



TIBCO® Offer and Price Engine

Installation and Configuration

Version 6.0.0
March 2023



Contents

| | |
|--|-----------|
| Contents | 2 |
| About This Product | 5 |
| Installation | 6 |
| Required Third-Party Jars | 7 |
| Postinstallation Tasks | 9 |
| Task 1: Copying Dependencies | 10 |
| Task 2: Creating the Database | 10 |
| Creating a PostgreSQL Database for the Admin User | 11 |
| Creating an Oracle Database for the Admin User | 12 |
| Creating a PostgreSQL Database for the Catalog User | 14 |
| Creating an Oracle Database for the Catalog User | 15 |
| Creating a PostgreSQL Database for the Shopping Cart User | 16 |
| Creating an Oracle Database for the Shopping Cart User | 18 |
| Task 3: Creating the TIBCO Enterprise Message Service Channel | 20 |
| Task 4: Configuring and Starting Authorization Service | 20 |
| Task 5: Creating Mandatory Users | 23 |
| Task 6: Configuring and Starting Configurator Service | 24 |
| Task 7: Configuring and Starting Configurator UI | 26 |
| Task 8: Uploading App Properties, Config Files through Configurator UI | 28 |
| Task 9: Configuring minimum requirements through Configurator UI | 33 |
| Task 10: Starting or Restarting the Services | 45 |
| Automating postinstallation configurations | 47 |
| Docker | 49 |
| Building a Docker Image Without an Internet Connection | 49 |
| Copying Files to Docker Context | 50 |

| | |
|--|-----------|
| Build EMS Client Image | 50 |
| EMS Setup | 51 |
| Setting Up the .env File | 52 |
| Building Docker Images | 52 |
| Preparing Docker Volumes | 53 |
| Running the Docker Containers | 54 |
| Extend Docker-Compose Files | 56 |
| Modifying a Container Time-Zone | 57 |
| Reading Container Logs | 59 |
| Troubleshooting Error from Building Docker Images | 59 |
| Deploying Individual TIBCO OPE services using Kubernetes scripts | 61 |
| TLS Support for Redis | 62 |
| Connecting to Redis through user credentials | 62 |
| Using OpenSSL or TLS | 62 |
| Red Hat Universal Base Image Implementation | 64 |
| Verifying Installation | 65 |
| Managing Health and Liveness Endpoints | 65 |
| Managing Health Endpoint | 65 |
| Managing Ready Endpoint | 65 |
| Hawkular Monitoring Support | 66 |
| Red Hat Openshift Deployment | 69 |
| TIBCO OPE Resource Allocation | 70 |
| Resource allocation for TIBCO OPE services on a cloud platform | 70 |
| Resource allocation of JVM for TIBCO OPE services on a non-cloud platform (baremetal box or VM) | 71 |
| Installing Helm Chart | 72 |
| External Property Enhancement | 73 |

| | |
|--|-----------|
| Migrating to TIBCO Offer and Price Engine 6.0.0 | 75 |
| Migrating from TIBCO OPE 5.1.0 HF-4 to TIBCO OPE 6.0.0 | 76 |
| Configuration | 77 |
| Application Properties | 77 |
| External Application Logging Configuration | 80 |
| Troubleshooting | 81 |
| TIBCO Documentation and Support Services | 84 |
| Legal and Third-Party Notices | 86 |

About This Product

TIBCO® Offer and Price Engine is a cloud-native, in-memory omnichannel server of offers and prices for digital service providers. It answers requests from a digital service provider's customer-facing channels for offers and prices, subject to business rules such as customer eligibility and product compatibility.

TIBCO Offer and Price Engine is the next generation of, and partially replaces, TIBCO® Fulfillment Order Management. To better align the direction of TIBCO Fulfillment Order Management with market demand, the product's capabilities have been reorganized into two new products:

- TIBCO® Order Management
- TIBCO® Offer and Price Engine

Customers who are currently on maintenance for TIBCO Fulfillment Order Management are entitled to upgrade to both TIBCO Order Management and TIBCO Offer and Price Engine. TIBCO will continue to support TIBCO Fulfillment Order Management, and there is currently no plan to retire TIBCO Fulfillment Order Management. New capabilities will be developed in TIBCO Order Management and TIBCO Offer and Price Engine.

Installation

Requirements

All Offer and Price Engine microservices configurations must be configured externally.

Downloading and Installing TIBCO Offer and Price Engine

Download the TIBCO Offer and Price Engine 6.0.0 build from [TIBCO eDelivery](#). The following project artifacts are downloaded in the OPE_HOME folder.

- TIB_ope_6.0.0.zip - extract to the OPE folder
- jdk-11.0.4_linux-x64_bin.tar.gz

The following components are present in the TIB_ope_6.0.0.zip file.

- database-scripts - Contains Oracle and PostgreSQL database scripts.
- ems - Contains files related to the creation or deletion of queues or topics.
- externalLib - Here you need to copy all the [external jars](#) required for all microservices. You also need to copy the tibjms.jar and tibcrypt.jar files for TIBCO Enterprise Message Service™ and ojdbc11.jar for Oracle.
- roles - Contains all the configuration files for all the microservices.
- Libs - Contains the opeClient.jar, which is available for Java customization changes.
- Samples - Sample models, web service requests, and response files.
- Specs - WSDL specification files.
- Helm - Contains all files required to deploy the authorization service and all other OPE services Helm charts.
- Kubernetes - Contains Kubernetes scripts for all OPE services.
- config-automation - Contains python scripts for automating postinstallation configurations.
- [Docker](#) - Contains files and artifacts to create and run Docker containers for all OPE microservices.

i Note: For a complete list of versions and supported platforms, see the TIB_ope_6.0.0_readme.txt file.

Required Third-Party Jars

The following table lists the required third-party jars:

| Jars | Download link |
|---|---|
| antlr-2.7.7 | https://repo1.maven.org/maven2/antlr/antlr/2.7.7/antlr-2.7.7.jar |
| dom4j-2.1.3 | https://repo1.maven.org/maven2/org/dom4j/dom4j/2.1.3/dom4j-2.1.3.jar |
| jms-2.0 | Copy from <EMS_HOME>/lib directory |
| tibjms.jar | Copy from <EMS_HOME>/lib directory |
| javassist-3.21.0-GA | https://repo1.maven.org/maven2/org/javassist/javassist/3.21.0-GA/javassist-3.21.0-GA.jar |
| hibernate-commons-annotations-5.1.2.Final | https://repo1.maven.org/maven2/org/hibernate/common/hibernate-commons-annotations/5.1.2.Final/hibernate-commons-annotations-5.1.2.Final.jar |
| hibernate-core-5.6.7.Final | https://repo1.maven.org/maven2/org/hibernate/hibernate-core/5.6.7.Final/hibernate-core-5.6.7.Final.jar |
| hibernate-entitymanager-5.6.7.Final | https://repo1.maven.org/maven2/org/hibernate/hibernate-entitymanager/5.6.7.Final/hibernate-entitymanager-5.6.7.Final.jar |
| hibernate-jpa-2.1-api-1.0.2.Final | https://repo1.maven.org/maven2/org/hibernate/javax/persistence/hibernate-jpa-2.1-api/1.0.2.Final/hibernate-jpa-2.1-api-1.0.2.Final.jar |

| Jars | Download link |
|---------------------------------|---|
| hibernate-validator-6.2.3.Final | https://repo1.maven.org/maven2/org/hibernate/validator/hibernate-validator/6.2.3.Final/hibernate-validator-6.2.3.Final.jar |
| ojdbc11 | https://repo1.maven.org/maven2/com/oracle/database/jdbc/ojdbc11/21.1.0.0/ojdbc11-21.1.0.0.jar |
| jboss-logging-3.4.3.Final | https://repo1.maven.org/maven2/org/jboss/logging/jboss-logging/3.4.3.Final/jboss-logging-3.4.3.Final.jar |



Note: When you obtain third-party software or services, it is your responsibility to ensure you understand the license terms associated with such third-party software or services and comply with such terms.

Postinstallation Tasks

This section gives an overview of the steps you have to perform in the given order on the successful installation of TIBCO Offer and Price Engine.

Recommended Environment Variables Setup

It is a good practice to set the following environment variables where TIBCO Offer and Price Engine is installed:

| Environment Variable | Value |
|----------------------|--|
| OPE_HOME | \$TIBCO_HOME/tibco/ope/6.0 |
| AF_CONFIG_HOME | \$OPE_HOME/roles/ope/standalone/config |
| EXTERNAL_LIB_PATH | \$OPE_HOME/externalLib |

The post-installation tasks can be carried out manually or through automation.

- To configure the installed components manually, complete the following steps:
 1. [Task 1: Copying Dependencies](#)
 2. [Task 2: Creating the Database](#)
 3. [Task 3: Creating the TIBCO Enterprise Message Service Channel](#)
 4. [Task 4: Configuring and Starting Authorization Service](#)
 5. [Task 5: Creating Mandatory Users](#)
 6. [Task 6: Configuring and Starting Configurator Service](#)
 7. [Task 7: Configuring and Starting Configurator UI](#)
 8. [Task 8: Uploading App Properties, Config Files through Configurator UI](#)
 9. [Task 9: Configuring minimum requirements through Configurator UI](#)
 10. [Task 10: Starting or Restarting the Services](#)
- To configure the installed components through automation, See [Automating](#)

[postinstallation configurations.](#)

i Note: Use the HTML documentation to copy code snippets to the XML files.

Task 1: Copying Dependencies

Offer and Price Engine does not include all the required third-party dependencies. A consolidated copyLib.sh script is provided at `$OPE_HOME/roles` for all the components.

Procedure

1. Copy the required [external jars](#) to the `$OPE_HOME/externalLib` directory.
2. Copy the `tibjms.jar` and `jms-2.0.jar` files for TIBCO Enterprise Message Service™ and `ojdbc11.jar` file for Oracle to the `$OPE_HOME/externalLib` directory.
3. Run the `$OPE_HOME/roles/copyLib.sh` script.

Task 2: Creating the Database

If this is your first time installing TIBCO Offer and Price Engine, then create the database by running the provided scripts.

i Note: To run the SQL scripts, you must have appropriate permission.

TIBCO Offer and Price Engine supports the PostgreSQL and Oracle databases. Depending on the database you use, choose the following tasks.

- PostgreSQL database
 - [Creating a PostgreSQL Database for the Admin User](#)
 - [Creating a PostgreSQL Database for the Catalog User](#)
 - [Creating a PostgreSQL Database for the Shopping Cart User](#)
- Oracle database

- [Creating an Oracle Database for the Admin User](#)
- [Creating an Oracle Database for the Catalog User](#)
- [Creating an Oracle Database for the Shopping Cart User](#)

Creating a PostgreSQL Database for the Admin User

Procedure

1. Open `$OPE_HOME/database-scripts/postgreSQL/admin/bin/postgres_admin_db.properties` file in a suitable editor and update the following values:

| Property | Update the Value for |
|------------------------------|-------------------------------|
| PG_HOME | PostgreSQL database home |
| PG_HOST | PostgreSQL database host |
| PG_PORT | PostgreSQL port |
| PG_SUPER_USER_NAME | PostgreSQL superuser name |
| PG_SUPER_USER_PASSWORD | PostgreSQL superuser password |
| PG_SUPER_USER_DATABASE | PostgreSQL superuser database |
| PG_ADMIN_USER | New admin user |
| PG_ADMIN_PASSWORD | Password for the admin user |
| PG_ADMIN_DATABASE | admin database name |
| PG_ADMIN_SCHEMA | admin schema |
| PG_ADMIN_TABLESPACE_LOCATION | admin tablespace location |

| Property | Update the Value for |
|---------------------------|--|
| PG_ADMIN_TABLESPACE | admin tablespace |
| IS_CLOUD_PLATFORM | Default value is false. You need to set it to true if the database is on a cloud platform. |
| EXECUTE_DDL_DML_ONLY | Default value is false. You need to set it to true if you want to execute only DDL/DML. |
| BASE_INSTALLATION_SCRIPTS | database_ddl.sql,proc_app_properties_change.sql,proc_get_timestamp.sql |
| 5.1.0_TO_6.0.0HF0_SCRIPTS | upgrade_5.1.0_to_6.0_ddl.sql,proc_app_properties_change.sql,proc_get_timestamp.sql |

2. Save and close the file.
3. Run the `$OPE_HOME/database-scripts/postgreSQL/admin/bin/db-setup.sh` script.

Creating an Oracle Database for the Admin User

Procedure

1. Open `$OPE_HOME/database-scripts/oracle/admin/oracle_admin_db.properties` file in a suitable editor and update the following values:

| Property | Update the Value for |
|---------------|---------------------------|
| ORCL_HOST | Oracle database host |
| ORCL_PORT | Oracle port |
| ORCL_USERNAME | Oracle superuser name |
| ORCL_PASSWORD | Oracle superuser password |

| Property | Update the Value for |
|-----------------------------------|---|
| ORCL_ SERVICENAME | Service name of the database |
| ORCL_ADMIN_USER | New admin user |
| ORCL_ADMIN_ PASSWORD | Password for the admin user |
| ORCL_ADMIN_ TABLESPACE | Admin tablespace |
| ORCL_ TABLESPACE_SIZE | Size of the tablespace |
| ORCL_MINSIZE | Minimum tablespace size |
| ORCL_MAXSIZE | Maximum tablespace size |
| ORCL_DATAFILE_ PATH | data file path of Oracle |
| IS_CLOUD_ PLATFORM | Default value is false. You need to set it to true if the database is on a cloud platform. |
| EXECUTE_DDL_ DML_ONLY | Default value is false. You need to set it to true if you want to execute only DDL/DML. |
| BASE_ INSTALLATION_ SCRIPTS | database_ddl.sql,proc_app_properties_change.sql:true,proc_get_timestamp.sql:true,blobConvertorFunction.sql:true |
| 5.1.0_TO_ 6.0.0HF0_ SCRIPTS | blobConvertorFunction.sql:true,proc_app_properties_change.sql:true,proc_get_timestamp.sql:true |

2. Save and close the file.
3. Run the \$OPE_HOME/database-scripts/oracle/admin/db-setup.sh script.

Creating a PostgreSQL Database for the Catalog User

Procedure

1. Open `$OPE_HOME/database-scripts/postgreSQL/catalog/bin/postgres_catalog_db.properties` file in a suitable editor and update the following values:

| Property | Update the Value for |
|--------------------------------|--|
| PG_HOME | PostgreSQL database home |
| PG_HOST | PostgreSQL database host |
| PG_PORT | PostgreSQL port |
| PG_SUPER_USER_NAME | PostgreSQL superuser name |
| PG_SUPER_USER_PASSWORD | PostgreSQL superuser password |
| PG_SUPER_USER_DATABASE | PostgreSQL superuser database |
| PG_CATALOG_USER | New Catalog user |
| PG_CATALOG_PASSWORD | Password for the Catalog user |
| PG_CATALOG_DATABASE | Catalog database name |
| PG_CATALOG_SCHEMA | Catalog schema |
| PG_CATALOG_TABLESPACE | Catalog tablespace |
| PG_CATALOG_TABLESPACE_LOCATION | Catalog tablespace location |
| IS_CLOUD_PLATFORM | Default value is false. You need to set it to true if the database is on a cloud platform. |

| Property | Update the Value for |
|---------------------------|---|
| EXECUTE_DDL_DML_ONLY | Default value is false. You need to set it to true if you want to execute only DDL/DML. |
| BASE_INSTALLATION_SCRIPTS | catalog_ds_ddl.sql |
| 5.1.0_TO_6.0.0HF0_SCRIPTS | upgrade_5.1.0_to_6.0_ddl.sql |

2. Save and close the file.
3. Run the `$OPE_HOME/database-scripts/postgreSQL/catalog/bin/db-setup.sh` script.

Creating an Oracle Database for the Catalog User

Procedure

1. Open `$OPE_HOME/database-scripts/oracle/catalog/oracle_admin_db.properties` file in a suitable editor and update the following values:

| Property | Update the Value for |
|-------------------|------------------------------|
| ORCL_HOST | Oracle database host |
| ORCL_PORT | Oracle port |
| ORCL_USERNAME | Oracle superuser name |
| ORCL_PASSWORD | Oracle superuser password |
| ORCL_SERVICENAME | Service name of the database |
| ORCL_CATALOG_USER | New Catalog user |

| Property | Update the Value for |
|---------------------------|--|
| ORCL_CATALOG_PASSWORD | Password for the Catalog user |
| ORCL_CATALOG_TABLESPACE | Catalog tablespace |
| ORCL_TABLESPACE_SIZE | Size of the tablespace |
| ORCL_MINSIZE | Minimum tablespace size |
| ORCL_MAXSIZE | Maximum tablespace size |
| ORCL_DATAFILE_PATH | data file path of Oracle |
| IS_CLOUD_PLATFORM | Default value is false. You need to set it to true if the database is on a cloud platform. |
| EXECUTE_DDL_DML_ONLY | Default value is false. You need to set it to true if you want to execute only DDL/DML. |
| BASE_INSTALLATION_SCRIPTS | OMS_DDL_CATALOG.sql |

2. Save and close the file.
3. Run the `$OPE_HOME/database-scripts/oracle/catalog/db-setup.sh` script.

Creating a PostgreSQL Database for the Shopping Cart User

Procedure

1. Open `$OPE_HOME/database-scripts/postgreSQL/shoppingCart/bin/postgres_shoppingcart_db.properties` file in a suitable editor and update the following values:

| Property | | Update the Value for |
|-------------------------------------|--|--|
| PG_HOME | | PostgreSQL database home |
| PG_HOST | | PostgreSQL database host |
| PG_PORT | | PostgreSQL port |
| PG_SUPER_USERNAME | | PostgreSQL superuser name |
| PG_SUPER_PASSWORD | | PostgreSQL superuser password |
| PG_SUPER_USER_DATABASE | | PostgreSQL superuser database |
| PG_SHOPPINGCART_USER | | New shopping cart user |
| PG_SHOPPINGCART_PASSWORD | | Password for the shopping cart user |
| PG_SHOPPINGCART_DATABASE | | Shopping cart database name |
| PG_SHOPPINGCART_SCHEMA | | Shopping cart schema |
| PG_SHOPPINGCART_TABLESPACE | | Shopping cart tablespace |
| PG_SHOPPINGCART_TABLESPACE_LOCATION | | Shopping cart tablespace location |
| IS_CLOUD_PLATFORM | | Default value is false. You need to set it to true if the database is on a cloud platform. |

| Property | Update the Value for |
|---------------------------|---|
| EXECUTE_DDL_DML_ONLY | Default value is false. You need to set it to true if you want to execute only DDL/DML. |
| BASE_INSTALLATION_SCRIPTS | shoppingcart_ds_ddl.sql |

2. Save and close the file.
3. Run the `$OPE_HOME/database-scripts/postgreSQL/shoppingCart/bin/db-setup.sh` script.

Creating an Oracle Database for the Shopping Cart User

Procedure

1. Open `$OPE_HOME/database-scripts/oracle/shoppingCart/bin/oracle_shoppingcart_db.properties` file in a suitable editor and update the following values:

| Property | Update the Value for |
|------------------|------------------------------|
| ORCL_HOST | Oracle database host |
| ORCL_PORT | Oracle port |
| ORCL_USERNAME | Oracle superuser name |
| ORCL_PASSWORD | Oracle superuser password |
| ORCL_SERVICENAME | Service name of the database |

| Property | Update the Value for |
|---------------------------------------|--|
| ORCL_SHOPPINGCART_USER | New shopping cart user |
| ORCL_SHOPPINGCART_PASSWORD | Password for the shopping cart user |
| ORCL_SHOPPINGCART_TABLESPACE | shopping cart tablespace |
| ORCL_SHOPPINGCART_TABLESPACE_LOCATION | shopping cart tablespace location |
| ORCL_MINSIZE | Minimum tablespace size |
| ORCL_MAXSIZE | Maximum tablespace size |
| ORCL_DATAFILE_PATH | data file path of Oracle |
| IS_CLOUD_PLATFORM | Default value is false. You need to set it to true if the database is on a cloud platform. |
| EXECUTE_DDL_DML_ONLY | Default value is false. You need to set it to true if you want to execute only DDL/DML. |
| BASE_INSTALLATION_SCRIPTS | SHOPPINGCART_DDL.sql |

2. Save and close the file.
3. Run the `$OPE_HOME/database-scripts/oracle/shoppingCart/bin/db-setup.sh` script.

Task 3: Creating the TIBCO Enterprise Message Service Channel

To create the TIBCO Enterprise Message Service channels, run the `tibemsadmin` command.

1. Go to `$EMS_HOME/bin` and run the following command:

```
$ tibemsadmin -server tcp://localhost:7222 -user admin -script  
$OPE_HOME/ems/CreateEMSChannel.txt
```

Task 4: Configuring and Starting Authorization Service

Procedure

1. Before starting the Authorization service, configure the application properties from the following table in `$OPE_HOME/roles/authorization-service/standalone/config/application.properties` file. To update the properties, you can refer to the sample file present under `$OPE_HOME/samples/authorization-services` directory for Oracle and PostgreSQL as per your requirement. The default values are set for PostgreSQL.

| Category | Element | Default Value |
|---|---------------------------|--|
| General | server.port | 9091 |
| | amPluggableCache | Relational |
| | default.tenant.id | TIBCO |
| | auth.superuser.appId | auth |
| | auth.superuser.appKey | ENC(P2yXphz4OVM=) |
| | allowedUserRoles | ROLE_ADMIN,ROLE_USER |
| Relational Database Connection Properties | datasourceDriverClassName | org.postgresql.Driver |
| | adminDsUrl | jdbc:postgresql://localhost:5432/admindbll?currentSchema=adminschema |
| | adminDsUsername | adminuserll |
| | adminDsPassword | ENC(O4UrXXgTEmyecFyHLo+lvw==) |
| | hibernateDialect | org.hibernate.dialect.PostgreSQLDialect |
| | hibernateDsDefaults | false |
| | adminHibernateShowSql | false |

Note: It is a good practice to change this default value and set your own key in an encrypted value. Refer 'Encrypt Password Utility' section in *TIBCO® Offer and Price Engine Security Guidelines*.

| Category | Element | Default Value |
|----------------------------------|---|----------------------|
| | adminDsInitialSize= | 10 |
| | adminDsMaxWait | 30000 |
| | adminDsMaxActive | 100 |
| | adminDsMaxIdle | 100 |
| | adminDsMinIdle | 10 |
| | datasourceValidationQuery | SELECT 1 |
| | adminDsTestOnBorrow | true |
| | adminDsValidationInterval | 5000 |
| Directory Service Configurations | directoryServiceDomainName | testad.com |
| | directoryServiceRootDistinguishedName | DC=testad,DC=com |
| | ldapURLForDirectoryService | ldap://localhost:389 |
| Actuator Endpoints Properties | management.endpoints.web.exposure.include | health,ready,loggers |

| Category | Element | Default Value |
|--|--------------------------------------|---|
| Authentication Token Generation Configuration | authentication.token.signing.key | ENC(nSa0k6lmjPPN8ZA5SO6BpQ==) |
| | authorization.access.token.validity | 43200 |
| | authorization.refresh.token.validity | 2592000 |
| | authorized.client.id | order-management-client |
| | authorized.client.secret | ENC(ggsmFvh5HBbeSD1j+l5Y0rP4qv0rJvEm) |
| | allowedCorsOrigins | http://localhost:9091,http://localhost:9090,http://localhost:9092,http://localhost:9094,http://localhost:9099,http://localhost:9095,http://localhost:9102,http://localhost:9100,http://localhost:9093,http://localhost:9089,http://localhost:9104,http://localhost:8090,http://localhost:8093,http://localhost:8090 |

2. Start the authorization service by running the `start.sh` script from the `$OPE_HOME/roles/authorization-service/standalone/bin` location.

Task 5: Creating Mandatory Users

Procedure

1. Create an Admin user with the required tenant Id and user roles as 'ROLE_ADMIN'. Refer 'Create User' section in *TIBCO® Offer and Price Engine User Guide*.
2. Create an apiUser with the tenant Id value as 'COMMON' and user roles as 'ROLE_ADMIN'.

Note: This apiUser is used for inter service communications only and it does not have access to any other functionalities.

Task 6: Configuring and Starting Configurator Service

Procedure

- Before starting the Configurator service, configure the application properties from the following table in \$OPE_HOME/roles/configurator/standalone/config/application.properties file. To update the properties, you can refer to the sample file present under \$OPE_HOME/samples directory for Oracle and PostgreSQL as per your requirement. The default values are set for PostgreSQL.

| Category | Element | Default Value |
|---------------------------------------|-------------------------|---|
| General | server.port | 9090 |
| | security.key | ENC(nSa0k6lmjPPN8ZA5SO6BpQ==) |
| | pluggableCache | Relational |
| | configuratorAccessRoles | ROLE_ADMIN |
| Relational Database Connection Proper | adminDsUrl | jdbc:postgresql://localhost:5432/admindbll?currentSchema=adminschema_v2 |

| Category | Element | Default Value |
|------------------------|---|---|
| ties | adminDsUsername | adminuserll |
| | adminDsPassword | ENC(O4UrXXgTEmyecFyHLo+lvw==) |
| | datasourceDriverClassName | org.postgresql.Driver |
| | hibernateDialect | org.hibernate.dialect.PostgreSQLDialect |
| | hibernateDsDefaults | false |
| | adminHibernateShowSql | false |
| | adminDsInitialSize= | 10 |
| | adminDsMaxWait | 30000 |
| | adminDsMaxActive | 100 |
| | adminDsMaxIdle | 100 |
| | adminDsMinIdle | 10 |
| | datasourceValidationQuery | SELECT 1 |
| | adminDsTestOnBorrow | true |
| | adminDsValidationInterval | 5000 |
| Actual or Endpoi | management.endpoints.web.exposure.include | health,refresh,loggers |

| Category | Element | Default Value |
|------------------------------------|--|-----------------------|
| nts Proper ties | management.endpoint.heath.show-details | ALWAYS |
| Notific ation Proper ties | notificationChannel | app_properties_events |
| | listenIntervalInMillis | 1000 |
| | purgeNotificationIntervalInMinutes | 15 |
| | purgeNotificationStartDelayInMinutes | 15 |
| | purgeNotificationOffsetInMinutes | 5 |

2. Start the authorization service by running the `start.sh` script from `$OPE_HOME/roles/configurator/standalone/bin` location.

Task 7: Configuring and Starting Configurator UI

Procedure

1. Before starting the Configurator UI, configure the application properties from the following table in `$OPE_HOME/roles/configurator-ui/standalone/config/application.properties` file. To update the properties, you can refer to the sample file present under `$OPE_HOME/samples/configurator-ui` directory for Oracle and PostgreSQL as per your requirement. The default values are set for PostgreSQL.

| Category | Element | Default Value |
|---------------------|--|---|
| General | server.port | 9104 |
| | configuratorServiceUrl | http://localhost:9090 |
| | configuratorServiceRetryCount | 5 |
| | configuratorServiceRetryDuration | 5 |
| | configuratorTrustStoreAbsolutePath | C:/Users/cacert |
| | configuratorTrustStorePassword | tibco123 |
| | configuratorTrustStoreType=jks security.key | ENC (nSa0k6lmjPPN8ZA5SO6BpQ==) |
| | authentication.token.signing.key | ENC (nSa0k6lmjPPN8ZA5SO6BpQ==) |
| | authorizationServiceTokenEndPoint | http://localhost:9091 |
| | authorization.client.id | order-management-client |
| | authorization.client.secret | ENC (ggsmFvh5HBbeSD1j+l5Y0rP4qv0rJvEm) |
| Actuator Properties | management.endpoints.web.exposure.include | health |

2. Start the authorization service by running the `start.sh` script from `$OPE_HOME/roles/configurator-ui/standalone/bin` location.

Task 8: Uploading App Properties, Config Files through Configurator UI

Once you start the Configurator UI, you can upload the metadata file, application properties, and configuration files through the UI.

Metadata: It defines the required application properties and configuration files for each application.

App properties: These are application properties in json format.

Config files: These are logback configuration files in XML format.

Logging in to Configurator UI

Procedure

1. To access the Login page, visit `http://<host>:<port number>`, where:
 - host is the computer where you have started Configurator UI service
 - port is the port number of the machine where Configurator UI listens for the requests. (**Default:** 9104)

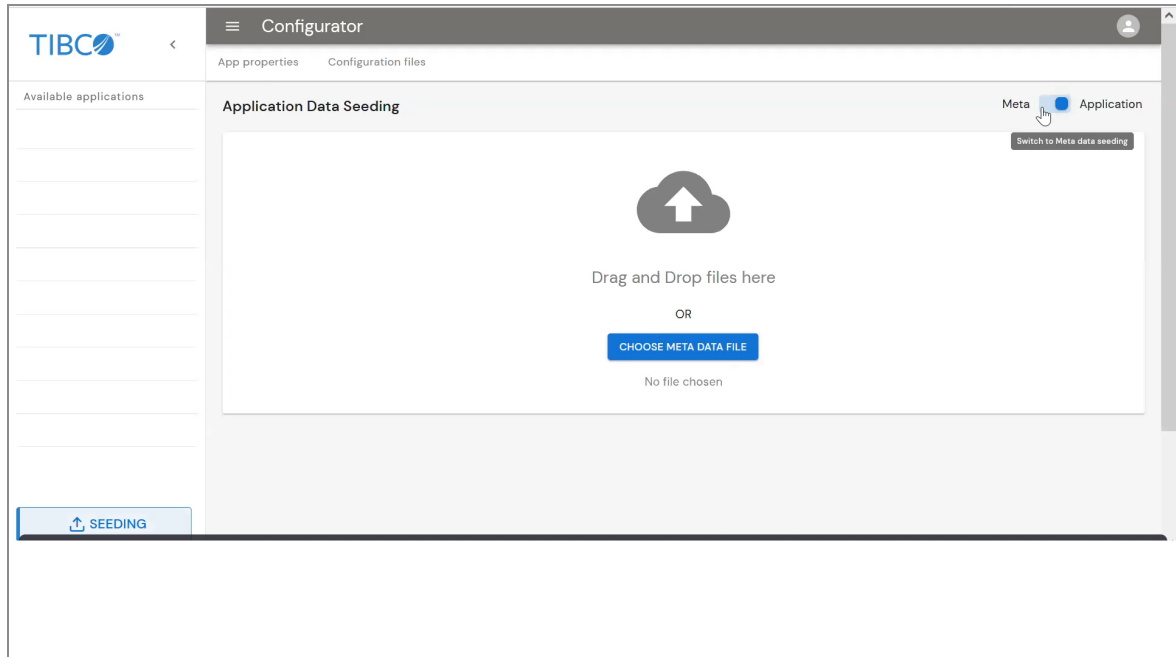
2. Enter the credentials in the **TenantID**, **Username**, and **Password** fields and then click **LOGIN**.

The UI dashboard opens.

Upload Metadata file

Procedure

1. Toggle the switch to select **Meta** and click **SEEDING** to upload the metadata file.

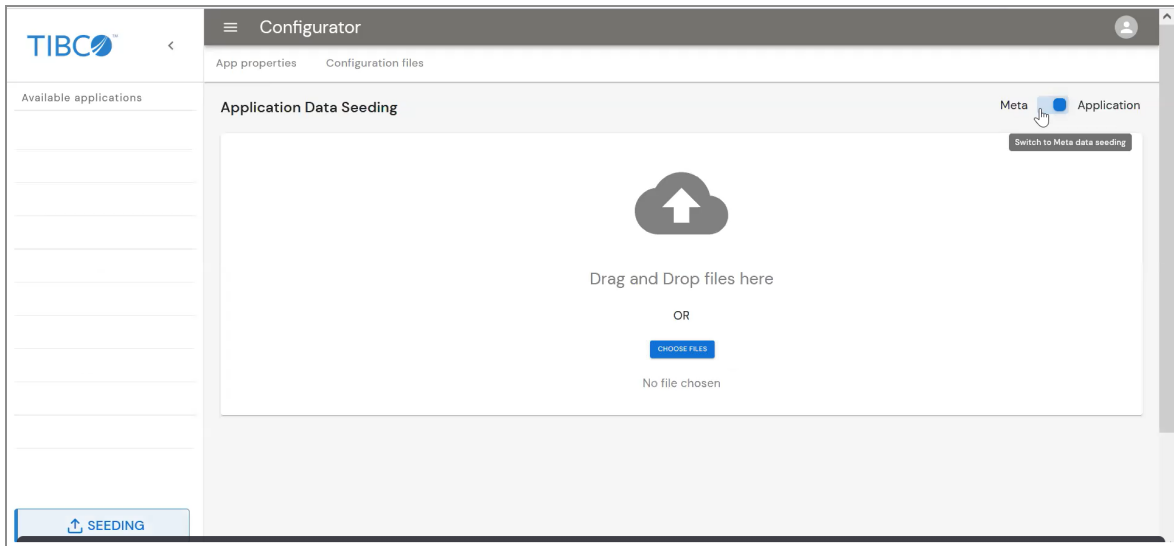


2. Use the drag-and-drop function or click **CHOOSE META DATA FILE** to select the application_metadata.json file from the \$OPE_HOME/seed-data directory.
3. Finally, click **UPLOAD** to upload the file.

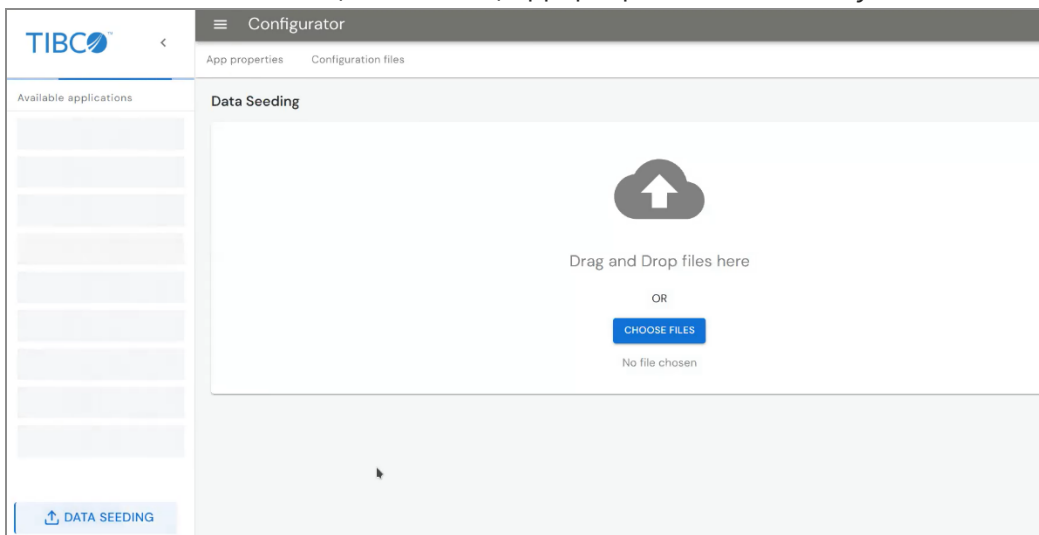
Upload Application Properties

Procedure

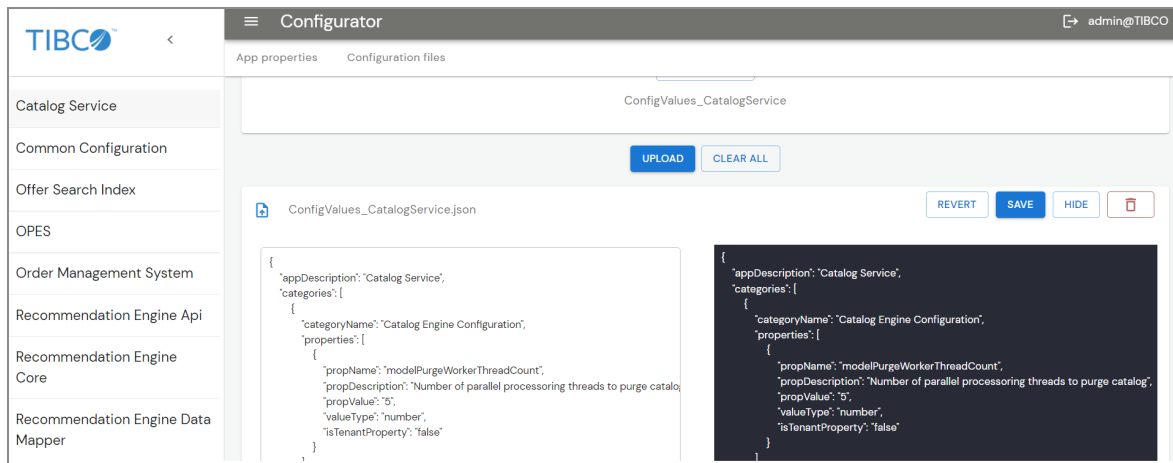
1. Toggle the switch to select **Application** and click **SEEDING** to upload the app properties files.



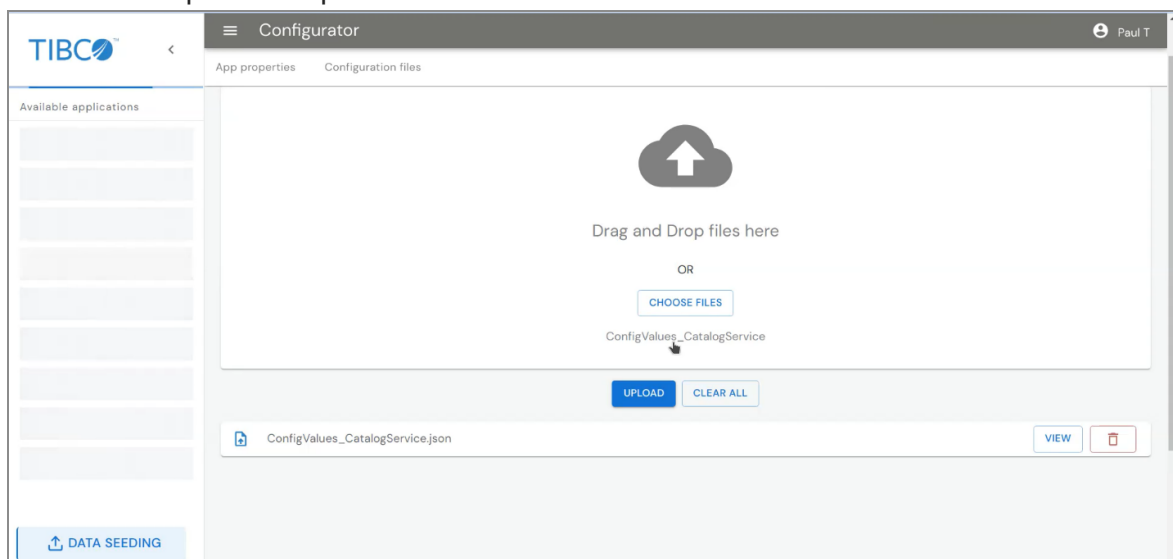
2. Use the drag-and-drop function or click **CHOOSE FILES** to select the app properties files from the `$OPE_HOME/seed-data/app-properties` directory.



3. To view an application property file, click **VIEW**. The file opens in text editor mode. The left pane is the editing pane and the right pane is the preview pane.



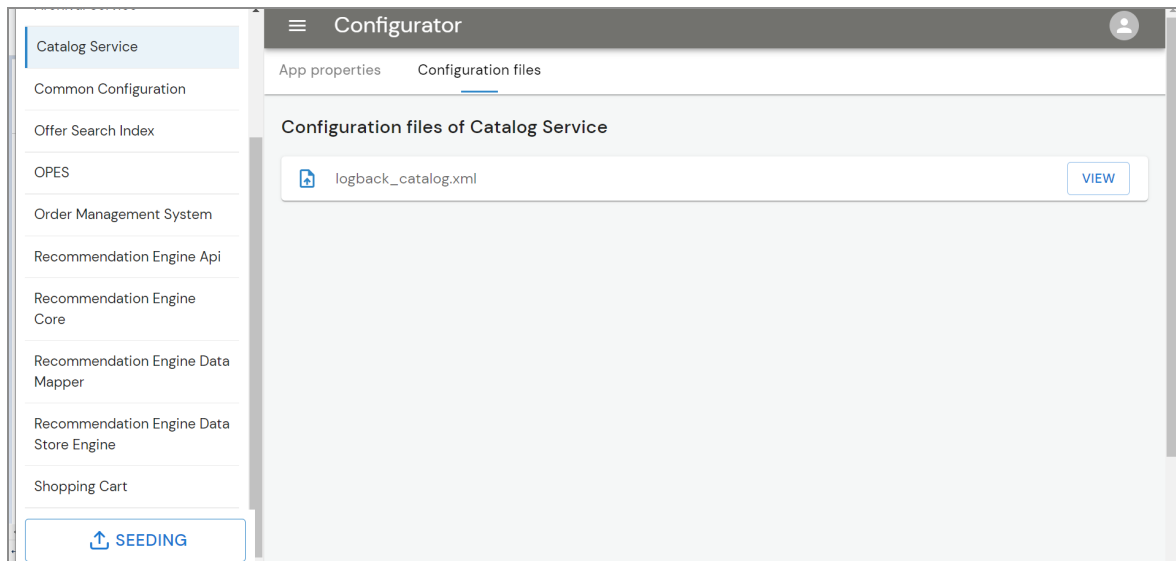
4. To discard or save the changes, click **DISCARD** or **SAVE**.
5. To hide or delete the files, click **HIDE** or **DELETE**.
6. Finally, click **UPLOAD** to upload the files. You can also use the **UPLOAD ALL** or **CLEAR ALL** option to upload or remove all the files at once.



Upload Configuration Files

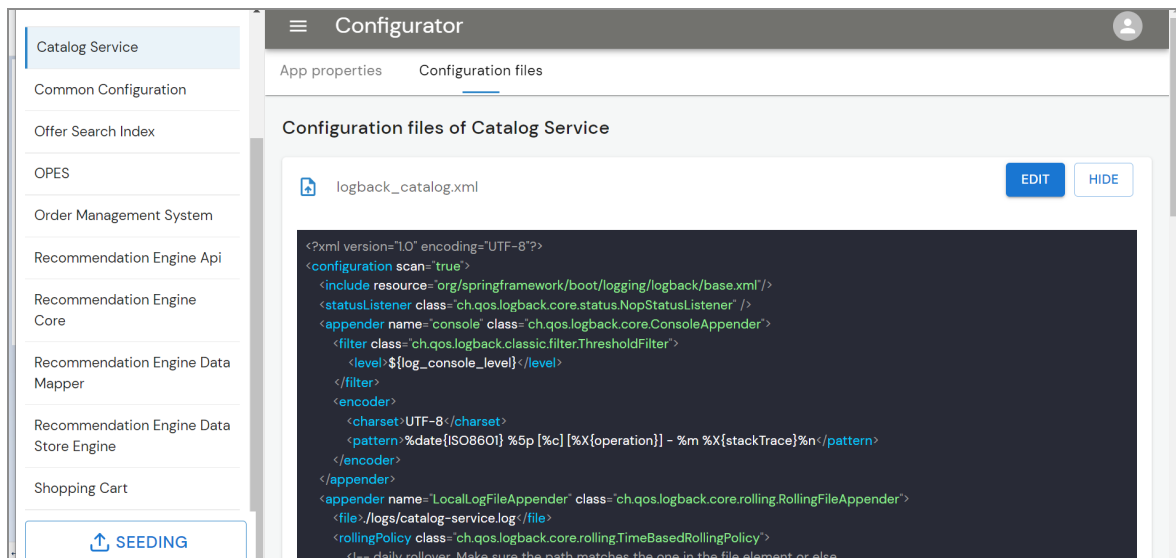
Procedure

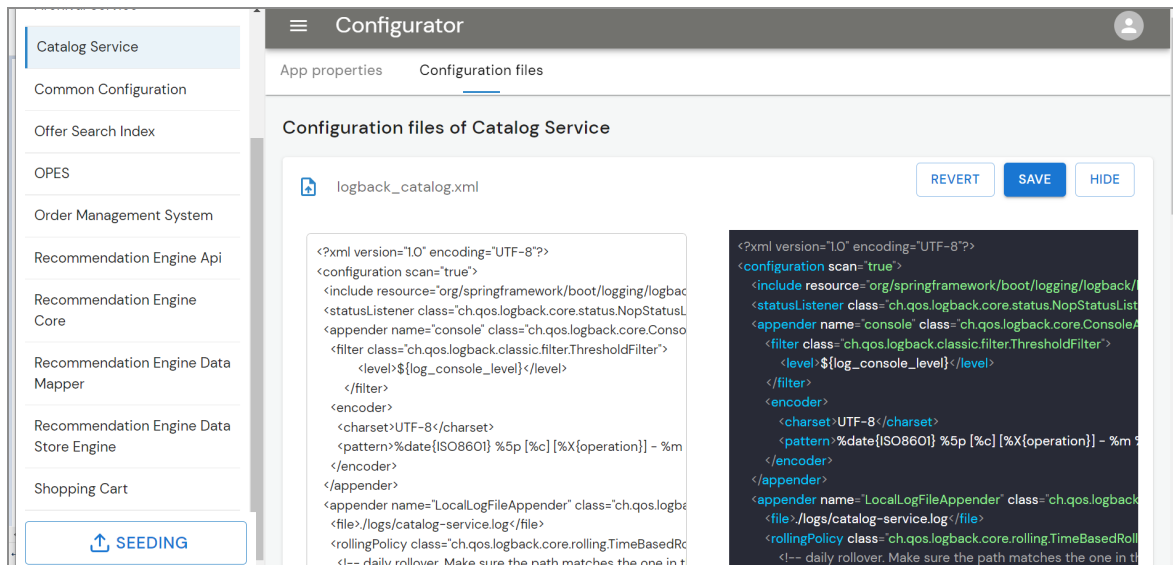
1. To add a configuration file from the `$OPE_HOME/seed-data/config-files` directory, go to the **Configuration files** tab and click **DATA SEEDING**.



2. To view a configuration file, click **VIEW** next to the respective configuration file.
3. To edit a configuration file, click **EDIT**.

The file opens in text editor mode. The left pane is the editing pane and the right pane is the preview pane.





4. To discard or save the changes, click **DISCARD** or **SAVE**.
5. Click **UPLOAD ALL** to upload all the config files.

Note: On the **Configuration files** tab, you can verify all the config files are present or not and accordingly you can add the missing files.

Task 9: Configuring minimum requirements through Configurator UI

Once you upload the metadata file, application properties, and configuration files, you have to configure the minimum required files for each service through Configurator UI to make them ready to start.

Common Configuration

Procedure

1. On the Configurator UI, navigate to **Common Configuration > App properties** and select **Authorization Server Configuration Properties Used for Swagger UI** category.

2. Update the property value as per the following table:

| Property name | Value |
|-----------------------------------|--|
| authorizationServiceTokenEndPoint | IP address or the DNS name Example: http://localhost:9091 |

Catalog Service

Procedure

1. On the Configurator UI, navigate to **Catalog Service > App properties**.
 - a. Select **EMS Configurations for Online Catalog Publishing from PSC** category and update the properties values as per the following table:

| Property name | Value |
|-------------------|--|
| emsServerURL | IP address or the DNS name where EMS is running |
| emsServerUsername | User name of the EMS server |
| emsServerPassword | Password of the EMS server |

- b. Select **Catalog Relational Data Source Configuration** category and update the properties values as per the following table:

| Property name | Value |
|---------------------------|---|
| catalogDsUrl | Data source URL (Oracle or PostgreSQL) |
| catalogDsUsername | User name of the catalog data source |
| catalogDsPassword | Password of the catalog data source |
| datasourceDriverClassName | Class name of the data source driver (Oracle or PostgreSQL) |
| datasourceValidationQuery | SQL query that is be used to validate connections |

- c. Select **Generic Relational Data Source Configuration** category and update the properties value as per the following table:

| Property name | Value |
|------------------|---|
| databaseType | Type of the database (Oracle or PostgreSQL) |
| hibernateDialect | Hibernate dialect (Oracle or PostgreSQL) Example: For PostgreSQL, it is <code>org.hibernate.dialect.PostgreSQLDialect</code> |

Offer Search Index

Procedure

1. On the Configurator UI, navigate to **Offer Search Index > App properties**.
 - a. Select **EMS Configuration** category and update the properties values as per the following table:

| Property name | Value |
|-------------------|---|
| emsServerURL | IP address or the DNS name where EMS is running |
| emsServerUsername | User name of the EMS server |
| emsServerPassword | Password of the EMS server |

- b. Select **Catalog Data Source Configuration** category and update the properties values as per the following table:

| Property name | Value |
|-------------------|--|
| catalogDsUrl | Data source URL (Oracle or PostgreSQL) |
| catalogDsUsername | User name of the catalog data source |

| Property name | Value |
|---------------------------|---|
| catalogDsPassword | Password of the catalog data source |
| datasourceDriverClassName | Class name of the data source driver (Oracle or PostgreSQL) |
| datasourceValidationQuery | SQL query that is be used to validate connections |

OPES

Procedure

1. On the Configurator UI, navigate to **OPES > App properties**.
 - a. Select **EMS Configurations for Global Cache Clean** category and update the properties values as per the following table:

| Property name | Value |
|-------------------|---|
| emsServerURL | IP address or the DNS name where EMS is running |
| emsServerUsername | User name of the EMS server |

| Property name | Value |
|-------------------|----------------------------|
| emsServerPassword | Password of the EMS server |



Note: Ensure that you have updated `isOfferSearchIndexEnabled` property value to true from the catalog service.

- b. Select **Catalog Data Source Configuration** category and update the properties values as per the following table:

| Property name | Value |
|---------------------------|---|
| catalogDsUrl | Data source URL (Oracle or PostgreSQL) |
| catalogDsUsername | User name of the catalog data source |
| catalogDsPassword | Password of the catalog data source |
| datasourceDriverClassName | Class name of the data source driver (Oracle or PostgreSQL) |
| datasourceValidationQuery | SQL query that is be used to |

| Property name | Value |
|---------------|----------------------|
| | validate connections |

- c. Select **Persistence** category and update the properties value as per the following table:

| Property name | Value |
|------------------|---|
| databaseType | Type of the database (Oracle or PostgreSQL) |
| hibernateDialect | Hibernate dialect (Oracle or PostgreSQL) |

Shopping Cart

Procedure

1. On the Configurator UI, navigate to **Shopping Cart > App properties**.
 - a. Select **EMS Configurations for Global Cache Clean** category and update the properties values as per the following table:

| Property name | Value |
|---------------|---|
| emsServerURL | IP address or the DNS name where EMS is running |

| Property name | Value |
|-------------------|-----------------------------|
| emsServerUsername | User name of the EMS server |
| emsServerPassword | Password of the EMS server |

- b. Select **Shopping Cart Data Source Configuration** category and update the properties values as per the following table:

| Property name | Value |
|---------------------------|--|
| hibernateDialect | Hibernate dialect (Oracle or PostgreSQL) |
| shopping cartDsUrl | Data source URL (Oracle or PostgreSQL) |
| shoppingCartDsUsername | User name of the shopping cart data source |
| shoppingCartDsPassword | Password of the shopping cart data source |
| datasourceDriverClassName | Class name of the data source driver |

| Property name | Value |
|---------------------------|---|
| | (Oracle or PostgreSQL) |
| datasourceValidationQuery | SQL query that is be used to validate connections |

- c. Select **Shopping Cart Initial Configuration** category and update the properties values as per the following table:

| Property name | Value |
|--------------------|--|
| cartPluggableCache | Pluggable Cache for ShoppingCart (Redis or Relational) |

Recommendation Engine Core

Procedure

1. On the Configurator UI, navigate to **Recommendation Engine Core > App properties**.
 - a. Select **Recommendation Engine Core Initial Configuration** category and update the properties values as per the following table:

| Property name | Value |
|---------------|----------------|
| dataStoreUrl | Recommendation |

| Property name | Value |
|--------------------------------|------------------------|
| | Engine Data Store Url |
| catalogServiceUrl | Catalog Service URL |
| authorization.service.username | Common User's Username |
| authorization.service.password | Common User's Password |

Recommendation Engine API

Procedure

1. On the Configurator UI, navigate to **Recommendation Engine API > App properties**.
 - a. Select **Data Store configuration** category and update the properties values as per the following table:

| Property name | Value |
|-----------------|--------------------------------------|
| dataStoreUrl | Recommendation Engine Data Store Url |
| shoppingCartUrl | URL for Shopping Cart |

- b. Select **Recommendation Engine Api Initial Configuration** category and update the properties values as per the following table:

| Property name | Value |
|--------------------------------|------------------------|
| authorization.service.username | Common User's Username |
| authorization.service.password | Common User's Password |

- c. Select **Shopping Cart Configurations** category and update the properties values as per the following table:

| Property name | Value |
|-----------------|---|
| shoppingCartUrl | Shopping cart URL for Recommendation Engine |

Recommendation Engine Data Mapper

Procedure

- On the Configurator UI, navigate to **Recommendation Engine Data Mapper > App properties**.
 - Select **Recommendation Engine Data Mapper** category and update the properties values as per the following table:

| Property name | Value |
|---------------|--------------------------------------|
| dataStoreUrl | Recommendation Engine Data Store Url |

- Select **Authorization Server Configuration Properties Used for Auth Token**

category and update the properties values as per the following table:

| Property name | Value |
|--------------------------------|------------------------------|
| authorization.service.username | Common User's Username |
| authorization.service.password | Common User's Password |

Recommendation Engine Data Store Engine

Procedure

1. On the Configurator UI, navigate to **Recommendation Engine Data Store Engine > App properties**.
 - a. Select **Authorization Server Configuration Properties Used for Auth Token** category and update the properties values as per the following table:

| Property name | Value |
|--------------------------------|------------------------------|
| authorization.service.username | Common User's Username |
| authorization.service.password | Common User's Password |

- b. Select **EMS Configurations for Global Product Clean** category and update the properties values as per the following table:

| Property name | Value |
|-------------------|---|
| emsServerURL | IP address or the DNS name where EMS is running |
| emsServerUsername | User name of the EMS server |
| emsServerPassword | Password of the EMS server |

- c. Select **Redis Configuration Properties** category and update the properties values as per the following table:

| Property name | Value |
|---------------------|--|
| redisHostPort | localhost:6379 |
| redisSslEnabled | Connect to Redis Cluster/Node via SSL |
| redisClusterEnabled | Whether redis is running in Cluster Mode |

Task 10: Starting or Restarting the Services

See [TIBCO OPE Resource Allocation](#), if you want to use resource allocation for TIBCO OPE services.

i Note: You can choose to provide 'Xms Xmx' or 'MinRAMPercentage MaxRAMPercentage' parameters based on your requirements.

Procedure

1. To restart the authorization service, navigate to the `$OPE_HOME/roles/authorization-service/standalone/bin` directory and run the `./start.sh` script.
2. To restart the configurator service, navigate to the `$OPE_HOME/roles/configurator/standalone/bin` directory and run the `./start.sh` script.
3. To restart the configurator-ui service, navigate to the `$OPE_HOME/roles/configurator-ui/standalone/bin` directory and run the `./start.sh` script.
4. To start the other services, navigate to the `$OPE_HOME/roles/<service-name>/standalone/config` and update the `application.properties` file for `configuratorServiceUrl`, value is the IP or DNS name of configurator service.
5. Navigate to the `$OPE_HOME/roles/<service-name>/standalone/bin/` directory and run the `./start.sh` script in the following sequence:
 - `catalog-service`
 - `ope`
 - `offersearchindex-service`
 - `shoppingcart`
 - `recommendation-engine-data-store`
 - `recommendation-engine-data-mapper`
 - `recommendation-engine-api`
 - `recommendation-engine-core`

i Note: You must start the Authorization and configurator service first and then all the other services.

i Note: For recommendation-engine-core, you must run the python scripts for the requirements.txt before running the `./start.sh` script. See `$OPE_HOME/roles/recommendation-engine-core/standalone/README.md` file.

Automating postinstallation configurations

This section describes the procedure of automating postinstallation configurations. Instead of performing each configuration one by one, you can now automate the task. To update the bare minimum properties that are already configured in the property file (`config.properties`) of `config-automation`, the authorization and configurator services are started through this automation procedure. Once these two services are started, you can choose to start the other services.

Before you begin

- Download the `TIB_ope_6.0.0.zip`
- Install Python 3.9.X or later on your machine and import the following Python packages or modules:
 - Requests
 - configobj
 - paramiko
 - scp
 - xmltodict
 - cx_Oracle
 - psycpg2

You can install these packages from the `requirements.txt` present under the `$OPE_HOME/config-automation` directory.

i Note:

- For PostgreSQL, you must create the tablespace location manually before executing these scripts.
- For Docker setup, the user must be part of the Docker group.

Procedure

1. Set all the configurations in the `$OPE_HOME/config-automation/properties/config.properties` file. See the `$OPE_HOME/config-automation/README.md` file for more information.
2. To automate the postinstallation configurations for all the microservices at once, run the `$OPE_HOME/config-automation/src/main.py` script.
3. To automate the postinstallation configurations for the microservices individually, run the following scripts in the given sequence:
 - a. To read config properties, run the `$OPE_HOME/config-automation/src/config_reader.py` script.
 - b. To set the environment variables, run the `$OPE_HOME/config-automation/src/environment_variables.py` script.
 - c. To create the EMS channel, run the `$OPE_HOME/config-automation/ems/ems_setup.py` script.
 - d. To set up the database, run the `$OPE_HOME/config-automation/oracle/oracle_setup.py` script for Oracle or `$OPE_HOME/config-automation/postgres/postgres_setup.py` script for PostgreSQL as per your requirement.
 - e. To create Docker images, run the `$OPE_HOME/config-automation/docker/docker_setup.py` script.
 - f. To create the default tenant users and to start the authorization service, run the `$OPE_HOME/config-automation/auth_server/auth_server_setup.py` script.
 - g. To seed and start Configurator, run the `$OPE_HOME/config-automation/configurator/configurator_setup.py` script.

Docker

microservices are containerized and run on hosts that support the Docker environment. The Docker files are delivered as part of the OPE zip file. You can build images using those Docker files and then run them as containers. You can see all the Docker files in their respective container folders. You must have an Internet connection to download the base Docker image.

You need to take care of the instructions under the comment "REQUIRED FILES TO BUILD THIS IMAGE" in each Docker file before building the Docker images.



Note: Some of the microservices are optional and you can choose not to run them in case they are not suitable for your use cases.

- **OfferSearchIndex Service** - You can choose not to deploy this service if you do not want to connect to Elasticsearch with your Offer and Price Engine deployment.

Building a Docker Image Without an Internet Connection

In `$OPE_HOME/docker/base/1.0/Dockerfile`, the 'FROM registry.access.redhat.com/ubi8/ubi-minimal' instruction initializes a new build stage and sets the base image for subsequent instructions. You can accept the default base image, which is the alpine image from Docker's public repository, or you can change the instruction and provide a valid source for a different base image. You can pull a valid base image from Docker's public repository or you can create your base image, push it to a public or private Docker registry, and then use the newly created image as a base image. For more information about creating your base Docker image, see the Docker documentation related to Creating a Base Image.

In `$OPE_HOME/docker/base/1.0/Dockerfile`, you can find instructions for downloading `wget` and `unzip` utilities. These instructions can be modified to pick up the installers of the utilities from the Docker context and install them in the image.

In this case, the Docker context for `$OPE_HOME/docker/base/1.0/Dockerfile` is `$OPE_HOME/docker/base/1.0`

The Docker context for `$OPE_HOME/docker/base/2.0/Dockerfile` is `$OPE_HOME/docker/base/2.0`

In `$OPE_HOME/docker/base/2.0/Dockerfile`, you can find instructions for creating Python docker image and installing Python modules required for starting Recommendation Engine Core service.

Copying Files to Docker Context

The term Docker Context refers to the directory where the Docker file is available.

You must copy files to the Docker context before building the required Docker images.

Procedure

1. Install Docker 20.10 (or later), Docker Compose 1.24.0 (or later), and TIBCO Offer and Price Engine on the host machine.
2. Run the `$OPE_HOME/roles/copyLib.sh` script. For more information, see the [Postinstallation Task 1: Copying Dependencies](#) topic.
3. Go to the `$OPE_HOME/docker` directory and run the `copy-required-files.sh` script.

This shell script copies all the required directories from `OPE_HOME` to a specific Docker context. These files are required to build Docker images. This script works on the `OPE_HOME` set in the environment or takes the value of `OPE_HOME` as user input.

Build EMS Client Image

To build the EMS Client image, you can choose one of the following options:

- Use the `Dockerfile` located under the `$OPE_HOME/docker/ems` path and perform a `docker build` command.
- Modify the `docker-compose-build-complete.yml` docker compose file adding a

dedicated EMS section as follows:

```
ems-client:  
  
build:  
  
context: ./ems  
  
dockerfile: Dockerfile  
  
args:  
  
  EMS_VERSION: "8.6"  
  
image: "tibco/ems-  
client:8.6.0"
```

Then you can use the docker-compose command to build all the images and the EMS Client.

EMS Setup

To perform the EMS configuration, you can use the EMS Client image that is already built.

In particular, you can use that image to run a container that connects to the EMS server and performs the needed EMS configuration for OPE services.

- Run the EMS Client container by running the following command:

```
$ docker run --name ems-client --rm -e 'INIT_FLAG=true' -e 'env_ems_svc=<ems_service_<br>external_ip>' -e 'env_ems_svc_port=<ems_service_port>' -e 'env_ems_user=<ems_admin_<br>user>' -e 'env_ems_password=<current_ems_password>' -e 'env_ems_new_password=<new_<br>ems_admin_password>' <image_registry>/<image_repository>:<image_tag>
```

- Alternatively, you can use the following command if you are using the Kubernetes:

```
$ kubectl run ems-client --env="INIT_FLAG=true" --env="<ems_service_fqdn>" --
env="env_ems_svc_port=<ems_service_port>" --env="env_ems_user=<ems_admin_user>" --
env="env_ems_password=<current_ems_password>" --env="env_ems_new_password=<new_ems_
admin_password>" --image=<image_registry>/<image_repository>:<image_tag>
```



Note: Replace the above placeholders with correct values. The env_ems_password is empty by default if no EMS Server configuration is performed. The env_ems_new_password is an option for an environment variable that can be used to configure a new EMS server password for the admin user.

Setting Up the .env File

Set the required variables, which vary according to the user's environment. All of these variables must be changed according to the user's environment.

All of these variables are in the .env file located in the \$OPE_HOME/docker directory.

Building Docker Images

After the required files are copied to the Docker context, you can build the Docker images. Use Docker Compose for building the Docker images.

Procedure

1. Go to the \$OPE_HOME/docker directory and run the following command:
`docker-compose --file docker-compose-build-complete.yml build`
2. Execute the following command to check the images that are created:
`$] docker images`

Preparing Docker Volumes

A Docker volume (commonly referred to as data volume) is a specially-designated directory within one or more containers that bypasses the Union File System. Docker volumes provide several useful features for persistent and shared data. Docker volumes can be shared and reused among containers.

Docker volumes persist even if the container itself is deleted. You can create a Docker volume container from an existing image. The volume container has the mapping of the host's directory to the container. For more information, see the Docker documentation on volume.

Before you start running the Docker containers, you are required to have the following directories on the host machine:

- A logs directory that you must create.

Procedure

- Create a logs directory where logs from all the containers are available on your host machine. For example, \$OPE_HOME/logs.



Note:

Since the Docker container reads and writes in the directory mentioned, it is mandatory to give read and write privileges to others on the host machine on the directories where you are creating volumes. Since Docker is writing to the existing files in the volume, each file in the volume must have read and write privileges. For example:

```
$] chmod  
o+rw -R
```

```
$OPE_  
HOME/logs
```

Running the Docker Containers

After building the Docker images, you can run the images as containers to start containerized TIBCO Offer and Price Engine.

Start the Docker container by using the specific docker compose file.

Procedure

1. Start the configurator-ui Docker container.

```
docker-compose --file docker-compose-run-configurator-ui.yml  
up -d
```

2. Access the configurator UI and configure it according to your environment and requirement.
4. Start the authorization service to fetch the token, which is then used across all Offer and Price Engine services to authorize and authenticate the user.

i Note: The authorization service and configuration service are needed for configurator-ui. Also, after uploading metadata, application properties, and configuration files, you need to perform the required configurations and start OPE services.

```
$> docker-compose --file docker-compose-run-authorization-service.yml  
up -d
```

You can start any of the Docker services by using the following compose commands:

- a. Start the catalog service container.

```
$> docker-compose --file docker-compose-run-catalog-service.yml  
up -d
```

- b. Start the Offer and Price Engine service container.

```
$> docker-compose --file docker-compose-run-ope.yml  
up -d
```

- c. Start the Recommendation Engine Api service container.

```
$> docker-compose --file docker-compose-run-recommendation-engine-api.yml  
up -d
```

- d. Start the Recommendation Engine Data Mapper service container.

```
$> docker-compose --file docker-compose-run-recommendation-engine-data-  
mapper.yml up -d
```

- e. Start the Recommendation Engine Data Store service container.

```
$> docker-compose --file docker-compose-run-recommendation-engine-data-  
store.yml up -d
```

- f. Start the shopping cart service container.

```
$> docker-compose --file docker-compose-run-shoppingcart.yml  
up -d
```

- g. Start the offerSearchIndex service container.

```
$> docker-compose --file docker-compose-run-offersearchindex-service.yml  
up -d
```

- h. Start the Recommendation Engine Core service container.

```
$> docker-compose --file docker-compose-run-recommendation-engine-core.yml  
up -d
```

5. Execute "\$] docker ps -a" to check the containers that are started.

Extend Docker-Compose Files

You can extend the Docker-Compose files provided as part of the TIBCO Offer and Price Engine installation.

This is mainly done to handle different environments. For example, this is done in case you require separate parameters for the containers based on your environments, such as a testing or production environment.

The suggested way to do this is to have multiple Docker compose files for each environment. For more information, see the [Docker documentation](#) on Multiple Compose Files.

Modifying a Container Time-Zone

The default time zone for any Docker container is UTC. In the case where you want the Docker container's time-zone to be in sync with the host machine's time-zone, you can apply these changes either in the Docker file or in the Docker-Compose YAML file.

Docker containers always use the system clock of the host machine but it sets its time-zone as UTC. The following steps are an example of changing the time zone for a TIBCO OPE Server container.

Procedure

- You can modify a container's time zone with either of the following two ways:
 - This approach can be applied when you have not created any images. Open the \$OPE_HOME/ docker/ope/6.0.0/Dockerfile in a suitable editor and modify the file as shown:

```
FROM tibco/base:1.0

COPY ope $OPE_HOME/ope ENV TZ=Asia/Kolkata

COPY config $OPE_HOME/config

RUN ln -snf /user/share/zoneinfo/$TZ etc/localtime "echo $TZ >
/etc/timezone

RUN  chmod 777 $OPE_HOME/ope/standalone/bin/* \

&& chmod -R a+w $OPE_HOME/ope/standalone/config

USER root

ENTRYPOINT ["sh","-c",
```

```
"$OPE_HOME/ope/standalone/bin/start.sh -
XX:MinRAMPercentage=$min_ram_percentage -
XX:MaxRAMPercentage=$max_ram_percentage --run=FG"]

EXPOSE 8090
```

In this example, the following has been modified:

```
ENV TZ=Asia/Kolkata
```

```
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo
$TZ /etc/timezone
```

Here you have to change the value of the TZ variable as per your time zone (in the example, the time zone is Asia/Kolkata).

- This approach can be applied if your images are already created and now you want to change the container time zone at runtime. Open \$OPE_HOME/docker/docker-compose-run-ope- service.yml in a suitable editor and modify the file as shown:

```
version: "3" services:ope:image: "tibco/ope:${OPE_VERSION_TAG}" ports:"${HOST_
OPE_SERVICE_PORT}:8090"volumes:"${HOST_LOG_ROOT_LOCATION_DIR_
PATH}:/opt/tibco/ope/6.0.0/ope/standalone/logs"environment:-
"TZ=Asia/Kolkata"command: sh -c "ln -snf/user/share/zoneinfo/$TZ
/etc/localtime && echo $TZ> /etc/timezone"
```

In this example, the following has been modified:

```
environment:
```

```
- "TZ=Asia/Kolkata"
```

```
command: >
```

```
sh -c "ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/
timezone"
```

Here you have to change the value of the TZ variable as per your time zone (in the example, the time zone is Asia/Kolkata).

Reading Container Logs

When all the desired containers are up and running, it is best practice to check the logs for all the running services.

Logs for all the started and exited containers are available at the path you have mentioned for the LOG_ROOT_LOCATION_DIR_PATH variable in the .env file. So all the logs are preserved on your host machine.

Troubleshooting Error from Building Docker Images

The following error might occur when building Docker images:

```
rm: cannot remove '/home/tibuser/tibco/ope/6.0.0/configurator/standalone/config/ backup':
Directory not empty
```

Perform the following steps to troubleshoot this error:

Procedure

1. Run the following command on the host machine:

```
$] docker info | grep 'Storage Driver' | awk -F':' '{print $2}'  
overlay
```

```
$]
```

2. If the output is overlaid, then apply the following workaround:
 - a. Stop the Docker engine.
 - b. Changed DOCKER_OPTS to set storage-driver value to device-mapper, edit /etc/docker/ daemon.json, and add "storage-driver" : "devicemapper" at the end of existing keys.
 - c. Start the Docker engine.

**Note:**

You can lose the existing Docker images due to the above change.

- d. Verify the fix by running the following command:

```
$] docker info | grep 'Storage Driver' | awk -F':' '{print $2}'  
devicemapper
```

```
$]
```

3. If the Recommendation Engine Data Store services fail to start from Docker container, check the Redis cluster configurations.

Deploying Individual TIBCO OPE services using Kubernetes scripts

Procedure

1. To deploy individual ope services using the Kubernetes scripts, choose from the `$OPE_HOME/kubernetes` directory.

Example: `kubernetes-deploy-run-ope.yml`

2. Modify the script for the image located at `image: tibco/ope:6.0.0` location. Also, modify the `min_ram_percentage` and `max_ram_percentage` script as per the requirement.

3. Finally, run the script from the location where scripts are saved using the following command:

```
kubectll apply -f kubernetes-deploy-run-ope.yml
```

(Make sure that all the required metadata, app_properties, and config-files are uploaded with correct values)

TLS Support for Redis

You can enable the security protocol by using any of the following methods:

- [Connecting to Redis through user credentials](#)
- [Using OpenSSL or TLS](#)

Connecting to Redis through user credentials

To secure Redis through user credentials, you must set the `adminRedisUsername`, `adminRedisDatabase`, `adminRedisClientname`, and `adminRedisPassword` properties in the `$OPE_HOME/roles/standalone/configurator/config/application.properties` and `$OPE_HOME/roles/authorization-service/standalone/config/application.properties` files.

Update `catalogRedisUsername`, `catalogRedisPassword`, `catalogRedisDatabase`, and `catalogRedisClientName` properties in the `$OPE_HOME/seed-data/appproperties/ConfigValues_OPES.json`, `$OPE_HOME/seed-data/appproperties/ConfigValues_OfferSearchIndexService.json`, and `$OPE_HOME/seeddata/app-properties/ConfigValues_CatalogService.json` files.

Also, update `shoppingCartRedisUsername`, `shoppingCartRedisPassword`, `shoppingCartRedisDatabase`, and `shoppingCartRedisClientName` properties in the `$OPE_HOME/seed-data/app-properties/ConfigValues_ShoppingCart.json` file.

Using OpenSSL or TLS

Redis 6.x or later versions support OpenSSL or TLS. For using OpenSSL or TLS, security certificates are required to be generated.

To secure Redis by using OpenSSL or TLS, you must set the following properties in the `$OPE_HOME/roles/standalone/configurator/config/application.properties` and `$OPE_HOME/roles/authorization-service/standalone/config/application.properties` files and files from `$OPE_HOME/seed-data/app-properties`:

| Property Name | Value |
|-----------------------------|--------------------------------|
| redisSslEnabled | true |
| redisKeyStoreType | pkcs12 |
| redisKeyStoreAbsolutePath | Key store absolute file path |
| redisKeyStorePassword | Password |
| redisTrustStoreType | jks |
| redisTrustStoreAbsolutePath | Trust store absolute file path |
| redisTrustStorePassword | Password |



Note: For redisKeyStoreType and redisTrustStoreType, only pkcs12 and jks are supported respectively. As of now, no other values are supported.

Red Hat Universal Base Image Implementation

The Red Hat Universal Base Image is implemented as the base layer for all of the containerized applications. The ubi-minimal offers a minimized pre-installed content set. Red Hat provides a set of command-line tools that can operate without a container engine like podman, buildah, skopeo, etc. You can use the Docker container engine for containerization. Also, all application images are built with the root user. By using ubi as a base image, the image or container size of an application is reduced effectively. Redis 6.x or later versions support OpenSSL or TLS. For using OpenSSL or TLS, security certificates are required to be generated.

i Note: To get an ubi-minimal image from a Red Hat container registry using registry service account token or using Red Hat login, modify the base:1.0 Dockerfile accordingly. For more information about the Red Hat ubi, see the documentation on the Red Hat website.

Verifying Installation

Make sure the required TIBCO OPE microservices have been installed and are running fine without any error.

Managing Health and Liveness Endpoints

TIBCO Offer and Price Engine supports the following health and liveness endpoints to check the overall health of the system and generate the response accordingly:

- [Health Endpoint](#)
- [Ready Endpoint](#)

The HTTP response code numbers are the same for both the endpoints:

- On Success - 200 (OK)
- On Failure - 503 (Service unavailability)

Managing Health Endpoint

The custom Health Endpoint provides information about the overall health status of the application resources like the Elastic search database and disk space.

`http://<host>:<port>/management/health/liveness`

```
{"status": "UP"}
```

Managing Ready Endpoint

The Ready Endpoint provides the Offer and Price Engine server liveness status. It also checks if the server is ready to accept the external requests.

`http://<host>:<port>/management/health/readiness`

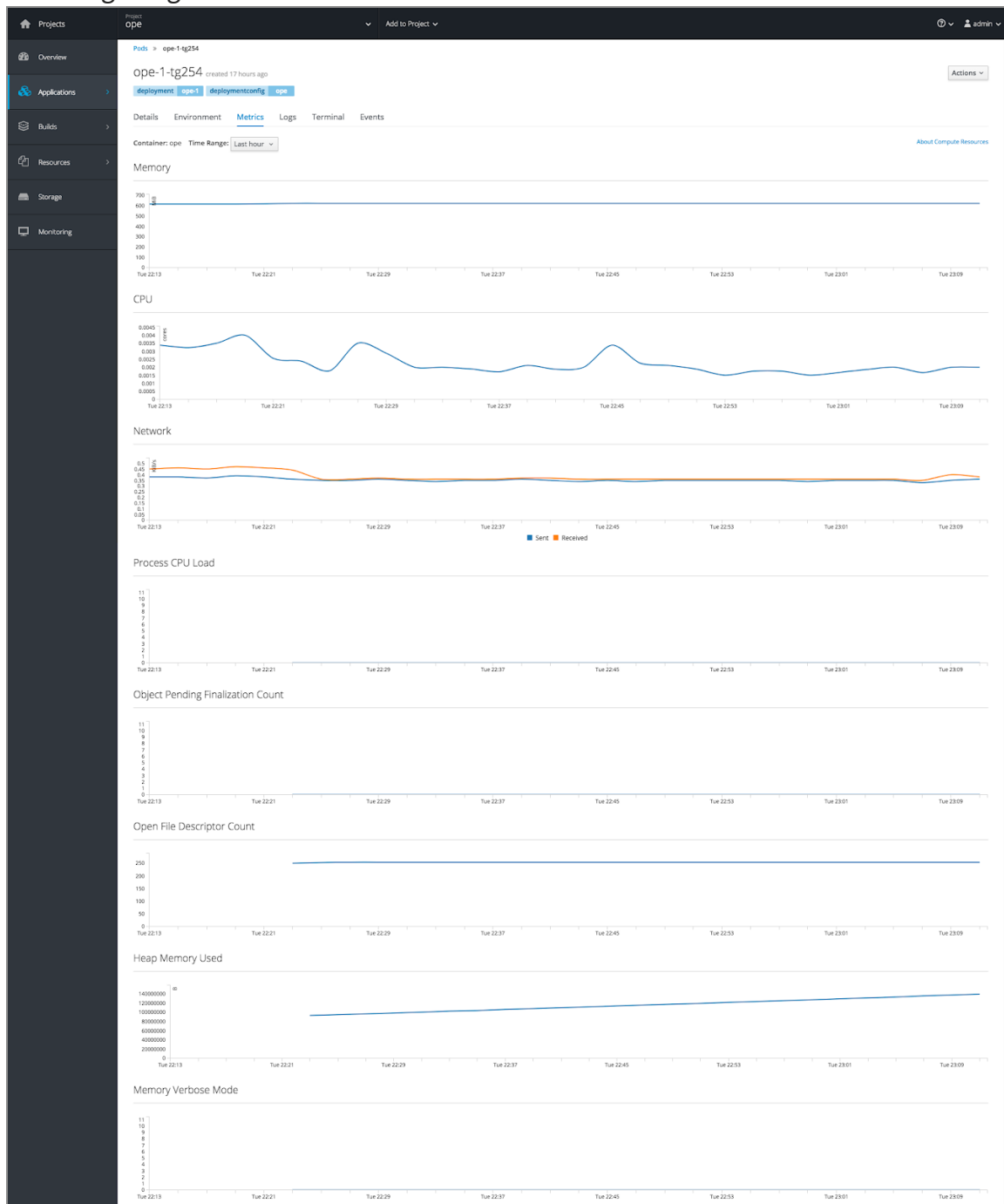
```
{"status": "UP"}
```

Hawkular Monitoring Support

TIBCO Offer and Price Engine provides support for exposing the application bean JMX endpoints on the HTTP protocol by using the Jolokia library.

Jolokia is a third-party bridge released under Apache license and enables exposing application JMX information on HTTP protocol. An OpenShift monitoring application such as Hawkular uses this information to display the additional metrics as shown in the

following image:

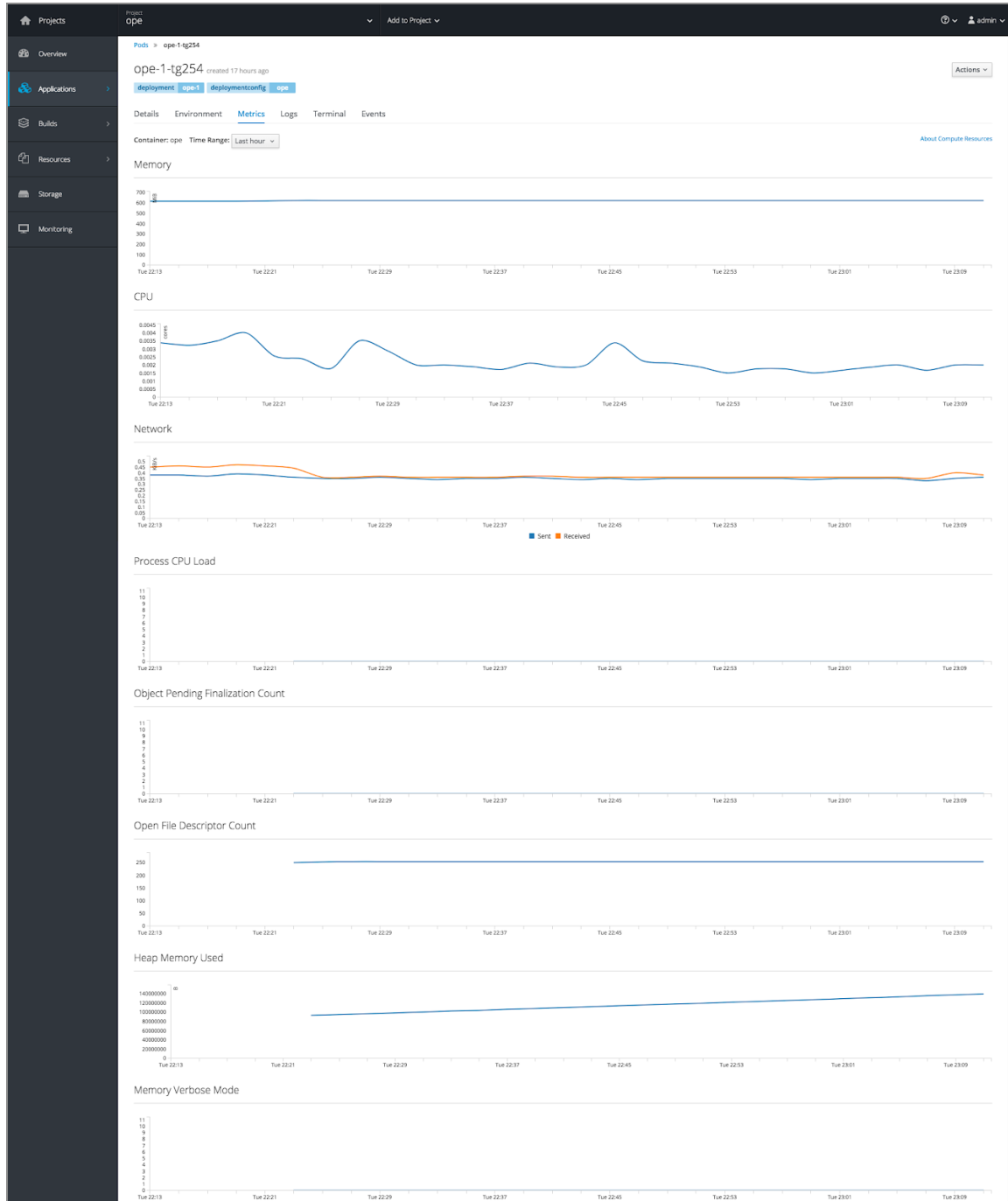


For an example of an OPE deployment yaml file with Jolokia configuration, see [Application Deployment](#)

Hawkular Monitoring Support

The TIBCO Offer and Price Engine provides support for exposing the application bean JMX endpoints on the HTTP protocol, by using the Jolokia bridge.

Jolokia is a third-party bridge released under Apache license and enables exposing application JMX information on HTTP protocol. An OpenShift monitoring application such as Hawkular uses this information to display the additional metrics as shown in the following image:



Red Hat Openshift Deployment

TIBCO Offer and Price Engine containers can be deployed on the Red Hat OpenShift 3.10 PaaS environment. The following topics describe TIBCO Offer and Price Engine compatibility with Red Hat OpenShift 3.10 deployment.

Make sure that your OpenShift environment is up and running and has access to its internal Docker registry.

TIBCO OPE Resource Allocation

Resource allocation for TIBCO OPE services on a cloud platform

1. Docker containers

Modify the .env file present in the \$OPE_HOME/docker directory for MinRAMPercentage and MaxRAMPercentage usage for the following values:

```
min_ram_percentage=25.0  
max_ram_percentage=100.0
```

2. Kubernetes

Kubernetes users need to modify the individual Kubernetes script files as required.

Example:

For the configurator service, modify the kubernetes-deploy-run-configurator.yml file:

```
env:  
- name: min_ram_percentage  
  value: "25.0"  
- name: max_ram_percentage  
  value: "100.0"
```

3. Helm Charts

Helm Charts users must change MinRAMPercentage and MaxRAMPercentage based on their requirements in the values.yaml file:

```
min_ram_percentage: 25.0  
max_ram_percentage: 100.0
```

Resource allocation of JVM for TIBCO OPE services on a non-cloud platform (baremetal box or VM)

When any TIBCO OPE service is started from a bare-metal Linux machine, the user must set the JVM parameters for each micro-service that starts for TIBCO OPE as per the following value:

```
./start.sh -XX:MinRAMPercentage=25.0 -XX:MaxRAMPercentage=100.0
```

i Note: The values for the `cpu/memory` resources change as per the load on a particular micro-service. If the user chooses any type of out-of-the-box caching offered along with TIBCO OPE, by default, values for the `cpu/memory` resources must be increased as required.

Installing Helm Chart

By using a Helm chart, you can deploy all the services and pods at once, instead of deploying each service and package manually. For TIBCO Offer and Price Engine, Helm chart supports on the following cloud platforms:

- Azure Kubernetes Service (AKS)
- Amazon Elastic Kubernetes Service (EKS)
- Google Kubernetes Engine (GKE)

Before you begin

1. Helm Client version 3.5.0 to 3.9.x must be installed on your Kubernetes cluster.
2. EMS server must be deployed on the same Kubernetes cluster (in the same subnet that has Kubernetes version 1.23.x) with all the required queues, topics, and bridges.
3. Create database users by running the scripts present under the `$OPE_HOME/db/dbscripts` directory. Ensure that the database is in the same subnet for the cloud instance that you use.
4. Create Docker images as follows:
 - a. Ensure that the <third-party libraries>link are present in the virtual machine that you are working on and then run the `$OPE_HOME/roles/copyLib.sh` script.
 - b. Run the `$OPE_HOME/docker/copy-required-files.sh` script.
 - c. Create Docker images for all the services. For supporting commands, see `$OPE_HOME/docker/Readme.txt` file.
 - d. Push the Docker images to the required Docker registry.

Procedure

1. Log in to the Kubernetes cluster.
2. Copy the `/ope_services` folder from the `$OPE_HOME/helm` directory to the Kubernetes cluster.
3. Set the environment variables and Docker image names for all the required services

in the `values.yaml` file that is present at the `/ope_services` folder.

4. Run the helm chart from the location where the copied directories are present.

Example: `helm install ope ./ope_services`

Note: The `values.yaml` file contains the required properties for starting authorization service, configurator service, and configurator UI services. Create required users from the authorization service and upload required `app_properties`, `metadata`, and `config` files as per components from the configurator service. Then you can deploy all services. For more details, refer to the `README.md` file from the helm directory.

5. Modify and configure `values.yaml` file to deploy required TIBCO OPE services by selecting components to be deployed as `true`.
Mostly changes are related to database, EMS, and intercommunication of microservices.
6. Choose your desired ingress controller by adding the value for `ingressClassName` field in the `values.yaml` file. The default supported one is `nginx`. You can configure all the services for ingress control in `ope_ingress.yaml` file present inside `templates` directory.
7. Run the helm chart again.

Result

A Helm chart is deployed with all the services present in the chart. See `$OPE_HOME/samples/helm/values.yaml` file for reference.

Note: If you want to enable SSL for TIBCO OPE services in helm chart, see 'Configuring SSL for TIBCO Offer and Price Engine' section in the *TIBCO® Offer and Price Engine Security Guidelines*.

External Property Enhancement

The following enhancement is applicable only for the helm chart. By setting the `authorizationServiceTokenEndPoint` property to the authorization service name (Kubernetes service), you cannot use the Authorize button in the Swagger UI for Offer and Price Engine services, as the browser, tries to call the auth Kubernetes service but it is not

reachable from the outside of the cluster. To resolve this issue, a new property (`authorizationServiceTokenSwaggerEndPoint`) has been added to set the authorization service hostname from ingress. Similarly one more property (`configuratorServiceExternalUrl`) is added to set the Configurator service hostname from ingress.

Migrating to TIBCO Offer and Price Engine 6.0.0

You can migrate TIBCO OPE data from TIBCO OPE 5.1.0 HF-4 to TIBCO OPE 6.0.0. Only the database upgradation and data seeding is done again.

Before you begin

This topic describes all the necessary steps that must be carried out before starting the migration to TIBCO OPE 6.0.0

Backing up the Database

Since the upgrade involves changes in the database, make sure to backup the database instance that has been used by the previous version.

Merging Configurations

The following applicationIds are renamed to make them more readable:

- offersearchindex is renamed to offer-search-index
- shoppingcart is renamed to shopping-cart

These applicationIds now must be updated in the back end store as well. To update the applicationIds for the existing configuration, complete the following steps:

Procedure

1. Use GET APIs to download the configuration files and application properties for the mentioned APIs.
2. Use DELETE APIs to delete the existing applications (offersearchindex and shoppingcart).
3. Upload the downloaded configuration files again and upload the latest application

properties from the \$OPE_HOME/seed-data/app-properties directory and reconfigure as per the existing values for the new applicationIds.

4. Configure all other applications again as per your requirements.

Merge the previous configurations with the current configurations. For more details, see [Task 8: Uploading App Properties, Config Files through Configurator UI](#) and [Task 9: Configuring minimum requirements through Configurator UI](#)

- Change authorizationServiceTokenEndPoint property value from `http://localhost:9091/oauth/token` to `http://localhost:9091` for "common" configuration.

Migrating from TIBCO OPE 5.1.0 HF-4 to TIBCO OPE 6.0.0

Before you start the migration, stop all the OPE 5.1.0 HF-4 running services.

Procedure

1. Install TIBCO OPE 6.0.0. Download the TIBCO OPE 6.0.0 build from TIBCO eDelivery and extract the `TIB_ope_6.0.0.zip` file to the `OPE_HOME` folder. See [Installation](#).
2. Navigate to the `$OPE_HOME/database-scripts/postgreSQL/<database-name>/bin` or `$OPE_HOME/database-scripts/oracle/<database-name>` directory.
3. Update the `postgres_<service-name>_db.properties` or `oracle_<service-name>_db.properties` for all the databases (except shopping-cart) with OPE 5.1.0 HF-4 database details.
4.
 - Run the `upgrade-5.1.0hf7-to-6.0_db-setup.sh` script for Oracle.
 - Run the `upgrade_5.1.0_to_6.0_db-setup.sh` script for PostgreSQL.
5. Start the Authorization, Configurator, and Configurator UI services.
6. Merge the configurations. See [Merging Configurations](#).

Configuration

Application Properties

Offer and Price Engine Integration Configuration

| Key | Description |
|---|---|
| <code>com.tibco.af.ope.flags.chkrelevantoludfs</code> | Validates User Defined Fields attached to the product that are defined as input characteristics of the product. The default value is false. |
| <code>com.tibco.af.ope.flags.chkvalidoludfs</code> | Validates mandatory characteristics attached to the product model that are found in the corresponding product instance as User Defined Fields in the request. The default value is false. |
| <code>com.tibco.af.ope.flags.chkvalidlinkudfs</code> | Validates mandatory linking User Defined Fields that are attached as User Defined Fields to the product in the order request. The default is false. |
| <code>com.tibco.af.ope.flags.chkrecordtypevalid</code> | Checks if the RecordType is valid. The default is false. |
| <code>com.tibco.af.ope.extn.enableRuleBasedEligibilityEvaluation</code> | Enables the Eligible rule for Eligible products. The default value is false. |

| Key | Description |
|--|--|
| <code>com.tibco.af. ope.extn.enableRuleBasedIneligibilityEvaluation</code> | Enables the InEligible rule for InEligible products. The default value is false. |
| <code>com.tibco.af. ope.extn.enableRuleBasedIncompatibilityEvaluation</code> | Enables the Incompatible rule for Incompatible products. The default value is false. |
| <code>com.tibco.af.ope.extn.missingRuleEntityDefault</code> | Used for missing rule entity. The default value is false. |
| <code>com.tibco.af.ope.extn.invConf.host</code> | The default value is localhost. |
| <code>com.tibco.af.ope.extn.invConf.port</code> | The default value is 8081. |
| <code>com.tibco.af.ope.extn.invConf.user</code> | The default value is admin@tibco. |
| <code>com.tibco.af.ope.extn.invConf.password</code> | The default value is admin. |
| <code>com.tibco.af.ope.flags.setnorecordstatustoactive</code> | Sets no RecordStatus to active. The default value is false. |
| <code>com.tibco.af.ope.flags.validateudfdatatype</code> | Validates the data types for orderline User Defined Fields. The default is false. |
| <code>com.tibco.af.ope.flags.validateudfrange</code> | Validates that the orderline User Defined Fields are within the specified range specified. The default is false. |
| <code>com.tibco.af.ope.flags.validateudfregex</code> | Validates that the orderline User Defined Fields have the values per the regex. The default is false. |

| Key | Description |
|--|---|
| <code>com.tibco.af.ope.flags.regexcurrency</code> | The regular expression for currency. The default value is <code>[0-9]*.[0-9]*</code> . |
| <code>com.tibco.af.ope.flags.regexdigits</code> | The regular expression for digits. The default value is <code>[0-9]*.[0-9]*</code> . |
| <code>com.tibco.af.ope.flags.regexdate</code> | The regular expression for date. The default value is <code>[0-3]?[0-9]/[0-1][0-9]/[1-2][0-9]{3}</code> . |
| <code>com.tibco.af.ope.flags.regextime</code> | The regular expression for time. The default value is <code>[0-2]?[0-9]:[0-5][0-9]</code> . |
| <code>com.tibco.af.ope.flags.regexboolean</code> | The regular expression for Boolean. The default value is <code>TRUE FALSE</code> . |
| <code>com.tibco.af.ope.flags.udfignorelist</code> | The User Defined Fields in this property are not shown as incompatible if they are in the same order: <code>EPMR_ACTION_PROVIDE</code> |
| <code>com.tibco.af.ope.cacheType.cache.lazyLoadProducts</code> | Lazy load models from database if not found |
| <code>com.tibco.af.ope.flags.filteroutduplicates</code> | Filter out duplicate products for offers |
| <code>com.tibco.af.ope.flags.enable.page</code> | Enable pagination for <code>getOffer</code> requests |
| <code>com.tibco.af.ope.flags.allow.empty.response</code> | Allow empty response for <code>GetOffer</code> API |

| Key | Description |
|---|---|
| <code>com.tibco.af.ope.flags.ineligible.reason</code> | Specify reason for ineligible records |
| <code>com.tibco.af.ope.flags.ineligible.price.reason</code> | Specify reason for ineligible prices |
| <code>com.tibco.af.ope.log.processingtime</code> | Enable this flag to print processing time for OPE API calls at ERROR level. By default it will log processing time at INFO level. |

External Application Logging Configuration

Offer and Price Engine logging can be configured using the environmental variable `CONSOLE_LEVEL` to specify the logging level.

Also, to configure the logging level through the `log4j2.xml` file under `config`, you can configure the logging level using the environmental variable `CONSOLE_LEVEL`.

You can set the variable and its values such as shown in the following example,

Variable Name: `CONSOLE_LEVEL`

Valid Values: `ALL / DEBUG / ERROR / FATAL / INFO / OFF / TRACE / WARN`

Default Value: `ERROR`

Troubleshooting

The following section describes troubleshooting information for TIBCO Offer and Price Engine.

| Error | Solution |
|--|---|
| <p>ERROR: for elasticsearch Cannot start service elasticsearch: OCI runtime create failed: container_linux.go:344: starting container process caused "process_linux.go:424: container init caused \"rootfs_linux.go:58: mounting \\\" to rootfs \\\"/var/lib/docker/devicemapper/mnt/ 2bf96ca7365c66c9e5f4a3a69a23c23eb5a 4b4f721c4b06c20db626fc7e49e63/ rootfs s and is the expected type ERROR: for elasticsearch Cannot start service elasticsearch: OCI runtime create failed: container_linux.go:344: starting container process caused "process_linux.go:424: container init caused \"rootfs_linux.go:58: mounting \\\" to rootfs \\\"/var/lib/docker/devicemapper/mnt/ 2bf96ca7365c66c9e5f4a3a69a23c23eb5a4 b4f721c4b06c20db626fc7e49e63 /rootfs\\\" at \\\"/var/lib/docker/devicemapper/mnt/ 2bf96ca7365c66c9e5f4a3a69a23c23eb5a 4b4f721c4b06c20db626fc7e49e63/ rootfs/usr/share/elasticsearch/config/elasticsearch.yml\\\" caused \\\"not a directory\\\"\\\": unknown: Are you trying to</p> | <p>Check if the specified host path exists and is the expected type.</p> <p>Create Elastic directory and copy the elasticsearch.yml from samples under Elastic directory.</p> <p>Also give 777 privileges on the Elastic directory:</p> <pre>sudo docker-compose -- file docker- compose.yml up elasticsearch</pre> |

| Error | Solution |
|---|---|
| <p>mount a directory onto a file (or vice-versa)? Check if the specified host path exists and is the expected type</p> <p>ERROR: Encountered errors while bringing up the project.</p> | |
| <p>ERROR: Encountered errors while bringing up the project.</p> | |
| <p>elasticsearch [2019-02-07T18:55:44,689][INFO] [o.e.b.BootstrapChecks] [es01] bound or publishing to a non-loopback address, enforcing bootstrap checks</p> <p>elasticsearch ERROR: [2] bootstrap checks failed</p> <p>elasticsearch [1]: initial heap size [536870912] not equal to maximum heap size [2634022912]; this can cause resize pauses</p> <p>and prevents mlockall from locking the entire heap</p> <p>elasticsearch [2]: max virtual memory areas vm.max_map_count [65530] is too low, increase to at least [262144]</p> | <p>Run the command:</p> <pre>sudo sysctl -w vm.max_map_count=262144</pre> |
| <p>Creating elasticsearch ...</p> <p>Creating elasticsearch ... error</p> <p>ERROR: for elasticsearch Cannot start service elasticsearch:</p> | <p>Enable firewall where Elastic Search is deployed, and open the ports needed:</p> |

| Error | Solution |
|--|--|
| <p>driver failed programming external connectivity on endpoint elasticsearch (447e4dcea1af447021e29cbaafe1c87ce08c5f8bfa8063e5faa8fa40eedd20c0):</p> <p>(iptables failed: iptables --wait -t nat -A DOCKER -p tcp -d 0/0 --dport 9300 -j DNAT --to-destination 172.20.0.2:9300 ! -i br-b99ab24b2e0e: iptables: error</p> <p>(exit status 1))</p> | <pre>firewall-cmd -- zone=work --add- port=9200/tcp -- permanent</pre> |

TIBCO Documentation and Support Services

For information about this product, you can read the documentation, contact TIBCO Support, and join TIBCO Community.

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the [TIBCO Product Documentation](#) website, mainly in HTML and PDF formats.

The [TIBCO Product Documentation](#) website is updated frequently and is more current than any other documentation included with the product.

Product-Specific Documentation

The following documentation for this product is available on the [TIBCO® Offer and Price Engine](#) documentation page:

- *TIBCO® Offer and Price Engine Release Notes*
- *TIBCO® Offer and Price Engine Installation and Configuration Guide*
- *TIBCO® Offer and Price Engine Concepts Guide*
- *TIBCO® Offer and Price Engine User Guide*
- *TIBCO® Offer and Price Engine Web services Guide*
- *TIBCO® Offer and Price Engine Security Guidelines*

How to Contact TIBCO Support

Get an overview of [TIBCO Support](#). You can contact TIBCO Support in the following ways:

- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the [TIBCO Support](#) website.
- For creating a Support case, you must have a valid maintenance or support contract with TIBCO. You also need a user name and password to log in to [TIBCO Support](#) website. If you do not have a user name, you can request one by clicking **Register** on

the website.

How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the [TIBCO Ideas Portal](#). For a free registration, go to [TIBCO Community](#).

Legal and Third-Party Notices

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE “LICENSE” FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of Cloud Software Group, Inc.

TIBCO, the TIBCO logo, and the TIBCO O logo are either registered trademarks or trademarks of Cloud Software Group, Inc. in the United States and/or other countries.

Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

This document includes fonts that are licensed under the SIL Open Font License, Version 1.1, which is available at: <https://scripts.sil.org/OFL>

Copyright (c) Paul D. Hunt, with Reserved Font Name Source Sans Pro and Source Code Pro.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

This software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the readme file for the availability of this software version on a specific operating system platform.

THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. CLOUD SOFTWARE GROUP, INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "READ ME" FILES.

This and other products of Cloud Software Group, Inc. may be covered by registered patents. Please refer to TIBCO's Virtual Patent Marking document (<https://www.tibco.com/patents>) for details.

Copyright © 2010-2023. Cloud Software Group, Inc. All Rights Reserved.