticate a user with a Kerberos ticket in Composer, you must either enable LDAP autoprovisioning or create a user with the same name in Composer.

Kerberos Authentication for Connectors

Kerberos is an enterprise authentication protocol that uses the concept of tickets and three-way authentication to enable users and computers to identify themselves and secure access to resources. Kerberos support does not apply to some connectors.

Support for this feature by connector is shown in the following table.

Key:Y - Supported; **N** - Not Supported; N/A - not applicable

Connector	Supported?	Notes
Amazon Redshift	**N**	
Amazon S3	**N**	
Apache Drill	**Y**	
Apache Phoenix	**Y**	Apache Phoenix supports Kerberos, but Apache Phoenix Query Server does not. For more information, see Enable Kerberos Authentication For Apache Phoenix Connectors.
Apache Phoenix Query Server (QS)	**N**	
Apache Solr	**Y**	
BigQuery	**N**	If you need to access a BigQuery partition, explicitly include an alias for the built in partition column in your select clause, such as `select *, _PARTITIONTIME as pt from projectId.datasetId.tableId`.
Cloudera Impala	**Y**	
Cloudera Search	**Y**	
Couchbase	N/A	
Dremio	**N**	
Elasticsearch 7.0	**N**	
Elasticsearch 8.0	**N**	
File Upload	**N**	
HDFS	**Y**	
Hive	**Y**	
Jira	**N**	
MemSQL	**N**	
Microsoft SQL Server	**N**	
MongoDB	**N**	
MySQL	**N**	

Connector	Supported?	Notes
Oracle	**N**	
PostgreSQL	**N**	
Python	**N**	
Real Time Sales	N/A	
Salesforce	**N**	
SAP Hana	**N**	
SAP S/4HANA	**N**	
SAP IQ	**Y**	
Spark SQL	**Y**	To enable Kerberos authentication for Spark SQL connectors, see Connect To Spark SQL Sources On A Kerberized HDP Cluster.
Snowflake	**N**	
Teradata	**N**	
TIBCO DV	**N**	
Trino	**N**	
File Upload (Upload API)	**N**	
Vertica	**N**	

Use Lightweight Directory Access Protocol (LDAP)

Lightweight Directory Access Protocol (LDAP) is an application protocol used over an IP network to manage and access directory information contained in an organization's secured network. Composer has been tested and can be used with Active Directory and OpenLDAP directory services. The Composer server can be configured to use one of these LDAP services to authenticate users. When LDAP is enabled, users can log into Composer using their familiar LDAP identity and credentials.

Note: LDAP configuration management tasks can be performed by a system admin or a member of the Supervisors group.

Note: User attributes set in regular Composer user definitions via the UI or in LDAP user definitions are encrypted when stored in metadata. To specify the encryption mode, see Change The Encryption Mode.

Connecting Composer to an LDAP directory requires coordination between the authorized LDAP administrator and the Composer administrator. After enabling LDAP authentication in Composer, you can import users in the LDAP directory of your organization. All user information will be maintained in LDAP, not in your Composer user accounts.

The following LDAP configuration information is needed to configure LDAP in Composer:

Screen Box	Description
URL	The LDAP connection string for connecting to the LDAP repository. Connection string is of the following format: `ldap://<ldap-server>:<ldap-port>` ■ Replace `<ldap-server>` with the DNS name or IP address of the LDAP repository server ■ Replace `<ldap-port>` with the port number where the LDAP service is listening on `<ldap-server>` (typically port 389)
Bind user	User name credential of the service account that (at minimum) has read access to the LDAP repository.
Bind password	Password credential for the Bind user (LDAP service account).
Search base	Identifies the Distinguished Name (DN) - the location in the LDAP directory tree where to begin queries for registered users in the LDAP directory.
Query	LDAP query that will resolve a specific set of users group found in the search base to be imported into Composer.
User ID attribute	Identifier attribute for users in LDAP implementation of your organization. The following user ID attributes are supported: **UID**, **CN**, **sAMAccountname**, and **userPrincipalName**. This attribute will determine how user names will be represented in Composer.

Configure LDAP

Configure LDAP

1. Log in as a system admin or a member of the Supervisors group.

   Note: The default **supervisor** user is no longer installed; add users to the **Supervisors** group instead.

2. Select **Security** from the menu ( ). The security tabs display.

3. On the Security Services tab, make sure that the LDAP security service is on. If it is not running, turn it on.

4. Select the **LDAP Settings** tab. The LDAP Settings tab has five sections: **General Settings**, **LDAP Server**, **User Provisioning**, **Mappings**, and **Mappings to Custom User Attributes**.

5. In the **General Settings** section, switch the **Enable LDAP** option on (slide it to the right).

6. Enter the LDAP connection URL (DNS or IP address) where the LDAP directory resides.

7. Enter the Bind User and Bind Password credentials. The authorized LDAP administrator needs to provide these credentials.

8. Specify the Search Base which is the DN or location in the LDAP directory tree where a search for registered users can begin. An example entry is provided in the text field: `OU=people,DC=zoomdata,DC=local`, where:
    i. OU means organizational unit
    ii. DC means domain controller
9. Provide a query string that can run to identify user nodes under the Search Base. An example is provided in the text field: `(objectclass=person)`. Keep in mind that you can only import individual users into Composer. As a result, your query should be limited to objects that are designated as a “person” or “user.” Use a search engine to look up 'common LDAP query strings'.
10. Optionally enable user provisioning, configure mappings and mappings to custom user attributes.
11. Manually import users from the LDAP directory. See Manually Importing Users from the LDAP Directory.
12. To use the secure LDAP connection, import the certificate to your local `jre` key store. See Using the Secure LDAP Connection.

Enable User Provisioning

Use the **User Provisioning** section to enable user provisioning. User provisioning allows you to verify the identity of users that log into Composer against the LDAP directory and automatically create new users in Composer if the user's credentials have been validated against the LDAP directory.

If disabled, you must manually import the users in order to allow them to log into Composer. See Manually Importing Users from the LDAP Directory.

When you have enabled the auto provisioning feature, you can select the default account for the provisioned users to be added.

The **Default Account** list contains all the account names, that are available within your Composer instance. If you want the users to be added to one of them, select the corresponding account. Otherwise, select the **User Account Mapping** option to configure the mappings with LDAP attributes for your users.

Configure Mappings

Use the **Mappings** subtab to define mappings that bind the user attributes from LDAP and Composer.

1. Select **Login Name Mapping** attribute from the list that will be used as a user login. There are four User ID Attributes supported: UID, CN, sAMAccountname, and userPrincipalName.
2. Account Mapping - select the account to which the user should be added. Account names are case-sensitive.
3. Active Account Mapping - select the account to which the user will log in for the first time

4. Full Name Mapping
5. Email mapping
6. If you want to import users and the groups which they are assigned to, in the **Group mapping attribute** box, type the name of the corresponding column in LDAP.
7. If you want Composer to automatically create groups for users if they don't exist in your environment yet, turn on **Auto Create Groups** (slide the switch to the right).

   After the credentials are verified, the user groups will be created in Composer and each user will be assigned to the corresponding group.

Manage Mappings to Custom User Attributes

You can associate custom user attributes with a Composer user on the **Mappings to Custom User Attributes** subtab. Custom user attributes can to store values used for credential pass through. This means that if users have access to a particular data source that has been connected to Composer, their credentials can be saved on this page so that their access privileges are maintained for that source within Composer.

Manually Import Users from the LDAP Directory

Manually import users from the LDAP directory to Composer

1. Log in as a system admin or a member of the Supervisors group.

   Note: The default **supervisor** user is no longer installed; add users to the **Supervisors** group instead.

2. Select **System Users** from the menu. The Manage Users work area opens.
3. On the **Users** tab, select **Import Users**. A list of users in the LDAP directory is displayed.
4. To import specific users, select them from the list. To import all users, select **Select All**.
5. Select **Import**.

If needed, you can delete imported users using the Users list in the left pane.

After users have been imported into Composer, they can be assigned groups and permissions. For an overview of how Composer manages users and groups and how to assign groups and permissions, see Authorize Composer Access.

Note: When a user is imported from Active Directory or if user provisioning is enabled and a new Active Directory user is added, the corresponding Composer user definition is automatically added. However, when a user is removed from Active Directory, the corresponding Composer user definition is not automatically removed. Composer authentication does not occur for the removed user, but you will need to manually remove the Composer user definition. See Delete Users.

Use the Secure LDAP Connection

To use the secure LDAP connection, you need to import the certificate to your local `jre` key store.

1. Run the following command:

```
sudo keytool -import -file <ca_file_name>.pem -keystore /opt/zoomdata/jre/lib/security/cacerts
```

2. Restart Composer after importing the certificate:

```
sudo service zoomdata restart
```

Now when using a secure connection to LDAP, the URL must be as follows:

```
ldaps://<ldap_server>:636
```

Trusted Access

Composer provides its own security methodology that allows for machine-to-machine authorization of Composer resources when embedded in your application (the “parent” application). This is a form of “delegated” authorization where the parent application can determine, on demand, how and when to authorize any given embedded Composer component to an end-user logged into the parent application. This methodology is called *Trusted Access*.

Note: insightsoftware recommends using Trusted Access for all embed-related workflows.

Similar to "single sign-on," this arrangement allows users to log in once to the parent application and yet have their security information propagated to Composer, creating a seamless and secure user experience. This, of course, means that users can't be allowed to "go around" the parent application and directly access Composer. In the stateless world of web applications, this requires some special mechanisms to ensure security that are provided for applications through our SecureKey technology.

On request from the parent application, Trusted Access provides a user access token with defined authorization rules that account for user privileges, object permissions, security filters and any specific user attributes used in interpolation. This user access token can then be used in the parent application to serve any Composer specific embedded components such as dashboards for the respective user. For information on how tokens are initiated and requested in your applications, see Embed ComposerSymphony Components Using JavaScript And Trusted Access.

Note: In environments where you use Typescript for your client side code, you can use Embed Manager as an npm package. See https://www.npmjs.com/package/logi-embed.

Trusted Access tokens are encrypted when stored in Composer metadata. The encryption mode used can be set as described in Change The Encryption Mode.

This topic also describes:

- Trusted Access Prerequisites
- Trusted Access Recommendations
- Register A Client
- Generate A User's Access Token

The following additional topics provide reference information:

- Trusted Access API Endpoints
- Trusted Access Client Properties

Trusted Access Prerequisites

- Every end user must have Composer user account defined, unless you are using LDAP autoprovisioning with Composer. See Manage Users.
- Trusted Access is enabled by default. If it is disabled, enable Trusted Access by selecting the **Trusted Access** option on the Security page. See Enable Trusted Access.

> **Note:** If you are installing or upgrading to v23.4 or later, the default admin user (system administrator) can perform all functions the former supervisor user previously performed, and provide authentication for embedded use cases.

Trusted Access Recommendations

For security reasons, we recommend that you use short-lived tokens. Tokens that are valid for less than 10 minutes are recommended. The validity time of a user access token is defined when you register a client with Composer.

Register a Client

To start using Trusted Access, you first need to register your application, as Composer refers to it, as a *client*.

Registering a client will generate a client ID and client secret. These credentials can then be used to generate user access tokens for any user in the Composer platform, as needed.

To register your application as a client, POST the `/api/trusted-access/clients` API endpoint. You can also patch, delete, and list Trusted Access clients using the `/api/trusted-access/clients` API endpoint. See Trusted Access API Endpoints.

Generate a User's Access Token

To generate a user's access token, pass the client ID and client secret to HTTP BasicAuth. To obtain the client ID and client secret, use the `/api/trusted-access/clients` API endpoint. See Trusted Access API Endpoints.

Generate a User's Access Token for Existing Composer Users

```
/********REQUEST TRUSTED ACCESS TOKEN ********/
const AccessToken = (ComposerUrl, Username, callback) => {
       var Client = GetClient();
       if(typeof Client === 'undefined' ||  Client === null)
               callback({ErrorMessage: 'Client Not Found', status : 500});
```

```
	else {
		var BasicAuth = Buffer.from(`${Client.client_id}:${Client.client_secret}`).toString('base64');
		Post(BasicAuth, `${ComposerUrl}/api/trusted-access/pull/tokens`, { "username": Username }).then((result) => {
			if(JSON.stringify(result).indexOf('error')>-1)
				callback(result, null);
			else
				callback(null, result);
		});
	}
};
```

Note: You can only generate tokens for regular users and for administrators.

Generate a User's Access Token for New Composer Users

```
/********REQUEST TRUSTED ACCESS TOKEN ********/
const UserContext = {
	"username": "joe",
	"account": "company",
	"fullname": "Example Inc",
	"email": "joe@example.inc",
	"groups": ["Store Manager", "Cashier"],
	"attributes": [{"key": "city", "values": ["London"]}]
};
const AccessToken = (ComposerUrl, UserContext, callback) => {
	var Client = GetClient();
	if (typeof Client === 'undefined' || Client === null)
		callback({ErrorMessage: 'Client Not Found', status: 500});
	else {
		var BasicAuth = Buffer.from(`${Client.client_id}:${Client.client_secret}`).toString('base64');
		Post(BasicAuth, `${ComposerUrl}/api/trusted-access/push/tokens`, UserContext).then((result) => {
			if (JSON.stringify(result).indexOf('error') > -1)
				callback(result, null);
			else
				callback(null, result);
		});
	}
};
```

Note: You can only generate tokens for regular users and for administrators.

Trusted Access API Endpoints

The following API endpoints can be used to manage Trusted Access.

Endpoint	Method	Description
`/api/trusted-access/clients`	GET	Returns all the Trusted Access client information in the metadata. Included with this information is the access token validity time (in seconds), client ID, client name, client secret expiration time (in seconds), and the token authentication method. For a description of these, see Trusted Access Client Properties.
`/api/trusted-access/clients`	POST	Creates a Trusted Access client. The request must specify the number of seconds for which the access token is valid and the client name. The client name must be unique. When you create the client, the client ID, client secret, secret expiration time, and the token authentication method are automatically generated.
`/api/trusted-access/clients/<id>`	GET	Returns the Trusted Access client information for a specific client. The request must specify the client ID.
`/api/trusted-access/clients/<id>`	DELETE	Deletes a specific Trusted Access client. The request must specify the client ID.
`/api/trusted-access/clients/<id>`	PATCH	Updates the Trusted Access client information for a specific client. The request must specify the client ID and the number of seconds for which the access token is valid.
`/api/trusted-access/pull/tokens`	POST	Use pull to request a user access token for users that already exist in Composer. The user must already exist, and have an active Composer user account (unless you are using LDAP with automatic provisioning for Composer). You can't update user context such as user attributes or groups using `pull`.
`/api/trusted-access/push/tokens`	POST	Use push to request a user access token for new and existing users by sending their context to Composer. Existing users are updated if their context has changed. Context must contain `username`, and `account`. It can optionally include `email`, `fullname`, `groups` and other individual `attributes` Several reserved keys are restricted from use in Composer as custom user attributes: `composerUserName` and `accountId`. If you push these reserved keys, a 400 Bad Request error is returned via API.

Trusted Access Client Properties

The following table describes the Trusted Access client properties. All of this information is encrypted when it is stored in the Composer metadata store, except the client secret, which is hashed using BCrypt.

Endpoint	Description
`client_name`	The human-readable string name of the client application. The client name must be unique.
`client_id`	The client ID issued to the client when the client is registered with Trusted Access. This is generated by the system and cannot be modified.
`client_secret`	The client secret string issued to the client when the client is registered with Trusted Access. This value is used by applications to authenticate to the token endpoint. The client secret is not available after it is initially created -- developers must store it locally in order to retain it.
`access_token_validity_seconds`	The validity (in seconds) of the access token. Short-lived tokens (less than 10 minutes) are recommended.
`client_secret_expires_at`	The time (in seconds) at which the `client_secret` will expire. A value of zero (0) means that the client secret never expires.
`token_endpoint_auth_method`	A string indicator of the authentication method used for the token endpoint. For Trusted Access, this value is always "`client_secret_basic`", which means that the client must always use HTTP basic authentication to request an access token.

Composer Service Monitor

Composer includes a Service Monitor microservice, which can be used to:

- Review all Composer microservices and their log files
- Produce a diagnostics bundle to send to Composer Support
- Set the logging level and properties for each microservice

Note: Composer recommends that you contact Composer Support for assistance with the installation and use of the Service Monitor.

Composer's Service Monitor provides administrators with real-time views of microservice health, configuration, and logs. The Service Monitor is not installed as part of a default Composer installation, so it must be installed and configured before you can use it.

Prerequisites

- Composer and its microservices must be version 3.5 or later.
- All microservices must have service discovery enabled.
- The Composer Service Monitor must be manually installed, configured and started.

The following topics provide complete information about the Composer Service Monitor:

- Install And Configure The Composer Service Monitor
- Access The Composer Service Monitor
- Service Monitor Views
- Download The Diagnostics Bundle
- Maintain Application Properties

Install and Configure the Composer Service Monitor

The Composer Service Monitor microservice is not installed as part of a default Composer installation. The microservice name is `zoomdata-admin-server`.

> **Note:** If you are installing Composer in a Windows environment, you can install the Service Monitor as part of running the initial bootstrap script. See Install Composer In A Windows Environment and Windows Bootstrap Reference.

Install, configure, and start the Service Monitor

1. Open your SSH client.
2. Use the following command to install the Composer Service Monitor in a CentOS environment:

```
sudo yum install zoomdata-admin-server -y
```

   Use the following command to install the Composer Service Monitor in an Ubuntu environment:

```
sudo apt-get install zoomdata-admin-server
```

3. After the Service Monitor is installed, you must specify a user name and password in its properties file. The properties file is called `admin-server.properties` and can be found in the `/etc/zoomdata/` directory (Linux) or the `<install-path>/conf-modify/` directory (Windows). If the properties file is not there, create it. The properties that must be defined are:

i. `monitor.user.name=<username>`

ii. `monitor.user.password=<password>`

   Edit the properties file with a text editor and substitute a Service Monitor user name for `<username>` and its associated password for `<password>`. The user name and password can be any user name and password you want.

   When you have finished, save the file.

4. Add the following properties to the `zoomdata.properties` file, located in the `/etc/zoomdata` directory (Linux) or the `<install-path>/conf-modify/` (Windows). These properties ensure that the Service Monitor has access to the Composer server actuator endpoints.

i. `actuator.user.name=<Composer-admin-username>`

ii. `actuator.user.password=<Composer-pswd>`

iii. `actuator.logging.external-file=<log-file-path>`

Edit the properties file and substitute the valid user name and password of a Composer administrator for `<Composer-admin-username>` and `<Composer-pswd>`. If the default Composer log file path is not used for your installation, substitute your custom log file path for `<log-file-path>`. The default log file path is `/opt/zoomdata/logs/zoomdata.log` for Linux and `<install-path>/logs/zoomdata.log` for Windows.

> **Note:** Setting these properties exposes valid Composer credentials as plain text in both the properties file and as tags in the Composer Consul. Anyone in your network with the ability to communicate directly with the Consul API or view the Consul UI will be able to see these values.

When finished, save the file.

5. Start the microservice. For example, use the following command to start the Composer Service Monitor using `systemd` in a CentOS or Ubuntu environment:

```
sudo systemctl start zoomdata-admin-server
```

See also Start Composer Microservices.

Access the Composer Service Monitor

After installing the Service Monitor and restarting its microservice, you can navigate to its UI in a web browser using default port 8050. For example, if running locally, the Composer instance would be **http://localhost:8080**, so the Service Monitor URL would be **http://localhost:8050**. If your Composer instance is IP address URL 10.2.3.24, the Service Monitor URL would be **http://10.2.3.24:8050**.

The user ID and password you should use to log into the Service Monitor are defined in properties you set up during the Service Monitor installation. See Install And Configure The Composer Service Monitor.

Service Monitor Views

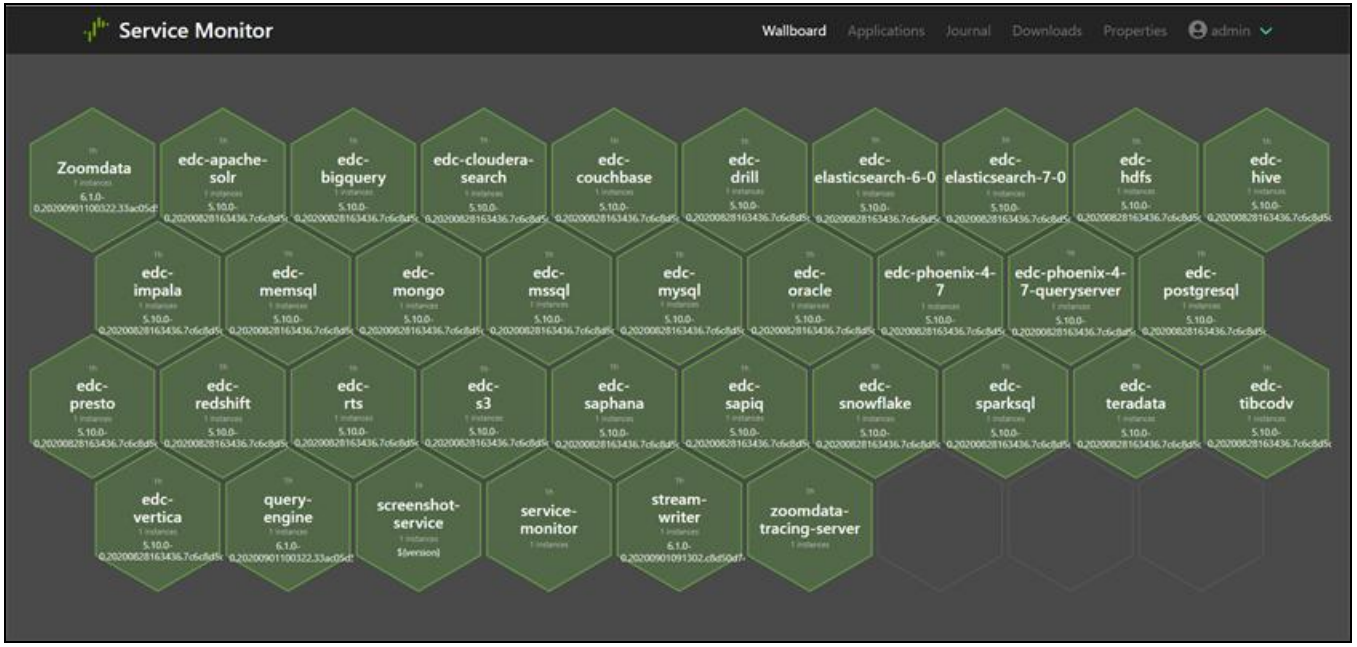
The Composer Service Monitor UI includes the following views, each accessible via a menu option in the main menu on the Service Monitor UI page:

Each view is described below.

- Wallboard View
- Applications View
- Journal View
- Downloads View
- Properties View

Wallboard View

The Wallboard view provides shows all the Composer microservice types used in your installation. You can access it by selecting the **Wallboard** menu option.

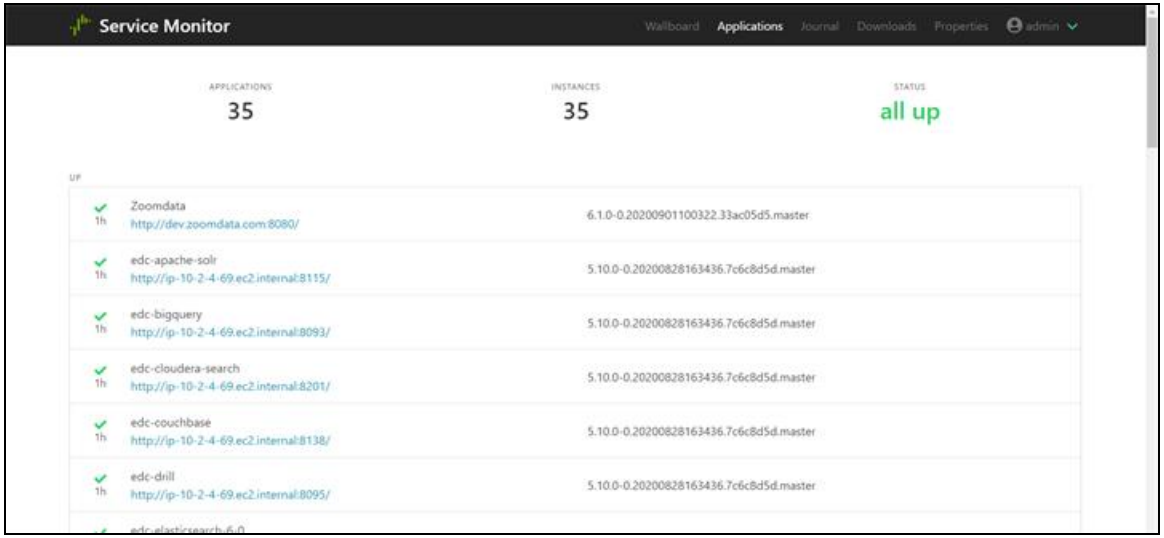


The Wallboard view shows how many instances of a Composer microservice are running and the length of time they have been running.

Select a microservice type to obtain detailed information about it. The detailed information available varies based on the microservice type, but may include metrics, health, environment, configuration properties, scheduled tasks, logging threads, audit log, and web mappings and HTTP traces for the microservice type.

Applications View

Select **Applications** to access the Applications view.

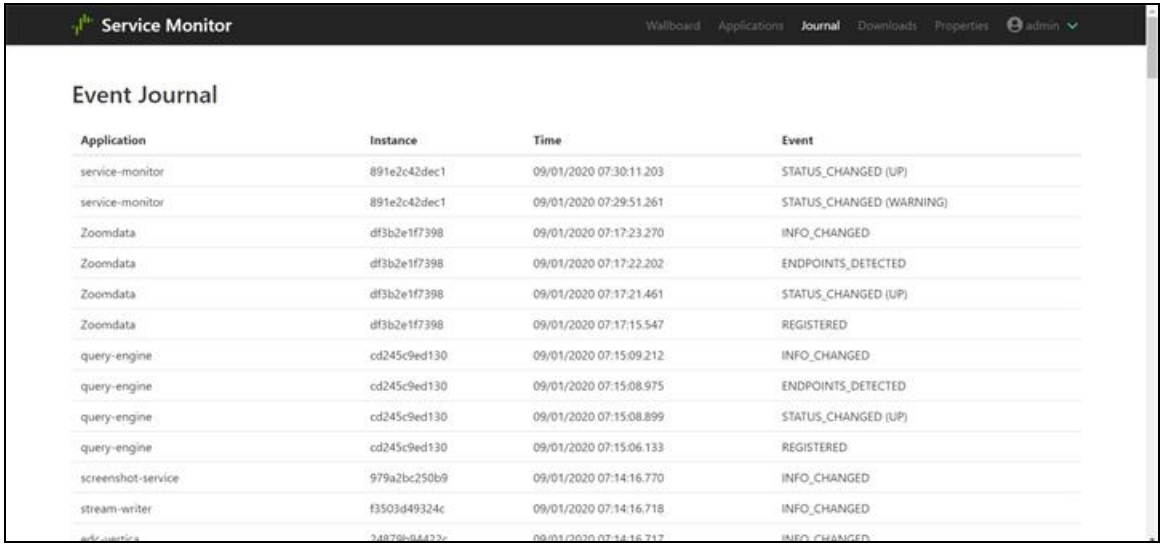


The Applications view shows all the Composer microservice types used in your installation. It also identifies the instance URLs, and indicates how many instances there are of each and how long they have been running. If more than one instance of a microservice is running, you can expand the microservice type to see the specific instance URLs.

If you select the URL for a Composer microservice instance, you will launch the Composer UI for that instance. If you select the URL for a Service Monitor instance, you will launch the Service Monitor for that instance. If you select the URL for an instance of any other microservice, a page of metrics and other information appears for that instance of the microservice.

Journal View

Select **Journal** to access the Journal view.



The Journal view allows you to review the journal entries for each Composer microservice type.

Downloads View

Select **Downloads** to access the Downloads view.

The Downloads view can be used to collect a diagnostics bundle for you to send to Composer Support when necessary. See Download The Diagnostics Bundle for more information.

Properties View

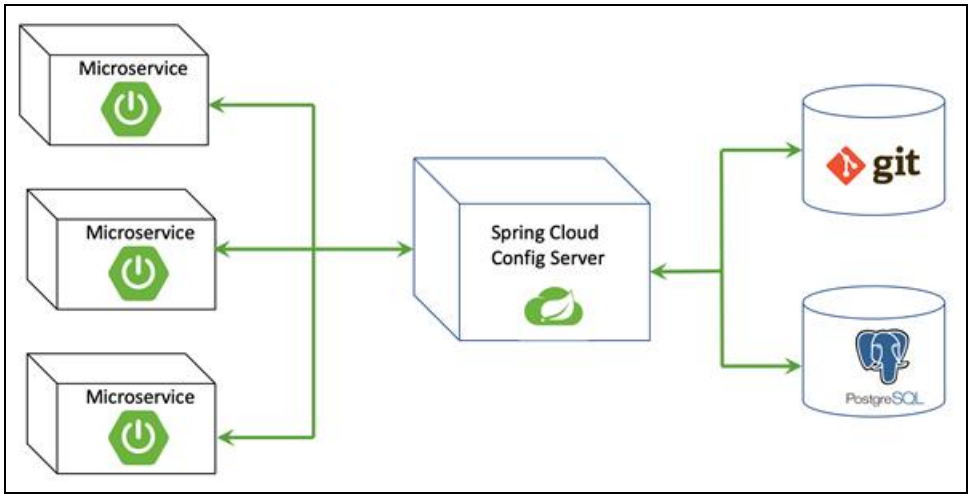
Select **Properties** to access the Properties view.

This view allows you to review and maintain the properties for each of the other Composer microservices. It requires that the Composer configuration microservice be installed and started first. See Use The Composer Configuration Microservice To Maintain Application Properties.

Use the Composer Configuration Microservice to Maintain Application Properties

You can use the Service Monitor to review and maintain the properties for each of the other Composer microservices. It provides a centralized location where Composer configuration properties can be maintained. However, it requires that the Composer configuration microservice be configured and started first.

The Composer configuration microservice is packaged with the Spring Cloud Configuration server, which allows Composer to easily integrate with its Spring-based microservices and provides the mechanism by which Composer property settings can be persisted in a supported PostgreSQL metastore or in a GitHub repository. The following diagram depicts this relationship.



After the Composer configuration microservice is installed and started, the properties can be maintained on the Service Monitor's Properties tab.

If you have the configuration microservice configured and running in a high availability environment, you can maintain properties for microservices of a given type in a single location in the Service Monitor. For example, if you have two query engine microservices running in your high availability environment, you can change the properties for both microservices in a single location, ensuring that the query engine microservices operate in the same manner across the product nodes. A `config-server-upload.jar` utility is provided that can be used to migrate the microservice properties from your standalone Composer servers to the Composer configuration data in the high availability PostgreSQL data store, where the configuration microservice can maintain them. For more information see Migrate Properties To The Configuration Server.

See the following topics:

- Set Up The Configuration Microservice Metadata Store Or Repository
- Configure And Start The Configuration Microservice

- Maintain Application Properties

Set Up the Configuration Microservice Metadata Store or Repository

Before you can install and start Composer's configuration microservice, you must set up a PostgreSQL metastore or a GitHub repository to store Composer property metadata. A separate PostgreSQL database or a separate GitHub repository must be configured.

PostgreSQL Database Setup Notes

If you elect to persist property metadata to a PostgreSQL metastore:

1. Configure a separate database and make it accessible to the connection user account:

   ```
   CREATE DATABASE <composer-config> WITH OWNER <db_username>;
   ```

   where `<composer-config>` is the name of the PostgreSQL database and `<db_username>` is the connection user account name.

2. Add the following properties to the Composer`config-server.properties` file, located in the `/etc/zoomdata` directory:

   ```
   # metadata storage settings
   spring.datasource.url=jdbc:postgresql://localhost:5432/<composer-config>
   spring.datasource.username=<db_username>
   spring.datasource.password=<db_password>
   ```

   Substitute the connection user account name and password you set up in Step 1 for `<db_username>` and `<db_password>`. Substitute the name of the PostgreSQL database for `<composer-config>`.

3. Save the properties file. You will restart the configuration microservice when you configure it. See Configure And Start The Configuration Microservice.

GitHub Repository Setup Notes

If you elect to persist property metadata to a GitHub repository:

1. Add the following properties to the Composer`config-server.properties` file, located in the `/etc/zoomdata` directory:

   ```
   # metadata storage settings
   spring.cloud.config.server.git.uri=<repo_uri>
   spring.cloud.config.server.git.skipSslValidation=true
   ```

```
spring.cloud.config.server.git.username=<repo_username>
spring.cloud.config.server.git.password=<repo_password>
```

Substitute the repository user account name and password for `<repo_username>` and `<repo_password>`. Substitute the URI of the repository for `<repo_uri>` (for example, `https://example.com/my/repo`).

Additional and advanced configuration information can be found in Spring.io's documentation.

2. Save the properties file. You will restart the configuration microservice when you configure it. See Configure And Start The Configuration Microservice.

Configure and Start the Configuration Microservice

Install, configure, and start the Composer configuration microservice

1. Verify that you have set up a PostgreSQL metastore or a GitHub repository to store the property metadata. See Set Up The Configuration Microservice Metadata Store Or Repository.

2. Open the SSH client associated with your Composer instance.

3. Configure all installed and enabled Composer microservices to use the Composer configuration microservice. Run the following script:

```
for i in $(systemctl list-unit-files | grep zoomdata | grep enabled | awk '{print $1}'|sed -e
's/\.service/.properties/g' -e 's/zoomdata\-//g');
do
echo "config-server.enabled=true">>/etc/zoomdata/$i;
done
```

4. Each Composer microservice supports two configuration properties related to the configuration microservice:

i. `config-server.enabled`: Enables or disables integration with the configuration microservice. Valid values are `true` (enable integration) and `false` (disable integration). The default is `true`.

ii. `config-client.retry.max-attempts`: Sets the maximum number of attempts that should be made to connect to the configuration microservice. The default is 20. The Composer microservice will fail if the number of attempts to connect to the configuration microservice exceeds this value. This property is useful in situations where the configuration microservice starts with a delay. If your Composer microservice fails while waiting for the configuration microservice and you want to give it more time, increase this value.

Update these properties, as appropriate, for each microservice.

5. Install, enable and start the Composer configuration microservice. Enter the following commands:

```
sudo yum install zoomdata-config-server \
&& systemctl enable zoomdata-config-server \
&& systemctl start zoomdata-config-server
```

Note: After starting the configuration microservice with a valid database configuration, the microservice should connect to the database and create a properties table.

6. Restart all the other Composer microservices. Enter the following command:

```
sudo systemctl restart $(systemctl list-unit-files | grep zoomdata | grep enabled | awk '{print $1}')
```

See also Restart Composer Microservices.

Maintain Application Properties

After the Composer configuration microservice has been configured and started, you can use the Properties tab in the Service Monitor to maintain the configuration properties of Composer's other microservices. For information about specific Composer properties and property files, see Configuration Property Files.

Maintain configuration properties using the Service Monitor

1. Using a web browser, navigate to the Service Monitor for the Composer instance. Its default port is 8050. For example, if running locally, the Composer instance would be **http://localhost:8080**, so the Service Monitor URL would be **http://localhost:8050**. If your Composer instance is at IP address 10.2.3.24, the Service Monitor URL would be **http://10.2.3.24:8050**.

   A login screen will appear.

2. Log into the Service Monitor using the Service Monitor user name and password you defined when the Service Monitor was installed. See Install And Configure The Composer Service Monitor.

3. Select **Properties** on the main menu bar.

   The Properties page appears.

4. Select a Composer microservice in the **Service** drop-down list. The properties for the microservice are listed. The screenshot above lists all the properties in the `zoomdata.properties` file used by the `zoomdata` microservice.

   See ComposerSymphony Data Discovery Microservice Name Reference for a list of Composer's microservices. If a microservice is not listed in the **Service** drop-down list, that Composer microservice is not yet set up to use the configuration server or the microservice is down.

5. Locate the microservice property in the list that you want to change.

i. The list of properties can be sorted alphabetically by key or value. Select the **Key** or **Value** list heading to sort the properties in ascending or descending order.

ii. You can locate a property in the list by typing all or part of its name in the **Search** box at the top of the page.

6. Change the value of the property in the **Value** box associated with the microservice property you located.

   > **Note:** Be careful changing properties. Some properties should not be changed, except with the help of Composer Technical Support. The results of property changes could be unpredictable, if they are not made correctly.

   See Configuration Property Files for a list of the property files and links to descriptions of the properties for which changes are approved.

7. Select **Save Properties** to save your property changes.

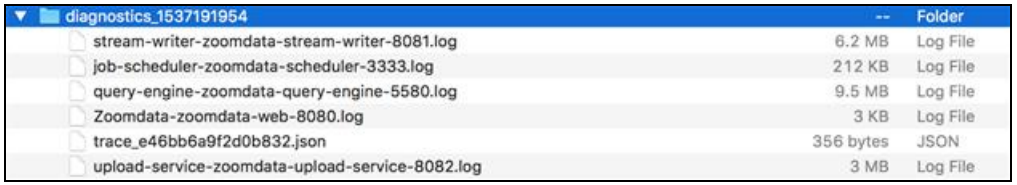
8. Select **Apply Updates** to apply the property changes to your running Composer instance. Composer dynamically stops and restarts the appropriate Composer microservices.

Diagnose Problems

Composer provides a "one-click" diagnostics bundle for the Composer platform as part of the Composer Service Monitor.

The diagnostics bundle is primarily useful as an easy way to send Composer the information needed to help you debug a problem. The bundle is a zip file containing all the current log files from all Composer's microservices. If distributed tracing is enabled and properly configured, the diagnostics bundle may also contain a JSON trace file with the trace ID you specify.

An example of the uncompressed contents of the diagnostic bundle might look like this:



Prerequisites

Before you can download the diagnostics bundle, the following conditions must be met:

- Composer and its microservices must be version 3.5 or later.
- All microservices must have service discovery enabled.
- The Composer Service Monitor must be manually installed, configured and started. See Install And Configure The Composer Service Monitor.

After the Service Monitor is installed, you can download the diagnostics bundle. See Download The Diagnostics Bundle.

Download the Diagnostics Bundle

After the Composer Service Monitor microservice is installed and running, you can use it to download a diagnostic bundle. If you want to include trace details in the diagnostics bundle, be sure to collect tracing information as described in Distributed Tracing For Composer.

Download a diagnostics bundle

1. Using a web browser, navigate to the Service Monitor. Its default port is 8050. For example, if running locally, the Composer instance would be **http://localhost:8080**, so the Service Monitor URL would be **http://localhost:8050**. If your Composer instance is at port 10.2.3.24, the Service Monitor URL would be **http://10.2.3.24:8050**.

   A login screen will appear.

2. Log into the Service Monitor using the Service Monitor user name and password you defined when the Service Monitor was installed. See Install And Configure The Composer Service Monitor.

3. Select **Downloads** on the main menu bar.

   A screen appears that you can use to download the diagnostics bundle:

4. Optionally enter a trace ID in the Trace ID box.

   If the tracing service is not configured, the Trace ID box will be disabled, with a relevant error message. When this happens, the diagnostic bundle contains only log files.

5. Select **Download Diagnostic Bundle**.

   If you did not specify a trace ID, the zip file that is produced contains only log files for all configured Composer microservices. If you specify a valid trace ID, the trace JSON file will also be included as part of the bundle.

   If you specify an invalid trace ID, a relevant error message displays.

In some cases, a previously valid trace ID may be reported as not found. Traces are best retrieved as soon as an error is encountered.

For more information about distributed tracing microservices, see Distributed Tracing For Composer.

Distributed Tracing for Composer

All Composer microservices support distributed tracing using the OpenTelemetry (OTel) specifications. Integrate with OTel javaagent or any other OTel compliant agent. Install a tracing server of your choice, an OTel collector, and configure the microservices to trace Composer front end and back end requests.

Tracing collects information about requests. Use it to associate logs of the other microservices with the collected information.

- Install A Tracing Server
- Install And Configure OTel Collector
- Configure The Collector
- Configure Composer Microservice Tracing

Install a Tracing Server

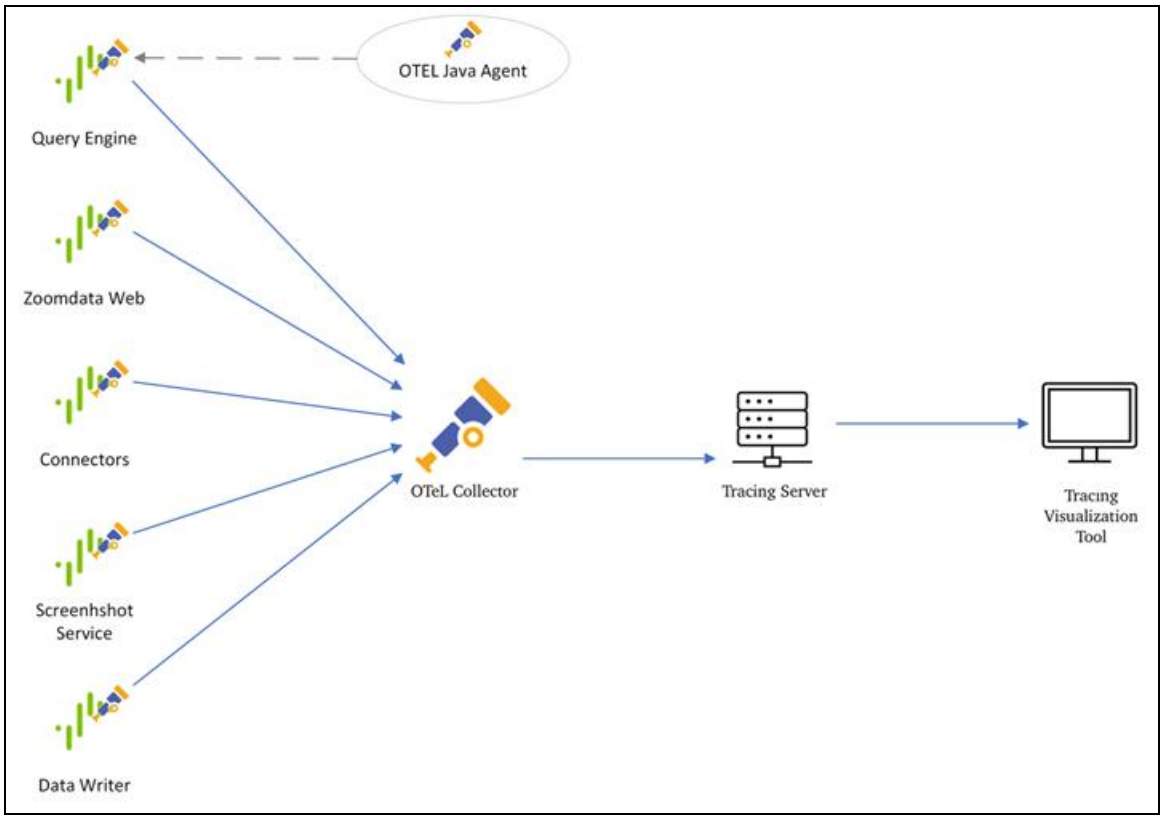
Composer integrates with any tracing server that supports OpenTelemetry (OTel) specifications and its export protocols (otlp, zipkin, jeager).

After you have installed a tracing server, install the otel collector.

Install and Configure OTel Collector

Install an OTel collector to act as a proxy between Composer microservices and your tracing server, both to simplify configuration and improve security by passing information through the proxy. See https://opentelemetry.io/docs/collector/ and https://opentelemetry.io/docs/collector/getting-started/.

After installation, configure your collector or collectors to export tracing information to the tracing server.



Configure the Collector

Provide the appropriate configuration information for your collector. The configuration information consists of three main sections, receivers, processors, and exporters. See https://opentelemetry.io/docs/collector/configuration/.

```
receivers:
       otlp:
               protocols:
                       grpc:

processors:
       batch:

exporters:
```

```
        otlp:
                endpoint: zoomdata-tempo:4317
                tls:
                        insecure: true

service:
        pipelines:
                traces:
                        receivers: [otlp]
                        processors: [batch]
                        exporters: [otlp]
```

- **Receivers**: Configure available protocols and endpoints for the receiver to receive tracing information from Composer's microservices. insightsoftware recommends you use the OTlp protocol.
- **Processor**: Enable additional processing of tracing information. In this example, batch processing is enabled to optimize network communication.
- **Exporters**: Configure export information. Provide a destination URL and other connection parameters to communicate with the tracing server. insightsoftware recommends you use the OTlp protocol.

After you have installed and configured the OTel collector, install and configure a java agent and Composer's microservices.

Configure Composer Microservice Tracing

You must install and configure the OTel java agent to trace Composer microservices. Optionally, next configure the Composer front end for tracing. After these steps are complete and you've restarted the microservices, you can extract the traceID in a REST request or from a web socket outbound message.

- Install The Java Agent
- Configure The Collector
- Configure The Composer Front End
- Extract The TraceID

Important: insightsoftware recommends you use the OTel java agent.

Install the Java Agent

1. Download the java agent to the server running the Composer microservice. https://github.com/open-telemetry/opentelemetry-java-instrumentation/releases/latest/download/opentelemetry-javaagent.jar

2. Define the path to the java agent for Composer. Select the option that works best for your environment.

   a. Specify the path in `SERVICE_NAME.jvm` (such as `zoomdata.jvm` or `edc-postgresql.jvm`) in the `/opt/zoomdata/conf/` folder, such as `-javaagent:/path/to/java/agent/opentelemetry-javaagent.jar`. See ComposerSymphony Data Discovery Microservice Name Reference.

   b. Define an environment variable, `_JAVA_OPTS`, that contains the path, such as `_JAVA_OPTIONS="-javaagent:/path/to/java/agent/opentelemetry-javaagent.jar"`.

Note: Some application performance monitoring tools provide their own java agents that are built on top of the OpenTelemetry java agent, or is compliant with the OTLP protocol. Use at your own discretion; the behavor may be different from the OTel java agent.

Configure the Java Agent

After you've installed the java agent, specify backend configuration properties for the agent using required and optional configuration properties. Include in the SERVICE_NAME.jvm or use environment variables. See https://opentelemetry.io/docs/instrumentation/java/automatic/agent-config/ and https://opentelemetry.io/docs/reference/specification/sdk-environment-variables/.

Required Configuration Properties

Property	Recommended Value	Description
OTEL_TRACES_EXPORTER	otlp	Defines the protocol to use to export tracing information.
OTEL_METRICS_EXPORTER	none	Disable metrics export using opentelemetry.
OTEL_LOGS_EXPORTER	none	Disable logs export using opentelemetry.
OTEL_EXPORTER_OTLP_TRACES_ENDPOINT	http://OTEL_COLLECTOR_HOST:4317	Defines the URL for the otel collector or tracing server.
OTEL_SERVICE_NAME	▪ zoomdata ▪ query-engine ▪ edc-postgresql	Define the names of the services.

Property	Recommended Value	Description
	■ edc-mssql ■ screenshot-service ■ data-writer	
OTEL_INSTRUMENTATION_SPRING_BOOT_ACTUATOR_AUTOCONFIGURE_ENABLED	false	Required, for now, to enable tracing. This may be updated after future otel releases.

Configure the Composer Front End

After you've provided back end configuration properties for tracking, provide any needed variables for the Composer front end as well.

Enable tracing and provide environment variables by specifying the fe.env configuration property for the zoomdata-web microservice using the following format: `fe.env=ENV_VARIABLE1=value1;ENV_VARIABLE2=value2;ENV_VARIABLE3=value3`.

The default value of `OTEL_SDK_DISABLED` is `false` in the OpenTelemetry SDK. When used in your Composer environment, installation changes this value to `true` for front end support. To enable tracing, specify `OTEL_SDK_DISABLED=false` in the environment variable.

Required Configuration Properties

Property	Recommended Value	Description
OTEL_TRACES_EXPORTER	otlp	Defines the protocol to use to export tracing information.
OTEL_METRICS_EXPORTER	none	Disable metrics export using opentelemetry.
OTEL_LOGS_EXPORTER	none	Disable logs export using opentelemetry.
OTEL_EXPORTER_OTLP_TRACES_ENDPOINT	http://OTEL_COLLECTOR_HOST:4317	Defines the URL for the otel collector or tracing server.
OTEL_SERVICE_NAME	■ zoomdata ■ query-engine ■ edc-postgresql	Define the names of the services.

Property	Recommended Value	Description
	■ edc-mssql ■ screenshot-service ■ data-writer	
OTEL_INSTRUMENTATION_SPRING_BOOT_ACTUATOR_AUTOCONFIGURE_ENABLED	false	Required, for now, to enable tracing. This may be updated after future otel releases.

OpenTelemetry is executed in the user's browser and all tracing information is sent to the collector from the browser. The `OTEL_EXPORTER_OTLP_TRACES_ENDPOINT` must be reachable by the browser to execute tracing collection. Depending on your environment, you may need to configure cors.

After you have installed and configured the java agent, restart all microservices.

Extract the traceID

When tracing is configured and enabled, you can extract the `traceID` in a REST request or from a web socket outbound message. `traceID` is embedded in the `traceparent` property. The structure is `traceparent` is `{version}-{traceId}-{spanId}-{sampleDecision}`.