TIBCO BusinessWorks™ Container Edition

Application Monitoring and Troubleshooting

Version 2.9.2 | August 2024

Contents

Changing Help Preferences

By default, documentation access from TIBCO Business Studio™ for BusinessWorks™ is online, through the TIBCO Product Documentation website that contains the latest version of the documentation. Check the website frequently for updates. To access the product documentation offline, download the documentation to a local directory or an internal web server and then change the help preferences in TIBCO Business Studio for BusinessWorks.

Before you begin

Before changing the help preferences to access documentation locally or from an internal web server, download the documentation.

1. Go to https://docs.tibco.com/

2. In the **Search** field, enter TIBCO ActiveMatrix BusinessWorks™ and press **Enter**.

3. Select the TIBCO ActiveMatrix BusinessWorks™ product from the list. This opens the product documentation page for the latest version.

4. Click **Download All**.

5. A compressed .zip file containing the latest documentation is downloaded to your web browser's default download location.

6. Copy the .zip file to a local directory or to an internal web server and unzip the file.

To point to a custom location:

Procedure

1. Perform one of the following steps in TIBCO Business Studio for BusinessWorks based on your operating system:

   - On Windows OS: Click **Window > Preferences**

   - On macOS: Click **TIBCO Business Studio > Preferences**.

2. In the Preferences dialog, click **BusinessWorks > Help**.

3. Click **Custom Location**, and then browse to the `html` directory in the folder where you extracted the documentation or provide the URL to the `html` directory on your

internal web server.

4. Click **Apply**, and then click **OK**.

Application Monitoring Overview

You can run the monitoring component of TIBCO BusinessWorks™ Container Edition on the same container platform where TIBCO BusinessWorks Container Edition applications are running. TIBCO BusinessWorks Container Edition applications can be registered with the monitoring application to view application metrics.
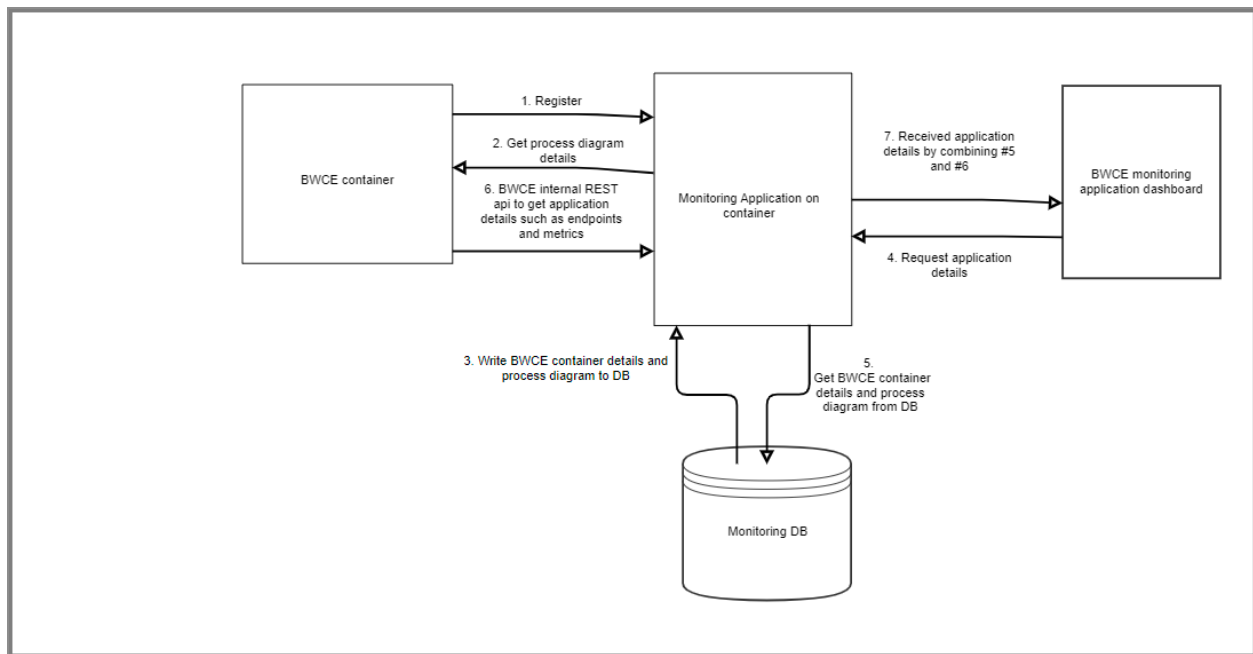
When TIBCO BusinessWorks Container Edition application registration fails at container startup because of an unhealthy monitoring container, the monitoring register mechanism is implemented. To configure the monitoring register mechanism, the following properties are used as environment variables:

- `BW_APP_MON_REGISTER_ATTEMPTS(#) (Default : 5)`

- `BW_APP_MON_REGISTER_DELAY(ms) (Default: 5000)`

Once the retry count exhausts, the monitoring application does not register and the user needs to restart the container.

To change log levels in application monitoring, the following log levels is used:

Property name: LOG_LEVEL

For example,

- LOG_LEVEL="debug"

- LOG_LEVEL="error"

- LOG_LEVEL="warning"

- LOG_LEVEL="info"

Health check for monitoring application is verified using the following API:
`http://host:port/api/v1/monitor/health`.

If the monitoring container is healthy it returns `200K`, else the returned response is 500 Internal Server Error.

At container startup, when the monitoring services fail to connect to the database due to database unavailability, the connection retry mechanism is implemented. To configure the database retry mechanism, the following properties are used as environment variables:

- `DB_RETRY_COUNT : default set to 10`

- `DB_RETRY_INTERVAL : default set to 10000 ms.`

Once the retry count exhausts, the monitoring application crashes automatically.

When a TIBCO BusinessWorks Container Edition application on Cloud Foundry takes time to generate an accessible routing URL, which causes registration failure at container startup, the retry mechanism is implemented for retrying the routing URL accessibility. To configure this retry mechanism, the following properties are used as environment variables:

- `ROUTING_ACCESSIBILITY_ATTEMPTS(#)(default : 20)`

- `ROUTING_ACCESSIBILITY_DELAY(ms)(default : 6000)`

Once the retry mechanism exhausts, the TIBCO BusinessWorks Container Edition container application does not register with the monitoring application, and the user needs to restart the container with higher values.

The DB_URL environment variable is used to provide database-specific information to run the application monitoring containers. The default format of the URL is:

```
docker run -it --rm -p 48080:8080 -e PERSISTENCE_TYPE="postgres" -e DB_
URL="postgres://<username:password>@<machine:port/database>" --name
<containerName><monitoringImageName:tag>
```

The DB_URL can also be segregated into the following environment variables:

- DB_NAME

- DB_USER

- DB_PWD

- DB_HOST

- DB_PORT

To use the above environment variables, refer to the below docker command:

```
docker run -it --rm -p 6151:8080 -e PERSISTENCE_TYPE="<DBTYPE>" -e DB_NAME="<DB_
Name>" -e DB_USER="<Username>" -e DB_PWD="<DB_Password>" -e DB_HOST="<HOST_
Address>" -e DB_PORT="<DB_Port>" --name <containerName><monitoringImageName:tag>
```

To avoid exposure of passwords in the free text form, we can obfuscate the password in the DB_PWD environment variable using the `npm run obfuscate plainpassword` on command line.

The *VALIDATE_DB_URL* boolean environment variable is added to skip DB_URL validation. The default value for the environment variable is True. If you want to use the DB_URL in any format other than its default format `[DBTYPE]://[UserName]:[Password]@[machineName]:[PORT]/[DBNAME]`, set the *VALIDATE_DB_URL* environment variable to False. When set to false, the DB_URL validation is skipped entirely.

Application Monitoring on Cloud Foundry

You can simply deploy a BusinessWorks application on the cloud foundry and enable application monitoring. The monitoring dashboard displays the running application details and its statistics collection.

> **ⓘ** **Note:** To display the application name on the monitoring dashboard from the `manifest.yml` file, instead of the application name from the EAR file, provide the property `DISPLAY_ALIAS: true` in the monitoring `manifest.yml` file.

Procedure

1. Run the monitoring application on a cloud foundry.

2. Enable monitoring by registering the TIBCO BusinessWorks Container Edition

application with the monitoring application by using a CUPS or an environment variable. For more information, see Enabling Monitoring on Cloud Foundry.

Configuring MySQL on Cloud Foundry

For persistence support with the monitoring application, you need to configure MySQL with either a marketplace service or a user-provided service.

Creating Service from the Marketplace

1. You can check the services available from the Cloud Foundry Marketplace by running the following command:

```
cf marketplace
```

```
xinpan-MBP15:app xinpan$ cf marketplace
Getting services from marketplace in org pcfdev-org / space pcfdev-space as admin...
OK

service        plans            description
local-volume   free-local-disk  Local service docs: https://github.com/cloudfoundry-incubator/local-volume-release/
p-mysql        512mb, 1gb       MySQL databases on demand
p-rabbitmq     standard         RabbitMQ is a robust and scalable high-performance multi-protocol messaging broker.
p-redis        shared-vm        Redis service to provide a key-value store

TIP:  Use 'cf marketplace -s SERVICE' to view descriptions of individual plans of a given service.
xinpan-MBP15:app xinpan$
```

2. Run the following command to create a service.

```
cf create-service <SERVICE_NAME><SERVICE_PLAN><SERVICE_INSTANCE_NAME>
```

> **Note:** Define the *<SERVICE_INSTANCE_NAME>* as bwcemon_mysql

```
xinpan-MBP15:bin xinpan$ cf create-service p-mysql 512mb bwcemon_mysql
Creating service instance bwcemon_mysql in org pcfdev-org / space pcfdev-space as admin...
OK
xinpan-MBP15:bin xinpan$
```

MySQL is now configured with the marketplace service on the Cloud Foundry environment.

Creating a User Provided Service for MySQL

1. To create the User Provided Service (CUPS) for the MySQL database, run the following command:

```
cf cups <service_instances_name> -p "host,username,password,database"
```


MySQL is now configured with the user-provided service on the Cloud Foundry environment.

Configuring PostgreSQL on Cloud Foundry

For persistence support with the monitoring application, you can also configure PostgreSQL with a user-provided service.

Creating a User Provided Service for PostgreSQL

Procedure

1. Run the following command to create the user-provided service.

   ```
   cf cups <service_instances_name> -p "db_url,db_username,db_password"
   ```

   > **Note:** Define the *<SERVICE_INSTANCE_NAME>* as `bwcemon_postgres`.

   PostgreSQL is now configured with the user-provided service on the Cloud Foundry environment.

Configuring MS SQL Server on Cloud Foundry

For persistence support with the monitoring application, you can also configure MS SQL Server with a user-provided service.

Creating User Provided Service for MS SQL Server

Procedure

Run the following command to create the user-provided service:

```
cf cups <service_instances_name> -p "db_url,db_username,db_password"
```


> **Note:** Define the *<SERVICE_INSTANCE_NAME>* as `bwcemon_mssql`.

MS SQL is now configured with the user-provided service on the Cloud Foundry environment.

To configure TIBCO BusinessWorks Container Edition monitoring with the Azure Managed MS SQL Server, set `DB_ENCRYPT` property to true. By default, this property is set to false.

Configuring Oracle on Cloud Foundry

For persistence support with the monitoring application, you need to configure Oracle with either a marketplace service or a user-provided service.

Creating nodeJs Buildpack with Oracle Client

Procedure

1. Create a folder on the same level as the `create-buildpack-nodejs-oracle.sh` file and name it `instantclient`.

2. Download Oracle client libraries from
   https://www.oracle.com/technetwork/topics/linuxx86-64soft-092277.html

3. Add the zip file to the `instantclient` folder. For example, `instantclient-basic-linux.x64-12.2.0.1.0.zip`.

4. Open the `create-buildpack-nodejs-oracle.sh` file and on line 11, point `CLIENT_FILENAME` to the zip file name that is downloaded.

5. On line 19, change the `libclntsh.so` file name according to the client download "`ln -s \$build_dir/oracle/lib/libclntsh.so.12.1 \$build_ dir/oracle/lib/libclntsh.so`". For 18.x and later, delete line 19.

6. Run the `create-buildpack-nodejs-oracle.sh` file.

   This creates the Nodejs build pack with an oracle client `nodejs-buildpack-master- oracle.zip` file.

7. Upload the build pack to cloud foundry. For example, nodejs-oracle (cf create-buildpack nodejs-oracle nodejs-buildpack-master-oracle.zip 1).

Creating User Provided Service for Oracle

Procedure

1. Run the following command to create the user-provided service.

   ```
   cf cups <service_instances_name> -p "db_url,db_username,db_password"
   ```
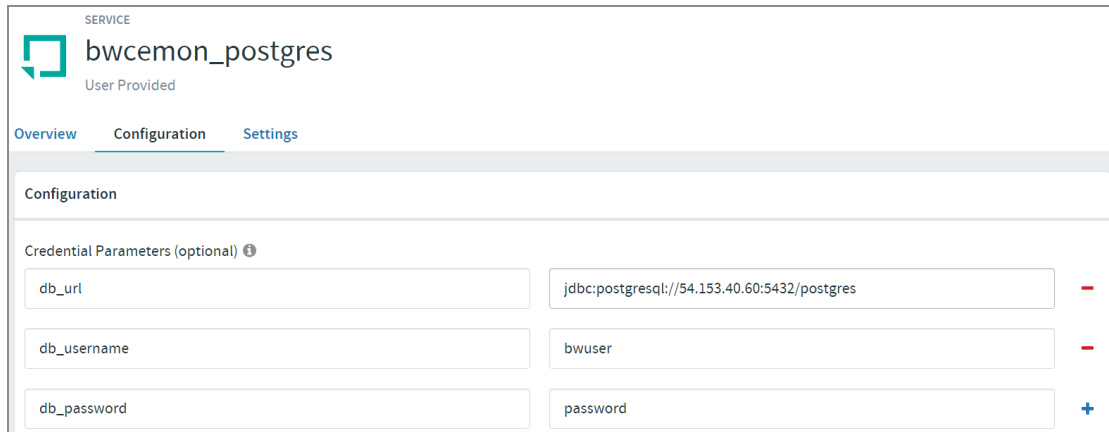
User Provided	bwcemon_mssql	0	User Provided

   > ℹ **Note:** Define the *<SERVICE_INSTANCE_NAME>* as `bwcemon_oracle`.

   Oracle is now configured with the user-provided service on the Cloud Foundry environment.

Configuring a Monitoring Application

Procedure

1. Open the `manifest.yml` file and change the below:

    ```
    buildpack: nodejs-oracle
    services:
            -bwcemon_oracle
    env:
                    PERSISTENCE_TYPE: oracle
    ```

2. Push the monitoring application to Cloud Foundry

    ```
    cf push
    ```

Setting up TIBCO BusinessWorks Container Edition Application Monitoring on Cloud Foundry

The following steps describe how to set up a TIBCO BusinessWorks Container Edition application on the Cloud Foundry.

Before you begin

1. Ensure that you install the Cloud Foundry Command Line Interface (CLI) successfully. Next, push the created TIBCO BusinessWorks Container Edition build pack to the Cloud Foundry environment.

2. Download the TIBCO BusinessWorks Container Edition monitoring zip file `bwce_mon-<version>.zip` from http://edelivery.tibco.com.

3. Ensure the MySQL, PostgreSQL, or MS SQL Server service is created on Cloud Foundry.

Procedure

1. Extract the `bwce_mon-<version>.zip` file.

2. Navigate to the **bwce_mon** directory.

3. Bind the service created earlier to the monitoring application. You must configure `manifest.yml` of the monitoring application to persist node registry information. You have to specify a database service and an environment variable for the MySQL database.

```
---
applications:
- name: com.tibco.bwce.monitoring
  command: node server/node-server.js
  memory: 512M
  buildpack: https://github.com/cloudfoundry/nodejs-buildpack
  services:
   - bwcemon_mysql
  env:
    PERSISTENCE_TYPE: mysql
```

> **ⓘ Note:** The environment variable for application monitoring has been changed from `persistence_DB` to `PERSISTENCE_TYPE`.

> **ⓘ Note:** For PostgreSQL or MS SQL or Oracle databases, ensure that the `PERSISTENCE_TYPE` environment variable value is set to `postgres` or `mssql` or `oracle`.

> **Note:** For the offline nodejs build pack uploaded on VMware Tanzu, edit the `manifest.yml` file with the uploaded build pack name. For example:
>
> ```
> buildpack: nodejs_buildpack
> ```

4. Run `cf push -f manifest.yml` to push the BWCE monitoring application on the Cloud Foundry.

   After the TIBCO BusinessWorks Container Edition application is running on Cloud Foundry, you can access the URL from a browser and monitor the application.
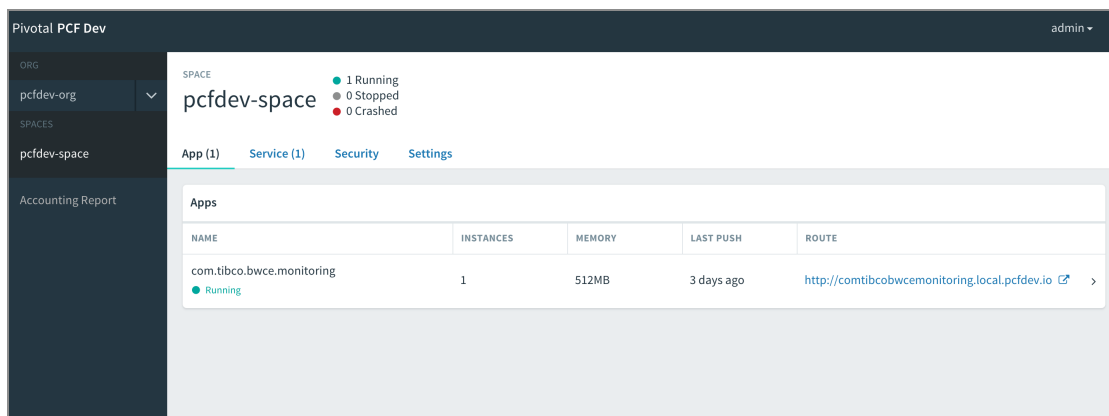
Binding BusinessWorks Application to Monitoring Application on Cloud Foundry

TIBCO BusinessWorks Container Edition application can be bound to the monitoring application by using **Create User Provided Service** (CUPS) or **environment variable**.

Using CUPS

You can monitor an application by using Create User Provided Service (CUPS).

Before you begin

Ensure that you create CUPS for TIBCO BusinessWorks Container Edition monitoring.

Procedure

1. In the cloud foundry CLI, run the command `cf cups <monitoring_app_name> -p"url"`.

   For example: `http://comtibcobwcemonitoring.local.pcfdev.io`

   > **ⓘ** **Note:** Ensure the name of the CUPS for the monitoring application must be `bwce-monitoring`.

   After running the command, you can see the service running on the VMware Tanzu management web UI.
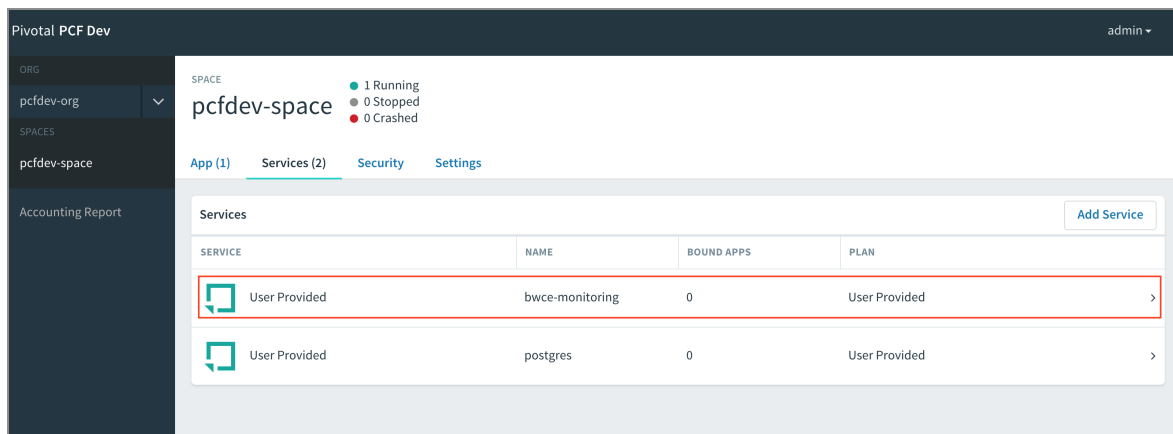
2. Create the `manifest.yml` file in the directory where the application EAR file is exported.

   > **ⓘ** **Note:** The application name displayed on the monitoring dashboard is provided by the `manifest.yml` file.

3. Add `bwce-monitoring` as a service in `manifest.yml`.

4. In the cloud foundry CLI, run the `cf push` command to deploy the application on the cloud foundry. After the application is deployed successfully, you can see the service running on the VMware Tanzu management web UI.

> **ⓘ** **Note:** After the application is successfully started, the TIBCO BusinessWorks Container Edition application gets registered with the monitoring application.

Using an Environment Variable

You can monitor an application by using an environment variable.
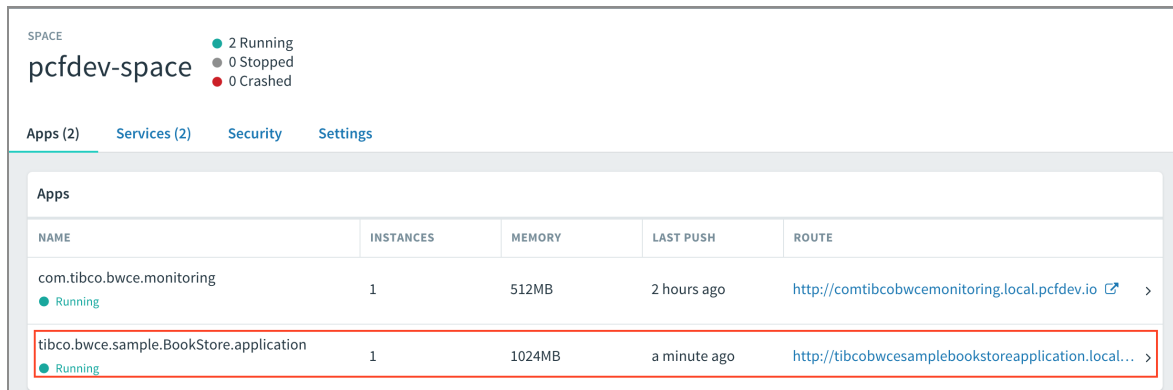
Procedure

1. Create a `manifest.yml` in the same directory where the application EAR file is exported.

2. Set the environment variable to bind the monitoring service. Add the `BW_APP_ MONITORING_CONFIG"` `<url>` environment variable in the `manifest.yml` file.

   ```
   applications:
   - name: RestBookStoreSample.application
           memory: 1024M
     path: tibco.bwce.sample.binding.rest.BookStore.application.ear
                   timeout: 60
                   buildpack: bw-buildpack
     env:
                           BW_LOGLEVEL: ERROR
                           BW_APP_MONITORING_CONFIG: "
   {\"url\":\"http://monitoring.tibcopcf110.com\"}"
   ```

> **i** **Note:** The application name displayed on the monitoring dashboard is provided by the `manifest.yml` file.

3. In cf CLI, run the command `cf push` to deploy the application on the Cloud Foundry.

4. After the application is deployed successfully, you can see the application running on the Cloud Foundry management web UI.

SPACE				
pcfdev-space	● 2 Running ● 0 Stopped ● 0 Crashed			

Apps (2)   Services (2)   Security   Settings

Apps

NAME	INSTANCES	MEMORY	LAST PUSH	ROUTE
com.tibco.bwce.monitoring ● Running	1	512MB	2 hours ago	http://comtibcobwcemonitoring.local.pcfdev.io ⤢ ›
tibco.bwce.sample.BookStore.application ● Running	1	1024MB	a minute ago	http://tibcobwcesamplebookstoreapplication.local... ›

User Authentication Using Cloud Foundry UAA

Cloud Foundry User Account and Authentication (UAA) is an open source identity server. It provides a centralized identity management service with a standalone OAuth2 server.

Application monitoring in TIBCO BusinessWorks Container Edition helps you to authenticate your Cloud Foundry credentials to access the monitoring URL, and act as a Single Sign-On (SSO) service by using those credentials. For more information on UAA, see the VMware Tanzu documentation.

> **ℹ** **Note:** By default, the value for the AUTHENTICATION_MODE environment
> variable is set to none in the manifest file. When set to none, the UAA is disabled
> for application monitoring.

```
- name: com.tibco.bwce.monitoring.sample
  command: node server/node-server.js
  memory: 512M
  buildpack: https://github.com/cloudfoundry/nodejs-buildpack
  services:
   - bwcemon_mysql
  env:
    PERSISTENCE_TYPE: mysql
    AUTHENTICATION_MODE: none
```

Enabling User Account and Authentication for VMware Tanzu

You can use User Account and Authentication (UAA) to authenticate the user with their Cloud Foundry user credentials to access the application monitoring URL.

Before you begin

Ensure that Cloud Foundry UAA Command Line Client (UAAC) is installed.

Procedure

1. Set the UAA target URL by running the following command:

    ```
    uaac target <UAA server path>
    ```

2. To authenticate and obtain an access token for the admin client from the UAA server, run the following command:

    ```
    uaac token owner get
    ```

    Enter the following details in the console:

    a. **Client ID**: Enter the client ID for the UAA admin client. By default, the client ID

is opsman.

  b. **Client Secret**: Enter the client secret for UAA. By default, the client secret is nullable.

  c. **Username**: Enter the VMware Tanzu Ops Manager username.

  d. **Password**: Enter the VMware Tanzu Ops Manager password.

3. Create a client for the monitoring application on the UAA server by running the following command:

```
uaac client add <client_ID> --secret <client_secret> --authorities <authorities> --
scope <allowed_scope_for_client>  --autoapprove <auto_approve> --authorized_
grant_types <grant_type_for_authorization_code> --redirect_uri <redirect_URL>
```

> **ℹ Note:** Ensure that the authorization grant type is authorization_code.

> **ℹ Note:** Redirect URL must be in the following format, where *<monitoring URL>* is the URL for the monitoring application.
>
> *<monitoring URL>*/*

4. Create a user on the UAA server by running the following command:

```
uaac user add <username> -p <user_secret> --emails <emailID>
```

You can use these user credentials to log in to the monitoring application.

> **ℹ Note:** You can also log into the monitoring application using VMware Tanzu Ops Manager credentials.

5. Create the VMware Tanzu user-provided service (CUPS) by configuring the following **Credential Parameter** in the **Configuration** tab of the user-provided service:

> **ℹ Note:** Ensure that the name of the service is bwcemon_UAAClient.

Fields	Description
client_id	Enter the client ID created in step 3.
client_secret	Enter the password of the client ID created in step 3.
url	Ops manager UAA URL.
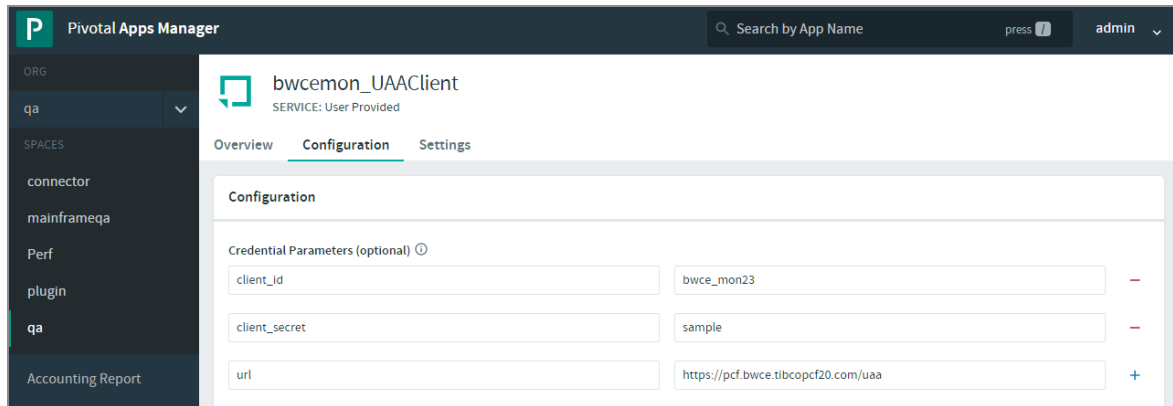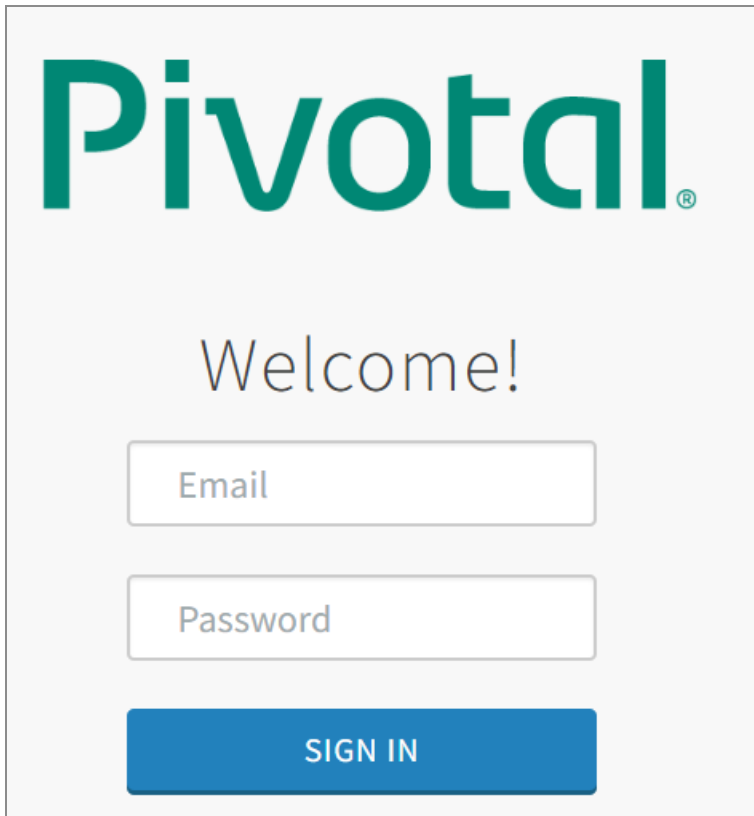
6. Bind the created user-defined service to the monitoring application.

> **i** **Note:** Ensure that the bwcemon_UAAClient service is created and the value for the AUTHENTICATION_MODE environment variable is set to UAA in the manifest file.

```
applications:
- name: com.tibco.bwce.monitoring.sample
  command: node server/node-server.js
  memory: 512M
  buildpack: https://github.com/cloudfoundry/nodejs-buildpack
  services:
   - bwcemon_mysql
   - bwcemon_UAAClient
  env:
    PERSISTENCE_TYPE: mysql
    AUTHENTICATION_MODE: UAA
```

7. After the monitoring application is deployed on Cloud Foundry, access the monitoring URL.

8. Enter the valid UAA user credentials and click **SIGN IN** to access the monitoring URL.

   You can log out from the monitoring UI by using the **Log Out** option available at the upper right corner of the monitoring UI.

Viewing Running Applications on Cloud Foundry

You can monitor the running application on Cloud Foundry by accessing the routable URL.

Before you begin

Ensure that the application is deployed on the container environment.

Procedure

1. Access the routable URL of the monitoring application in the browser to view the monitoring dashboard. You can view the following details for the running application:

   - Application name

- Status of the application
- Version of the application
- Application Instances

Application Monitoring on Docker

You can run the BusinessWorks application on Docker and enable Application Monitoring to monitor the application. The monitoring dashboard displays the running application details and application statistics.

Setting Up TIBCO BusinessWorks Container Edition Application Monitoring On Docker

The following steps describe how to set up an TIBCO BusinessWorks Container Edition application on Docker.

Before you begin

Download the `bwce_mon-<version>.zip` file from http://edelivery.tibco.com.

Procedure

1. Extract the `bwce_mon-<version>.zip` file.

2. Navigate to the `bwce_mon` directory and build the docker image by running the following command:

   ```
   docker build -t bwce/monitoring:latest .
   ```

```
REPOSITORY          TAG             IMAGE ID            CREATED             SIZE
bwce/monitoring     latest          018a62e2f6bb        55 seconds ago      99.8 MB
tibco/bwce          latest          f73ee3db6e78        2 days ago          352 MB
tibco/bwce          v2.3.0.23       f73ee3db6e78        2 days ago          352 MB
```

3. Ensure that MySQL or PostgreSQL or MS SQL Server is running and create the user with all the privileges. You can use a standalone Docker to run the monitoring application by passing the two environment variables.

    a. To start the application monitoring successfully, provide the following two environment variables:

    ```
    PERSISTENCE_TYPE
    DB_URL
    ```

    b. To run the monitoring application on a Docker container, run one of the following commands:

    For MySQL

    ```
    docker run -p 8080:8080 -e PERSISTENCE_TYPE="mysql" -e DB_
    URL="mysql://<user name:password>@<machine:port/database>" --name
    <containerName><monitoringImageName:tag>
    ```

    For PostgreSQL

    ```
    docker run -p 8080:8080 -e PERSISTENCE_TYPE="postgres" -e DB_
    URL="postgresql://<user name:password>@<machine:port/database>" --name
    <containerName><monitoringImageName:tag>
    ```

    For MS SQL Server

    ```
    docker run -p 8080:8080 -e PERSISTENCE_TYPE="mssql" -e DB_
    URL="mssql://<user name:password>@<machine:port/database>" --name
    <containerName><monitoringImageName:tag>
    ```

    To configure TIBCO BusinessWorks Container Edition Monitoring with the Azure Managed MS SQL Server, set the property DB_ENCRYPT to true. By default, this property is set to false.

    c. To configure TIBCO BusinessWorks Container Edition monitoring with MySQL database using SSL Configuration, you must configure the SSL at the Server and Client side. When starting the monitoring container, pass the

following environment variables in the Docker run command:

- `DB_SSL_MYSQL=true`

- `DB_SSL_CA=ca.pem`

- `DB_SSL_KEY=client-key.pem`

- `DB_SSL_CERT=client-cert.pem`

For example, run TIBCO BusinessWorks Container Edition monitoring by providing the hostname as an IP address.

```
docker run -p 9054:8080 -e PERSISTENCE_TYPE="mysql" -e DB_
URL="mysql://<user name:password>@<machine:port/database>" -e DB_SSL_
CA="true" -e DB_SSL_CA="ca.pem" -e DB_SSL_KEY="client-key.pem"
-e DB_SSL_CERT="client-cert.pem" --name
<containerName><monitoringImageName:tag>
```

4. To view the running container, run the following command:

```
docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
d67534c36237	bwce/monitoring:latest	"npm start"	6 seconds ago	Up 5 seconds	0.0.0.0:8080->8080/tcp	bwce-monitoring

5. After the monitoring container runs successfully, you can access the monitoring UI by using the following URL in the browser:

`http://<docker-host-ip>:8080`
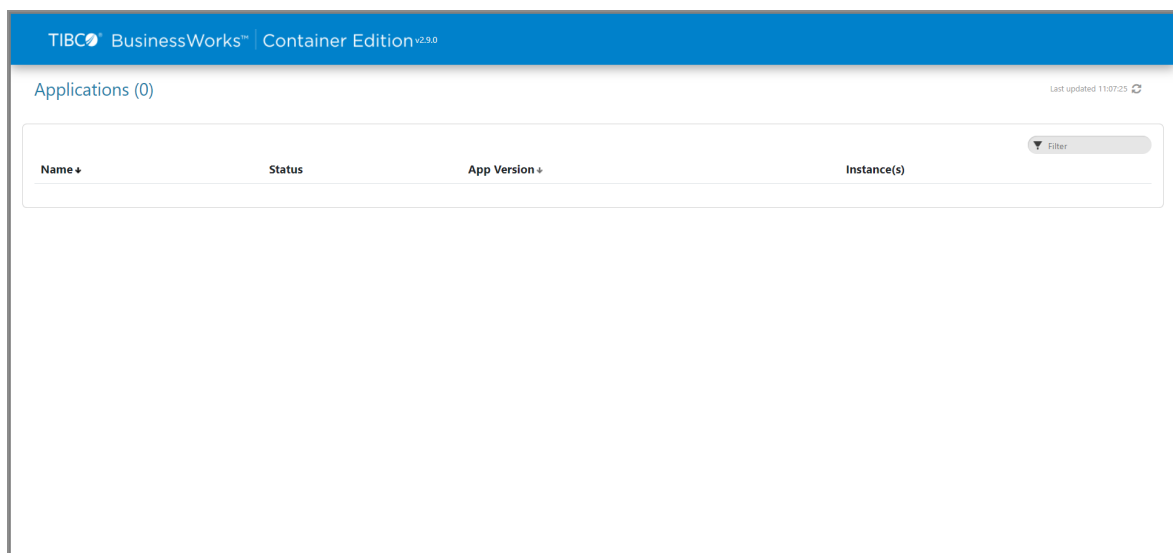
> ℹ **Note:** The environment variable for application monitoring has been changed to `PERSISTENCE_TYPE`.

Configuring and running a monitoring application with Oracle On Docker

Procedure

1. Create a folder on the same level as the root directory and name the folder `instantclient`.

2. Download Oracle client libraries from https://www.oracle.com/technetwork/topics/linuxx86-64soft-092277.html .

3. Add the zip file to the `instantclient` folder. For example, `instantclient-basic-linux.x64-19.5.0.0.0.zip`.

4. Rename the Dockerfile_Oracle to Dockerfile.

5. On line 20, change the `ENV CLIENT_FILENAME` according to the client download in step 2.

6. On line 27, change the `libclntsh.so` file name according to the client download and for 18.x or 19.x keep the line commented.

7. Run the following command for Docker build.

   ```
   docker build -t bwcemonoracle
   ```

8. Run the following command for Docker.

   ```
   docker run -p 18080:8080 -e PERSISTENCE_TYPE="oracle" -e DB_
   URL="oracle://user:pass@machine:1521/orclpdb1" bwcemonoracle
   ```

> ℹ **Note:** Use URL encoding for special characters while providing special characters for the database username or password.

Using Docker Compose for MySQL

You can use Docker Compose to run a monitoring application along with the MySQL database on Docker.

> ℹ️ **Note:** Running the monitoring application using Docker Compose is not recommended for production deployment.

Procedure

1. Navigate to the `bwce_mon` directory.

2. Run the following command to build the application monitoring image.

   ```
   docker-compose build
   ```

3. Run the following command, which downloads the MySQL image and configures the database with admin user and bwcemon database.

   ```
   docker-compose up mysql_db
   ```

```
mysql_db:
  image: mysql:latest
  container_name: mon-mysql
  network_mode: bridge
  ports:
    - "3306:3306"
  environment:
    MYSQL_DATABASE: bwcemon
    MYSQL_ROOT_PASSWORD: admin
  volumes:
    - mysql_data:/var/lib/mysql
    - ./dbscripts/mysql:/docker-entrypoint-initdb.d
```

```
C:\svn\bw6mon>docker-compose up mysql_db
Pulling mysql_db (mysql:latest)...
latest: Pulling from library/mysql
9f0706ba7422: Already exists
2290e155d2d0: Already exists
547981b8269f: Already exists
2c9d42ed2f48: Already exists
55e3122f1297: Already exists
abc10bd84060: Already exists
aa37081010bb: Pull complete
aadaa7b95bc6: Downloading [===>                              ]  5.406MB/79.61MB
8781ef2786a7: Download complete
b5c96613e09e: Download complete
3eac97813dda: Download complete
```

> **ⓘ Note:** Ensure that the volume is removed before setting up a MySQL database on Docker.

4. Run the following command to start the monitoring server on `8080` port.

```
docker-compose up mon_app
```

```
mon_app:
  build: .
  ports:
    - "8080:8080"
  links:
    - mysql_db
  environment:
    #DB_URL: mongodb://mongodb:27017/bwcemon
    #PERSISTENCE_TYPE: mongo
    DB_URL: mysql://admin:admin@mon-mysql:3306/bwcemon
    PERSISTENCE_TYPE: mysql
  network_mode: bridge
  -
```

```
C:\svn\641bw6mon>docker-compose up mon_app
mon-mysql is up-to-date
Creating 641bw6mon_mon_app_1 ...
Creating 641bw6mon_mon_app_1 ... done
Attaching to 641bw6mon_mon_app_1
mon_app_1    | npm info it worked if it ends with ok
mon_app_1    | npm info using npm@3.10.10
mon_app_1    | npm info using node@v6.9.5
mon_app_1    | npm info lifecycle com.tibco.bwce.monitoring@2.3.0~prestart: com.tibco.bwce.monitoring@2.3.0
mon_app_1    | npm info lifecycle com.tibco.bwce.monitoring@2.3.0~start: com.tibco.bwce.monitoring@2.3.0
mon_app_1    |
mon_app_1    | > com.tibco.bwce.monitoring@2.3.0 start /usr/src/app
mon_app_1    | > NODE_ENV=dev PORT=8080 node server/node-server.js
mon_app_1    |
mon_app_1    | info: Initializing mysql DB....
mon_app_1    | info: Listening on port 8080
mon_app_1    | info: table created
```

Using Docker Compose for PostgreSQL

You can use Docker Compose to run the monitoring application along with the PostgreSQL database on Docker.

Procedure

1. Navigate to the `bwce_mon` directory.

2. Run the following command to build the application monitoring image.

   ```
   docker-compose build
   ```

3. Run the following command, which downloads the PostgreSQL image and configures the database with admin user and bwcemon database.

   ```
   docker-compose up postgres_db
   ```

```
postgres_db:
   image: postgres:latest
   container_name: mon-postgres
   network_mode: bridge
   ports:
     - "5432:5432"
   environment:
     POSTGRES_DB: bwcemon
     POSTGRES_PASSWORD: admin
   volumes:
     - postgres_data:/var/lib/postgres
     - ./dbscripts/postgres:/docker-entrypoint-initdb.d
```

```
D:\WorkSpaces\BWCE\2.3.2\V30\bwce-mon>docker-compose up postgres_db
Creating mon-postgres ...
Creating mon-postgres ... done
Attaching to mon-postgres
mon-postgres    | The files belonging to this database system will be owned by user "postgres".
mon-postgres    | This user must also own the server process.
mon-postgres    |
mon-postgres    | The database cluster will be initialized with locale "en_US.utf8".
mon-postgres    | The default database encoding has accordingly been set to "UTF8".
mon-postgres    | The default text search configuration will be set to "english".
mon-postgres    |
mon-postgres    | Data page checksums are disabled.
mon-postgres    |
mon-postgres    | fixing permissions on existing directory /var/lib/postgresql/data ... ok
mon-postgres    | creating subdirectories ... ok
mon-postgres    | selecting default max_connections ... 100
mon-postgres    | selecting default shared_buffers ... 128MB
mon-postgres    | selecting dynamic shared memory implementation ... posix
mon-postgres    | creating configuration files ... ok
mon-postgres    | running bootstrap script ... ok
mon-postgres    | performing post-bootstrap initialization ... ok
mon-postgres    | syncing data to disk ...
mon-postgres    | WARNING: enabling "trust" authentication for local connections
mon-postgres    | You can change this by editing pg_hba.conf or using the option -A, or
mon-postgres    | --auth-local and --auth-host, the next time you run initdb.
mon-postgres    | ok
mon-postgres    |
mon-postgres    | Success. You can now start the database server using:
mon-postgres    |
mon-postgres    |     pg_ctl -D /var/lib/postgresql/data -l logfile start
mon-postgres    |
mon-postgres    | waiting for server to start....2017-11-07 11:18:56.854 UTC [36] LOG:  listening on IPv4 address "127.0.0.1", port 5432
mon-postgres    | 2017-11-07 11:18:56.854 UTC [36] LOG:  could not bind IPv6 address "::1": Cannot assign requested address
mon-postgres    | 2017-11-07 11:18:56.854 UTC [36] HINT:  Is another postmaster already running on port 5432? If not, wait a few seconds and retry.
mon-postgres    | 2017-11-07 11:18:56.859 UTC [36] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
mon-postgres    | 2017-11-07 11:18:56.878 UTC [37] LOG:  database system was shut down at 2017-11-07 11:18:56 UTC
mon-postgres    | 2017-11-07 11:18:56.885 UTC [36] LOG:  database system is ready to accept connections
mon-postgres    |  done
mon-postgres    | server started
mon-postgres    | CREATE DATABASE
```

> **Note:** Ensure that the volume is removed before setting up the PostgreSQL database on Docker.

4. Run the following command to start the monitoring server on `8080` port.

```
docker-compose up mon_app
```

```
mon_app:
  build: .
  ports:
    - "8080:8080"
  links:
    #- mysql_db
    ├ postgres_db
  environment:
    #DB_URL: mongodb://mongodb:27017/bwcemon
    #PERSISTENCE_TYPE: mongo
    #DB_URL: mysql://admin:admin@mon-mysql:3306/bwcemon
    #PERSISTENCE_TYPE: mysql
    DB_URL: postgresql://admin:admin@mon-postgres:5432/bwcemon
    PERSISTENCE_TYPE: postgres
  network_mode: bridge
```

```
D:\WorkSpaces\BWCE\2.3.2\V30\bwce-mon>docker-compose up mon_app
Starting mon-postgres ...
Starting mon-postgres ... done
Creating bwcemon_mon_app_1 ...
Creating bwcemon_mon_app_1 ... done
Attaching to bwcemon_mon_app_1
mon_app_1    | npm info it worked if it ends with ok
mon_app_1    | npm info using npm@3.10.10
mon_app_1    | npm info using node@v6.9.5
mon_app_1    | npm info lifecycle com.tibco.bwce.monitoring@2.3.0~prestart: com.tibco.bwce.monitoring@2.3.0
mon_app_1    | npm info lifecycle com.tibco.bwce.monitoring@2.3.0~start: com.tibco.bwce.monitoring@2.3.0
mon_app_1    |
mon_app_1    |
mon_app_1    | > com.tibco.bwce.monitoring@2.3.0 start /usr/src/app
mon_app_1    | > NODE_ENV=dev PORT=8080 node server/node-server.js
mon_app_1    |
mon_app_1    | info: Initializing postgres DB....
mon_app_1    | info: Listening on port 8080
mon_app_1    | info: noderegistry table created
mon_app_1    | info: ProcessInstanceLoggingStats table created
mon_app_1    | info: ActivityLoggingStats table created
mon_app_1    | info: process table created
```

Setting up TIBCO BusinessWorks Container Edition Application Monitoring for HTTPS Server on Docker

The following steps describe how to set up a TIBCO BusinessWorks Container Edition monitoring application on the HTTPS Server for Docker.

Before you begin

Download the `bwce_mon-<version>.zip` TIBCO BusinessWorks Container Edition monitoring zip file from http://edelivery.tibco.com.

Procedure

1. Extract the `bwce_mon-<version>.zip` file.

2. Navigate to the `bwce_mon` directory and add the keys and certificates files in the `certs` folder. Next, update the `https_config.json` file. For more information on updating the JSON file, see Updating HTTP Config JSON file.

   > ℹ **Note:** In the `https_config.json` file, ensure that either the "key" or "pfx" keys are present. If not, the HTTPS server fails to start.

3. Run the following command to build the application monitoring image.

   ```
   docker build -t bwce/monitoring:latest
   ```

4. To establish a connection between the monitoring application and the database pass the two environment variables.

   ```
   PERSISTENCE_TYPE
   DB_URL
   ```

   a. You must also provide the following environment variable to start the application monitoring on the HTTPS Server.

      ```
      HTTPS
      ```

      The value of the environment variable is true.

   b. To run the monitoring application on a Docker container on the HTTPS server, run the following command.

      For MySQL

      ```
      docker run -p 8080:8080 -p 443:443 -e PERSISTENCE_TYPE="mysql"
      -e DB_URL="mysql://<user name:password>@<machine:port/database>" -e
      HTTPS=true --name <containerName><monitoringImageName:tag>
      ```

      For PostgreSQL

```
docker run -p 8080:8080 -p 443:443 -e PERSISTENCE_
TYPE="postgres" -e DB_URL="postgresql://<user
name:password>@<machine:port/database>" -e HTTPS=true --name
<containerName><monitoringImageName:tag>
```

For MS SQL Server

```
docker run -p 8080:8080 -p 443:443 -e PERSISTENCE_TYPE="mssql"
-e DB_URL="mssql://<user name:password>@<machine:port/database>" -e
HTTPS=true --name  <containerName><monitoringImageName:tag>
```

5. Optional. If the monitoring application is using a self-generated CA certificate, this CA certificate should be added in the `<BWCE_HOME>`/docker/resources/addons/certs folder. The format for the certificate must be non-encrypted binary.

6. Run the following command to view the running container.

```
docker ps -a
```

7. After the monitoring container runs successfully, you can access the monitoring UI by using the following URL in the browser:

https://<docker-host-ip>:443
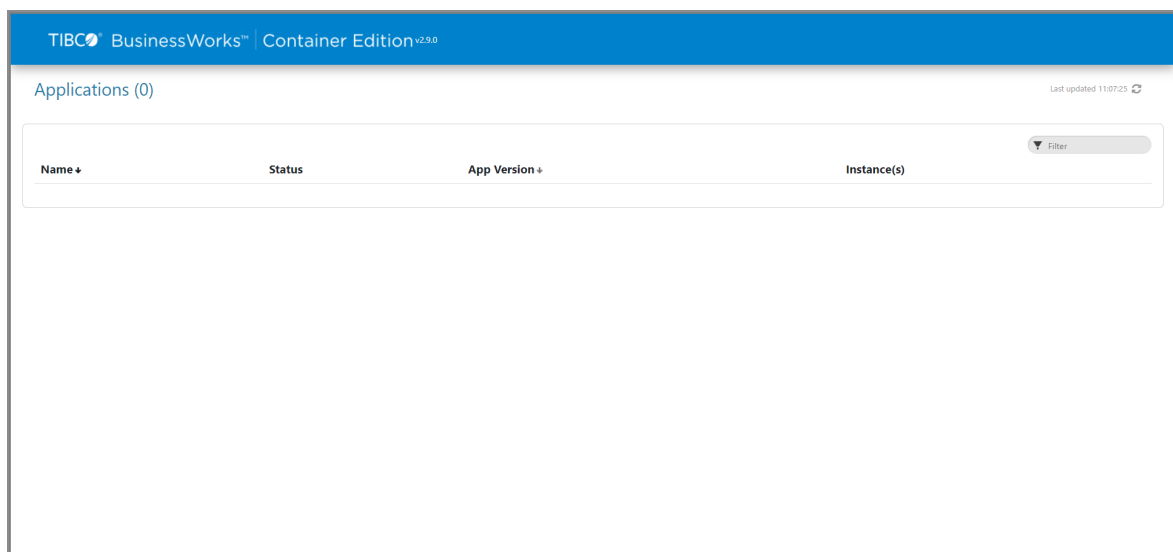
Updating the HTTPS Config JSON file

While updating the `https_config.json` file, different keys and values need to be passed in the JSON file. The following are the keys that can be passed in the JSON file:

Keys	Description
key	While using the `Base64` ASCII format key, pass the name of the file containing the private key in the `https_config.json` file in the following format: `"key": <file_name>` The default value is **server-key.pevm** **Note:** If the private key is encoded with a password, pass the `"passphrase"` key in the JSON file.
cert	While using a separate file for the certificate, pass the name of the file in the `https_config.json` file

Keys	Description
	in the following format: `"cert": <file_name>` The default value is **`server-cert.pem`**
`ca`	This key contains the name of a single file that holds all the `ca` chain certificates. The format is as follows: `"ca": <file_name>`
`pfx`	While using an encoded binary format key and certificate, pass the name of the file in the `https_config.json` file in the following format: `"pfx": <file_name>`

Keys	Description
	Note: If the "pfx" key is used, then the "`passphrase`" key is mandatory. The "`key`" and "`cert`" keys cannot be used along with the "`pfx`" key.
`passphrase`	The "`passphrase`" key is used when the key or certificate files are encoded with a password. The default value is an empty string ("").

Binding TIBCO BusinessWorks Container Edition to Monitoring Application on Docker

TIBCO BusinessWorks Container Edition application can be bound to the monitoring application by using the environment variable `BW_APP_MONITORING_CONFIG`.

Before you begin

Ensure that the monitoring application is running on the Docker container.

Procedure

1. Create a Dockerfile to deploy TIBCO BusinessWorks Container Edition an application on Docker. For more information about creating the Dockerfile, see "Application Development for Docker" in the *TIBCO BusinessWorks Container Edition Application Development*.

   ```
   FROM tibco/bwce:latest MAINTAINER Tibco ADD <application name>.ear / EXPOSE
   8080
   ```

2. Run the Docker terminal and navigate to the directory where the EAR and Dockerfile are stored.

3. Run the following command to build the application image:

   ```
   docker build -t <application name> .
   ```

4. In the Docker run command, set the environment variable BW_APP_MONITORING_ CONFIG to enable monitoring.

5. Run the command in the Docker terminal using Docker machine IP or using link.

   a. **Using the Application Monitoring URL**

   ```
   docker run -d -p 18050:8080 -e
   BW_APP_MONITORING_CONFIG='{"url":"http://<docker-host-
   IP>:<port>"}'<appname>:<tag>
   ```

> **ℹ Note:**
> - For Docker on the Windows platform, use the BW_APP_ MONITORING_CONFIG environment variable changes to:
>
>   ```
>   '{\"url\":\"http://<docker-host-IP>:8080\"}'
>   ```
>
> - To configure batch size and publish timer, you can pass additional parameters:
>
>   ```
>   -Dbw.monitor.batchsize=10
>   -Dbw.monitor.publishtimer=15000
>   ```
>
>   Batch size: This property specifies the batch size for the data. Process monitoring data is published in batches.
>
>   Publish timer: This property specifies the time interval for publishing Process Monitoring data.
>
> - To register with Monitoring UI applications running on the HTTPS Server, run the below command:
>
>   ```
>   docker run -d -p 18050:8080 -e
>   BW_APP_MONITORING_CONFIG='
>   {"url":"https://<docker-host-IP>:<https_
>   port>"}'<appname>:<tag>
>   ```

b. **Using Link on the Same Docker Host**

```
docker run --link=<name or id>:alias -p 18080:8080 -e BW_APP_
MONITORING_CONFIG='{"url":"http://<alias>:8080"}'
<applicationName>
```

> **ℹ Note:** The use of links is deprecated by Docker.

> **Note:** For Docker on the Windows platform, the BW_APP_
> MONITORING_CONFIG environment variable changes to:
>
> `'{\"url\":\"http://<alias>:8080\"}'`

```
xinpan-MBP15:HTTP xinpan$ docker run -P -e MESSAGE='Welcome to BWCE 2.3 !!!!!!!!' --link bwce-monitoring:bwcemonitoringservice -e BW_APP_MONITORING_CONFIG='{"url":
 "http://bwcemonitoringservice:8080"}' bwce-http-app
set bw.frwk.event.subscriber.metrics.enabled to true

BW_PROFILE is set to 'default.substvar'
TIBCO BusinessWorks Container Edition version 2.3.0, build V23, 2017-04-18
23:22:51.965 INFO  [main] com.tibco.thor.frwk - bwappnode TIBCO BusinessWorks Container Edition version 2.3.0, build V23, 2017-04-18 initialized using logging conf
ig /tmp/tibco.home/bwce/2.3/config/logback.xml
------------------------------------------
  Starting AppNode framework
------------------------------------------
23:23:08.000 INFO  [main] com.tibco.bw.frwk.engine.BWEngine - TIBCO-BW-FRWK-300002: BW Engine [Main] started successfully.
23:23:08.947 INFO  [main] com.tibco.thor.frwk - AppNode (OSGi Framework) started in 8 seconds
------------------------------------------
AppNode (OSGi Framework) started in 8 seconds
------------------------------------------
23:23:08.952 INFO  [Framework Event Dispatcher: Equinox Container: 00396749-2026-0017-1c5e-932b80584c42] com.tibco.thor.frwk.Deployer - TIBCO-THOR-FRWK-300001: Sta
rted OSGi Framework of AppNode [standalone] in AppSpace [standalone] of Domain [standalone]
23:23:09.202 INFO  [Framework Event Dispatcher: Equinox Container: 00396749-2026-0017-1c5e-932b80584c42] com.tibco.thor.frwk.Application - TIBCO-THOR-FRWK-300018:
Deploying BW Application [docker.http.application:1.0].
23:23:09.308 INFO  [Framework Event Dispatcher: Equinox Container: 00396749-2026-0017-1c5e-932b80584c42] com.tibco.thor.frwk.Application - Application bundle [dock
er.http.application_1.0.0.20151215223135 [423]] is resolved, but not started
23:23:09.312 INFO  [Thread-21] com.tibco.thor.frwk.Application - TIBCO-THOR-FRWK-300008: Stopped BW Application [docker.http.application:1.0]
23:23:09.318 INFO  [Framework Event Dispatcher: Equinox Container: 00396749-2026-0017-1c5e-932b80584c42] com.tibco.thor.frwk.Application - TIBCO-THOR-FRWK-300005:
Starting BW Application [docker.http.application:1.0]
23:23:09.329 INFO  [Framework Event Dispatcher: Equinox Container: 00396749-2026-0017-1c5e-932b80584c42] com.tibco.thor.frwk.Application - TIBCO-THOR-FRWK-300021:
All Application dependencies are resolved for Application [docker.http.application:1.0]
23:23:10.434 INFO  [Thread-26] com.tibco.thor.frwk.Application - TIBCO-THOR-FRWK-300006: Started BW Application [docker.http.application:1.0]
```
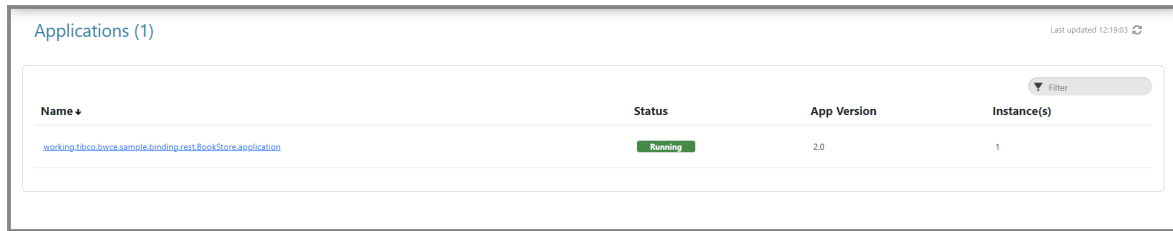
Viewing Running Applications on Docker

You can monitor the application running on Docker by accessing the Docker URL.

Before you begin

Ensure that the application is deployed on the Docker environment.

Procedure

1. Access the url `http://<docker-host-ip>:8080` or `https://<docker-host-ip>:<https_port>` to monitor the application on the TIBCO BusinessWorks Container Edition monitoring web UI. You can view the following details for the running application:

   - Application name

   - Status of the application

   - Version of the application

   - Application Instances

Applications (1)			Last updated 12:19:03 ⟳
			▼ Filter
Name ↓	Status	App Version	Instance(s)
working.tibco.bwce.sample.binding.rest.BookStore.application	Running	2.0	1

Application Monitoring on Kubernetes

You can run the BusinessWorks application on Kubernetes and enable Application Monitoring to monitor the application. The monitoring dashboard displays the running application details and application statistics.

Setting up TIBCO BusinessWorks Container Edition Application Monitoring on Kubernetes

The following steps describe how to set up the TIBCO BusinessWorks Container Edition application on Kubernetes.

Before you begin

Download the `bwce_mon-<version>.zip`TIBCO BusinessWorks Container Edition monitoring zip file from http://edelivery.tibco.com.

Procedure

1. Extract the `bwce_mon-<version>.zip` file.

2. Navigate to the `bwce_mon` directory and build the docker image by running the following command.

   ```
   docker build -t bwce/monitoring:latest.
   ```

3. Tag the monitoring application image by running the following command:

```
docker tag <monitoring_application_name> your_docker_container_registry/<your_
project_name>/<monitoring_application_name>
```

4. Push your monitoring application image to the Docker Container Registry. For example, to push your monitoring application docker image on the Google Cloud Registry, run the following command:

```
gcloud docker -- push gcr.io/<your_project_name>/<monitoring_application_name>
```

5. Confirm that the image is present in the Docker Container Registry.

6. Create the `manifest.yml` file and update the monitoring application image name. Ensure that the image name follows the following format:

```
<your_docker_container_registry>/<your_gcloud_project_name>/<monitoring_
application_image_name>
```

7. To configure a monitoring application with an external database, add the following two environment variables to the `manifest.yml` file.

- PERSISTENCE_TYPE

- DB_URL

The below sample is of a `manifest.yml` file:

```
apiVersion: v1
kind: Service
metadata:
  name: <monitoring_image>
  labels:
    app: <monitoring_image>
spec:
  type: LoadBalancer
  ports:
  - port: 80
    targetPort: 8080
  selector:
    app: <monitoring_image>
---
apiVersion: v1
```

```
kind: ReplicationController
metadata:
  name: <monitoring_image>
spec:
  replicas: 1
  selector:
    app: <monitoring_image>
  template:
    metadata:
      name: <monitoring_image>
      labels:
        app: <monitoring_image>
    spec:
      containers:
        - name: <monitoring_image>
          image: gcr.io/<project_name>/<monitoring_image>
          resources:
            limits:
                memory: 512Mi
            requests:
                memory: 512Mi
          imagePullPolicy: Always
          env:
          - name: PERSISTENCE_TYPE
            value:postgres
          - name: DB_URL
            value: postgres://<DB_USERNAME>:<PASSWORD>@<DB_IP>:<DB_
PORT>/<DB_NAME>
          ports:
            - containerPort: 8080
```

8. To create the monitoring service and replication controller, run the following
   command:

```
kubectl create -f manifest.yml
```

> **ⓘ Note:** Please find the monitoring sample manifest file for configuring the
> two environment variables.
>
> - PERSISTENCE_TYPE
>
> - DB_URL

9. To verify that the monitoring application has started successfully, run the following command:

```
kubectl logs pod-name
```

10. To get the external IP of the running monitoring service, run the following command:

```
kubectl get svc
```

Access the monitoring dashboard in the browser by using the external IP.

Setting up TIBCO BusinessWorks Container Edition Application Monitoring for HTTPS Server on Kubernetes

The following steps describe how to set up the TIBCO BusinessWorks Container Edition monitoring application on HTTPS for Kubernetes.

Before you begin

Download the `bwce_mon-<version>.zip` TIBCO BusinessWorks Container Edition monitoring zip file from http://edelivery.tibco.com.

Procedure

1. Extract the `bwce_mon-<version>.zip` file.

2. Navigate to the `bwce_mon` directory and add the keys and certificates files in the `certs` folder and update the `https_config.json` file. For more information on updating the HTTPS file, see Updating HTTP Config JSON file.

3. Tag the monitoring application image by running the following command:

```
docker tag <monitoring_application_name> your_docker_container_registry/<your_
project_name>/<monitoring_application_name>
```

4. Push your monitoring application image to the Docker Container Registry. For example, to push your monitoring application docker image on the Google Cloud Registry, run the following command:

```
gcloud docker -- push gcr.io/<your_project_name>/<monitoring_application_name>
```

5. Confirm that the image is present in the Docker Container Registry.

6. Create the `manifest.yml` file and update the monitoring application image name. Ensure that the image name follows the following format:

```
<your_docker_container_registry>/<your_gcloud_project_name>/<monitoring_application_image_name>
```

7. To configure a monitoring application with an external database, add the following two environment variables to the `manifest.yml` file.

   - PERSISTENCE_TYPE

   - DB_URL

   The below sample is of the `manifest.yml` file:

```
apiVersion: v1
kind: Service
metadata:
  name: <monitoring_image>
  labels:
    app: <monitoring_image>
spec:
  type: LoadBalancer
  ports:
  - port: 80
    name: http
    targetPort: 8080
  - port: 443
    name: https
    targetport: 443
  selector:
    app: <monitoring_image>
---
apiVersion: v1
kind: ReplicationController
metadata:
  name: <monitoring_image>
spec:
  replicas: 1
```

```
selector:
  app: <monitoring_image>
template:
  metadata:
    name: <monitoring_image>
    labels:
      app: <monitoring_image>
  spec:
    containers:
      - name: <monitoring_image>
        image: gcr.io/<project_name>/<monitoring_image>
        resources:
          limits:
              memory: 512Mi
          requests:
              memory: 512Mi
        imagePullPolicy: Always
        env:
        - name: PERSISTENCE_TYPE
          value:postgres
        - name: DB_URL
          value: postgres://<DB_USERNAME>:<PASSWORD>@<DB_IP>:<DB_
PORT>/<DB_NAME>
        - name: HTTPS
          value: 'true'
        ports:
          - containerPort: 8080
          - containerPort: 443
```

8. To create the monitoring service and replication controller, run the following
   command:

```
kubectl create -f manifest.yml
```

> ⓘ **Note:** Find the monitoring sample manifest file for configuring the
> environment variables.
>
> - PERSISTENCE_TYPE
>
> - DB_URL
>
> - HTTPS

9. To verify that the monitoring application has started successfully, run the following command:

```
kubectl logs pod-name
```

10. To get the external IP of the running monitoring service, run the following command:

```
kubectl get svc
```

Access the monitoring dashboard in the browser by using the external IP.

Binding TIBCO BusinessWorks Container Edition to Monitoring Application on Kubernetes

To bind TIBCO BusinessWorks Container Edition applications to monitoring applications on Kubernetes, follow the steps below:

Before you begin

- Ensure that you have configured the database for application monitoring.

- Ensure that you have created the TIBCO BusinessWorks Container Edition base docker image. For more information about creating a base docker image, see "Creating TIBCO BusinessWorks Container Edition base docker image" in the *TIBCO BusinessWorks™ Container Edition Application Development*.

- Ensure that you have created the TIBCO BusinessWorks Container Edition application docker image. For more information about building an application docker image, see "Building the application docker image" in the *TIBCO BusinessWorks™ Container Edition Sample*.

Procedure

1. Tag the application docker image by running the following command:

```
docker tag <application_image_name>your_docker_container_registry/<your_project_name>/<application_image_name>
```

2. Push your application image to the Docker Container Registry.

For example: To push your application docker image on the Google Cloud Registry, run the following command:

```
gcloud docker -- push gcr.io/<your_project_name>/<application_image_name>
```

3. Confirm that the image is present in the Docker Container Registry

4. Create the `manifest.yml` file and update the application image name. Ensure that the image name follows the following format:

```
<your_docker_container_registry>/<your_gcloud_project_name>/<image_name>
```

To configure a monitoring application with an external database, add the following two environment variables to the manifest file and port 80 with the monitoring URL:

- BW_APP_MONITORING_CONFIG
- BW_JAVA_OPTS

The below sample is of a `manifest.yml` file:

```
apiVersion: v1
kind: Service
metadata:
  name: <BW_APP_IMAGE_NAME>
  labels:
    app: <BW_APP_IMAGE_NAME>
spec:
  type: LoadBalancer
  ports:
  - port: 80
    targetPort: 8080
  selector:
    app: <monitoring_image>
---
apiVersion: v1
kind: ReplicationController
metadata:
  name: <BW_APP_IMAGE_NAME>
spec:
  replicas: 1
  selector:
    app: <BW_APP_IMAGE_NAME>
```

```
template:
  metadata:
    name: <BW_APP_IMAGE_NAME>
    labels:
      app: <BW_APP_IMAGE_NAME>
  spec:
    containers:
      - name: <BW_APP_IMAGE_NAME>
      image: gcr.io/<PROJECT_ID>/<BW_APP_IMAGE_NAME>
      resources:
        limits:
          memory: 512Mi
        requests:
          memory: 512Mi
      imagePullPolicy: Always
      env:
      - name: BW_PROFILE
        value:default
      - name: BW_LOGLEVEL
        value: DEBUG
      - name: BW_APP_MONITORING_CONFIG
        value: '{"url":"http://<MONITORING_IP>:80"}'
      ports:
        - containerPort: 8080
```

> **ⓘ** **Note:** To bind your TIBCO BusinessWorks Container Edition application to
> monitoring running on a HTTPS server, add the pertaining URL in the BW_
> APP_MONITORIN_CONFIG property

5. To create the monitoring service and replication controller, run the following
   command:

   ```
   kubectl create -f manifest.yml
   ```

6. To verify that the application has started successfully, run the following command:

   ```
   kubectl logs pod-name
   ```

7. To get the external IP of a running monitoring service, run the following command:

```
kubectl get svc
```

Access the monitoring dashboard in the browser by using the external IP

Viewing Running Applications on Kubernetes

You can monitor the running application on Kubernetes by accessing the external IP of the running service.

Before you begin

Ensure that the application is deployed on the Kubernetes environment.

Procedure

1. Access the web UI of the monitoring application by using the external IP of the running service of the monitoring application. You can view the following details for the running application:

   - Application name

   - Status of the application

   - Version of the application

   - Application Instances

Viewing Application Monitoring Dashboard

You can view App Instances, Endpoints, and Processes for a running application from the application monitoring dashboard.

Procedure

1. View the application status on the **Application** page. The monitoring dashboard displays the following information as per grouped in Cloud Foundry spaces:

   - Total number of application instances, the application instances (container),

and the running number of instances.

- Application version

- REST Doc URL

> **ⓘ Note:** The REST Doc URL is shown, if the application has a Swagger endpoint. The REST Doc URL for Docker is an internal endpoint and cannot be accessed externally. If you are exposing the TIBCO BusinessWorks Container Edition application externally, then the REST Doc URL can be accessed at `http://<External URL>/swagger.`

- The number of jobs created, running jobs, faulted jobs, canceled jobs, and scheduled jobs.

- To upload or download a Logback file click the Upload or Download link from the monitoring dashboard.

2. On the **Applications** page, select the running application you want to view.

3. To view app instances of an application, click the **App Instances** tab. You can also upload and download the Logback file from the **App Instances** tab.

4. Click the **Endpoints** tab to view the endpoints exposed by the application. The type of endpoint is displayed at the top of the tab.

> **ⓘ Note:** For endpoint URL in Docker-based platforms, replace the container ID with the external IP on which the TIBCO BusinessWorks Container Edition application is accessible.

5. Open the **Processes** tab to view an application process diagram.

> **ⓘ Note:** To view the process diagram, ensure that the version of the EAR file is 2.3.1 or later.

   a. Use the **Instance** drop-drown to select the instances of an application.

   b. You can enlarge a process diagram by clicking the **Zoom In** and **Zoom Out** button.
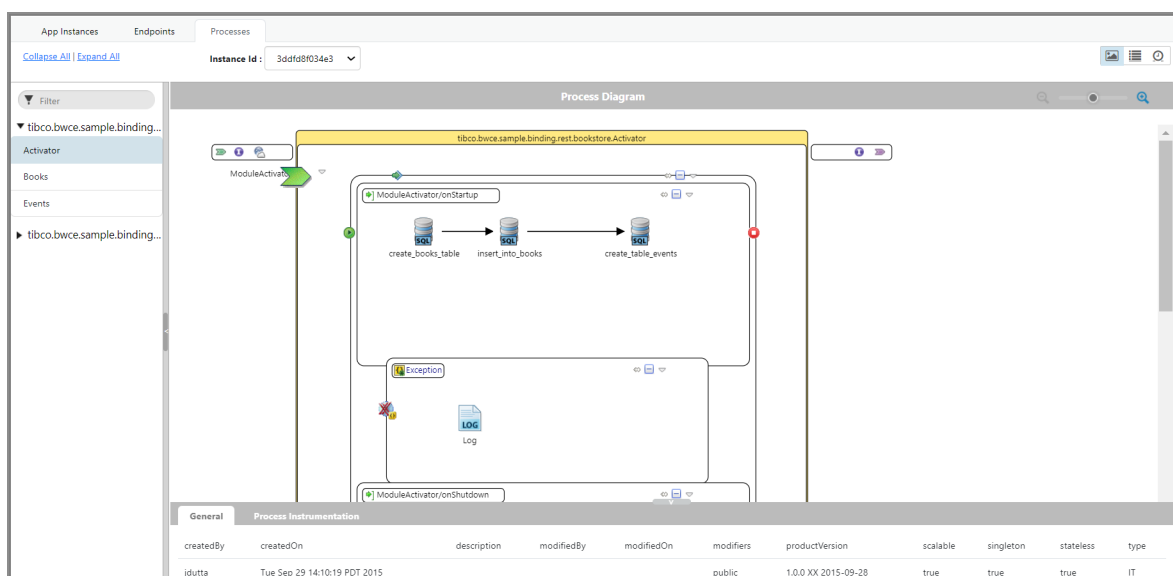
Application Statistics Collection

Application statistics collection can be enabled or disabled from the monitoring dashboard by setting the following property:

Property	Description
Process Instrumentation	To enable the monitoring of an application running on multiple app containers, click the application name and click **ON** the Process Instrumentation property. Process instrumentation statistics is collected for all applications.
	To enable process instrumentation statistics, set the `bw.frwk.event.subscriber.instrumentation.enabled` property to TRUE. Configure this BWEngine property in the `BW_JAVA_OPTS` environment variable to enable or disable the collection of statistical data for all processes running at application startup time.
	The process instrumentation statistics is disabled at the application startup time, if the property is set to FALSE.
	If the property is not set, the previous state of the process instrumentation persists.
Process Monitor	To enable process monitoring, click **ON** to view the process instances. This enables process monitoring for all the App instances.

> **Note:**
> 1. The **App Stats Collection** is ON or OFF, only when all the instances of an application are ON or OFF.
>
> 2. The newly registered instance shows the same statistics status as the **App Stats Collection** status.
>
> 3. The change in the **App Stats Collection** status triggers the same changes in all the instances of that application.
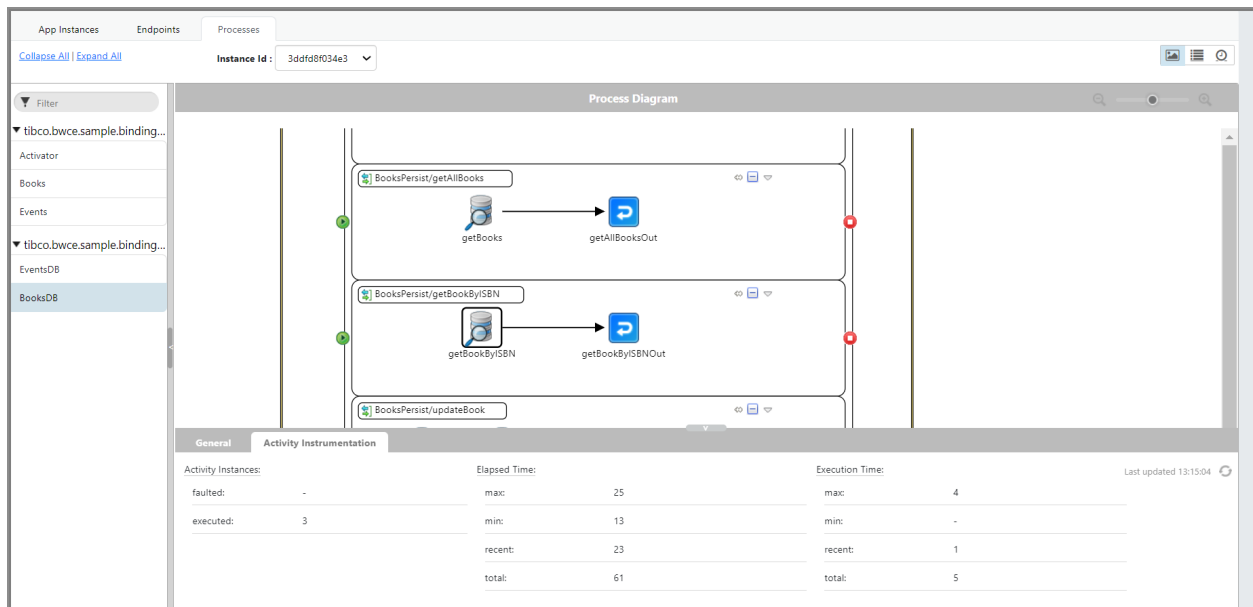
View Process Data

To view process instrumentation data, click an individual process. The process diagram, along with process instrumentation data is displayed.
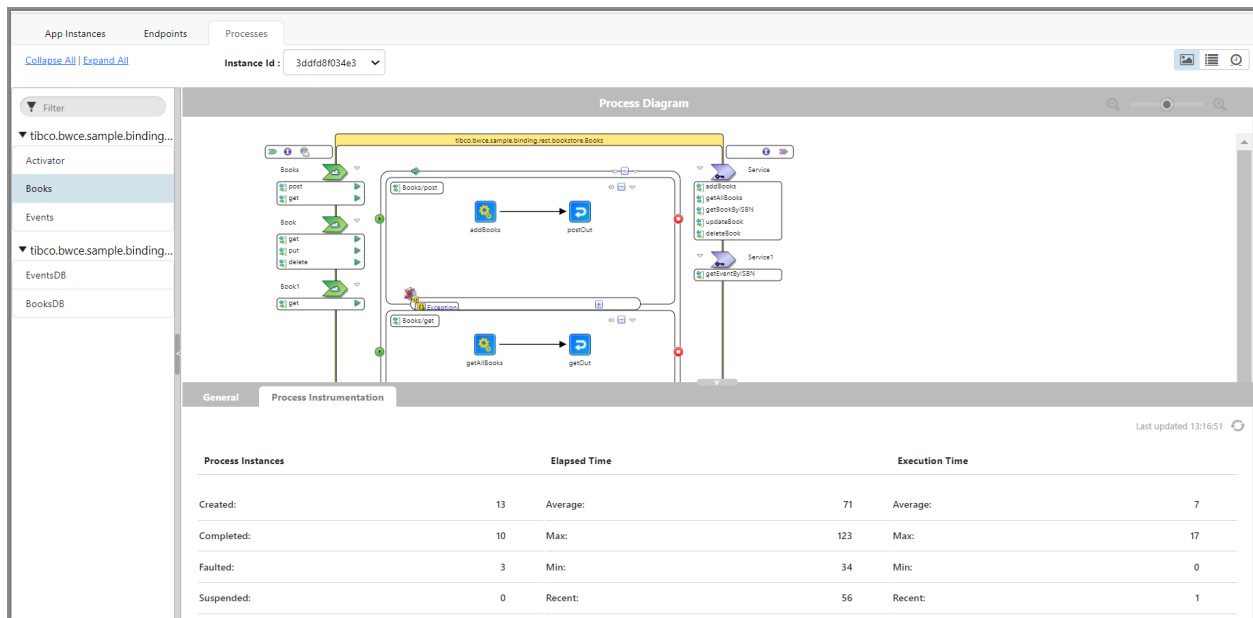
View Activity Data

Select the **Processes** tab to view the process diagram. You can view the activity instrumentation data by clicking an activity in the process diagram. The activity instrumentation data for all activities is visible by clicking the icon in the upper right corner of the **Processes** tab.

Viewing Application Properties

You can view the application properties of an application along with its value.

Monitoring Processes

Using the process monitoring feature you can observe and check the status of process instances from the Monitoring UI.

All the process instances in the application are grouped by packages. You can monitor the status of the process instances and subprocesses that were successfully executed, canceled, or faulted.

Details such as input data, output data, fault data, and other configuration details for the activities are also available by viewing the process diagram for the instances.

Enabling Process Monitoring

Process monitoring can be configured by using the environment variable BW_APP_ MONITORING_CONFIG. For more information, see Binding BusinessWorks Application to Monitoring Application on Docker or Binding BusinessWorks Application to Monitoring Application on CF or Binding BusinessWorks Application to Monitoring Application on Kubernetes

To access the landing page of process monitoring, go to the **Application Level 2** page, navigate to the **Process** tab, click the **Process Instance** icon .

All the instances, processes, and subprocesses of the selected application are displayed on the landing page.

You can begin monitoring your process instances once you enable the Process Monitor button after deploying the application.
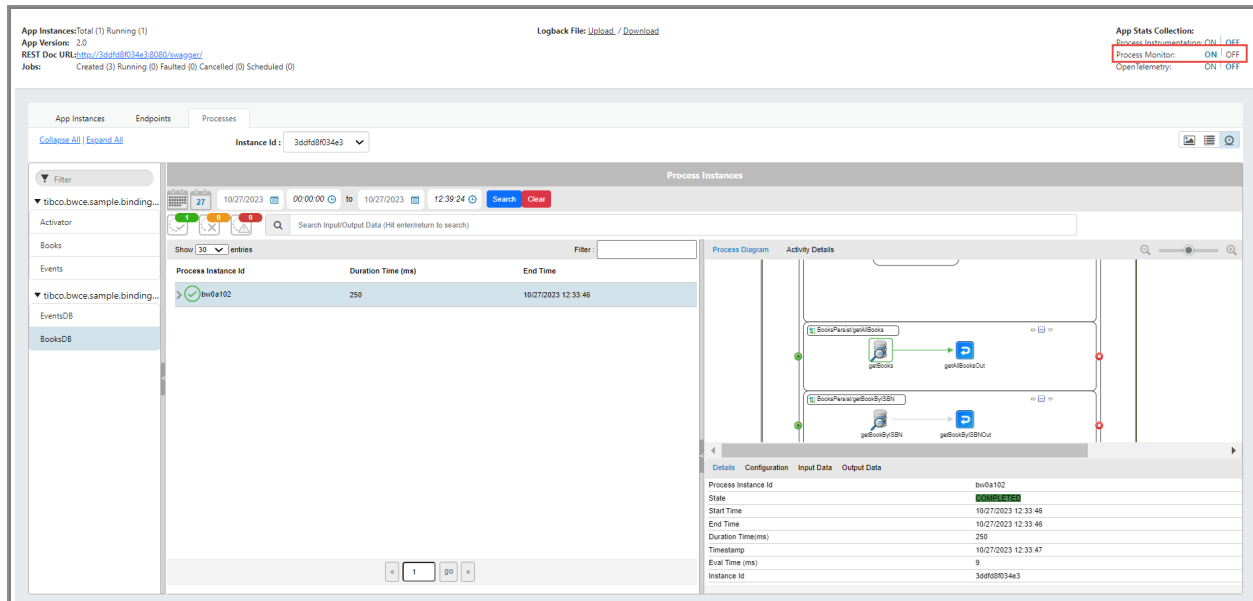
By default, all the instances in the selected process are displayed.

In the above example, click the process Books. Job data related to the Books process is displayed in a tabular form, and the process diagram of the process is also displayed.

In the Monitoring UI, the following details are displayed in the default view.

- **Process Instance Id** - displays all the (instance ids of the) process instances.

- **DurationTime (ms)** - displays the total time taken to run the process instance (in milliseconds).

- **EndTime**- displays the time when the process instance ended.

The columns displayed in the default view can also be customized to display additional information about the process instances. Use the **Select Columns** filter ⏢ to add the columns, **StartTime** and **EvalTime (ms)**.

The other filters provided in Monitoring UI are:

- Job Status filters - Job data can be filtered based on their completion status. Select the ✅ icon to filter the jobs that were completed. The ❌ icon displays only the jobs that are canceled. The ⚠ icon filters the jobs that are faulted.

- Filter - This filter searches through the column for values provided in this filter text box that are available on that page.

The process diagram and activity details for each process instance is displayed in the extreme right panel. Click the process instance in the second panel, and the process diagram for that instance is displayed.
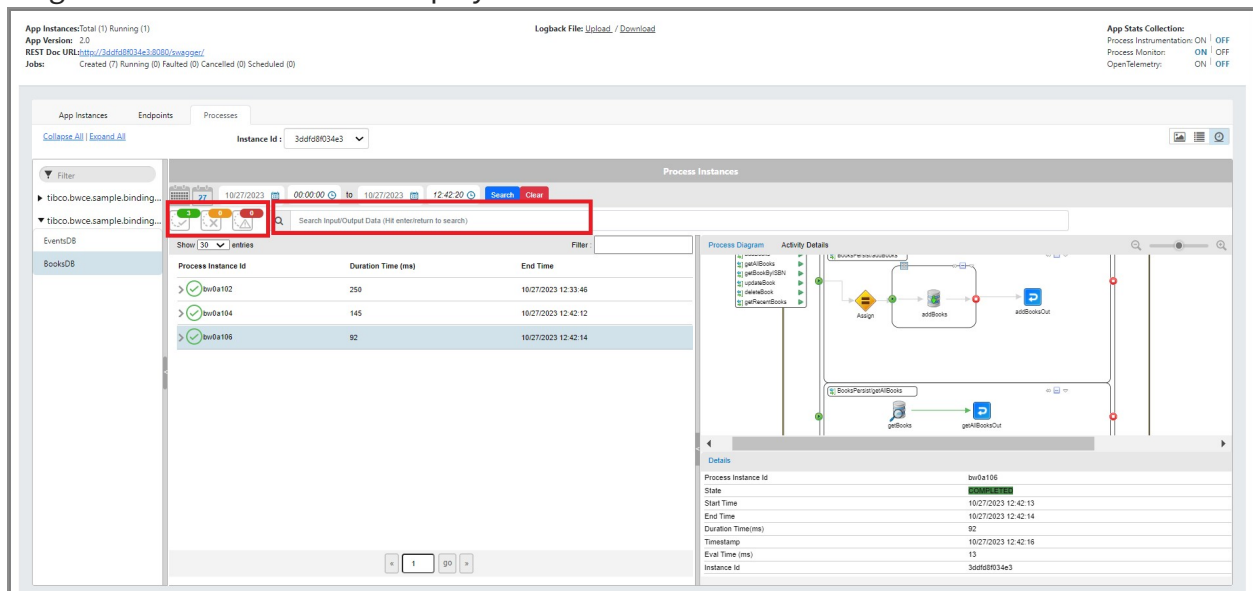
The **Activity Details** tab contains the **ActivityName**, **State**, **Timestamp**, **StartTime**, **DurationTime (ms)** and **EvalTime (ms)** of the particular activity selected.

The **State** of the selected activity can either be **Completed**, **Faulted** or **Canceled**. If the activity is in a canceled state, the details of only those activities are displayed before the Canceled state in the **Details** tab.

The **Details** tab, **Configuration**, **Input Data**, and **Output Data** tabs contain the configuration, input, and output details of the process instance.

> **Note:**
> - When a process contains multiple constructors and you minimize one while creating the EAR file, the activities in the constructor are not visible in the Monitoring UI. Expand the constructors and regenerate the EAR file to view the activities inside the constructors.
>
> - Fix any ActivityID-related warnings that are displayed in TIBCO Business Studio for BusinessWorks. Next, create the EAR file to ensure that the input and output data is correctly displayed.

In the image above, the **Output Data** tab displays the error due to which the process is faulted.

To enable the input and output of data storage for audit events when collecting statistics, use the following BWEngine REST API:

```
http://<host>:<appnode
port>/bwm/monitor.json/enableinputoutputdataforauditevents
```

To disable the input and output of data storage for audit events when collecting statistics, use the following BWEngine REST API:

```
http://<host>:<appnode
port>/bwm/monitor.json/disableinputoutputdataforauditevents
```

OpenTelemetry

OpenTelemetry is an open source, vendor neutral standard for distributed systems that are used to track the current state of the job. OpenTelemetry is a set of APIs, SDKs, tooling, and integrations designed to create and manage telemetry data such as traces and metrics.

> **ℹ Note:** OpenTelemetry does not support checkpointing.

For more information about OpenTelemetry, see OpenTelemetry documentation.

Enabling or Disabling OpenTelemetry

OpenTelemetry can be enabled or disabled through the Monitoring UI and the `BW_JAVA_OPTS` environment variable.

Monitoring UI

Enable or disable OpenTelemetry.
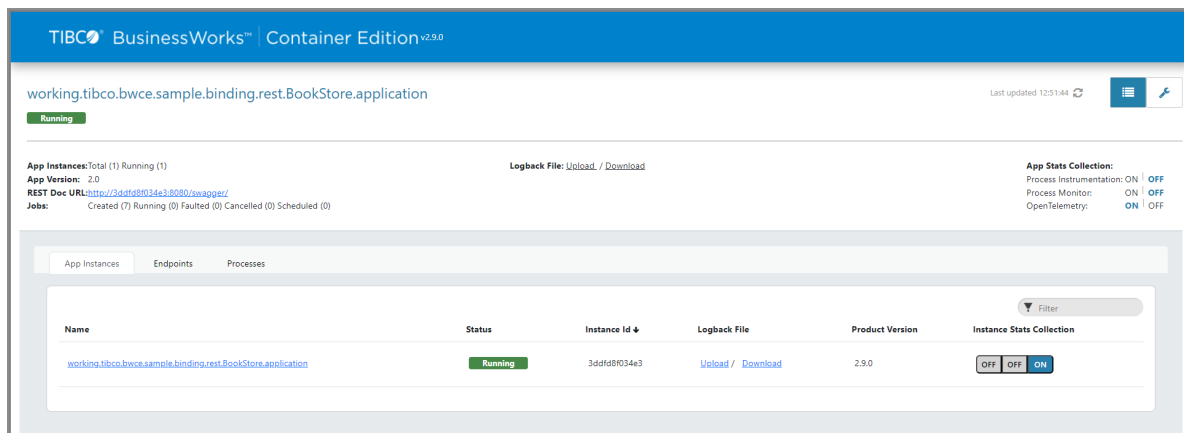
`BW_JAVA_OPTS` Environment Variable

Configure the following engine property in the `BW_JAVA_OPTS` environment variable while running the application to enable and disable OpenTelemetry:

- `bw.engine.opentelemetry.enable=true`.

> **ℹ Note:** By default, the property is false.

The following table describes how you can enable trace, metric, or both the variants simultaneously by setting up the BWEngine properties accordingly:

Properties	Trace	Metric
`bw.engine.opentelemetry.enable=true` `bw.engine.opentelemetry.trace.enable=true` or blank `bw.engine.opentelemetry.metric.enable=false` or blank	Enable	Disable

`bw.engine.opentelemetry.enable=true`
 	
`bw.engine.opentelemetry.trace.enable=false`
 	
`bw.engine.opentelemetry.metric.enable=true`
	Disable	Enable

`bw.engine.opentelemetry.enable=true`
 	
`bw.engine.opentelemetry.trace.enable=true` or blank
 	
`bw.engine.opentelemetry.metric.enable=true`
	Enable	Enable

OpenTelemetry via OpenTelemetry-Collector

1. Set up the OpenTelemetry-collector service. You can further integrate OpenTelemetry with a tracing service provider that is compliant with OpenTelemetry.

> **ℹ Note:** The OpenTelemetry via OpenTelemetry-Collector is the recommended approach.

To configure OpenTelemetry native properties for traces, set the `bw.opentelemetry.autoConfigured` system property to `True`. Once if this property is set to true, you can use the environment variables https://opentelemetry.io/docs/specs/otel/configuration/sdk-environment-variables/ listed out here.

> **ⓘ** **Note:** For this release, only the Span environment variable is supported.

To send data over the HTTP protocol for OpenTelemetry traces and metrics, set the `bw.engine.opentelemetry.http.protocol` system property to `True`.

To configure AWS_XRAY for OpenTelemetry traces and metrics, set the `bw.engine.opentelemetry.enable`, `bw.opentelemetry.aws.xrayIdGenerator`, and `bw.engine.opentelemetry.metric.enable` system property to `True`.

To enable logging for the OpenTelemetry traces and metrics, set the `io.opentelemetry` logger in the `logback.xml` file. Pass this logger as an environment variable. On enabling this logger, all the detailed information about traces and metrics is available in the logs.

Logback file example:

```
<logger name="io.opentelemetry">
      <level value="ALL"/>
</logger>
```

Container (environment variable) example:

`BW_LOGGER_OVERRIDES="io.opentelemetry=ALL"`

The OpenTelemetry TraceId and SpanId parameters can be mapped from an application's ProcessContext to any output activity such as Log or Write File.
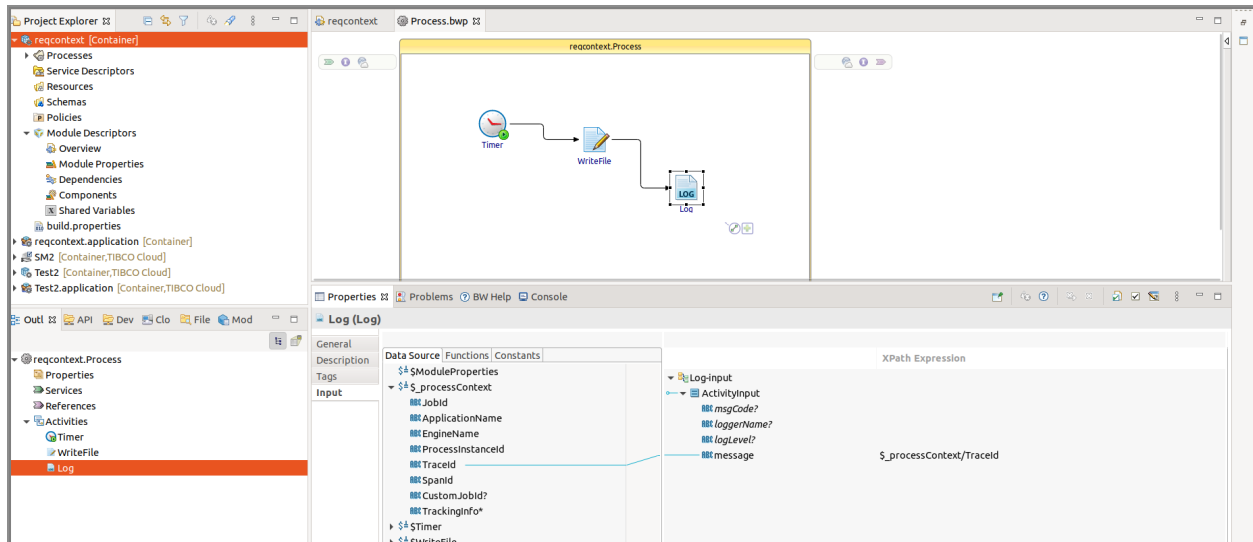
Traces

TIBCO BusinessWorks Container Edition supports all OpenTelemetry-compliant telemetry backends to display a span for each activity and process instance during job execution. Span corresponds to a process instance as well as an activity instance that has information such as ActivityName, JobID, process instance ID. For every process instance, a root span is created and all the activity instances are child spans of it.

Traces represent multiple related process instance spans.

> **Note:** In the case of HTTP palette, JMS palette, REST binding, SOAP binding, client, and server process the instances are shown in one trace. For all other palettes, every process instance is a trace.

> **Note:** Traces can be enabled by enabling the `bw.engine.opentelemetry.enable` property. By default, it is false.

You can configure the following properties specific to OpenTelemetry:

Property	Value	Description
bw.engine.opentelemetry.span.proces sor	Possible values are SPAN or BATCH. The default value is BATCH.	Configure Span Processor type.
bw.engine.opentelemetry.span.proces sor.delay	Value in milliseconds	Sets the delay interval between two consecutive exports.
bw.engine.opentelemetry.span.proces sor.timeout	Value in milliseconds	Sets the maximum time an export is allowed to run before being canceled.
bw.engine.opentelemetry.span.proces sor.batch.size	Integer value in kb.	Sets the maximum batch size for every export. This must be smaller or equal to maxQueuedSpans.
bw.engine.opentelemetry.span.proces sor.queue.size	Queue size in kb	Sets the maximum number of spans that are kept in the queue before start dropping. More memory than this value may be allocated to optimize queue access.
bw.engine.opentelemetry.span.sampl er	ON, OFF, 0.0 to 1.0. The default value is ON.	Configure Span Sampler type.
bw.engine.opentelemetry.span.export er	OTLP-GRPC	This property helps you to set a custom exporter injected as a service. The value of this property should be the component name of the service. For the Jaeger exporter, the value

Property	Value	Description
		for this property should be set to `com.tibco.bw.opentelemetry.exporter.jaeger.`
bw.engine.opentelemetry.span.exporter.endpoint	http://<host>:<port>	Sets the OTLP or Jaeger endpoint to connect to. **Note:** In the case of TIBCO BusinessWorks Container Edition, it is mandatory to set this property.
bw.engine.opentelemetry.span.exporter.timeout	Value in milliseconds	Sets the maximum time to wait for the collector to process an exported batch of spans.

Supported tags for querying on OpenTelemetry

Currently, the following tags are supported for querying on OpenTelemetry:

Tag	Description
SpanInitiator	Name of the process starter activity.
DeploymentUnitName	Name of the application.
DeploymentUnitVersion	Version of the application.
AppnodeName	Name of an AppNode on which an application is running.
Hostname	Name of the machine on which a TIBCO ActiveMatrix BusinessWorks™ application is running. This tag is applicable for Jaeger exporter UI.
IP	IP address. This tag is applicable for Jaeger exporter UI.

Tag	Description
ActivityName	Name of an activity in a process.
ActivityID	Id of an activity.
ProcessInstanceId	Process instance ID.
JobId	Job ID of the process.
ProcessName	Name of the process displayed for starter activities.

OpenTelemetry via Jaeger Span Exporter

1. Set up a Jaeger service.

2. To configure OpenTelemetry with Jaeger span exporter by using the following properties:

```
bw.engine.opentelemetry.enable=true

bw.engine.opentelemetry.span.exporter=com.tibco.bw.opentelemetry.exporter.jaeger

bw.engine.opentelemetry.span.exporter.endpoint=http://localhost:14250
```

> ⚠ **Caution:** With OpenTelemetry Span Exporter, the tags under process detail such as hostname, IP, Jaeger version are not displayed on the Jaeger UI. If you use the Jaeger exporter service instead of the default OpenTelemetry exporter service, the tags are visible on the Jaeger UI.

By default, the OpenTelemetry traces by using Jaeger Span Exporter and OpenTelemetry Collector are available on Jaeger UI at `http://localhost:16686/`.

OpenTelemetry Tags from Palettes

To get more information about the current job in execution, activity level tags are also supported. These tags are pre-defined tags.

The following sections show the list of pre-defined tags supported by each activity:

Basic Activities Palette

Activity name	Supported Tags
Invoke	• Service name • Operation Name

General Palette

Activity Name	Supported Tags
Confirm	Confirm Event
Call Process	• Spawned • Called Process Name
External Command	• Command • Environment
Log	Log Level
Sleep	Interval In MilliSec

File Palette

Activity Name	Supported Tags
Copy File	• From File • To File
Create File	File Name

Activity Name	Supported Tags
File Pollar	• File Name • Polling Interval(sec)
List Files	• File Name Pattern • Number of Files • Mode
Read File	• File Name • Content Style
Remove File	File Name
Rename File	• From File • To File
Write File	• File Name • Write As
Wait For File Change	• File Name • Polling Interval(sec)

FTP Palette

Activity Name	Supported Tags
FTP Change Default Directory	• peer.hostname • peer.port
FTP Delete File	• peer.hostname • peer.port
FTP Dir	• peer.hostname

Activity Name	Supported Tags
	• peer.port
FTP Get	• peer.hostname • peer.port
FTP Get Default Directory	• peer.hostname • peer.port
FTP Make Remote Directory	• peer.hostnam • peer.port
FTP Put	• peer.hostname • peer.port
FTP Quote	• peer.hostname • peer.port
FTP Remove Remote Directory	• peer.hostname • peer.port
FTP Rename File	• peer.hostname • peer.port
FTP SYS Type	• peer.hostname • peer.port

HTTP Palette

Activity Name	Supported Tags
HTTP Receiver	• peer.hostname • peer.port

Activity Name	Supported Tags
	• http.url • span.kind • error • ErrorMessage
Send HTTP Request	• span.kind • http.url • HTTPRequestQuery • HTTPPostDataType • HTTPCookiePolicy • http.method • IsSecureHTTP • error • ErrorMessage • ErrorCode • ErrorStatus
Send HTTP Response	• span.kind • http.status_code • peer.hostname • peer.port • http.method • peer.ipv4 • HttpServerProtocol • ContentType • IsSecureHTTP • error

Activity Name	Supported Tags
	• HTTPServerErrorMessage • HTTPServerErrorCode • ErrorCode • ErrorMessage
Wait For HTTP Request	• peer.hostname • peer.port • http.url • span.kind • error • ErrorMessage

Java Palette

Activity Name	Supported Tags
Java Invoke	• Class Name • Method Name • CleanUp method • Global Instance • Method Return • IsMultipleOutput • Construct Declared • Cache Declared
Java To XML	• Class Name • Constructor Declared • Cache Declared

Activity Name	Supported Tags
XML To Java	Class Name

JDBC Palette

Activity Name	Supported Tags
JDBC Call Procedure	• ActivitySharedResourceURL • ActivityIsOverrideSharedResource • ActivityOverrideSharedResourceUR • ActivityInTransaction • ActivityExecutionStatus
JDBC Query	• ActivitySharedResourceURL • ActivityIsOverrideSharedResource • ActivityOverrideSharedResourceUR • ActivityInTransaction • ActivityExecutionStatus
JDBC Update	• ActivitySharedResourceURL • ActivityIsOverrideSharedResource • ActivityOverrideSharedResourceURL • ActivityInTransaction • ActivityExecutionStatus
SQL Direct	• ActivitySharedResourceURL • ActivityIsOverrideSharedResource • ActivityOverrideSharedResourceURL • ActivityInTransaction • ActivityExecutionStatus

JMS Palette

Activity Name	Supported Tags
Get JMS Queue Message	• messaging.destination • MessagingStyle • MessageType • AcknowledgementMode
JMS Receive Message	• messaging.destination • MessagingStyle • MessageType • span.kind
JMS Request Reply	• messaging.destination • MessagingStyle • MessageType • span.kind
JMS Send Message	• messaging.destination • MessagingStyle • MessageType • span.kind
Reply to JMS Message	• MessagingStyle • MessageType • span.kind • ReplyQueue
Wait for JMS Request	• messaging.destination • MessagingStyle • MessageType

Mail Palette

Activity Name	Supported Tags
Receive mail	• peer.hostname • peer.port • From Address • Reply To Address • To Address
Send Mail	• peer.hostname • peer.port • From Address • Reply To Address • To Address • CC Address • BCC Address • Sent Date

Parse Palette

Activity Name	Supported Tags
Mime Parser	• InputStyle • OutputStyle
Parse Data	• FormatType • Encoding • LineLength • SkipBlankLines • ColumnSeperator

Activity Name	Supported Tags
	• StringValue or FileName - Depending on input type • NumberOfRecord
Render Data	• FormatType • LineLength • ColumnSeperator • FillCharacter

REST and JSON Palette

Activity Name	Supported Tags
Invoke REST API	• http.status_code • http.url • net.peer.name • net.peer.port • http.method • error • ErrorType • ErrorMessage
Parse JSON	• SchemaType • OutputRootElementName • IsBadgerfishEnabled • error • ErrorType • ErrorMessage

Activity Name	Supported Tags
Render JSON	• IsJsonRenderException - This tag is populated only when some exception occurs • SchemaType • RemoveRoot • IsBadgerfishEnabled • error • ErrorType • ErrorMessage
Transform JSON	• error • ErrorType • ErrorMessage

TCP Palette

Activity Name	Supported Tags
Read TCP Data	• Data Type • Timeout • net.peer.name • net.peer.port
TCP Open Connection	• net.peer.name • net.peer.port
Wait For TCP Request	• net.peer.name • net.peer.port
Write TCP Data	• Data Type

Activity Name	Supported Tags
	• net.peer.name • net.peer.port

XML Palette

Activity Name	Supported Tags
Parse XML	• IsOutputValidationEnabled • Input Style • error • ErrorType • ErrorMessage
Render XML	• IsInputValidationEnabled • Encoding • OutputStyle • DefaultNamespaceFormat • error • ErrorType • ErrorMessage
Transform XML	• InputOutputStyle • StyleSheet • error • ErrorType • ErrorMessage

OpenTelemetry Tags From SOAP Bindings

The following tags are supported for SOAP service and reference binding. Here, **Invoke** activity represents client-side tags and **Receive** activity represents server-side tags.

SOAP with HTTP

Side	Supported Tags
Service	• RequestURI • TransportType • http.method • peer.hostname • peer.port
Client	• TransportType • LocationURI • AttachementStyle • WSDLPort • ServiceName • OperationName

SOAP with JMS

Side	Supported Tags
Service	• ReplyTo • span.kind • messaging.destination • MessagingStyle • MessageType • Operation

Side	Supported Tags
Client	• TransportType • EndpointReference • ReplyTo • MessagingStyle • Service Name • Operation Name • messaging.destination • span.kind • MessageType

OpenTelemetry Tags From REST Binding

The following tags are supported for REST service and reference binding. Here, **Invoke** activity represents client-side tags and **Receive** activity represents server-side tags.

Side	Supported tags
Service	• http.url • isUsingSSL • error • errorMessage • errorStatus • net.peer.port • span.kind • net.peer.name • clientResponseFormat • http.method
Client	• http.url

Side	Supported tags
	• isUsingSSL
	• error
	• errorMessage
	• errorStatus
	• net.peer.port
	• http.status_code
	• span.kind
	• net.peer.name
	• isRequestBuffered
	• contentType
	• http.method

Custom Tags for OpenTelemetry

For OpenTelemetry, you can add custom tags. To add custom tags, use the **Tags** tab added in each activity in TIBCO Business Studio for BusinessWorks.
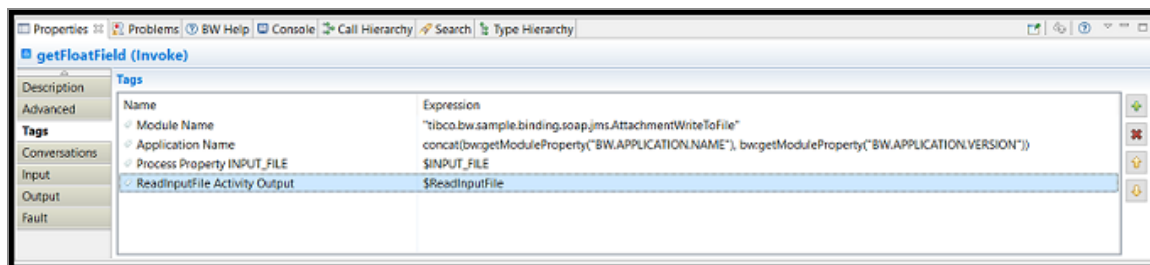
You can add **Expression** such as hardcoded values, XPath expressions for custom tags.

At run time, an asterisk (*) prefix is added for the names of the custom tags. It avoids the overriding of pre-defined engine tags.

Metrics

TIBCO BusinessWorks Container Edition can export metrics data to OpenTelemetry that can be used by the OpenTelemetry backend-supported client.

The following data are sent to OpenTelemetry:

- App data (TOTAL_JOB_COUNT, and so on)

- System data (ACTIVE_THREAD_COUNT, and so on)

- Process and Activity data (ACTIVITY_MAX_ELAPSED_TIME, and so on)

The following properties must be enabled for exporting the metrics data to OpenTelemetry:

- `bw.engine.opentelemetry.enable=true`

- `bw.engine.opentelemetry.metric.enable=true`

> **ⓘ Note:**
> - Metrics (fields and their values) are displayed only if the relevant data is available.
>
> - To enable the process and activity data in the OpenTelemetry metric, first enable the Process Instrumentation data property.
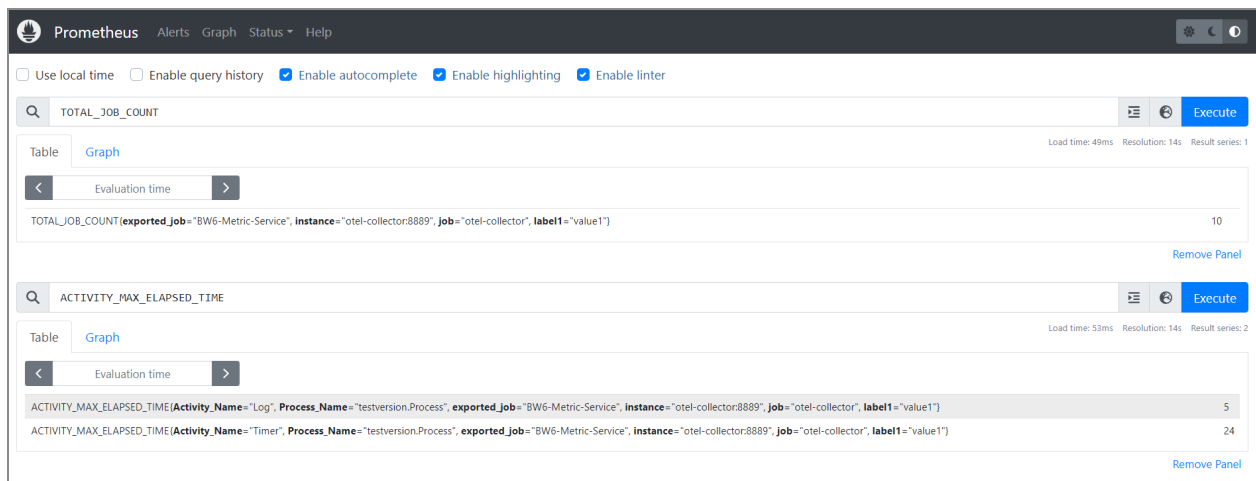
The `bw.engine.opentelemetry.metric.exporter.endpoint` property is used to set up the OpenTelemetry for metrics with remote machines or custom endpoints. When the OpenTelemetry Collector and the TIBCO BusinessWorks Container Edition application are running on two different machines, this property is added to get the metrics exported to the OpenTelemetry Collector at a specified IP/Host and Port.

If this property is not provided, the OpenTelemetry Collector picks `localhost:4317` as a default endpoint.

For example, `bw.engine.opentelemetry.metric.exporter.endpoint=http://<host ip>:<port>`.

Binding TIBCO BusinessWorks Container Edition Application to OpenTelemetry on Cloud Foundry

You can enable open tracing on the cloud foundry using an environment variable.

Before you begin

Ensure the OpenTelemetry agent is running.

Procedure

1. Create a `manifest.yml` file in the same directory where the application EAR file is exported.

2. Add environment variables for `BW_JAVA_OPTS` in `manifest.yml` file. The following is a sample of the manifest file:

   ```
   applications:
   -name: Httpapp
    memory: 1024M
    path: httpgreetings.application.ear
    timeout: 60
    buildpack: opentelemetry
   env:
    BW_LOGLEVEL: ERROR
    BW_PROFILE: default
    BW_JAVA_OPTS: "-Dbw.engine.opentelemetry.enable=true -

   Dbw.engine.opentelemetry.span.exporter.endpoint=http://192.168.225.
   224:4317"
   ```

3. In cf CLI, run the command `cf push` to deploy the application on the cloud foundry.

4. After the application is deployed successfully, it is registered under `<services>` in JAEGER UI.

Binding BusinessWorks Application to OpenTelemetry on Docker

You can enable open tracing on Docker using an environment variable.

Before you begin

Ensure the OpenTelemetry agent is running.

Procedure

1. Create a Dockerfile to deploy the TIBCO BusinessWorks Container Edition application on Docker. For more information about creating the Dockerfile, see "Application Development for Docker" in the *TIBCO BusinessWorks Container Edition Application Development*.

   ```
   FROM tibco/bwce:latest
   MAINTAINER Tibco
   ADD <application name>.ear/
   EXPOSE 8080
   ```

2. Run the Docker terminal and navigate to the directory where the EAR and Dockerfile are stored.

3. Run the following command to build the application image:

   ```
   docker build -t <application name>
   ```

4. In the docker run command, set the environment variables for `BW_JAVA_OPTS` to enable open tracing.

5. Run the command in the Docker terminal using Docker machine IP or using link.

   a.

```
docker run -d -p 18050:8080
-e BW_JAVA_OPTS= "-Dbw.engine.opentelemetry.enable=true -
Dbw.engine.opentelemetry.span.exporter.endpoint=http://localho
st:4317" <application name>
```

Smart Engine

TIBCO BusinessWorks Container Edition collects engine data on container. Based on the engine data collected, it generates HTML reports and provides analysis and recommendations for improving your application performance.

Generating Reports for Engine Data

You can generate various reports under some conditions such as increased memory usage, high CPU usage, more live threads for certain time.

By default, TIBCO BusinessWorks Container Edition generates reports in `{BWCE_HOME}\<product_version>\reports`. You can change the report location by setting the `bw.smartengine.report.path={path_to_report_folder}` property in the `BW_JAVA_OPTS` environment variable while running the application. You can also maintain a history or reports.

Before you begin

- Configure the following Engine property in the `BW_JAVA_OPTS` environment variable while running the application to enable and disable the smart engine.

  ```
  bw.smartengine.enabled=true
  ```

  You can also enable the smart engine feature dynamically by using the following REST API:

  ```
  http://<host>:<port>/monitor/systemproperties/enableSmartEngine
  ```

Procedure

1. To get the application statistics in reports, set the `BW_JAVA_OPTS` environment

variable `bw.smartengine.appStatistics.enabled` property to `true`. You can also enable the application statistics dynamically by using the following REST API:

```
http://<host>:<port>/monitor/systemproperties/enableSmartEngine?bw.
smartengine.appStatistics.enabled=true
```

> ⚠ **Warning:** You may observe performance degradation after setting the property.

2. Based on your requirements to get data in the report, several triggers are available. For more information, see the list of available Triggers.

3. To keep a specific number of reports for each type of performance use case at `{BWCE_HOME}\<product_version>\reports` location, set the below `BW_JAVA_OPTS` environment variables while running the application:

   `bw.smartengine.keepRecentReports.enabled=true`

   `bw.smartengine.keepRecentReports.size=5`

   By default, the smart engine stores the previous five reports for each performance use case.

Result

The report is stored at your specified location in the .zip format. The .zip file contains a report in an HTML format. The report has the following layout:

The HTML report has the following sections:

Section	Description
BW Applications	This section shows the TIBCO BusinessWorks Container Edition applications in a table. Each row shows the number of jobs for the TIBCO BusinessWorks Container Edition processes in an application, such as: • Created Jobs • Running Jobs • Completed Jobs • Faulted Jobs • Canceled Jobs After the TIBCO BusinessWorks Container Edition applications table, line charts are shown for each application, such as: • Total Job Count chart • New Job Count chart When an application has incoming HTTP requests, the Total HTTP Connector Calls chart, and the New HTTP Connector Calls chart are shown. For each TIBCO BusinessWorks Container Edition application, the TIBCO BusinessWorks Container Edition processes in the TIBCO BusinessWorks Container Edition application are shown in a table. Each row shows the number of jobs for a TIBCO BusinessWorks Container Edition process, such as • Created • Completed • Faulted • Suspended After the TIBCO BusinessWorks Container Edition processes table, line charts are shown for each TIBCO BusinessWorks Container Edition process, such as Total Job Count chart and New Job Count chart.

Section	Description
	For each TIBCO BusinessWorks Container Edition process, the TIBCO BusinessWorks Container Edition activities in the TIBCO BusinessWorks Container Edition process are shown in a table.
	Each row shows the runtime information of an activity such as:
	- Recent Status - Executed - Faulted - Recent Elapsed Time (ms) - Min Elapsed Time (ms) - Max Elapsed Time (ms) - Total Elapsed Time - Recent Activity Output Memory (bytes) - Min Activity Output Memory (bytes) - Max Activity Output Memory (bytes)
	The processes and activities statistics data is available when the application statistics feature is enabled. For example, `bw.smartengine.appStatistics.enabled=true`.
	The activity output for memory data is available when the engine analyzer feature is enabled. For example, `bw.engine.analyzer.subscriber.enabled=true`.
Operating System	This section shows the operating system information in a table, such as: - OS Name - OS Version - OS Architecture - Available Processors

Section	Description
	• Committed Virtual Memory • Free Physical Memory • Total Physical Memory • Free Swap Space • Total Swap Space • JVM Process CPU Time • JVM CPU Load • System CPU Load • System Load Average After this table, line charts are shown, such as • Free Physical Memory and Free Swap Space chart • JVM CPU Load and System CPU Load chart • System Load Average chart
Runtime Information	This section shows the runtime JVM information in a table, such as: • Process Name • Spec Name • Spec Vendor • Spec Version • VM Name • VM Version • VM Vendor • Management Spec Version • Start Time • Up Time

Section	Description
	• Class Path • Library Path • Input Arguments • System Properties
JVM Information	This section shows the overall JVM information in a table such as: • PID • Java Vendor • Java Name • Java Version • OS User • CPU Load • Up Time • GC Time • GC Count • GC Load • Max Heap • Used Heap • Used Non-Heap • Total Loaded Class Count • Thread Count • Peak Thread Count • Total Started Thread Count After this table, Top Threads information is shown in a table. Each row shows the data of a thread, such as:

Section	Description
	• TID
	• Name
	• State
	• Thread CPU Usage(%)
	• Thread Total CPU Usage(%)
	• Blocked Thread
	After that, Top Methods information is shown in a table. Each row shows the data of a method, such as:
	• Class Name
	• Method Name
	• Total CPU Time(ms)
Memory Information	This section shows the JVM memory information in a table, such as:
	• Max Heap Size
	• Committed Heap Size
	• Init Heap Size
	• Used Heap Size
	• Max Non-Heap Size
	• Committed Non-Heap Size
	• Init Non-Heap Size
	• Used Non-Heap Size
	After the table, line charts are shown, such as:
	• Heap Memory Usage chart
	• Non-Heap Memory Usage chart
Thread Information	This section shows the overall JVM thread information in a table, such

Section	Description
	as: • Thread Count • Daemon Thread Count • Peak Thread Count • Total Started Thread Count • Current Thread CPU Time • Current Thread User Time After this table, a Thread State Count table is shown. Each row shows the number of threads in a thread state, such as: • New • Runnable • Blocked • Waiting • Timed Waiting After that, line charts are shown, such as: • JVM Thread Count chart • JVM Thread State Count chart
Thread List	This section shows the JVM threads in a table. Each row shows the data of a thread, such as: • TID • Name • State • CPU Time(ms) • Allocated Heap Size

Section	Description
Thread Dump	This section shows the JVM threads dump in a table. Each row shows the thread dump of a thread, such as: • TID • Thread Name • Thread State • Thread Allocated Heap • Stack Trace
Class Loading	This section shows the JVM class loading information in a table, such as: • Loaded Class Count • Total Loaded Class Count • Unloaded Class Count After the table, a line chart of Classes Count is shown.
Objects Snapshot	This section shows the JVM objects in a table. Each row shows the data of an object, such as: • Number of instances • Allocated Heap Size • Class name
Analysis	This section shows the analysis of various performance use cases. When the triggers are evaluated, if a trigger condition is met for a performance use case, a corresponding analysis is provided and shown in the report.
Recommendations	This section shows the recommendations for various performance use cases. When the triggers are evaluated, if a trigger condition is met for a performance use case, related recommendations are provided by corresponding recommendation providers and shown in the report.

Triggers

You can populate the data in a report based on certain conditions. When those conditions are met, the trigger run's. Based on your requirements, you can modify threshold values by using REST APIs.

> **ⓘ Note:** Use `http://<host>:<port>/monitor` as a base URL for all the REST APIs provided.

The following triggers are available:

High CPU Trigger

ID	bw.montr.trigger.HighCPUTrigger
Threshold	highCpuThresholdPercent: 80 highCpuDurationMins: 5
Description	The trigger measures the high CPU usage situation. The trigger conditions are met when CPU usage is equal to or greater than 80% and the situation has lasted for more than (including) 5 minutes.

High Memory Trigger

ID	bw.montr.trigger.HighMemoryTrigger
Threshold	highMemoryThresholdPercent: 80 highMemoryDurationMins: 5
Description	The trigger measures the high memory usage situation. The trigger conditions are met when memory usage is equal to or greater than 80% and the situation has lasted for more than (including) 5 minutes.

Out of Memory Trigger

ID	bw.montr.trigger.OutOfMemoryTrigger
Threshold	outOfMemoryThresholdPercent: 95
Description	The trigger measures a very high memory usage situation (very close to out of memory). The trigger condition is met when memory usage is equal to or greater than 95%.

High Live Threads Trigger

ID	bw.montr.trigger.HighLiveThreadsTrigger
Threshold	highLiveThreadsThreshold: 500 highLiveThreadsDurationMins: 5
Description	The trigger measures a high number of live threads situation. The trigger conditions are met when the number of live threads (including both daemon and non-daemon threads) is equal to or greater than 500 and the situation has lasted for more than (including) 5 minutes.

High JMS Queue Pending Messages Trending Trigger

ID	bw.sharedresource.trigger.HighQueuePendingMessagesTrendingTriggerAction
Threshold	queuePendingMessagesCountMinValueThreshold: 1000 queuePendingMessagesTrendingPercentThreshold: 300 queuePendingMessagesDurationMinutesThreshold: 5
Description	The trigger measures the delay of processing JMS messages situation by checking the trending of pending messages in JMS queues that are accessed by activities in each TIBCO BusinessWorks Container Edition application. The trigger conditions are met when the pending messages in a JMS queue has

increased by more than (including) 300 percent in recent 5 minutes with a minimum pending messages of 1000.

High JMS Queue Pending Messages Count Trigger

ID	bw.sharedresource.trigger.HighQueuePendingMessagesCountTriggerAction
Threshold	queuePendingMessagesCountThreshold: 10000 queuePendingMessagesDurationMinutesThreshold: 5
Description	The trigger measures the delay of processing JMS messages situation by checking the number of pending messages in JMS queues that are accessed by activities in each TIBCO BusinessWorks Container Edition application. The trigger conditions are met when the number of pending messages in a JMS queue is equal to or greater than 10000 and the situation has lasted for more than (including) 5 minutes.

High JMS Queue Pending Messages Count Trigger

ID	bw.sharedresource.trigger.HttpConnectorAcceptorThreadCountThresholdTriggerAction
Threshold	-
Description	The default value of the HTTP Acceptor Thread Count Configuration on the HTTP Connector Shared Resource is 1. Jetty provides a formula for the maximum number of acceptor threads that can be allocated based on the available machine processors. The trigger checks the under-utilized acceptor threads. This means that the trigger condition is met when the configured value is less than the MAX allowed acceptor thread count value.

HTTP Connector Acceptor Thread Count Threshold Trigger

ID	bw.sharedresource.trigger.HttpConnectorAcceptorThreadCountThresholdTriggerAction
Threshold	-
Description	The default value of the HTTP Acceptor Thread Count Configuration on the HTTP Connector Shared Resource is 1. Jetty provides a formula for the maximum number of acceptor threads that can be allocated based on the available machine processors. The trigger checks the under-utilized acceptor threads. This means that the trigger condition is met when the configured value is less than the MAX allowed acceptor thread count value.

HTTP Connector Executor Threadpool Utilization Threshold Trigger

ID	bw.sharedresource.trigger.HttpConnectorExecutorThreadpoolUtilizationThresholdTriggerAction
Threshold	executorThreadpoolUtilizationThreshold: 85.0 highThreadpoolUtilizationDurationMinutesThreshold: 5.0
Description	The trigger measures the threadpool utilization while processing the incoming HTTP requests. The trigger conditions are met when the threadpool utilization is more than (including) 85% over the span of 5 minutes by default.

HTTP Connector Queue Utilization Threshold Trigger

ID	bw.sharedresource.trigger.HttpConnectorQueueUtilizationThresholdTriggerAction
Threshold	connectorThreadpoolQueueUtilizationThreshold: 85.0
Description	The blocking queue size for the Jetty server in TIBCO BusinessWorks Container

Edition 2.x can be set using the System property, bw.engine.http.jetty.blockingQueueSize=<Integer Value>. The trigger measures this jetty blocking queue utilization percentage. The trigger conditions are met when the blocking queue size is more than (including) 85% full by default.

Triggers REST API

This section has the following Triggers REST APIs:

- /triggers
- /triggers/{triggerId}/properties

/triggers

Method	GET
Description	Get a list of triggers of the smart engine.
Path Parameters	None
Query Parameters	None
Header Parameters	None
Output	- Code = 200 Message = "Returns a list of triggers." - Code = 503 Message = "Internal Server Error".

/triggers/{triggerId}/properties

Method	PUT
Description	Update the properties of a trigger.
Path Parameters	• Parameter: triggerId • Type: String(required) • Description: The id of a trigger
Query Parameters	None
Header Parameters	None
Body Parameters	{ "{propertyName1}": {propertyValue1}, "{propertyName2}": {propertyValue2}, ... "{propertyNameN}": {propertyValueN} }
Output	• Code = 200 Message = "Trigger's properties are updated." • Code = 503 Message = "Internal Server Error".
Sample Output	{ "highIdleTimeoutPerMinuteThreshold": 60, "highIdleTimeoutDurationMinutesThreshold": 5 } { "code": "200", "message": "Trigger's properties are updated.", "status": "success" }

REST API Reports

This section has the following reports on REST APIs:

- /reports

- /reports/generate

- /reports/{reportId}/download

- /reports/{reportId}/delete

- /reports/deleteall

/reports

Method	GET
Description	Get a list of reports
Path Parameters	None
Query Parameters	None
Header Parameters	None
Output	• Code = 200 Message = "Returns a list of reports." • Code = 503 Message = "Internal Server Error".
Sample Output	[{ "id": "Report-2021-08-26T12-56-50-0700", "date": "2021-08-26 12:56:50" }, { "id": "Report-2021-08-26T16-42-14-0700", "date": "2021-08-26 16:42:14" }]

/reports/generate

Method	GET
Description	Generate a report manually
Path Parameters	None
Query Parameters	- Parameter: engineData - Type: Boolean(Optional) - Description: Whether to generate a APPNODE_DATA file in the report zip file. By default, the value is false
Header Parameters	None
Output	- Code = 200 Message = "Reports are generated." - Code = 503 Message = "Internal Server Error".

Sample Output	{
	"code": "200",
	"message": "Reports are generated.",
	"status": "success"
	}

/reports/{reportId}/download

Method	GET
Description	Download a report zip file.
Path Parameters	Parameter: reportId

Method	GET
	• Type: String (required) • Description: The id of a report
Query Parameters	None
Header Parameters	None

Output	• Code = 200
	
	    Message = "Download a report zip file."
	
	• Code = 503
	
	    Message = "Internal Server Error".

Sample Output	{
	
	"code": "200",
	
	"message": "Reports are generated.",
	
	"status": "success"
	
	}

/reports/{reportId}/delete

Method	GET
Description	Delete a report.
Path Parameters	• Parameter: reportId • Type: String (required) • Description: The id of a report
Query Parameters	None
Header Parameters	None

Method	GET

Output	• Code = 200
	
	  Message = "Delete a report."
	
	• Code = 503
	
	  Message = "Internal Server Error".

Sample Output	{
	
	"code": "200",
	
	"message": "Report is deleted.",
	
	"status": "success"
	
	}

/reports/deleteall

Method	GET
Description	Delete all reports.
Path Parameters	None
Query Parameters	None
Header Parameters	None

Output	• Code = 200
	
	  Message = "Reports are deleted."
	
	• Code = 503
	
	  Message = "Internal Server Error".

Sample Output	{ "code": "200", "message": "Reports are deleted.",

Method	GET

	`"status": "success"`
	
	`}`

Properties REST API

This section has the following REST APIs properties:

- /systemproperties/enableSmartEngine?bw.smartengine.appStatistics.enabled=true

- /systemproperties/disableSmartEngine

/systemproperties/enableSmartEngine?bw.smartengine.appStatistics.enabled=true

Method	GET
Description	Enable smart engine
Path Parameters	None
Query Parameters	- Parameter: bw.smartengine.appStatistics.enabled. - Type: Boolean (optional). - Description: Whether to enable application statistics. The default value is false.
Header Parameters	None
Output	- Code = 200 Message = "System property is set with old value and new value." - Code = 503

Method	GET
	Message = "Internal Server Error".

Sample Output	{
	
	"code": "200",
	
	"message": "Smart engine is enabled.",
	
	"status": "success"
	
	}
	
	{
	
	"code": "200",
	
	"message": "Smart engine (with application statistics) is enabled.",
	
	"status": "success"
	
	}

/systemproperties/disableSmartEngine

Method	GET
Description	Disable smart engine
Path Parameters	None
Query Parameters	None
Header Parameters	None

Output	• Code = 200
	
	    Message = "System property is set with old value and new value."
	
	• Code = 503
	
	    Message = "Internal Server Error".

Method	GET
Sample Output	{
	"code": "200",
	"message": "Smart engine is disabled.",
	"status": "success"
	}

Running OSGi Commands

You can run commands to gather data about running AppNodes and applications. For more information, see **Using HTTP Client to Connect to the Runtime**.

Command Reference

- To view all commands, use

```
curl -v http://localhost:8090/bw/framework.json/osgi?command=help
```

- To view command syntax, use

```
curl -v
http://localhost:8090/bw/framework.json/osgi?command=help%20<comman
d_name>
```

For example,

```
curl -v
http://localhost:8090/bw/framework.json/osgi?command=help%20pauseap
p
```

The following table lists some of the commands.

OSGi Commands

Command	Description
`bw:dsr`	Diagnoses shared resource issues.
`bw:geticon`	Tests for availability of TIBCO BusinessWorks Container Edition activity icons with a given ID and type.
`bw:lais`	Retrieves statistics for activities that have been run in one of the processes for the application.
`bw:lapi`	Retrieves information about all process instances for the application based on the applied filters. **Note:** You can see the output of the `lapi` command on the console. The output can be exported in the CSV format.
`bw:las`	Lists all instantiated

Command	Description
	activities.
`bw:lat`	Lists all registered activity types.
`bw:lbwes`	Lists all subscribers that are currently listening to TIBCO BusinessWorks Container Edition statistics events.
`bw:le`	Prints information about TIBCO BusinessWorks Container Edition engines.
`bw:lec`	Prints information about TIBCO BusinessWorks Container Edition engine configurations.
`bw:lendpoints`	Lists endpoints exposed by the TIBCO BusinessWorks Container Edition engine.
`bw:les`	Lists all instantiated `EventSources.`
`bw:lmetrics`	Prints job metrics

Command	Description
	for applications running on the AppNode.
bw:lpis	Prints statistics of one of the processes that run for the application.
bw:lr	Lists all resource details.
bw:lrhandlers	Lists all resource handlers.
bw:lrproxies	Lists all resource proxies.
bw:startesc	Starts a collection of execution statistics for a given entity (activity/process) for applications.
bw:stopesc	Stops execution statistics collection of a given entity (process/activity) for applications.
bw:startpsc	Starts collection of process statistics for applications.
bw:stoppsc	Stops collection of process statistics

Command	Description
	for applications.
`bw:lapis`	Prints summary of an active process instance.
`frwk:appnodeprocessinfo`	Prints information about AppNode system processes.
`frwk:dc`	Delete a configuration with a given PID.
`frwk:dc`	Delete all configurations.
`frwk:la`	Print information about all applications.
`frwk:lap`	Print all application properties.
`frwk:lb`	List installed bundles matching a substring.
`frwk:lb`	List all installed bundles.
`frwk:lcfg`	Print all CAS configuration details.
`frwk:lp`	Print information about all known

Command	Description
	TIBCO BusinessWorks Container Edition processes.
`frwk:ll`	Print information about all libraries.
`frwk:lloggers`	Print all loggers currently configured on the AppNode.
`frwk:lp`	Print information about all known TIBCO BusinessWorks Container Edition processes.
`frwk:pauseapp`	Stop the process starters and their bindings and pause all jobs of an TIBCO BusinessWorks Container Edition application.
`frwk:resumeapp`	Start the process starters and their bindings and resume all jobs of an TIBCO BusinessWorks Container Edition application.

Command	Description
`frwk:setloglevel`	Sets the log level for a given logger.
`frwk:startcomps`	Start all process starters and their bindings of an TIBCO BusinessWorks Container Edition application.
`frwk:startps`	Start the process starters of an TIBCO BusinessWorks Container Edition application.
`frwk:stopps`	Stop the process starters of an TIBCO BusinessWorks Container Edition application.
`frwk:startapp`	Start a TIBCOTIBCO BusinessWorks Container Edition application gracefully.
`frwk:stopapp`	Stop a TIBCOTIBCO BusinessWorks Container Edition application gracefully.

Command	Description
`frwk:td`	Print a full thread dump.

> **Note:** To run some of the statistics retrieval commands such as `lapi`, you must first run the `startpsc` statistics activation command.

Disabling OSGi Commands

To disable OSGi commands, use the `bw.osgi.disable` property as an environment variable.

When the property `bw.osgi.disable=true` it disables the OSGi commands on the port and you can still start the AppNode, but it does not accept any OSGi requests.

Connecting to TIBCO BusinessWorks Container Edition Runtime using the HTTP Client

You can connect to the TIBCO BusinessWorks Container Edition runtime environment by using the HTTP Client. The preferred way to use is curl.

Procedure

1. Open a terminal window to start an interactive session with the application container.

   Run the following commands for applications deployed in VMware Tanzu

   ```
   cf ssh <application_name>
   ```

   Run the following commands for applications deployed in Docker

```
docker exec -it <container_id> bash
```

2. Run the following OSGi command in the container terminal window in the following format:

```
curl -v localhost:8090/bw/framework.json/osgi?command=<osgi_
command>
```

For example:

- To print information about BWEngine:

```
curl -v http://localhost:8090/bw/framework.json/osgi?command=le
```


> **ⓘ Note:** For applications deployed in Cloud Foundry, if the applications are not using HTTP connection resource or FTL connection resource or TCP connection resource then port 8080 is used for connecting to OSGI.

- To pause all jobs of TIBCO BusinessWorks Container Edition applications:

```
curl -v
http://localhost:8090/bw/framework.json/osgi?command=pauseapp%2
0-v
%201.0%20tibco.bw.sample.binding.rest.BookStore.application
```

Auto Collecting Engine Data

The collection of data requires multiple engine API (OSGi commands). These APIs are invoked internally and output is exported in file format at a specified location.

A REST API is provided to collect engine data. Invoke the REST API as POST:
`http://<host>:<port>/bw/framework.json/collect/.`
You can run the REST API commands similar to the OSGi commands. For more information, see Using HTTP Client to Connect to the Runtime.

The default path in TIBCO BusinessWorks Container Edition is:
`user.dir\..\debug\APPNODE_DATA_<TIME_STAMP>.zip`

REST API

API context	`http://<host>:<port>/bw/framework.json/collect/{operation}`
Method	POST
Authorization required	YES
Header-parameter	login
Operations	• `ALL` • `INCLUDE` • `EXCLUDE` • `DOWNLOAD` • `LIST` • `DELETE` For example: `http://<host>:<port>/bw/framework.json/collect/ALL`

The operation details are as follows:

Operation	Description
ALL	This API is used for running the default set of operations. The default set of operations is as follows: ["THREAD_DUMP", "HEAP_DUMP", "VM_ARGUMENTS","ENVIRONMENT_ VARIABLES", "SYSTEM_PROPERTIES", "THREAD_SNAPSHOT","MEMORY_ SNAPSHOT", "SYSTEM_PROCESS_INFORMATION","CPU_ INFORMATION","LMETRICS", "LCFG", "LP", "LA", "LENDPOINTS","LAPI *"]
INCLUDE	This API accepts a list of commands or operations as an input in the form of a JSON list. Only the listed operations run.
EXCLUDE	This API accepts a list of commands or operations as an input in the form of a JSON list. All default set operations excluding the set of operations given as input runs.
DOWNLOAD	This API is available to download all collected data as a stream APPLICATION_ OCTET_STREAM
LIST	This API is available to list the files present.
DELETE	This API is available to delete data files created.

Header Parameter	Description
PATH	An optional parameter to provide a directory path where the data is collected or is downloaded.
OVERRIDE	An option for collect data operation [ALL, INCLUDE, EXCLUDE], where the data collected previously is overwritten by the new data. The default value is TRUE.
ALL	An option for operation DOWNLOAD, where all files present are compressed at one file with the name APPNODE_DATA.zip and downloaded at once. The default value is FALSE.

Header Parameter	Description
DOWNLOADANDDELETE	An option for operation DOWNLOAD, where the file is deleted after the download operation. The default value is FALSE.
LOGIN	This option is required for the authorization of the user. This option is mandatory. For TIBCO BusinessWorks Container Edition use `login = admin`.

API consumes entity: INPUT

Required header parameter: `Content-Type=application/json`

JSON list of commands: Sample input: `["command1", "command2"]`.

Applicable for `INCLUDE` and `EXCLUDE` operations.

To use the REST API on Docker or Kubernetes with default settings, applications must be deployed with root users. Non-root users can use the PATH header parameter to collect data at the given path.

On Cloud Foundry, the REST API supports route URL with HTTP Protocol only.

The options to copy data from container to host machine in TIBCO BusinessWorks Container Edition are as follows:

- **Mount Volume to the container:**

  Docker command: `docker run -v <host_dir>:<container_dir> <image_name>`

- **Copy command to copy a file:**

  Docker command: `docker cp <containerID>:<file_path> <host_destination>`

- **Using the REST API:**

  Docker command: `REST API context path: http://<host>:<port>/bw/framework.json/collect/download`

Updating Flow Limit Dynamically

You can update the Flow limit value dynamically without restarting an application. Additionally, you can use the following REST API:

The base path for all REST APIs exposed is `http://<host or IP address>:<port>/` where port is of running AppNode.

bw/app.json/updateflowlimit/

Method	POST
Description	Update the Flow limit without restarting an application.
Path Parameters	None
Query Parameters	- Parameter: Flow limit. - Type: Integer (mandatory). - Description: The new value of the Flow limit. - Parameter: name. - Type: String. - Description: Application name. This property is mandatory for BW 6.x applications but it is optional when using for BWCE or TCI applications. - Parameter: version. - Type: Integer. - Description: Application version. This property is mandatory for BW 6.x applications but it is optional when using for BWCE or TCI applications. - Parameter: component. - Type: String (Optional. - Description: Component name of an application.

For example	http://<host or IP address>:<port>/bw/app.json/updateflowlimit?flowLimit=<new_flow_limit>&name=<app_name>.application&version=<app_version>&component=<component_name>

bw/app.json/flowlimit/

Method	GET
Description	Get the latest Flow limit applied to the application or the component without restarting an application.
Path Parameters	None
Query Parameters	- Parameter: name. - Type: String. - Description: Application name. This property is mandatory for BW 6.x applications but it is optional when using for BWCE or TCI applications. - Parameter: version. - Type: Integer. - Description: Application version. This property is mandatory for BW 6.x applications but it is optional when using for BWCE or TCI applications. - Parameter: component. - Type: String (Optional). - Description: Component name of an application.
For example	http://<host or IP address>:<port>/bw/app.json/flowlimit?name=<app_name>.application&version=<app_version>&component=<component_name>

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 Product Documentation website, mainly in HTML and PDF formats.

The 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 BusinessWorks™ Container Edition page:

- *TIBCO BusinessWorks™ Container Edition Release Notes*

- *TIBCO BusinessWorks™ Container Edition Installation*

- *TIBCO BusinessWorks™ Container Edition Application Development*

- *TIBCO BusinessWorks™ Container Edition Application Monitoring and Troubleshooting*

- *TIBCO BusinessWorks™ Container Edition Bindings and Palettes Reference*

- *TIBCO BusinessWorks™ Container Edition Concepts*

- *TIBCO BusinessWorks™ Container Edition Error Codes*

- *TIBCO BusinessWorks™ Container Edition Getting Started*

- *TIBCO BusinessWorks™ Container Edition Migration*

- *TIBCO BusinessWorks™ Container Edition Performance Benchmarking and Tuning*

- *TIBCO BusinessWorks™ Container Edition REST Implementation*

- *TIBCO BusinessWorks™ Container Edition Refactoring Best Practices*

- *TIBCO BusinessWorks™ Container Edition Samples*

How to Contact Support for TIBCO Products

You can contact the Support team in the following ways:

- To access the Support Knowledge Base and getting personalized content about products you are interested in, visit our product Support website.

- To create a Support case, you must have a valid maintenance or support contract with a Cloud Software Group entity. You also need a username and password to log in to the product Support website. If you do not have a username, 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

Cloud SG 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 a specific version of Cloud SG software on a specific operating system platform.