TIBCO ActiveMatrix BusinessWorks™

Administration

Software Release 6.6.1
May 2020

Document Updated: July 2020

TIBCO®

Important Information

Contents

Figures

TIBCO Documentation and Support Services

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the TIBCO Product Documentation website, mainly in HTML and PDF formats.

The TIBCO Product Documentation website is updated frequently and is more current than any other documentation included with the product. To access the latest documentation, visit https://docs.tibco.com.

Product-Specific Documentation

Documentation for TIBCO products is not bundled with the software. Instead, it is available on the TIBCO Documentation site.

Access the following *TIBCO ActiveMatrix BusinessWorks™* guides on the TIBCO Documentation site:

- Concepts
- Installation
- Getting Started
- Application Development
- Administration
- Bindings and Palettes Reference
- Samples
- Error Codes
- Migration
- Performance Benchmarking and Tuning
- REST Reference Guide
- Refactoring Best Practices

How to Contact TIBCO Support

You can contact TIBCO Support in the following ways:

- For an overview of TIBCO Support, visit http://www.tibco.com/services/support.
- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the TIBCO Support portal at https://support.tibco.com.
- For creating a Support case, you must have a valid maintenance or support contract with TIBCO. You also need a user name and password to log in to https://support.tibco.com. If you do not have a user name, you can request one by clicking Register on the website.

How to Join TIBCO Community

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

Administration Architecture Overview

Applications are deployed into runtime environments and managed using the bwadmin utility.

TIBCO ActiveMatrix BusinessWorks™ provides a flexible framework that allows you to scale your runtime environment as needed. The runtime also provides an option to execute the engine so that the risk of a single point of failure when running an application is reduced.

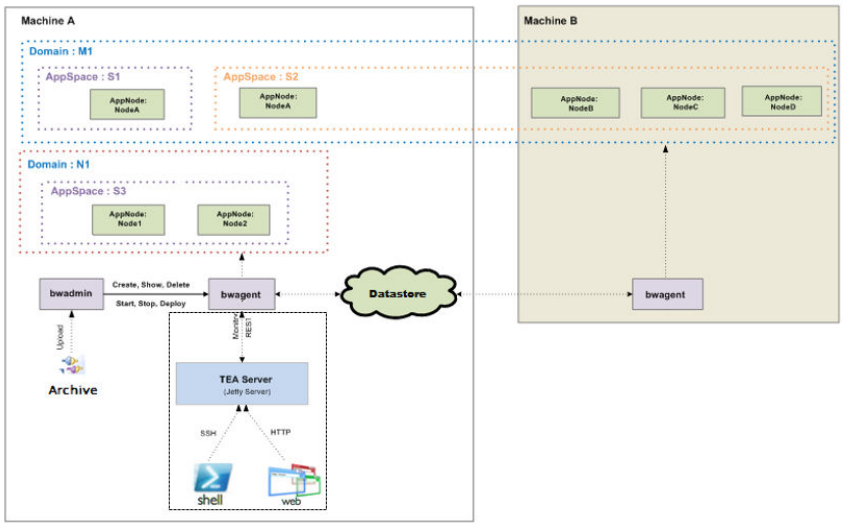
The following are the key administrative components:

- An Application Archive is the deployment unit for an application that is generated in TIBCO Business Studio™ for BusinessWorks™.
- A domain is a logical group that provides an isolated environment for applications and their resources to reside.
- An AppSpace is a group of one or more AppNodes, which are runtime entities that host ActiveMatrix BusinessWorks™ applications. AppSpaces are contained within a domain. One or more than one application can be deployed to an AppSpace.
- An AppNode is a runtime entity that hosts applications. AppNodes are contained in an AppSpace.
- The bwagent is a daemon that runs on every ActiveMatrix BusinessWorks installation. When multiple installations across machines are configured as a network, the bwagents interact with each other using a datastore. They also synchronize the data from the datastore with the local file system.

The Administration Architecture illustration below shows an example of runtime entities created across two bwagents in a network. In the illustration, domain M1 spans two machines, Machine A and Machine B. Domain N1 is on Machine A only. Domain M1 contains two AppSpaces, S1 and S2. AppSpace S2 spans both machines. The bwagent on Machine A is configured to interact with the bwagent on Machine B through the datastore.

The Admin UI is a web UI that runs in TIBCO® Enterprise Administrator (TEA). Using the Admin UI is optional. To enable the Admin UI, the bwagent must be registered with a running TEA server. In the Administration Architecture illustration below, the bwagent on Machine A is registered with the TIBCO Enterprise Administrator (TEA) server. If the registered bwagent becomes unavailable, the connection between the TEA server and the agent network is automatically recovered. The bwagent on Machine B will autoregister with the server.

Administration Architecture



The runtime entities manifest as a hierarchical folder structure on the local file system. Every action performed on the runtime entities results in an update to the file system. The location of the default

domains folder in the local file system can be changed by editing the *BW_HOME*/domains/DomainHomes.properties file.

When runtime entities span machines, the bwagent synchronizes the data from the datastore with the local file system. At any given point in time, the data in the file system is the source of truth. This ensures that in case of a failure in the communication channel, the runtime is not affected as it refers to the data on the local file system.

In your production environment, ensure you are using an external database and either TIBCO FTL® or TIBCO Enterprise Message Service™ (EMS) for data persistence and communication transport.

For more information about administration concepts, see the *TIBCO ActiveMatrix BusinessWorks™ Concepts* guide.

Getting Started

Deploy and manage applications created in TIBCO Business Studio™ for BusinessWorks™ using the bwadmin console or the Admin UI, and the bwagent.

The bwadmin console and the bwagent are executables located in the `bin` folder of the product installation. For more information about the bwadmin console and the bwagent, see Administrator and Agent.

For information about the Admin UI, see Using the Admin UI.

There are several ways to deploy an application:

- bwadmin: For more information about deploying with bwadmin, see Deploying an Archive.
- Admin UI: For more information about deploying with the Admin UI, see Deploying an Archive.
- Deployment servers in TIBCO Business Studio for BusinessWorks: For more information about deploying with deployment servers, see Deploying an Application in the *TIBCO ActiveMatrix BusinessWorks™ Application Development* guide.
- Rest API

In this document, *BW_HOME* points to *TIBCO_HOME*`\bw\n.n`

Execution Modes

The execution mode is set using the bwadmin command line console or in the bwagent's configuration file.

The execution mode determines whether bwadmin communicates with the bwagent. There are two modes: local and enterprise. The default mode is set to local, meaning that there is no communication with the bwagent.

Local mode

In local mode, bwadmin modifies the local file system directly instead of delegating the work to a bwagent. Local mode does not provide data storage and runtime entities are created in the file system. This mode is useful for developers during development and testing cycles. For more information, see Running in Local Mode.

Enterprise mode

In enterprise mode, bwadmin communicates with the bwagent. bwagents can communicate across machines and can be configured to form a bwagent network. Instead of working on the file system directly, bwadmin sends commands to the bwagent. The bwagent dispatches the command to targeted agent. That agent then completes the command on the local file system. For more information, see Running in Enterprise Mode Using the Command Line. In enterprise mode, the bwagent can be registered with the TEA server.

To change the mode, navigate to *BW_HOME*`\bin` (Windows) or `${`*BW_HOME*`}/bin` (Unix) and issue the following command: **`bwadmin mode local`** (to switch to local mode) or **`bwadmin mode enterprise`** (to switch to enterprise mode).

Changing the mode sets the **`bw.admin.mode`** property in the bwagent configuration file. This file is called `bwagent.ini` and is located in the *BW_HOME*`\config` folder (Windows) or `${`*BW_HOME*`}/config` folder (Unix).

Running in Local Mode

Local mode allows application testing and debugging on the local file system.

This procedure shows you how to create runtime entities and deploy and run an application using bwadmin local mode. You will learn how to set the bwadmin mode; create a domain, AppSpace, and AppNode; upload an application archive; start the AppSpace; and deploy and start the uploaded application.

The runtime entities created in local mode are not visible to bwagents when they are started.

Procedure

1. In a terminal, navigate to the following paths:
   - Windows:
     ```
     BW_HOME\bin
     ```
   - Unix:
     ```
     ${BW_HOME}/bin
     ```
2. Set the bwadmin mode to local.
   - Windows:
     ```
     BW_HOME\bin>bwadmin mode local
     ```
   - Unix:
     ```
     [root@BW_HOME bin]# ./bwadmin mode local
     ```
3. Create a domain. For more information, see Creating a Domain.
   - Windows:
     ```
     BW_HOME\bin>bwadmin create domain D1
     ```
   - Unix:
     ```
     [root@BW_HOME bin]# ./bwadmin create domain D1
     ```
4. Show the domain.
   - Windows:
     ```
     BW_HOME\bin>bwadmin show domain
     ```
   - Unix:
     ```
     [root@BW_HOME bin]# ./bwadmin show domain
     ```
5. Create an AppSpace in the domain. For more information, see Creating an AppSpace.
   - Windows:
     ```
     BW_HOME\bin>bwadmin create -d D1 appspace AS1
     ```
   - Unix:
     ```
     [root@BW_HOME bin]# ./bwadmin create -d D1 appspace AS1
     ```
6. Create an AppNode in the AppSpace. When creating an AppNode, you must specify the HTTP management port that allows communication with the AppNode.
   - Windows:
     ```
     BW_HOME\bin>bwadmin create -domain D1 -appspace AS1 -httpPort 8060 appnode
     AN1
     ```
   - Unix:
     ```
     [root@BW_HOME bin]# ./bwadmin  create -domain D1 -appspace AS1 -httpPort
     8060 appnode AN1
     ```

The HTTP management port must be unique across all defined AppNodes on a machine. If the specified port is already in use, an error is issued and the AppNode cannot be created.

For more information, see Creating an AppNode.

7. Use the **show** command for the AppSpace after you've created the AppNode.

```
BW_HOME\bin>bwadmin show -domain D1 -appspace AS1 appnodes
```

8. Upload an application archive into the domain. The following command uploads the Bookstore sample application archive. Use forward slash in the Windows command line.

   - Windows:

```
BW_HOME\bin>bwadmin upload -domain D1 ../samples/core/admin/ears/bookstore/
tibco.bw.sample.binding.rest.BookStore.application_1.0.0.ear
```

   - Unix:

```
[root@BW_HOME bin]# ./bwadmin upload -domain D1 ../samples/core/admin/ears/
bookstore/tibco.bw.sample.binding.rest.BookStore.application_1.0.0.ear
```

   For more information, see Uploading an Archive.

9. Show that the application archive was uploaded.

   - Windows:

```
BW_HOME\bin>bwadmin show -domain D1 archives
```

   - Unix:

```
[root@BW_HOME bin]# ./bwadmin show -domain D1 archives
```

10. Start the AppSpace. This starts the AppNode in the AppSpace.

    - Windows:

```
BW_HOME\bin>bwadmin start -domain D1 appspace AS1
```

    - Unix:

```
[root@BW_HOME bin]# ./bwadmin start -domain D1 appspace AS1
```

    For more information, see Starting an AppSpace.

11. Deploy the application into the AppSpace. This deploys the application to the AppNode.

    - Windows:

```
BW_HOME\bin>bwadmin deploy -domain D1 -appspace AS1
tibco.bw.sample.binding.rest.BookStore.application_1.0.0.ear
```

    - Unix:

```
[root@BW_HOME bin]# ./bwadmin deploy -domain D1 -appspace AS1
tibco.bw.sample.binding.rest.BookStore.application_1.0.0.ear
```

    For more information, see Deploying an Application.

12. See the deployed application using show command.

    - Windows:

```
BW_HOME\bin>bwadmin show -domain D1 application
```

    - Unix:

```
[root@BW_HOME bin]# ./bwadmin show -domain D1 application
```

13. Start the application. Each uploaded application maintains a version. The version number is required for starting and stopping the application.

    - Windows:

```
BW_HOME\bin>bwadmin start -d D1 -appspace AS1 application
tibco.bw.sample.binding.rest.BookStore.application 1.0
```

- Unix:

```
[root@BW_HOME bin]# ./bwadmin start -d D1 -appspace AS1
```

To find the version number, you can use the **show** command, for example: `bwadmin show -domain D1 application`

For more information, see Starting an Application.

Use the **-csv** command to print the table content as a comma separated value table. The first row contains the headers and applies to domains, AppSpaces, AppNodes, Applications, Archive, Archives, Machines and Installations. For example, `bwadmin[admin@d1]> show -csv apps` to show all the Applications in the csv format.

14. Optionally, stop and undeploy the application, stop the AppSpace, and delete the entities (archive, AppNode, AppSpace, and domain).

Result

You used bwadmin in local mode to create a domain (D1), an AppSpace (AS1), and an AppNode (AN1). You uploaded an application archive to the domain, deployed the application, and started and stopped the application. Spend some time experimenting with bwadmin commands. For more information about domains, AppSpaces, AppNodes, and applications, see Administration Tasks and Reference.

Running in Enterprise Mode Using the Command Line

In enterprise mode, bwagents can communicate across machines and can be configured to form a bwagent network.

This procedure shows you how to set up a network using bwagents on two machines and use bwadmin to create runtime entities across machines.

Enterprise mode requires a data persistence and communication transport layer to keep bwagents in sync across machines. By default, TIBCO FTL® is used for communication transport and external database for data persistence. The software also provides the option of using TIBCO Enterprise Message Service™ (EMS) for communication transport.

For more information about configuring bwagent, see Configuring bwagent .

The following example uses TIBCO FTL®, the default configuration.

Prerequisites

Install the software on two machines. Machines are noted as M1 and M2 in the instructions. Make a note of the host name or IP address for each machine.

Procedure

1. Configure the bwagent on machine M1.

   For more information about how to configure bwagent with database TIBCO FTL®, see Database with TIBCO FTL®

2. Repeat the configuration for the bwagent on machine M2.

   Ensure that same database, TIBCO FTL® and network name is used for M1 and M2.

3. Start the bwagent on M1.
   a) Open a terminal on M1 and navigate to the *BW_HOME*`\bin` folder (Windows) or `[root@`*BW_HOME*` bin]#` (Unix).
   b) Type `bwagent` (Windows) or `./bwagent` (Unix).
   The bwagent starts.

4. Start the bwagent on M2.
   a) Open a terminal on M2 and navigate to *BW_HOME*\bin folder (Windows) or ${*BW_HOME*}/bin (Unix).
   b) Type `bwagent` (Windows) or `./bwagent` (Unix).
   The bwagent starts.
5. Start the bwadmin console on M1 machine and show the bwagents.

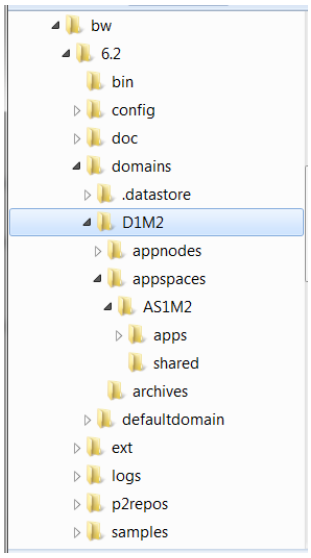
   - Windows:
     ```
     BW_HOME\bin>bwadmin show agents
     ```
   - Unix:
     ```
     [root@BW_HOME bin]# ./bwadmin show agents
     ```

   The **`TEA Server URL`**, **`Registered TEA Agent`**, and **`Auto Registration`** settings are important when you are using the Admin UI. These settings can be ignored for this example. For more information about the Admin UI, see Running Applications in Enterprise Mode using the Admin UI and Using the Admin UI.
6. Now the bwagents on machines M1 and M2 are communicating with each other. You can use the bwadmin console on M1 to create a domain on M2 by specifying the bwagent in the command line.

   - Windows:
     ```
     BW_HOME\bin>bwadmin create -agent M2 domain D1M2
     ```
   - Unix:
     ```
     [root@BW_HOME bin]# ./bwadmin create -agent M2 domain D1M2
     ```
7. On M1, use the **`show domains`** command to show the domain.

   - Windows:
     ```
     BW_HOME\bin>bwadmin show domains
     ```
   - Unix:
     ```
     [root@BW_HOME bin]# ./bwadmin show domains
     ```
8. Create an AppSpace on M2 using bwadmin on M1.
   a) Create an AppSpace on M2 by specifying the bwagent. Two AppNodes are specified. The **`-minNodes`** option specifies the number of nodes in the AppSpace. The AppSpace cannot be started until this value is met. For more information about AppSpaces and AppNodes, see the Administration Concepts topic in the *TIBCO ActiveMatrix BusinessWorks™ Concepts* guide.

      - Windows:
        ```
        BW_HOME\bin>bwadmin create -agent M2 -domain D1M2 -minNodes 2 appspace AS1M2
        ```
      - Unix:
        ```
        [root@BW_HOME bin]# ./bwadmin create -agent M2 -domain D1M2 -minNodes 2
        appspace AS1M2
        ```
   b) View the file system on machine M2 to verify that the AppSpace was created. The example below shows the file system on a Windows machine.



Each runtime entity is created in the file system. It is critical that all runtime entities are managed using bwadmin so that they are in sync with the datastore.

9. On machine M1, create the 2 AppNodes for the AppSpace `AS1M2`, specifying bwagent `M2` for the AppNodes. The HTTP management port must be unique. A list of defined AppNodes for a given domain, including port numbers, is available with the **`show`** command: `show -d <DomainName> appnodes`

When an AppNode is created, an optional port for the OSGi console can be specified to monitor the AppNode (Only enable this port for troubleshooting purposes.). For more information, see Enabling the OSGi Console for an AppNode.

Windows:

```
BW_HOME\bin>bwadmin create -agent M2 -domain D1M2
-appspace AS1M2 -httpPort 8070 appnode AN1M2
```

```
BW_HOME\bin>bwadmin create -agent M2 -domain D1M2
-appspace AS1M2  -httpPort 8071 appnode AN2M2
```

Unix:

```
[root@BW_HOME bin]# ./bwadmin create -agent M2 -domain D1M2
-appspace AS1M2 -httpPort 8070 appnode AN1M2
```

```
[root@BW_HOME bin]# ./bwadmin create -agent M2 -domain D1M2
-appspace AS1M2  -httpPort 8071 appnode AN2M2
```

The **`-httpPort`** option is case sensitive.

10. From the bwadmin console on M1, upload an application archive into the domain on M2.

Windows:

```
BW_HOME\bin>bwadmin upload -domain D1M2 ../samples/core/admin/ears/bookstore/
tibco.bw.sample.binding.rest.BookStore.application_1.0.0.ear
```

Unix:

```
[root@BW_HOME bin]# ./bwadmin upload -domain D1M2 ../samples/core/admin/ears/
bookstore/tibco.bw.sample.binding.rest.BookStore.application_1.0.0.ear
```

11. From M1, start the AppSpace on M2. This starts the AppNodes in the AppSpace on M2.

Windows:

```
BW_HOME\bin>bwadmin start -domain D1M2 appspace AS1M2
```

Unix:

```
[root@BW_HOME bin]# ./bwadmin start -domain D1M2 appspace AS1M2
```

12. From the bwadmin console on M1, verify that the AppNodes are running:

    Windows:

    ```
    BW_HOME\bin>bwadmin show -domain D1M2 -appspace AS1M2 appnodes
    ```

    Unix:

    ```
    [root@BW_HOME bin]# ./bwadmin show -domain D1M2 -appspace AS1M2 appnodes
    ```

13. Use bwadmin on M1 to stop the AppSpace on M2. This will stop AppNodes `AN1M2` and `AN2M2`.

    Windows:

    ```
    BW_HOME\bin>bwadmin stop -domain D1M2 appspace AS1M2
    ```

    Unix:

    ```
    [root@BW_HOME bin]# ./bwadmin stop -domain D1M2 appspace AS1M2
    ```

14. Back up the domain. The backup command exports the persisted state of runtime entities into a command file. This command file can be used to recreate the environment. For more information, see Backing Up and Restoring from the Backup.
    The **`backup`** command requires the name of the specific entity being backed up (domain, agent, AppSpace or AppNode) as well as the path to a destination file. In this example (Windows), D1M2 is backed up.

    ```
    BW_HOME\bin>bwadmin backup -s backup.cmd domain D1M2
    ```

The bwadmin **`backup`** command and the bwadmin **`restore`** command are not complimentary. The **`backup`** command exports the current state of the environment to a command file. The **`restore`** command restores the file system of a bwagent to the state of the persistent datastore. For more information, see Backing Up and Restoring from the Backup and Restoring the File System of Runtime Entities.

Result

You set up a network with two bwagents on two machines. You used the bwadmin console on one machine to create runtime entities on the other machine. You uploaded an application archive to the domain and started the AppSpace. You also backed up the environment.

You can continue experimenting by adding additional machines to the network, adding more runtime entities, or deploying the archive (you will need to start the AppSpace again).

When you are done, you can force delete the domain using bwadmin on either machine with the following bwadmin command: **`delete -force domain D1M2`**

After you delete the domain, you can recreate the environment from the backup by feeding the backup command file to bwadmin, for example: **`bwadmin -f backup.cmd`** (Windows).

To exit the bwagent, type `^C` (this may take a few seconds). At the command line, type **`bwagent stop`** to completely stop the agent.

Running in Enterprise Mode Using the Admin UI

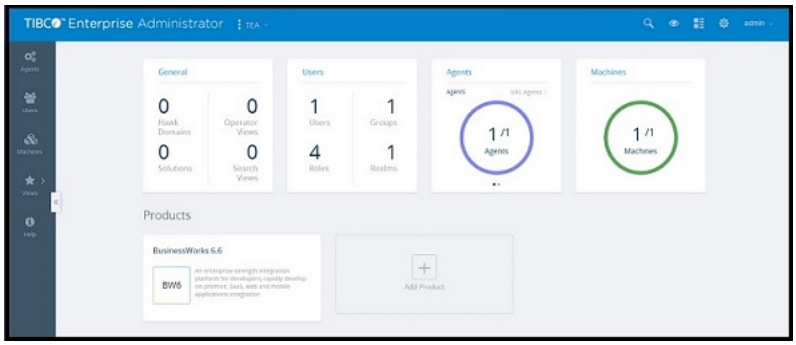
Use the Admin UI to manage and monitor runtime entities.

The Admin UI is a web UI that runs in TIBCO® Enterprise Administrator (TEA). To enable the Admin UI, the bwagent must be registered with a running TEA server.

This procedure shows you how to create runtime entities and deploy and run an application using the Admin UI. You will learn how to register the TEA agent with the bwagent; open the Admin UI; create a domain, AppSpace, and AppNode; upload an archive; start the AppSpace; and deploy and start the uploaded application.

Procedure

1. Install TEA and start the TEA server.
   - Windows:
     ```
     TEA_HOME\2.0\bin>tea.exe
     ```
   - Unix:
     ```
     [root@TEA_HOME bin]# ./tea.sh
     ```
2. In a terminal, navigate to the following location:
   - Windows:
     ```
     BW_HOME\bin
     ```
   - Unix:
     ```
     BW_HOME/bin
     ```
3. Set the bwadmin mode to enterprise.
   - Windows:
     ```
     BW_HOME\bin>bwadmin mode enterprise
     ```
   - Unix:
     ```
     [root@BW_HOME bin]# ./bwadmin mode enterprise
     ```
4. Open a new terminal and navigate to *BW_HOME*`\bin` for Windows or `${`*BW_HOME*`}/bin` for Unix. Register the bwagent TEA agent with the TEA server. This allows the bwagent to be available to the TIBCO Enterprise Administrator server. The URL to the TEA server is required in the command. The URL is available from the terminal where the TEA server was started.
   - Windows:
     ```
     BW_HOME\bin>bwadmin registerteaagent http://M1:8777/
     ```
   - Unix:
     ```
     [root@BW_HOME bin]# ./bwadmin registerteaagent http://M1:8777/
     ```
5. Open a web browser and go to the TEA URL. Sign in, using `admin` for the user name and `admin` password.
   BusinessWorks is displayed in the **Products** list.



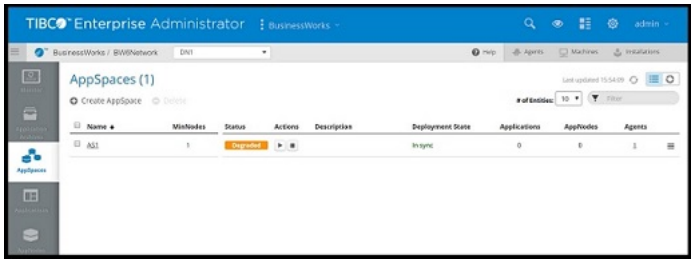
6. Click the BusinessWorks icon to go to ActiveMatrix BusinessWorks. The Domain Management page is displayed. If you completed the steps in the "Running in Enterprise Mode Using the Command Line," you see a domain listed on the Domain Management page. Otherwise the page is empty.
   a) Click **Create Domain** to open the **Create Domain** dialog box.
   b) Enter the domain name in the **Name** field.
   c) Choose the bwagent registered with the TEA server from the **Agent** drop-down.
   d) Click **Create** to create the domain.

   The domain is created and displayed on the Domain Management page.

7. Click the domain name to open the domain.
8. Add an AppSpace to the domain.

   a) Click the **AppSpaces** icon  to open the AppSpaces page.
   b) Click **Create AppSpace**.
   c) In the Create AppSpace dialog box, enter AppSpace name in the **Name** field. Accept the value of 1 in the **MinNodes** field.
   d) Select the agent registered with the TEA server and click **Create**.

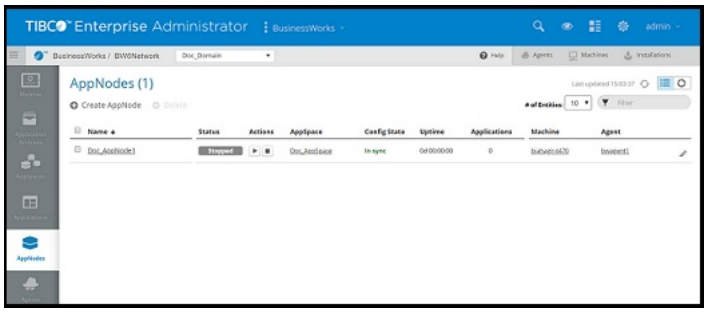
   The AppSpace is created. A success notification is displayed at the top of the page. Notice that the AppSpace status is displayed as Degraded, because there are no AppNodes yet for the AppSpace. This will change to Stopped when an AppNode is added to the AppSpace. The AppSpaces page will look similar to this:



9. Add an AppNode to the AppSpace.

   a) Click the **AppNodes** icon  to open the AppNodes page.
   b) Click **Create AppNode**.
   c) In the Create AppNode dialog box, enter name of the AppNode in the **Name** field.
   d) Select the agent registered with the TEA server from the **Agent** drop-down.
   e) Enter a value in the **HTTP Port** field, for example: `8075`. This port must be available; each AppNode on the machine must be assigned to a unique HTTP management port.
   f) Leave the OSGi fields empty. (This are optional fields for debugging the AppNode. For more information, see Enabling the OSGi Console for an AppNode.)
   g) Select the AppSpace from the **AppSpace** drop-down.
   h) Click **Create**.

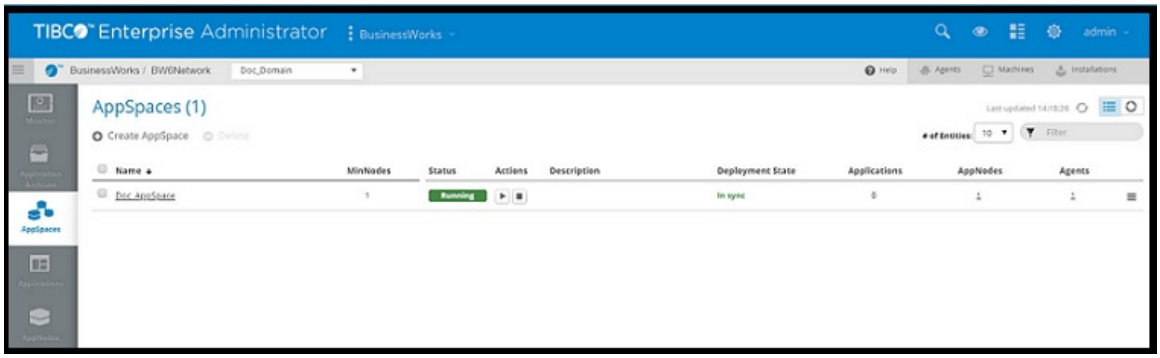
   The AppNode is created. A success notification is displayed at the top of the page and the AppNode is displayed. The AppNodes page will look similar to the following image:



10. Open the AppSpaces page and notice that the status has been updated to Stopped.

    a) Start the AppSpace by clicking the **Start** icon ▶.

    The AppSpace starts and starts the AppNode. The status changes to Starting, then Running to indicate that both the AppNode and AppSpace are running.



11. Upload an application archive to the AppSpace.

    a) Click the **Application Archives** icon 

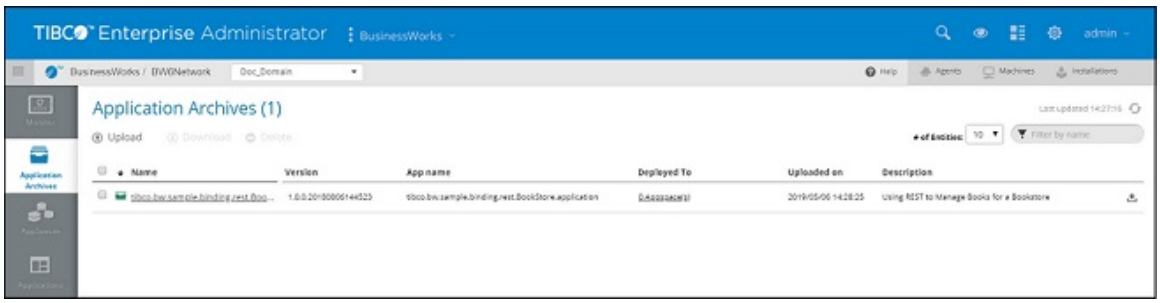
.

    b) In the Application Archives page, click the **Upload** link and drag the BookStore sample application archive from *BW_HOME*`\samples\AppSpace\core\admin\ears\bookstore\ears` to the Upload Ear File dialog box.

    c) Click **Upload**.

    d) A success message is displayed in the dialog box. Click **Done** to close the dialog box.

    The application archive is displayed on the Application Archives page:



12. Deploy the application.

    a) Click the application archive link on the Application Archives page to pivot to the application view, then click **Deploy** to deploy the BookStore application.

    b) In the Deploy Applications dialog box, make sure `UI-AppSpace1` is selected in the **AppSpace** drop-down.

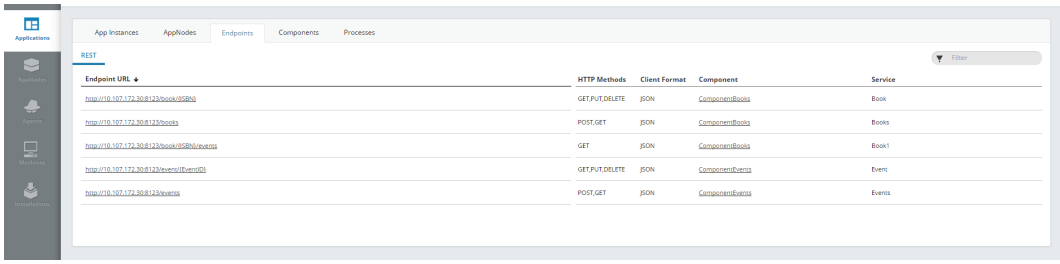
    c) Select `WindowsProfile.substvar (Default)` from the **Profile** drop-down.

    d) Check the **Start applications on AppNodes after deployment** option. This option starts the application after successful deployment.

    e) Click **Deploy.**

    The application is deployed to the selected AppSpace and is started on the AppNode.

13. Click the **REST Doc URL** link to view the application's REST API. You can also open the **Endpoints**, **Components**, or **Processes** tabs to drill down into the running application.



Result

You registered the bwagent with the TEA server. You used the Admin UI to create a domain, AppSpace, and AppNode; start the AppSpace; upload an application archive; deploy the application; and start the application. Spend some time experimenting with the Admin UI. The web interface allows you to drill down into entities and pivot views.

For more information, see Using the Admin UI.

Core Admin Sample Scripts

The sample scripts provide a simple and fast way to run the core Admin samples.

These are bash scripts. On Windows, install Cygwin64.

Admin scripts are located in the following folders: *$BW_HOME*`/samples/core/admin` and *$BW_HOME*`/scripts/admin`

For information about running the sample scripts, see the "Running Admin Sample Scripts" section in the *TIBCO ActiveMatrix BusinessWorks™ Getting Started* guide.

This sets the *TIBCO_HOME*, *BW_HOME*, *TEA_HOME*, *EMS_HOME*, and *JAVA_HOME* environment variables necessary to run the admin scripts.

All scripts support the `-h` and command-line argument with full documentation of what each script does.

Location of the Admin Scripts

The admin scripts are located in the following folders:

- The sample scripts are located in `$BW_HOME/samples/core/admin`
- The scripts that are generic for Activematrix BusinessWorks™ are located in `$BW_HOME/scripts/admin`

The scripts are updated to rely on the PATH setting to find the generic scripts. To make this easier to configure, after installation you can generate `$BW_HOME/scripts/bashrc.sh` that can be sourced from your `~/.bashrc`.

Source the `$BW_HOME/scripts/bashrc.sh` to setup the following environment variables required to run the scripts mentioned in the table below:

Variable	Required
TIBCO_HOME	Yes
BW_HOME	Yes
TEA_HOME	No. But required if you run TIBCO® Enterprise Administrator on this machine.
EMS_HOME	No. But required if Enterprise Message Service™ is configured on this machine.
PATH	This variable is auto-populated based on the values set for the above variables.

Core Admin Scripts

The following table lists some of the available scripts; browse the folder to see the complete list.

Core Admin Scripts

Script	Description	Script Location
`AppManage.sh`	This is a ActiveMatrix BusinessWorks™ 6.x utility program that emulates ActiveMatrix BusinessWorks™ 5.x AppManage commands. The main purpose of this utility is to demonstrate how the AppManage commands from ActiveMatrix BusinessWorks™ 5.x translate to corresponding TIBCO ActiveMatrix BusinessWorks 6.x bwadmin commands. This utility creates a `cmd/AppManage_deploy.cmd` folder that contains bwadmin commands and uses `bwadmin -f cmd/AppManage_deploy.cmd` to run it.  Not all AppManage commands are implemented in this emulation utility. **ActiveMatrix BusinessWorks™ Augmented Options**: • `-appSpace` or `- a` - AppSpace name to be used for Application lifecycle. • `-profile` or `-p` - Configuration Profile to use for deployment. This profile must be available in the EAR file. • `-profileFile`- Configuration Profile file to use for deployment. • `-debug` - Turn on debug tracing for this utility. • `-sapp` - Single Application per AppSpace deployment mode. Each AppSpace supports only one application deployment. • `-mapp` - Multiple Applications per AppSpace deployment mode. Each AppSpace supports one or more application deployments.  ActiveMatrix BusinessWorks™ supports both `-sapp` and `-mapp` modes. The default is `-mapp` mode.	*`$BW_HOME`*`/samples/core/admin`

Script	Description	Script Location

`bootstrap.sh`	Usage: **`bootstrap.sh [-h|-help] [-clean] [-forceClean|-force|-forceclean]`**
	
	This utility is a wrapper script around the following scripts:
	
	• `killtea.sh`
	• `killbwagent.sh`
	• `teaclean.sh` only if `-clean` or `-forceClean` options is used.
	• `bwclean.sh` if and only if `-clean` or `-forceClean` options is used.
	• `genbwagentini.sh`
	• `tea.sh`
	• `bwagent.sh`
	• `registeragent.sh`
	
	`[-h]` or `[-help]` - Prints this usage message.
	
	`-clean` Cleans TIBCO Enterprise Administrator Server Data Store and ActiveMatrix BusinessWorks™ Domain Data Store.
	
	 
	The -clean command on the data store is not reversible, so back up your data stores before using the command. Use this option carefully, as you may lose all your configurations if you do not have a backup.
	
	`-forceClean` Same as `-clean`, except it avoids prompting user to confirm with clean.
	
	`-force` Same as `-forceClean`
	
	`-forceclean` Same as `-forceClean`
	
	This script assumes that the following products are installed correctly and the environment variables are set accordingly:
	
	TIBCO_HOME = *TIBCO_HOME* directory where you installed ActiveMatrix Businessworks™.
	
	TEA_HOME = Parent directory to TIBCO Enterprise Administrator's `/bin` directory.
		$BW_HOME`/scripts/admin`

Script	Description	Script Location
	Supports generation of `bwagent.ini` file for either Database/ TIBCO EMS™, or Database/ TIBCO FTL® as the technology type.	
`bounce.sh`	This utility does the following: 1. Stops TIBCO Enterprise Administrator Server and bwagent Processes. 2. Restarts TIBCO Enterprise Administrator Server and bwagent Processes. 3. Registers bwagent to TIBCO Enterprise Administrator Server. `[-h]` or `[-help]` - Prints this help message and exits.	*`$BW_HOME`*`/scripts/admin`
`bounceagent.sh`	Kills and restarts bwagent Process. `[-h]` or `[-help]` - Prints this help message and exits.	*`$BW_HOME`*`/scripts/admin`

Script	Description	Script Location
bwadmin.sh	This is a utility script that wraps around the bwadmin executable. [-h] or [-help] - Prints this help message and exits. **[-network <bwagent Network Name>]** - Connects to a named bwagent Network. This is an optional argument. By default, this script uses $BW_HOME/config/bwagent.ini **[<bwadminArgs> ...]** - Use bwadmin to run commands found in the input files. Start bwadmin in the interactive mode if cmdFile is not specified. A bwagent Network Name is a named directory under **${TIBCO_HOME}/bw/networks** and contains the corresponding bwagent.ini. **How to Set Up a Newly Named Network** 1. Obtain a bwagent.ini created for the named bwagent network. For example, a named network called "acmeNetwork" 2. Create the acmeNetwork directory under ${TIBCO_HOME}/bw/networks. For example, mkdir ${TIBCO_HOME}/bw/networks/acmeNetwork 3. Copy bwagent.ini to the above directory. 4. Rerun bwadmin.sh -network acmeNetwork	*$BW_HOME*/scripts/admin

Script	Description	Script Location
bwagent.sh	This script starts bwagent in the background and waits until it is fully initialized, or the maxWait time (<n> * 2 sec) expires. `[-h]` or `[-help]` - Prints this usage message. **`[-network <Network>]`** - Starts up bwagent using the configuration of a named network. **`[-maxWait <n>]`** - Maximum amount of wait time (2 sec increment) for bwagent start up success. The dDefault value for <n> is 30, which means 30 * 2 sec = 60 seconds	*$BW_HOME*`/scripts/admin`
bwclean.sh	This utility script cleans up ActiveMatrix BusinessWorks™ Domain Data and internal Data Store. The end effect of this clean up is similar to a fresh installation of ActiveMatrix BusinessWorks™. `[-force]` or `[-forceClean]` - Proceeds with wiping ActiveMatrix BusinessWorks™ Domain Data and internal Data Store without prompting user reconfirmation. By default, the script prompts user confirmation.	*$BW_HOME*`/scripts/admin`

Script	Description	Script Location
configureBWEngineGroup.sh	This utility configures AppNodes in a Domain/AppSpace to form a fault-tolerant group and cross engine persistence [-h] or [-help] - Prints this usage message. [-c] or [-clean] - Cleans up and drops all the previously configured database tables. Use this option carefully. This operation cannot be undone. Do not specify both -setup and -cleanup on the same run. [-s] or [-setup] - Does the one time setup of bwengine Database. When this option is used, -domain and -appspace arguments are not needed and are not used even if specified. ${BW_HOME}/config/sqlscripts/<dbtype>/create.sql is used to set up the database tables and configuration. [-b] or [-bootstrap] - Does clean up then setup. [-t] or [-dbtype] - This is the default value is postgresql. -cf <config.sh> - Sources configuration from the specified <config.sh> file. By default, *<$BW_HOME>*/scripts/admin/config/bwengine-group-<dbtype>.sh [-d] or [-domain] - Domain Name [-a] or [-appspace <appspace>] - AppSpace Name. All AppNodes in the specified Domain and AppSpace are configured to form a Fault-Tolerant group and across engine persistence.	*$BW_HOME*/scripts/admin

Script	Description	Script Location
deploy.sh	Usage: **deploy.sh -ear <EARFile> [-h|-help] [-domain <DomainName>] [-appspace <AppSpaceName>] [-redeploy|-force] [-profile <Profile>]** Deploys the specified ActiveMatrix BusinessWorks™ EAR File into -domain <DomainName> -appspace <AppSpaceName> [-h] or [-help]- Prints this help message. -ear *<EARFile>* - Enterprise Archive file to deploy **[-domain <DomainName>]** - Domain Name - Optional parameter If it is not specified, DomainName is computed from ${USER}-Domain This utility creates the Domain if it does not already exist. **[-appspace <AppSpaceName>]** - AppSpace Name - Optional parameter If it is not specified, AppSpaceName is computed from the name of the EAR file. This utility creates the AppSpace and AppNode if they do not already exist. **[-redeploy|-force]** - Redeploy if the application has been previously deployed. The application is not redeployed if it already exists and this option is not specified. **[-profile <Profile>]** : Profile name to use for this deployment. If it is not specified, the default Profile as packaged in the Enterprise Archive file is used. **[-mapp]** - Optional flag to set Multiple Applications per AppSpace Mode. This is the default mode for ActiveMatrix BusinessWorks™. **[-debug]** - Prints debug tracing for this script **./deploy.sh**	*$BW_HOME*/scripts/admin

Script	Description	Script Location
genbwagentini.sh	This script auto generates `${BW_HOME}/config/bwagent.ini` based on configurations defined in `./config/bwadmin-default- config.sh` `-h` or `-help` - Prints this help message. The following variables are required from `./config/bwadmin-default-config.sh`: • BWAgentNetworkName - Name of BWAgent Network. • BWMachines - Defined as a list of machine names (as obtained through `hostname -f`). If you have only one machine to configure, do not add it to this list because this script auto-configures it as a standalone BWAgent Network. This script uses `hostname -f` to determine the name of the machine it is run on. It then determines whether this machine is in the BWMachines list. You can assume that the `discoveryURL` of the bwagent.ini is comparable to that of a Database Server's URL, and `BWAgentNetworkName` is then comparable to the Database Name. You can configure both to uniquely access the specific instance of the Database. If the `KEEP_BWAGENT_INI` environment variable is defined, `bwagent.ini` generation is skipped. You can edit either the `./config/bwadmin-default- config.sh` file, or make a copy of it, edit it, and then set environment variable `BWADMIN_CONFIG` to point to it. For example, export `BWADMIN_CONFIG=~/config/bwadmin-my-config.sh` Generates `bwagent.ini` file for either Database/EMS, or Database/ TIBCO FTL® as the technology type.	*$BW_HOME*`/scripts/admin`

Script	Description	Script Location
kill.sh	Kills all processes that match the specified <*process name*> -h or -help - Prints this help message. <*process name*> - name of the process you want to kill. This script kills all instances of the pid that matches this name.	*$BW_HOME*/scripts/admin
killall.sh	This script finds and kills all instances of processes that match the following names: • tea • bwagent • bwappnode • bwadmin -h or -help - Prints this help message.	*$BW_HOME*/scripts/admin
killbwagent.sh	This script finds and kills all instances of processes that match "bwagent" . -h or -help - Prints this help message.	*$BW_HOME*/scripts/admin
killbwappnodes.sh	This script finds and kills all instances of processes that match "bwappnode". -h or -help - Prints this help message.	*$BW_HOME*/scripts/admin
killtea.sh	This script finds and kills all instance of processes that matches "tea". -h or -help - Prints this help message.	*$BW_HOME*/scripts/admin
killtibemsd64.sh	This script finds and kills all instances of processes that match "tibemsd". -h or -help - Prints this help message.	*$BW_HOME*/scripts/admin
recreatedb.sh	This script cleans up and recreates the Postgres DB needed by ActiveMatrix BusinessWorks™ BookStore REST sample located in: ${BW_HOME}/samples/binding/rest/BookStore -h and -help - Prints this help message.	*$BW_HOME*/samples/core/admin

Script	Description	Script Location
registeragent.sh	This utility registers the local bwagent with TIBCO Enterprise Administrator server. -h or -help - Prints this help message. This utility assumes that the following environment variables have been set: export TIBCO_HOME="<Where ActiveMatrix BusinessWorks™ is installed>" At least one of the following environment variable is set: export TEA_HOME="Where TIBCO Enterprise Administrator is installed in the form of $TIBCO_HOME/tea/<version>" Or, export TEA_HOSTNAME=<HostName> If *TEA_HOSTNAME* environment variable is set, it assumes the TIBCO Enterprise Administrator server is running remotely from the local bwagent instance. If *TEA_HOSTNAME* environment variable is not set, this script registers the local bwagent to the locally running TIBCO Enterprise Administrator server.	*$BW_HOME*/scripts/admin

Script	Description	Script Location
runAcme.sh	Creates <domain> and deploys all EAR files found under `${BW_HOME}/samples/core/admin/ears/acme`. `-h` or `-help` : Displays this usage message `<domain>` - can be "Acme-QA-Domain" or "Acme-UAT-Domain". When not specified, the default is "Acme-QA-Domain" `<mode> -[-sapp ]` or `[-mapp]` `-sapp`- Single App AppSpace deployment mode. Each AppSpace supports only one application deployment. `-mapp` - Multiple App AppSpace deployment mode. Each AppSpace supports one or more application deployment.  ActiveMatrix BusinessWorks™ supports both -sapp and -mapp modes. The default is -mapp mode. This script dynamically creates a **bwadmin** command file in `cmd/<domain>-<mode>.cmd` and executes it.	*$BW_HOME*`/samples/core/admin`

Script	Description	Script Location
runAll.sh	This utility is a wrapper script that performs the following: • bootstrap.sh - only if running in a single machine setup • runBookStore.sh • runSamples.sh • runAcme.sh -domain Acme-QA-Domain • runAcme.sh -domain Acme-UAT-Domain `-h` or `-help` - Displays this usage message and exits `-clean` - Cleans the TIBCO Enterprise Administrator Server Data Store and ActiveMatrix BusinessWorks™ Domain Data Store. These data store clean is not reversible. Make sure you back up your data stores before running this command. Use this option with utmost care, otherwise you risk losing all your configurations. `-forceClean` - Same as **`-clean`**, except it avoids prompting you to confirm with clean. `-force` - Same as **`-forceClean`** `<mode>` - [-sapp | -mapp] `-sapp` - Single Application per AppSpace deployment mode. Each AppSpace supports only one application deployment. `-mapp` - Multiple Applications per AppSpace deployment mode. Each AppSpace supports one or more application deployments.  ActiveMatrix BusinessWorks™ supports both -sapp and -mapp modes. The default is -mapp mode. Generates the `bwagent.ini` file for either Database/EMS™, or Database/FTL® technology type.	*$BW_HOME*`/samples/core/admin`

Script	Description	Script Location
runBookStore.sh	Creates BookStore-Domain and deploys all EAR files found under `${BW_HOME}/samples/core/admin/ ears/bookstore` `-h` or `-help` - Displays this usage message. `<mode>` - [-sapp | -mapp] `-sapp` - Single Application per AppSpace deployment mode. Each AppSpace supports only one application deployment. `-mapp` - Multiple Application per AppSpace deployment mode. Each AppSpace supports one or more application deployments. ActiveMatrix BusinessWorks™ supports both -sapp and -mapp modes. The default is -mapp mode.  This script dynamically creates a **bwadmin** cmd file in `cmd/Samples-Domain-<mode>.cmd` and executes it.	*$BW_HOME*`/samples/core/admin`
runSamples.sh	Creates Samples-Domain and deploys all EAR files found under `${BW_HOME}/samples/core/admin/ ears/samples` `-h` or `-help` - Displays this usage message. `<mode>` : `[-sapp]` or `[-mapp]` `-sapp` : Single Application per AppSpace deployment mode. Each AppSpace supports only one application deployment. `-mapp` : Multiple Application per AppSpace deployment mode. Each AppSpace supports one or more application deployments.  ActiveMatrix BusinessWorks™ supports both -sapp and -mapp modes. The default is -mapp mode. This script dynamically creates a `bwadmin` command file in `cmd/Samples-Domain-<mode>.cmd` and executes it.	*$BW_HOME*`/samples/core/admin`

Script	Description	Script Location
showprocs.sh	Shows process ID and complete binary path of all processes required in ActiveMatrix BusinessWorks™: • tibemsd • tea • bwagent • bwappnode • bwadmin	*$BW_HOME*/scripts/admin
tea.sh	This script starts TIBCO Enterprise Administrator in the background and waits until it is completely initialized, or the maxWait time (<n> * 2 sec) expires. -h or -help - Prints this usage message. [-maxWait <n>] - Max number of wait time (2 sec increment) for TIBCO Enterprise Administrator Server startup success. The default value for <n> is 30, which means 30 * 2 sec = 60 seconds.	*$BW_HOME*/scripts/admin
teaclean.sh	This utility script cleans TIBCO Enterprise Administrator Server's configuration data store. The end effect of this clean up is similar to a fresh installation of TIBCO Enterprise Administrator. -h or -help - Prints this usage message -force or -forceClean - Proceeds with wiping ActiveMatrix BusinessWorks™ Domain Data and internal data store without prompting user reconfirmation. By default, the script prompts user confirmation.	*$BW_HOME*/scripts/admin

Script	Description	Script Location
tibemsd64.sh	This script starts `tibemsd64` in the background and waits until it is completely initialized, or the maxWait time (<n> * 2 sec) expires. `-h` or `-help` - Prints this usage message `[-maxWait <n>]` - Max number of wait time (2 seconds increment) for `tibemsd64` start up success. The default value for <*n*> is 30, which means 30 * 2 sec = 60 seconds. This script is only supported on UNIX based systems. For Windows, use Windows Systems Services to start or stop `tibemsd64`.	*$BW_HOME*`/scripts/admin`

Each **runAcme.sh**, **runBookStore.sh**, **runSamples.sh**, **deploy.sh**, and **AppManage.sh** generates **bwadmin** commands before execution.

The generated **bwadmin** command files are found under `cmd` subdirectory.

Administrator and Agent

bwadmin and bwagent are used to create, manage, and monitor domains, AppSpaces, AppNodes, archives, and applications.

For more information, see the topics called bwadmin and bwagent.

Runtime entities are created in the local file system in the *BW_HOME*/domains folder. The default location of this folder can be changed. For information, see Configuring the Location of the Default Domains Folder.

bwadmin

bwadmin provides a command line console that can be used in local mode or enterprise mode to create and manage domains, AppSpaces, AppNodes, archives, and applications. Collectively, the entities provide the logical and physical structure for the runtime environment.

bwadmin provides the following features:

- One tool for both local and enterprise mode with identical commands
- Interactive shell
- Batch/silent mode by passing a command file as argument
- Ability to execute commands locally as well as remotely
- Ability to address different bwagent networks
- Simple and intuitive command structures
- Nested commands
- Unix-style commands for complex scripting
- Command completion

A full range of commands is available. Command can be executed stand-alone from the command line or from the bwadmin console. Unix-style scripts can be created to run bwadmin commands. When scripting, you may need to include conditions for possible error codes.

For more information about error codes and the corrective action to take, see the *TIBCO ActiveMatrix BusinessWorks™ Error Codes* guide.

Commands can be issued from:

- Interactive Mode: Useful for exploration. Commands are executed from the bwadmin shell. Any number of commands can be executed in a sequence.
- Command Line: Useful for execution of single commands. Commands are executed stand-alone from the command line with the provided syntax.
- Batch Mode: Useful for execution of repetitive commands.

To get help on a command, including syntax information, type **help** followed by the command name, from either interactive mode or the command line, for example:

```
bwadmin help
bwadmin help create
bwadmin help registerteaagent
```

Interactive Mode

Interactive mode is used for exploring runtime entities. Enter interactive mode by typing **bwadmin** at the command line. To view a list of available commands, press tab.

The cd command sets the runtime entity context so you can omit runtime entity options for commands like **create**, **delete**, **start**, or **stop**.

Command Line

bwadmin commands can be issued from the command line in the format: **`bwadmin [options] command <arguments>`**

To see the list of all bwadmin commands, type **`bwadmin help`** at the command line.

The following options can be specified for bwadmin at the command line:

bwadmin Command Options

Option	Description	Example
`-b / --batch`	Reads a series of commands from the standard input.	**`bwadmin --batch | bwadmin get admin.mode`**
`-config`	Applies the configuration in the specified file to the server instance.	**`bwadmin -config -d myDomain -a myAppSpace -cf`** *file_path*`/config.ini`
`-D <property=value>`	Applies the specified value to the specified property. Use the bwadmin **`get`** command to retrieve the value.	**`bwadmin -D name=User1`**
`-f <file[,<file>]`	Reads commands from the specified file or from the comma-separated list of files. The specified file can contain one command or multiple commands. Exits after command execution is completed.	**`bwadmin -f backupMyAppNode.cmd`**
`-l / --login <arg>`	Specifies the login ID to use for the session.	Given the following command: **`bwadmin -l User1`** bwadmin in interactive mode displays: `bwadmin[User1]>`
`-logconfig <file>`	Uses the specified file for logback configuration.	**`bwadmin -logconfig mylogback.xml`** For more information about logging, see Logging.

`-x,--xtrace`	Echoes the command to the terminal.	Given the following command **`bwadmin -x create domain MyDomain1`** the following sample output is issued:
		`TIBCO ActiveMatrix BusinessWorks version 6.2.0, build V20, 2014-10-09`
		`+ create domain MyDomain1`

`exit`	Exits the command line console.	**`bwadmin>exit`** It will exit the command line console.

For more information about bwadmin commands for different administration tasks, see the "bwadmin Command Line" topics in the Administration Tasks and Reference section.

Batch Mode

A command file can be passed to bwadmin at the command line with the **-f** option. The batch file should contain all required inputs. An example of a command file is a backup file created with the **backup** command.

The **-f eoe** command is an optional command and you can execute the bwadmin commands in a batch mode. If any of the commands fail, the subsequent commands are not executed. Syntax for the command is **bwadmin.exe -f eoe <command file>**

For example: **bwadmin.exe -f eoe bwadmin.sh**

bwagent

A bwagent is a daemon process that is responsible for provisioning AppNodes and applications, performing administration commands, and synchronizing data from the datastore with the local file system.

There is one bwagent for each installation. The bwagent enables communication between agents located on different machines. When multiple bwagents are configured to communicate with each other using a common datastore, they form a bwagent network. bwagents can communicate using TIBCO FTL® for communication transport and TIBCO Enterprise Message Service for communication transport, and by using an external database for data persistence .

For information about configuring the bwagent, see Configuring bwagent.

When multiple bwagents belong to a network and one of the system fails, the failed system can be restored after a restart by using the bwadmin **restore** command to force the file system to be synchronized with the datastore.

There are multiple ways to access the bwagent: bwadmin, the Admin UI, or the REST API.

- bwadmin: In enterprise mode, bwadmin sends commands to the bwagent. The bwagent dispatches the command to the targeted agent. For more information, see the "bwadmin Command Line" tasks under Administration Tasks and Reference section in the *TIBCO ActiveMatrix BusinessWorks™ Administration* guide.
- Admin UI: When the bwagent is registered with the TEA server, the Admin UI can be used to create and manage runtime entities. For more information, see the "Admin UI" tasks under "Administration Tasks and Reference" section in the *TIBCO ActiveMatrix BusinessWorks™ Administration* guide.
- REST API: View the bwagent REST API in the Swagger UI. For more information, see the section "Accessing the bwagent REST API with the Swagger UI" in the *TIBCO ActiveMatrix BusinessWorks™ Administration* guide.

bwagent supports its own set of commands. Commands are issued from the command line in the format: **bwagent [options] command <arguments>**

bwagent commands are listed below.

bwagent Commands

Command	Description
apiserver	Starts the apiserver that hosts the REST API in the Swagger UI. Open a browser and go to the following URL: `http://localhost:5555`
startagent	Starts the bwagent. This is the same as the default command when no command is given.

Command	Description
stop	Stops the bwagent gracefully.

The following options can be specified for bwagent:

bwagent Command Options

Option	Description	Example
-config	Applies the configuration in the specified file to the server instance.	**bwagent -config bwagent.ini**
-logconfig <file>	Uses the specified file for logback configuration.	**bwagent -logconfig mylogback.xml**
-x,--xtrace	Echoes the command to the terminal.	Given **bwagent -x** , the text `+startagent` is echoed to the console when the agent starts.

Configuring bwagent

The bwagent can be configured for a multi-agent, multi-machine environment.

The bwagent is configured using the `bwagent.ini` file in the *BW_HOME*`\config` folder. The bwagent.ini file template is a configuration file and contains the BW Admin data store configuration properties. Properties are pushed to the configuration file using JSON files.

Properties	Description
BW Agent general configuration	
`bw.admin.mode`	The Admin mode. BW Administration tools can work in two modes, enterprise mode or local mode. In the enterprise mode it works with the agents across machines. In the local mode it works only with local machine and assumes no data store and transport and agents are available. The actions performed in local mode are not visible to the agents when ever they are started or even the admin tool when it is started in the enterprise mode.
`bw.agent.networkName`	The name of the network. Must be the same for all the bwagents in the network. For more information, see Creating an Agent Network.
`bw.agent.memberName`	The name of the bwagent. Must be unique within the network. For more information, see Creating an Agent Network.

Properties	Description
`bw.agent.technology.db.create.schema`	Add this property manually to `bwagent.ini` file. Set the property to `true` to allow BWAgent to run create table script on startup. When you set the property to `false`, the default behavior of BWAgent changes, and it restricts BWAgent from runing create table script on startup.
Logging configuration	
`logback.configurationFile`	The logback configuration file to be used by the agent.
`bw.agent.http.port`	The HTTP port.
`bw.agent.http.host`	The HTTP interface (default=localhost).
`bw.appnode.agent.http.communication.port`	The internal HTTP communication port the Thor engine uses to communicate with bwagent to send the status of AppNodes and applications. Update this property to specify a port to start the internal server on. The default port number is 56565.
`bw.agent.http.access.log.config`	The HTTP Request Access Log Configuration file.
`bw.agent.bw.auth`	The authentication mechanism used by the REST API, BASIC (default) or DIGEST.
`bw.agent.https.port`	The secure port.
`bw.agent.https.truststorepath`	The truststore.
`bw.agent.https.truststorepassword`	The truststore password
`bw.agent.https.keystorepath`	The keystore.
`bw.agent.https.keystorepassword`	The keystore password.
`bw.agent.https.excludeprotocols`	The protocols to be excluded.
`bw.agent.https.includeprotocols`	The protocols to be included.
Configuration for AppNode to agent communication	
`bw.agent.appnode.user`	The user used by the AppNodes to communicate with the bwagent.

Properties	Description
`bw.agent.appnode.password`	The password for user used by AppNodes to communicate with the bwagent. If not set, the obfuscated password is read from the configured realm file. For example, $BW_HOME/config/realm.properties.
`bw.agent.appnode.status.notify.timeout`	Time interval in seconds when the AppNode reports its status to the bwagent.
TEA Agent configuration	
`bw.agent.tea.agent.host`	Identifies the bwagent for TEA to be registered.
`bw.agent.tea.agent.port`	Identifies the bwagent for TEA to be registered.
`bw.agent.tea.agent.context.path=/bwta`	Used to create bwagents register URL for TEA.
`bw.agent.tea.server.url`	The bwagent uses this URL to identitfy which server to be registered to.
Technology Type Configuration. Supported types are DBEMS or DBFTL	
`bw.agent.technology.type`	The provider to use for the datastore such as an external database (PostgreSQL , MySQL, Microsoft SQL, Oracle, db2 or MariaDB,) with transport as TIBCO FTL® or TIBCO Enterprise Message Service (EMS). Set to either: • `DBFTL` • `DBEMS` For more information, see the following topics: • Database with FTL Configuration for bwagent • Database with EMS Configuration for bwagent
`bw.agent.technology.requestTimeout`	Timeout for requests sent to other bwagents. The default value is 60000 ms.
DBEMS technology type	
`bw.agent.technology.dbems.db.provider`	Database provider. Supported options are postgresql, mysql and oracle database 12c, MS sqlserver, db2 and MariaDB.

Properties	Description
`bw.agent.technology.dbems.db.driver`	The DB driver. Example: dbDriver=org.postgresql.Driver Example: dbDriver=com.mysql.jdbc.Driver Example: dbDriver=oracle.jdbc.OracleDriver Example: dbDriver=com.microsoft.sqlserver.jdbc.SQLServerDriver Example: dbDriver=com.ibm.db2.jcc.DB2Driver Example: dbDriver=org.mariadb.jdbc.Driver
`bw.agent.technology.dbems.db.connectionURL`	The DB connection URL. Example: dbConnectionURL=jdbc:postgresql://db:5432/bwadmindb Example: dbConnectionURL=jdbc:mysql://db:3306/bwadmindb Example: dbConnectionURL=jdbc:oracle:thin:@db:1521:bwadmindb Example: dbConnectionURL=jdbc:sqlserver://db:1433;databaseName=bwadmindb Example: dbConnectionURL=jdbc:db2://db:50000/bwadmindb Example: dbConnectionURL=jdbc:mariadb://db:3306/databaseName=bwadmindb
`bw.agent.technology.dbems.db.userName`	The DB user. Example: dbUserName=bwuser
`bw.agent.technology.dbems.db.password`	The DB password. Example: dbPassword=bwuser
`bw.agent.technology.dbems.ems.serverUrl`	The EMS server URL. Example: emsServerUrl=tcp://ems:7222 Example: ldap://nn.nn.nnn.nnn:nnnnn/CN=admin,ou=users,o=tibco  Provide a comma-separated list to add multiple EMS servers.
`bw.agent.technology.dbems.ems.connectionFactoryName`	The EMS connection Factory name. Example: ldap.connectionFactoryName=QCFN
`bw.agent.technology.dbems.ems.userName`	The EMS user. Example: emsUserName=admin

Properties	Description
`bw.agent.technology.dbems.ems.password`	The EMS user password. Example: emsPassword=
`bw.agent.technology.dbems.ems.requestQueueName`	The EMS member queue. Example: requestQueueName=bw6.admin.operations.queue.{{membername}}
`bw.agent.technology.dbems.ems.qin.EMSPrefix`	The BW Agent Qin group name prefix. This property is optional and the default value is "EMSGMS".
`bw.agent.technology.dbems.ems.ssl.trust.identity`	The EMS ssl configuration. client identity consisting of the certificate, private key and optionally extra issuer certificates can be included into a single data block using PKCS12, KeyStore or Entrust Store encodings. Example:bw.agent.technology.dbems.ems.ssl.trust.identity={EMS_HOME}/samples/certs/client_identity.p12
`bw.agent.technology.dbems.ems.ssl.trust.cert.location`	The set of Trusted Certificates represents all trusted issuers of the server certificate. It must be specified by the client application unless the host certificate verification is completely disabled. Example: bw.agent.technology.dbems.ems.ssl.trust.location={EMS_HOME}/samples/certs/server_root.cert.pem
`bw.agent.technology.dbems.ems.ssl.trust.password`	EMS SSL connection trust password. This property is required if the JMS server protocol is ssl. The password may be clear text or supplied as an obfuscated string.
`bw.agent.technology.dbems.ems.ssl.disable.verify.host.name`	The trusted certificate commonname must match the ems server hostname if set to false.
`bw.agent.technology.dbems.ems.ssl.disable.verify.host`	The client and server certificates must match if set to false.
`bw.agent.technology.dbems.ems.reconnection.interval`	Interval for EMS reconnection. Value is in milliseconds (default: 10s).
DBFTL technology type	
`bw.agent.technology.dbftl.db.provider`	The Database provider. Supported options are postgresql, mysql and oracle database 12c, MS sqlserver, db2 and MariaDB.

Properties	Description
`bw.agent.technology.dbftl.db.driver`	The DB driver. Example: dbDriver=org.postgresql.Driver Example: dbDriver=com.mysql.jdbc.Driver Example: dbDriver=oracle.jdbc.OracleDriver Example: dbDriver=com.microsoft.sqlserver.jdbc.SQLServerDriver Example: dbDriver=com.ibm.db2.jcc.DB2Driver Example: dbDriver=org.mariadb.jdbc.Driver
`bw.agent.technology.dbftl.db.connectionURL`	The DB connection URL. Example: dbConnectionURL=jdbc:postgresql://db:5432/bwadmindb Example: dbConnectionURL=jdbc:mysql://db:3306/bwadmindb Example: dbConnectionURL=jdbc:oracle:thin:@db:1521:bwadmindb Example: dbConnectionURL=jdbc:sqlserver://db:1433;databaseName=bwadmindb Example: dbConnectionURL=jdbc:db2://db:50000/bwadmindb Example: dbConnectionURL=jdbc:mariadb://db:3306/databaseName=bwadmindb
`bw.agent.technology.dbftl.db.userName`	The DB user. Example: dbUserName=bwuser
`bw.agent.technology.dbftl.db.password`	The DB password. Example: dbPassword=bwuser
`bw.agent.technology.dbftl.ftl.realmserver`	The FTL Realm server URL. Example: ftlRealmServerUrl=http://localhost:8070
`bw.agent.technology.dbftl.ftl.application`	The FTL application name. Example: ftlApplicationName=bwadmin
`bw.agent.technology.dbftl.ftl.identifier`	The FTL identifier. Example: ftlIdentifier=
`bw.agent.technology.dbftl.ftl.secondary`	The FTL secondary realm server. Example: ftlSecondaryUrl=http://localhost:8070
`bw.agent.technology.dbftl.ftl.username`	The FTL user. Example: ftlUserName=admin

Properties	Description
`bw.agent.technology.dbftl.ftl.password`	The FTL user password. Example: ftlPassword=
`bw.agent.technology.dbftl.ftl.endpoint`	The FTL endpoint. Example: ftlEndpoint=bw-endpoint
`bw.agent.technology.dbftl.ftl.dataformat`	The FTL data format. Example: ftlDataformat=
`bw.agent.technology.dbftl.ftl.inbox`	The FTL inbox name. Example: ftlInbox=
Statistics Provider Configuration	
`bw.agent.technology.statsProvider`	The stats provider technology.
`bw.agent.technology.statsProvider.db.provider`	The Database provider. Supported options are postgresql, mysql and oracle database 12c, MS sqlserver, db2 and MariaDB.
`bw.agent.technology.statsProvider.db.driver`	The DB driver. Example: dbDriver=org.postgresql.Driver Example: dbDriver=com.mysql.jdbc.Driver # Example: dbDriver=oracle.jdbc.OracleDriver Example: dbDriver=com.microsoft.sqlserver.jdbc.SQLServerDriver Example: dbDriver=com.ibm.db2.jcc.DB2Driver Example: dbDriver=org.mariadb.jdbc.Driver
`bw.agent.technology.statsProvider.db.connectionURL`	The DB connection URL. Example: dbConnectionURL=jdbc:postgresql://db:5432/bwadmindb Example: dbConnectionURL=jdbc:mysql://db:3306/bwadmindb Example: dbConnectionURL=jdbc:oracle:thin:@db:1521:bwadmindb Example: dbConnectionURL=jdbc:sqlserver://db:1433;databaseName=bwadmindb Example: dbConnectionURL=jdbc:db2://db:50000/bwadmindb Example: dbConnectionURL=jdbc:mariadb://db:3306/databaseName=bwadmindb

Properties	Description
bw.agent.technology.statsProvider.db.userName	The DB user. Example: dbUserName=bwuser
bw.agent.technology.statsProvider.db.password	The DB password. Example: dbPassword=bwuser
Governance and Policy Director Configuration - The properties in this section are applicable to Governance Lifecycle Event Listener and it is used to communicate with the TIBCO Policy Director Administrator.	
bw.governance.enable	To enable or disable the Governance Lifecycle Event Listener. This property is optional and specifies whether the Governance Lifecycle Event Listener should be enabled or disabled in the BW Agent. The supported values are: true or false. The default value is 'false'.
bw.governance.jms.server.url	The Policy Director Administrator JMS URL. This property is optional and is used to specify the JMS server URL used to communicate with the Policy Director Administrator. If this property is not set, then the Lifecycle Event Listener will not attempt to connect to the JMS server. The URL is expected to start with 'tcp://' or 'ssl://' and the failover URLs can be specified as a ',' or '+' separated list.
bw.governance.jms.server.username	The Policy Director Administrator JMS User Name. This property is required if the Policy Director Administrator JMS URL is specified.
bw.governance.jms.server.password	The Policy Director Administrator JMS User Password. This property is required if the Policy Director Administrator JMS URL is specified.
bw.governance.jms.ssl.trust.store.type	The Policy Director Administrator JMS SSL connection trust store type. This property is required if the JMS server protocol is ssl. The supported values are 'JKS' and 'JCEKS'. The default value is 'JKS'.
bw.governance.jms.ssl.trust.store.location	The Policy Director Administrator JMS SSL connection trust store location. This property is required if the JMS server protocol is ssl.
bw.governance.jms.ssl.trust.store.password	The Policy Director Administrator JMS SSL connection trust store password. This property is required if the JMS server protocol is ssl. The password may be clear text or supplied as an obfuscated string.
bw.governance.jms.reconnect.attempt.count	The Policy Director Administrator JMS Connection attempt count. This property is required if the Policy Director Administrator JMS URL is specified and it specifies the number of JMS connection attempts the Lifecycle Event Listener will make. The default value is '120'.

Properties	Description
`bw.governance.jms.reconnect.attempt.timeout`	The Policy Director Administrator JMS Connection attempt timeout. This property # is required if the Policy Director Administrator JMS URL is specified and # it specifies the timeout between the attempt to reestablish connection to # the JMS server. The default value is '500'.
`bw.governance.jms.reconnect.attempt.delay`	The Policy Director Administrator JMS Connection attempt delay. This property is required if the Policy Director Administrator JMS URL is specified and it specifies the delay in milliseconds between attempts to establish reestablish connection the JMS server. The default value is '500'.
`bw.governance.jms.queue.pd.receiver.name`	The Policy Director Administrator JMS receiver queue name prefix. This property is required if the Policy Director Administrator JMS URL is specified and it specifies receiver queue name prefix for the Lifecycle Event Listener and Policy Director Administrator communication. This property value must match the value specified in the Policy Director Administrator configuration. The default value is 'governance.de.bw.default'.
`bw.governance.jms.application.property.<UserCustomProperty>`	The Policy Director Administrator JMS JNDI custom property. This property is optional and it provides the ability to specify custom property for the JMS JNDI Initial Context. For example to provide a custom property called "myProperty" for the JNDI Initial Context, then specify a property "bw.governance.jms.application.property.myProperty=".

The default location of the domains folder, where runtime entities are stored, can be changed. For information, see Configuring the Location of the Default Datastore.

Database with TIBCO FTL for bwagent

The bwagent can be configured to use TIBCO FTL for transport among bwagents. PostgreSQL, MySQL, Microsoft SQL, Oracle, and DB2 are the supported databases.

Use of TIBCO FTL® with TIBCO ActiveMatrix BusinessWorks™ for configuring bwagent and for configuring group provider for engine does not require TIBCO FTL® licenses.

Regularly back up domain data using the bwadmin **`backup`** command. For more information about backing up and restoring domain data, see Backing up and Restoring a Domain

For a multi-agent, multi-machine environment using an external database and TIBCO FTL, modify the following properties in the `bwagent.ini` file .

bwagent properties for Multi-Agent, Multi-Machine Environments using Database/FTL

Property Name	Description
bw.agent.technology.dbftl.db.provider	Set one of the following supported database providers: • **postgresql** • **mysql** • **mssql** • **oracle** • **db2**
bw.agent.technology.dbftl.db.driver	The database driver.
bw.agent.technology.dbftl.db.connectionURL	The URL to connect to the database.
bw.agent.technology.dbftl.db.userName	The user name to authenticate to the database.
bw.agent.technology.dbftl.db.password	The password to authenticate to the database.
bw.agent.technology.type	Set `dbftl` as the technology type for the bwagent to use.

bw.agent.technology.dbftl.ftl.realmserver	Set The FTL realm server.
	Example:
	`bw.agent.technology.dbftl.ftl.realmserver= http://localhost:8070`
	In case of FTL 6.x server in FT mode, set multiple realmserver values separated by pipe. (`|`).
	For example:
	`bw.agent.technology.dbftl.ftl.realmserver= http://10.97.240.76:8050 | http://10.97.240.76:8051 | http://10.97.240.76:8052`

bw.agent.technology.dbftl.ftl.application	Set the application name. Example: `bw.agent.technology.dbftl.ftl.application=bwadmin`
bw.agent.technology.dbftl.ftl.identifier	Set the FTL identifier.

Property Name	Description
bw.agent.technology.dbftl.ftl.secondary	Set the secondary realm server. This property is optional for FTL 5.x. This property is available in the `bwagent.ini` file only when you set the `ftlsecondary` property to `true` in the `bwagent_ftl.json` file. By default, the property is set to `false`.
bw.agent.technology.dbftl.ftl.username	Set the FTL user name.
bw.agent.technology.dbftl.ftl.password	Set the FTL password.
bw.agent.technology.dbftl.ftl.endpoint	Set the FTL endpoint. Example: `bw.agent.technology.dbftl.ftl.endpoint=bwadmin-endpoint`
bw.agent.technology.dbftl.ftl.dataformat	Set the FTL data format. Example: `bw.agent.technology.dbftl.ftl.dataformat=bw-format`
bw.agent.technology.dbftl.ftl.inbox	Set the FTL inbox. Example: `bw.agent.technology.dbftl.ftl.inbox=bw-inbox`
bw.agent.technology.requestTimeout	Timeout for requests sent to other BWAgents. The default value is 6000 milliseconds.
bw.agent.technology.remote.status.requestTimeout	Timeout for requests sent to BWAgents to find the status of AppNodes, applications, and other BWAgents. The default value is 3000 milliseconds.

For information about setting properties, see:

- PostgreSQL — For instructions, see Configuring bwagent for PostgreSQL and TIBCO FTL.
- MySQL — For instructions, see Configuring bwagent for MySQL and TIBCO FTL.
- Microsoft SQL — For instructions, see Configuring bwagent for Microsoft SQL and TIBCO FTL.
- Oracle — For instructions, see Configuring bwagent for Oracle and TIBCO FTL.
- DB2 — For instructions, see Configuring bwagent for DB2 and TIBCO FTL.

Configuring bwagent for PostgreSQL and TIBCO FTL®

The bwagent can be configured to use PostgreSQL database with TIBCO FTL for transport.

Use of TIBCO FTL with TIBCO ActiveMatrix BusinessWorks™ for configuring bwagent and for configuring group provider for engine does not require TIBCO FTL licenses.

The database name must be unique per agent network if multiple networks share the same physical database. bwagent and bwengine supports sharing the same database, users and schemas.

Prerequisites

- Install and configure TIBCO FTL on the same machine that you have installed TIBCO ActiveMatrix BusinessWorks™ 6.x on. For more information, see the "Setting Up TIBCO FTL® for bwagent Transport" topic in the *TIBCO ActiveMatrix BusinessWorks™ Installation* guide.

  For the version of TIBCO FTL that is supported with the version of ActiveMatrix BusinessWorks™ 6.x, see the ActiveMatrix BusinessWorks™ readme.
- Install PostgreSQL. The PostgreSQL driver is available by default.

Procedure

1. After installing PostgreSQL, create a database `bwadmindb` and the database owner, `bwuser` as described in the following steps:
   a) Run the following commands on the psql terminal:
   ```
   > psql -p 5432 -c "CREATE USER bwuser WITH CREATEDB PASSWORD 'bwuser';"
   > psql -p 5432 -c "CREATE DATABASE bwadmindb WITH OWNER bwuser;"
   ```
   b) Open the pgAdmin III utility and expand **Schemas > Tables** in the **Object Browser** to view the tables in the database.
   c) To add a password for the database owner, expand **Login Roles > bwuser**, right-click and choose **Properties**. Choose the **Definition** tab and create and save a password.
2. Stop the bwagent if it is running.
3. Open the `bwagent_ftl.json` file located in *`BW_HOME`*`\config` (Windows) or `${`*`BW_HOME`*`}/config` (Unix).
4. Update the following properties for your environment:

Property Name	PostgreSQL Value
dbtype	postgresql
dbdriver	org.postgresql.Driver
dbConnectionURL	jdbc:postgresql://localhost:5432/bwadmindb
dbuser	bwuser
dbpassword	bwuser

5. Run the bwadmin **`config`** command with the **`-cf`** option to push the changes from the JSON file to the `bwagent.ini` file.
   ```
   BW_HOME\bin>bwadmin config -cf ../config/bwagent_ftl.json agent
   ```
6. Restart the bwagent.

Configuring bwagent for MySQL and TIBCO FTL®

The bwagent can be configured to use MySQL Server database with TIBCO FTL for transport.

Use of TIBCO FTL with TIBCO ActiveMatrix BusinessWorks™ for configuring bwagent and for configuring group provider for engine does not require TIBCO FTL licenses.

The database name must be unique per agent network if multiple networks share the same physical database. bwagent and bwengine supports sharing the same database, users and schemas.

Prerequisites

- Install and configure TIBCO FTL on the same machine that you have installed ActiveMatrix BusinessWorks™ 6.x on. For more information, see the "Setting Up TIBCO FTL® for bwagent Transport" topic in the *TIBCO ActiveMatrix BusinessWorks™ Installation* guide.

  For the version of TIBCO FTL that is supported with the version of ActiveMatrix BusinessWorks™ 6.x, see the ActiveMatrix BusinessWorks™ readme.
- Download the MySQL server package `MySql_server_5.5.40` from http://dev.mysql.com/downloads/mysql/ and install MySQL. Configure the server configuration by following the prompts in the MySQL Server Configuration wizard. Ensure that you select the following values:
  - Database: Multifunctional
  - Type of connectivity: Manual
  - Default port: 3306
- Download the following JDBC driver and connector JAR files for MySQL to the *`BW_HOME`*`\config\drivers\shells\jdbc.mysql.runtime\runtime\plugins\com.tibco.bw.jdbc.datasourcefactory.mysql\lib` folder:
  - `com.mysql.jdbc.Driver.jar` from http://www.java2s.com/Code/Jar/c/Downloadcommysqljdbc515jar.htm
  - `MySql-connector-java-5.1.30-bin.jar` from http://dev.mysql.com/downloads/connector/j/.
- Install the MySQL driver by running the command **`bwinstall mysql-driver`** from the `/bin` folder.

After installing MySQL Server, create a database `bwadmindb` and grant privileges to the default database owner `root` as described in the following steps:

Procedure

1. Create a database `bwadmindb` and grant privileges to the default database owner `root` as described in the following steps:
   a) Run the following command on the MySQL terminal:
   ```
   mysql>create database bwadmindb
   ```
   b) Run the following command to view the tables included in the newly created database:
   ```
   mysql>use bwadmindb;
   ```
   c) Run the following command to grant all privileges to the root user for that database after replacing the value for the host IP address:
   ```
   mysql>GRANT ALL PRIVILEGES ON *.* TO 'root'@<host_IP> IDENTIFIED BY
   'Tibco123'WITH GRANT OPTION;
   ```
   To grant the minimum amount of permissions to the bwuser for that database, run the following command, where *<host_IP>* is replaced with the value for the host IP address:
   ```
   grant create,select,update,insert,delete ON *.* to 'bwuser'@<host_IP>
   IDENTIFIED BY 'bwuser';
   ```

2. Stop the bwagent if it is running.
3. Open the `bwagent_ftl.json` file located in *BW_HOME*`\config` (Windows) or `${`*BW_HOME*`}/config` (Unix).
4. Update the following properties for your environment:

Property Name	MySQL Value
dbtype	mysql
dbdriver	com.mysql.jdbc.Driver
dbconnectionurl	jdbc:mysql://localhost:3306/bwadmindb
dbuser	bwuser
dbpassword	bwuser

5. Run the bwadmin **`config`** command with the **`-cf`** option push the changes from the JSON file to the `bwagent.ini` file.

```
BW_HOME\bin>bwadmin config -cf ../config/bwagent_ftl.json agent
```

6. Restart the bwagent.

Configuring bwagent for Microsoft SQL Server and TIBCO FTL®

The bwagent can be configured to use Microsoft SQL database with TIBCO FTL for transport.

Use of TIBCO FTL with TIBCO ActiveMatrix BusinessWorks™ for configuring bwagent and for configuring group provider for engine does not require TIBCO FTL licenses.

The database name must be unique per agent network if multiple networks share the same physical database. bwagent and bwengine supports sharing the same database, users and schemas.

Prerequisites

- Install and configure TIBCO FTL on the same machine that you have installed ActiveMatrix BusinessWorks™ 6.x on. For more information, see the "Setting Up TIBCO FTL® for bwagent Transport" topic in the *TIBCO ActiveMatrix BusinessWorks™ Installation* guide.

  For the version of TIBCO FTL that is supported with the version of ActiveMatrix BusinessWorks™ 6.x, see the ActiveMatrix BusinessWorks™ readme.

- Install Microsoft SQL server. The Microsoft SQL driver is available by default.

After installing Microsoft SQL Server, create a database `bwadmindb` and grant privileges to the default database owner `root` as described in the following steps:

Procedure

1. Open Microsoft SQL Server Management Studio.
2. From the **Object Explorer** pane, right-click **Databases** > **New Database** to create the database `bwadmindb`.
3. Right-click **Security** > **Logins** > **New Login...**.
4. Make the following configurations from the Login-New window:
   a) Type `bwuser` in the **Login name** field.

b) Select **SQL Server Authentication**.
c) **Optional.** Unselect **Enforce password policy**, **Enforce password expiration**, or **User must change password at next login**.
d) In the **Default database** field, select **bwadmindb**.
e) From the Select a page pane, on the left side of the Login - New window, click on the **Server Roles** tab.
f) To configure the bwagent with MS SQL Server, set the values for Minimum Server Role and Database Role required for a user, for a particular database. In MS SQL Server Management Server, navigate to **Security** > **Logins**. Right click **Login Properties** > **Server Roles**. The minimum server role required for a particular user is *public*. Under User Mapping, the minimum database role membership for the selected database for a user mapped to the login should be one of the following two combinations: public and db_owner OR public, db_datawriter, db_datareader, and db_ddladmin.
g) Click **OK**.

5. Stop the bwagent if it is running.
6. Open the `bwagent_ftl.json` file located in *`BW_HOME`*`\config` (Windows) or `${`*`BW_HOME`*`}/config` (Unix).
7. Update the following properties for your environment:

Property Name	Microsoft SQL Value
dbtype	sqlserver
dbdriver	com.microsoft.sqlserver.jdbc.SQLServerDriver
dbconnectionurl	jdbc:sqlserver:// localhost:1433;databaseName=bwadmindb
dbuser	bwuser
dbpassword	bwuser

8. Run the bwadmin **`config`** command with the **`-cf`** option push the changes from the JSON file to the `bwagent.ini` file.

```
BW_HOME\bin>bwadmin config -cf ../config/bwagent_ftl.json agent
```

9. Restart the bwagent.

Configuring bwagent for Oracle and TIBCO FTL®

The bwagent can be configured to use Oracle database with TIBCO FTL for transport.

Use of TIBCO FTL with TIBCO ActiveMatrix BusinessWorks™ for configuring bwagent and for configuring group provider for engine does not require TIBCO FTL licenses.

The database name must be unique per agent network if multiple networks share the same physical database. bwagent and bwengine supports sharing the same database.

Prerequisites

Install and configure TIBCO FTL on the same machine that you have installed ActiveMatrix BusinessWorks™ 6.x on. For more information, see the "Setting Up TIBCO FTL® for bwagent Transport" topic in the *TIBCO ActiveMatrix BusinessWorks™ Installation* guide.

For the version of TIBCO FTL that is supported with the version of ActiveMatrix BusinessWorks™ 6.x, see the ActiveMatrix BusinessWorks™ readme.

Ensure you have installed FTL client libraries. For more information, see the "Integrating with TIBCO FTL" section in the *TIBCO ActiveMatrix BusinessWorks™ Installation* guide.

Install Oracle Database 12c:

1. Download and install Oracle Database 12c from http://www.oracle.com/technetwork/database/enterprise-edition/downloads/index.html.
2. Configure the server configuration by following the prompts in the Oracle Configuration wizard.
3. Accept the default port value 1521, or enter your own port number.
4. Download the following JDBC driver connector JAR files to the *BW_HOME*`\config\drivers\shells\jdbc.oracle.runtime\runtime\plugins\com.tibco.bw.jdbc.datasourcefactory.oracle\lib` folder for Windows or the `${`*BW_HOME*`}/system/lib` for Unix:
   a. `ojdbc7.jar` from http://www.oracle.com/technetwork/database/features/jdbc/default-2280470.html
5. Install the Oracle driver by running the command **`bwinstall oracle-driver`** from the `/bin` folder.

If you are using Oracle Database 11g, execute the oracle11g_create.sql script at *BW_HOME*`/config/dbscripts/admin/oracle` and restart the bwagent.

Procedure

1. After installing Oracle Database 12c, create a database `bwadmindb` and grant privileges to the default database owner `bwuser` as described in the following steps:
   a) Run the following commands in SQLPlus as a root user:

```
CREATE DATABASE bwadmindb;
create USER C##bwuser identified by "bwuser";
GRANT CREATE SESSION TO C##bwuser;
grant create sequence to C##bwuser;
ALTER USER C##bwuser quota unlimited on USERS;
grant create table to C##bwuser;
```

2. Stop the bwagent if it is running.
3. Open the `bwagent_ftl.json` file located in *BW_HOME*`\config` (Windows) or `${`*BW_HOME*`}/config` (Unix).
4. Update the following properties for your environment:

Property Name	Oracle Value
dbtype	oracle
dbdriver	oracle.jdbc.OracleDriver
dbconnectionurl	jdbc:oracle:thin:@db:1521:bwadmindb
dbuser	Cbwuser
dbpassword	bwuser

5. Run the bwadmin **`config`** command with the **`-cf`** option to push the changes from the JSON file to the `bwagent.ini` file.

```
BW_HOME\bin>bwadmin config -cf ../config/bwagent_ftl.json agent
```

If you are creating a user with '##' (e.g. c##bwuser), then you need to keep the **username** field as empty in the `bwagent_db.json` file and later update the `bwagent.ini` file manually.

6. Restart the bwagent.

Configuring bwagent for DB2 and TIBCO FTL®

The bwagent can be configured to use DB2 database with TIBCO FTL for transport.

Use of TIBCO FTL with TIBCO ActiveMatrix BusinessWorks™ for configuring bwagent and for configuring group provider for engine does not require TIBCO FTL licenses.

The database name must be unique per agent network if multiple networks share the same physical database.

Prerequisites

- Install and configure TIBCO FTL on the same machine that you have installed ActiveMatrix BusinessWorks™ 6.x on. For more information, see the "Setting Up TIBCO FTL® for bwagent Transport" topic in the *TIBCO ActiveMatrix BusinessWorks™ Installation* guide.

  For the version of TIBCO FTL that is supported with the version of ActiveMatrix BusinessWorks™ 6.x, see the ActiveMatrix BusinessWorks™ readme.
- Download the DB2 server package 10.5 from http://www-01.ibm.com/software/data/db2/linux-unix-windows/downloads.html and install DB2. Configure the server configuration by following the prompts in the DB2 Server Configuration wizard.
- Download the following JDBC driver and connector JAR files for DB2 to the *BW_HOME*`\config\drivers\shells\jdbc.db2.runtime\runtime\plugins\com.tibco.bw.jdbc.datasourcefactory.db2\lib` folder for Windows or the `${`*BW_HOME*`}/system/lib` folder for Unix:

  – `db2jcc4.jar from http://www-01.ibm.com/support/docview.wss?uid=swg21363866.`
- Install the DB2 driver by running the command **`bwinstall db2-driver`** from the `/bin` folder.

Procedure

1. Log in to DB2 and create database by executing the following command:
   `CREATE DATABASE <database name> USING CODESET UTF-8 TERRITORY US COLLATE USING SYSTEM PAGESIZE 16384`

   Set the page size to 16K (16384) or higher.
2. Stop the bwagent if it is running.
3. Open the `bwagent_ftl.json` file located in *BW_HOME*`\config` (Windows) or `${`*BW_HOME*`}/config` (Unix).
4. Update the following properties for your environment:

Property Name	DB2 Value
dbtype	db2
dbdriver	com.ibm.db2.jcc.DB2Driver

Property Name	DB2 Value
dbconnectionurl	jdbc:db2://localhost:50000/bwadmindb
dbuser	bwuser
dbpassword	bwuser

5. Run the bwadmin **`config`** command with the **`-cf`** option to push the changes from the JSON file to the `bwagent.ini` file.
```
BW_HOME\bin>bwadmin config -cf ../config/bwagent_ftl.json agent
```
6. Restart the bwagent.

Configuring bwagent for MariaDB and TIBCO FTL®

The bwagent can be configured to use MariaDB database with TIBCO FTL for persistence and transport.

The database name must be unique per agent network if multiple networks share the same physical database. bwagent and bwengine supports sharing the same database, users and schemas.

Prerequisites

- Install TIBCO Enterprise Messaging Server 8.1 and start the server.
- Download the MariaDB server package `MariaDB_server_10.1` from https://downloads.mariadb.org/ and install MariaDB. Configure the server configuration by following the prompts in the MariaDB Server Configuration wizard. Ensure that you select the following values:
  - Database: Multifunctional
  - Type of connectivity: Manual
  - Default port: 3306
- Download the following JAR files for MariaDB to the *`BW_HOME`*`\config\drivers\shells\jdbc.mariadb.runtime\runtime\plugins\com.tibco.bw.jdbc.datasourcefactory.mariadb\lib` folder:
  - mariadb-java-client-2.0.1.jar from https://downloads.mariadb.org/connector-java/2.0.1/.
- Install the MariaDB driver by running the command **`bwinstall mariadb-driver`** from the `/bin` folder.
- Ensure you have installed EMS client libraries. For more information, see "Integrating with TIBCO Enterprise Message Service™" in the *TIBCO ActiveMatrix BusinessWorks™ Installation* guide.
- Optional. If you are upgrading to ActiveMatrix BusinessWorks™ 6.3.2, or later versions of the software, upgrade the database schema. For more information, see "Updating the Database Schema to Enable bwagent to Use Database with TIBCO Enterprise Message Service" in the *TIBCO ActiveMatrix BusinessWorks™ Installation* guide.

Procedure

1. Create a database `bwadmindb` and grant privileges to the default database owner `root` as described in the following steps:
   a) Run the following command on the MariaDB terminal:
```
mariadb>create database bwadmindb
```

b) Run the following command to view the tables included in the newly created database:

```
mariadb>use bwadmindb;
```

c) Run the following command to grant all privileges to the root user for that database after replacing the value for the host IP address:

```
mariadb>GRANT ALL PRIVILEGES ON *.* TO root@<host_IP> IDENTIFIED BY
'Tibco123'WITH GRANT OPTION;
```

To grant the minimum amount of permissions to the bwuser for that database, run the following command, where <*host_IP*> is replaced with the value for the host IP address:

```
grant create,select,update,insert,delete ON *.* to bwuser@<host_IP> IDENTIFIED
BY "bwuser";
```

2. Stop the bwagent if it is running.
3. Open the `bwagent_db.json` file located in *BW_HOME*`\config` (Windows) or `${`*BW_HOME*`}/config` (Unix).
4. Update the following properties for your environment:

Property Name	MariaDB Value
dbtype	mariadb
dbdriver	org.mariadb.jdbc.Driver
dbconnectionurl	jdbc:mariadb://localhost:3306/bwadmindb
dbuser	bwuser
dbpassword	bwuser

5. Run the bwadmin **`config`** command with the **`-cf`** option to create an `.ini` file in the correct location.

```
BW_HOME\bin>bwadmin config -cf ../config/bwagent_db.json agent
```

6. Restart the bwagent.

Database with TIBCO Enterprise Message Service™ Configuration for bwagent

The bwagent can be configured to use an external relational database for persistence and TIBCO Enterprise Message Service™ (EMS) for transport among bwagents. PostgreSQL, MySQL, Microsoft SQL, and Oracle are the supported databases.

In your production environment, ensure you are using an external database for data persistence and either TIBCO FTL® or TIBCO Enterprise Message Service™ for communication transport.

Regularly back up domain data using the bwadmin **`backup`** command. For more information about backing up and restoring domain data, see Backing up and Restoring a Domain

For a non admin EMS user, the following permissions are to be given to use bwagent:

```
grant topic "$sys.monitor.>" user=<username> all
grant admin user=<username> view-connection,view-server,create-destination
```

Make sure that TIBCO Enterprise Message Service™ (EMS) server has a dynamic permission for topics or queues. For more information, see "Wildcards and Dynamically Created Destinations" in the *TIBCO Enterprise Message Service™ User's Guide*.

For a multi-agent, multi-machine environment using an external database and TIBCO Enterprise Message Service, the following properties in the `bwagent.ini` file are important.

bwagent properties for Multi-Agent, Multi-Machine Environments using Database/EMS

Property Name	Description
bw.agent.technology.dbems.db.provider	Database provider. One of: • **postgresql** • **mysql** • **mssql** • **oracle** • **db2**
bw.agent.technology.dbems.db.driver	The database driver. Example: `dbDriver=org.postgresql.Driver`
bw.agent.technology.dbems.db.connectionURL	The URL to connect to the database. Example: `jdbc:postgresql://localhost:5432/bwadmindb`
bw.agent.technology.dbems.db.userName	The user name to authenticate to the database.
bw.agent.technology.dbems.db.password	The password to authenticate to the database.
bw.agent.technology.dbems.ems.serverUrl	The URL to connect to the EMS server. Example: `tcp://localhost:7222`
bw.agent.technology.dbems.ems.userName	The user name to authenticate to the EMS server. The default is **admin.** To authenticate a non-admin user, create a user and set the password. Run the following two commands in the TIBCO EMS admin console: • **grant topic "$sys.monitor.>" user=sri2 all** • **grant admin user=sri2 view-connection,view-server**
bw.agent.technology.dbems.ems.password	The password to authenticate to the EMS server. There is no password by default. You can provide obfuscated password. For more information about how to obfuscate passwords, see Obfuscating or Encrypting Password for Database, EMS, and FTL Users.

Property Name	Description
bw.agent.technology.dbems.ems.requestQueueName	Member Queue Name. Set the value as **bw6.admin.operations.queue.<memberQueueName>** where **memberQueueName** is the value of **bw.agent.memberName**. For example, If `bw.agent.memberName=machine1`, `bw.agent.technology.dbems.ems.requestQueueName=bw6.admin.operations.queue.machine1`
bw.agent.technology.requestTimeout	Timeout for requests sent to other bwagents The default value is 6000 milliseconds.
bw.agent.technology.remote.status.requestTimeout	Timeout for requests sent to bwagents to find the status of AppNodes, applications, and other bwagents. The default value is 3000 milliseconds.
bw.agent.technology.dbems.ems.reconnection.interval	Set the **bw.agent.technology.dbems.ems.reconnection.interval** property to specify, in milliseconds, how often the bwagent checks its connection with the EMS server. The default value is 10000 milliseconds.

For information about setting properties, see:

- PostgreSQL — For instructions, see Configuring bwagent for PostgreSQL and TIBCO Enterprise Message Service.
- MySQL — For instructions, see Configuring bwagent for MySQL and TIBCO Enterprise Message Service.
- Microsoft SQL — For instructions, see Configuring bwagent for Microsoft SQL Server and TIBCO Enterprise Message Service.
- Oracle — For instructions, see Configuring bwagent for Oracle and TIBCO Enterprise Message Service.
- DB2 — For instructions, see Configuring bwagent for DB2 and TIBCO Enterprise Message Service.

Configuring BWAgent for PostgreSQL and TIBCO Enterprise Message Service

The BWAgent can be configured to use PostgreSQL database with TIBCO Enterprise Message Service (EMS) for persistence and transport.

The database name must be unique per agent network if multiple networks share the same physical database. BWAgent and BW Engine supports sharing the same database, users and schemas.

Prerequisites

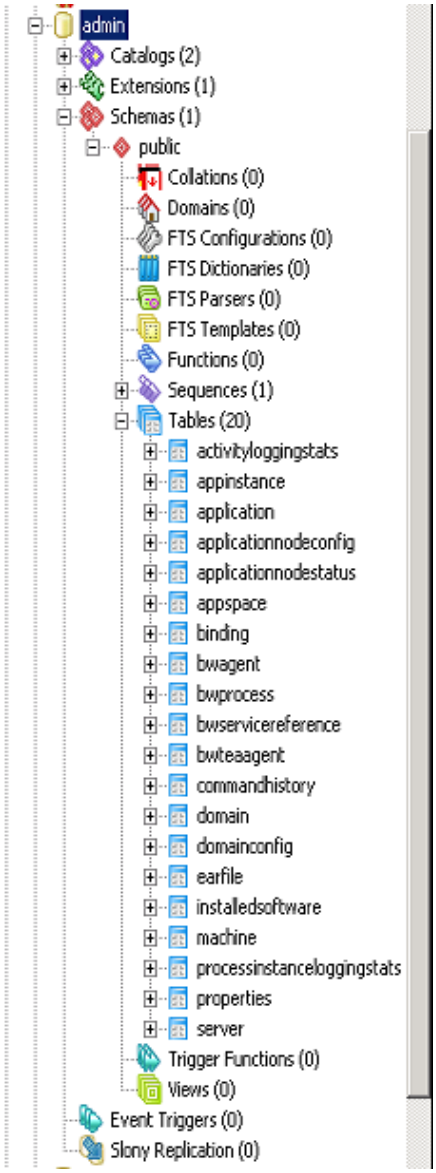
- Install TIBCO Enterprise Messaging Server 8.1 and start the server.
- Install PostgreSQL. The PostgreSQL driver is available by default.
- Ensure you have installed EMS client libraries. For more information, see the "Integrating with TIBCO Enterprise Message Service™" section in the *TIBCO ActiveMatrix BusinessWorks™ Installation* guide.
- Optional. If you are upgrading to ActiveMatrix BusinessWorks™ 6.3.2, or later versions of the software, upgrade the database schema. For more information, see the "Updating the Database Schema to Enable bwagent to Use Database with TIBCO Enterprise Message Service" section in the *TIBCO ActiveMatrix BusinessWorks™ Installation* guide.

Procedure

1. After installing PostgreSQL, create a database `bwadmindb` and the database owner, `bwuser` as described in the following steps:
   a) Run the following commands on the psql terminal:
   ```
   > psql -p 5432 -c "CREATE USER bwuser WITH CREATEDB PASSWORD 'bwuser';"
   > psql -p 5432 -c "CREATE DATABASE bwadmindb WITH OWNER bwuser;"
   ```
   b) To add a password for the database owner, expand **Login Roles > bwuser**, right-click and choose **Properties**. Choose the **Definition** tab and create and save a password.
2. Stop the bwagent if it is running.
3. Open the `bwagent_db.json` file located in *BW_HOME*`\config` (Windows) or `${`*BW_HOME*`}/config` (Unix).
4. Update the following properties for your environment:

Property Name	PostgreSQL Value
dbtype	postgresql
dbdriver	org.postgresql.Driver
dbconnectionURL	jdbc:postgresql://localhost:5432/bwadmindb
dbuser	bwuser
dbpassword	bwuser

5. Run the bwadmin **`config`** command with the **`-cf`** option push the changes from the JSON file to the `bwagent.ini` file.
   ```
   BW_HOME\bin>bwadmin config -cf ../config/bwagent_db.json agent
   ```
6. Restart the bwagent.
7. Open the pgAdmin III utility and expand **Schemas** > **Tables** in the **Object Browser** to view the tables in the database:



Configuring BWAgent for MySQL and TIBCO Enterprise Message Service

The BWAgent can be configured to use MySQL database with TIBCO Enterprise Message Service (EMS) for persistence and transport.

The database name must be unique per agent network if multiple networks share the same physical database. BWAgent and BW Engine supports sharing the same database, users and schemas.

Prerequisites

- Install TIBCO Enterprise Messaging Server 8.1 and start the server.
- Download the MySQL server package `MySql_server_5.5.40` from http://dev.mysql.com/downloads/mysql/ and install MySQL. Configure the server configuration by following the prompts in the MySQL Server Configuration wizard. Ensure that you select the following values:
  - Database: Multifunctional
  - Type of connectivity: Manual

  - Default port: 3306
- Download the following connector JAR files for MySQL to the *BW_HOME*\config\drivers\shells\jdbc.mysql.runtime\runtime\plugins\com.tibco.bw.jdbc.datasourcefactory.mysql\lib folder:
  - MySql-connector-java-5.1.30-bin.jar from http://dev.mysql.com/downloads/connector/j/.
- Install the MySQL driver by running the command **bwinstall mysql-driver** from the /bin folder.
- Ensure you have installed EMS client libraries. For more information, see the "Integrating with TIBCO Enterprise Message Service™" section in the *TIBCO ActiveMatrix BusinessWorks™ Installation* guide for additional details.
- Optional. If you are upgrading to ActiveMatrix BusinessWorks™ 6.3.2, or later versions of the software, upgrade the database schema. For more information, see the "Updating the Database Schema to Enable bwagent to Use Database with TIBCO Enterprise Message Service" section in the *TIBCO ActiveMatrix BusinessWorks™ Installation* guide.

Procedure

1. Create a database bwadmindb and grant privileges to the default database owner root as described in the following steps:
   a) Run the following command on the MySQL terminal:
   ```
   mysql>create database bwadmindb
   ```
   b) Run the following command to view the tables included in the newly created database:
   ```
   mysql>use bwadmindb;
   ```
   c) Run the following command to grant all privileges to the root user for that database after replacing the value for the host IP address:
   ```
   mysql>GRANT ALL PRIVILEGES ON *.* TO 'root'@<host_IP> IDENTIFIED BY
   'Tibco123'WITH GRANT OPTION;
   ```
   To grant the minimum amount of permissions to the bwuser for that database, run the following command, where <*host_IP*> is replaced with the value for the host IP address:
   ```
   grant create,select,update,insert,delete ON *.* to 'bwuser'@<host_IP>
   IDENTIFIED BY 'bwuser';
   ```
2. Stop the bwagent if it is running.
3. Open the bwagent_db.json file located in *BW_HOME*\config (Windows) or ${*BW_HOME*}/config (Unix).
4. Update the following properties for your environment:

Property Name	MySQL Value
dbtype	mysql
dbdriver	com.mysql.jdbc.Driver
dbconnectionurl	jdbc:mysql://localhost:3306/bwadmindb
dbuser	bwuser
dbpassword	bwuser

5. Run the bwadmin **config** command with the **-cf** option to create an .ini file in the correct location.
   ```
   BW_HOME\bin>bwadmin config -cf ../config/bwagent_db.json agent
   ```

6. Restart the bwagent.

Configuring bwagent for Microsoft SQL Server and TIBCO Enterprise Message Service

The bwagent can be configured to use Microsoft SQL Server database with TIBCO Enterprise Message Service (EMS) for persistence and transport.

The database name must be unique per agent network if multiple networks share the same physical database. bwagent and bwengine supports sharing the same database, users and schemas.

Prerequisites

- Install TIBCO Enterprise Messaging Server 8.1 and start the server.
- The Microsoft SQL driver is available by default.
- Ensure you have installed EMS client libraries. For more information, see the "Integrating with TIBCO Enterprise Message Service™" section in the *TIBCO ActiveMatrix BusinessWorks™ Installation* guide.
- Optional. If you are upgrading to ActiveMatrix BusinessWorks™ 6.3.2, or later versions of the software, upgrade the database schema. For more information, see the "Updating the Database Schema to Enable bwagent to Use Database with TIBCO Enterprise Message Service" section in the *TIBCO ActiveMatrix BusinessWorks™ Installation* guide for additional details.

After installing Microsoft SQL Server, create a database `bwadmindb` and grant privileges to the default database owner `root` as described in the following steps:

Procedure

1. Open Microsoft SQL Server Management Studio.
2. From the **Object Explorer** pane, right-click **Databases > New Database** to create the database `bwadmindb`.
3. Right-click **Security > Logins > New Login...**.
4. Make the following configurations from the Login-New window:
   a) Type `bwuser` in the **Login name** field.
   b) Select **SQL Server Authentication**.
   c) **Optional.** Unselect **Enforce password policy**, **Enforce password expiration**, or **User must change password at next login**.
   d) In the **Default database** field, select **bwadmindb**.
   e) From the Select a page pane, on the left side of the Login - New window, click on the **Server Roles** tab.
   f) To configure the bwagent with MS SQL Server, set the values for Minimum Server Role and Database Role required for a user, for a particular database. In MS SQL Server Management Server, navigate to **Security** > **Logins**. Right click **Login Properties** > **Server Roles**. The minimum server role required for a particular user is *public*. Under User Mapping, the minimum database role membership for the selected database for a user mapped to the login should be one of the following two combinations: public and db_owner OR public, db_datawriter, db_datareader, and db_ddladmin.
   g) Click **OK**.
5. Stop the bwagent if it is running.
6. Open the `bwagent_db.json` file located in *BW_HOME*`\config` (Windows) or `${`*BW_HOME*`}/config` (Unix).
7. Update the following properties for your environment:

Property Name	Microsoft SQL Value
dbtype	sqlserver
dbdriver	com.microsoft.sqlserver.jdbc.SQLServerDriver
dbconnectionurl	jdbc:sqlserver:// localhost:1433;databaseName=bwadmindb
dbuser	bwuser
dbpassword	bwuser

8. Run the bwadmin **`config`** command with the **`-cf`** option push the changes from the JSON file to the `bwagent.ini` file.
```
BW_HOME\bin>bwadmin config -cf ../config/bwagent_db.json agent
```
9. Restart the bwagent.

Configuring bwagent for Oracle and TIBCO Enterprise Message Service

The bwagent can be configured to use Oracle database with TIBCO Enterprise Message Service (EMS) for persistence and transport.

The database name must be unique per agent network if multiple networks share the same physical database. bwagent and bwengine supports sharing the same database.

Prerequisites

- Install TIBCO Enterprise Messaging Server 8.1 and start the server.
- Install Oracle Database 12c:
  1. Download and install Oracle Database 12c from http://www.oracle.com/technetwork/database/enterprise-edition/downloads/index.html.
  2. Configure the server configuration by following the prompts in the Oracle Configuration wizard.
  3. Accept the default port value 1521, or enter your own port number.
  4. Download the following JDBC driver connector JAR files to the *BW_HOME*`\config\drivers\shells\jdbc.oracle.runtime\runtime\plugins\com.tibco.bw.jdbc.datasourcefactory.oracle\lib` folder:
     a. `ojdbc7.jar` from http://www.oracle.com/technetwork/database/features/jdbc/default-2280470.html
- Install the Oracle driver by running the command **`bwinstall oracle-driver`** from the `/bin` folder.

If you are using Oracle Database 11g, execute the oracle11g_create.sql script at *BW_HOME*`/config/dbscripts/admin/oracle` and restart the bwagent.

- Ensure you have installed EMS client libraries. For more information, see the "Integrating with TIBCO Enterprise Message Service™" section in the *TIBCO ActiveMatrix BusinessWorks™ Installation* guide.
- Optional. If you are upgrading to ActiveMatrix BusinessWorks™ 6.3.2, or later versions of the software, upgrade the database schema.For more information, see the "Updating the Database Schema to Enable bwagent to Use Database with TIBCO Enterprise Message Service" section in the *TIBCO ActiveMatrix BusinessWorks™ Installation* guide.

Procedure

1. After installing Oracle Database 12c, create a database `bwadmindb` and grant privileges to the default database owner `bwuser` as described in the following steps:
   a) Run the following commands in SQLPlus as a root user:

```
CREATE DATABASE bwadmindb;
create USER C##bwuser identified by "bwuser";
GRANT CREATE SESSION TO C##bwuser;
grant create sequence to C##bwuser;
ALTER USER C##bwuser quota unlimited on USERS;
grant create table to C##bwuser;
```

2. Stop the bwagent if it is running.
3. Open the `bwagent_db.json` file located in *BW_HOME*`\config` (Windows) or `${`*BW_HOME*`}/config` (Unix).
4. Update the following properties for your environment:

Property Name	Oracle Value
dbtype	oracle
dbdriver	oracle.jdbc.OracleDriver
dbconnectionurl	jdbc:oracle:thin:@localhost:1521:bwadmindb
dbuser	Cbwuser
dbpassword	bwuser

5. Run the bwadmin **`config`** command with the **`-cf`** option push the changes from the JSON file to the `bwagent.ini` file.

```
BW_HOME\bin>bwadmin config -cf ../config/bwagent_db.json agent
```

If you are creating a user with '##' (e.g. c##bwuser), then you need to keep the **`username`** field as empty in the `bwagent_db.json` file and later update the `bwagent.ini` file manually.

6. Restart the bwagent.

Configuring bwagent for DB2 and TIBCO Enterprise Message Service

The bwagent can be configured to use DB2 database with TIBCO Enterprise Message Service (EMS) for persistence and transport.

The database name must be unique per agent network if multiple networks share the same physical database.

Prerequisites

- Install TIBCO Enterprise Messaging Server 8.1 and start the server.
- Download the DB2 server package 10.5 from http://www-01.ibm.com/software/data/db2/linux-unix-windows/downloads.html and install DB2. Configure the server configuration by following the prompts in the DB2 Server Configuration wizard.

- Download the following JDBC driver and connector JAR files for DB2 to the *BW_HOME*\config\drivers\shells\jdbc.db2.runtime\runtime\plugins\com.tibco.bw.jdbc.datasourcefactory.db2\lib folder:
  - db2jcc4.jar from http://www-01.ibm.com/support/docview.wss?uid=swg21363866.
- Install the DB2 driver by running the command **bwinstall db2-driver** from the /bin folder.
- Ensure you have installed EMS client libraries. For more information, see the "Integrating with TIBCO Enterprise Message Service™" section in the *TIBCO ActiveMatrix BusinessWorks™ Installation* guide.
- Optional. If you are upgrading to ActiveMatrix BusinessWorks™ 6.3.2, or later versions of the software, upgrade the database schema. For more information, see the "Updating the Database Schema to Enable bwagent to Use Database with TIBCO Enterprise Message Service" section in the *TIBCO ActiveMatrix BusinessWorks™ Installation* guide.

Procedure

1. Log in to DB2 and create database by executing the following command:
   CREATE DATABASE <database name> USING CODESET UTF-8 TERRITORY US COLLATE USING SYSTEM PAGESIZE 16384

   Set the page size to 16K (16384) or higher.
2. Stop the bwagent if it is running.
3. Open the bwagent_db.json file located in *BW_HOME*\config (Windows) or ${*BW_HOME*}/config (Unix).
4. Update the following properties for your environment:

Property Name	DB2 Value
dbtype	db2
dbdriver	com.ibm.db2.jcc.DB2Driver
dbconnectionurl	jdbc:db2://localhost:50000/bwadmindb
dbuser	bwuser
dbpassword	bwuser

5. Run the bwadmin **config** command with the **-cf** option push the changes from the JSON file to the bwagent.ini file.
   ```
   BW_HOME\bin>bwadmin config -cf ../config/bwagent_db.json agent
   ```
6. Restart the bwagent.

Configuring bwagent for MariaDB and TIBCO EMS

The bwagent can be configured to use MariaDB database with TIBCO Enterprise Message Service (EMS) for persistence and transport.

The database name must be unique per agent network if multiple networks share the same physical database. bwagent and bwengine supports sharing the same database, users and schemas.

Prerequisites

- Install TIBCO Enterprise Messaging Server 8.1 and start the server.
- Download the MariaDB server package `MariaDB_server_10.1` from https://downloads.mariadb.org/ and install MariaDB. Configure the server configuration by following the prompts in the MariaDB Server Configuration wizard. Ensure that you select the following values:
  - Database: Multifunctional
  - Type of connectivity: Manual
  - Default port: 3306
- Download the following JAR files for MariaDB to the *BW_HOME*`\config\drivers\shells\jdbc.mariadb.runtime\runtime\plugins\com.tibco.bw.jdbc.datasourcefactory.mariadb\lib` folder:
  - mariadb-java-client-2.0.1.jar from https://downloads.mariadb.org/connector-java/2.0.1/.
- To install the MariaDB driver, run the command **`bwinstall mariadb-driver`** from the `/bin` folder.
- Ensure you have installed EMS client libraries. For more information see Integrating with TIBCO Enterprise Message Service™ in the *TIBCO ActiveMatrix BusinessWorks™ Installation* guide.
- Optional. If you are upgrading to ActiveMatrix BusinessWorks™ 6.3.2, or later versions of the software, upgrade the database schema. For more information, see "Updating the Database Schema to Enable bwagent to Use Database with TIBCO Enterprise Message Service" in the *TIBCO ActiveMatrix BusinessWorks™ Installation* guide.

Procedure

1. Create a database `bwadmindb` and grant privileges to the default database owner `root` as described in the following steps:
   a) Run the following command on the MariaDB terminal:
   ```
   mariadb>create database bwadmindb
   ```
   b) Run the following command to view the tables included in the newly created database:
   ```
   mariadb>use bwadmindb;
   ```
   c) Run the following command to grant all privileges to the root user for that database after replacing the value for the host IP address:
   ```
   mariadb>GRANT ALL PRIVILEGES ON *.* TO root@<host_IP> IDENTIFIED BY
   'Tibco123'WITH GRANT OPTION;
   ```
   To grant the minimum amount of permissions to the bwuser for that database, run the following command, where *<host_IP>* is replaced with the value for the host IP address:
   ```
   grant create,select,update,insert,delete ON *.* to bwuser@<host_IP> IDENTIFIED
   BY "bwuser";
   ```
2. Stop the bwagent if it is running.
3. Open the `bwagent_db.json` file located in *BW_HOME*`\config` (Windows) or `${`*BW_HOME*`}/config` (Unix).
4. Update the following properties for your environment:

Property Name	MariaDB Value
dbtype	mariadb
dbdriver	org.mariadb.jdbc.Driver

Property Name	MariaDB Value
dbconnectionurl	jdbc:mariadb://localhost:3306/bwadmindb
dbuser	bwuser
dbpassword	bwuser

5. Run the bwadmin **config** command with the **-cf** option to create an `.ini` file in the correct location.
   ```
   BW_HOME\bin>bwadmin config -cf ../config/bwagent_db.json agent
   ```
6. Restart the bwagent.

Obfuscating or Encrypting Password for Database, EMS, and FTL Users

By default, the Database, EMS, and FTL users do not have a password. You can set the password however, this password is not encrypted.
To obfuscate the password, perform the following steps using Admin CLI.

Procedure

1. Generate obfuscated password using bwadmin utility.
   ```
   bwadmin[admin]> obfuscate <user password>
   ```
   It shows the generated obfuscated password.
2. Use this obfuscated password in the `bwagent.ini` file.

Creating an Agent Network

This topic shows how to configure bwagents so they can be members of the same agent network.

When using multiple machines, the runtime status of the bwagents and AppNodes cannot be computed reliably if the machine clocks in the agent network are not in sync with each other. Make sure that the clocks for machines in the network are synchronized.

All agents in a network should be of same 4 part version.

For example, If there is one agent with version 6.4.2_HF009, all the other agents should be of the version 6.4.2_HF009 only.

Complete the following steps for each bwagent that is to join the agent network.

Procedure

1. Stop bwagent.
2. For each bwagent, open the JSON configuration file, located in *BW_HOME*`\config` (Windows) or `${`*BW_HOME*`}/config` (Unix). Use the configuration file specific to the technology type used by the bwagents in the network.
   a) Edit parameters in this file as follows:

Parameter	Property in bwagent.ini File	Setting
bwagentnetworkname	**bw.agent.network.name**	The name of the network. Must be the same setting for each bwagent in the network.
	bw.agent.technology.dbems.ems.serverUrl/ bw.agent.technology.dbftl.ftl.realmserver	Use Same transport layer URL
	bw.agent.technology.dbems.db.connectionURL and bw.agent.technology.dbftl.db.connectionURL	Same database

b) Set other parameters in the JSON file. For more information about parameters, see Configuring bwagent.

c) Save the file and use the bwadmin **config** command to push the changes from the JSON file to the `bwagent.ini` file.
Use the `bwagent_db.json` or `bwagent_ftl.json` file as follows:

```
BW_HOME\bin>bwadmin config -cf ../config/bwagent_db.json agent
```

```
BW_HOME\bin>bwadmin config -cf ../config/bwagent_ftl.json agent
```

3. Restart bwagent.

Result

Use the **show agents** command to show all discovered bwagents. Agents in a network can be managed by any other bwagent.

Accessing the bwagent REST API with the Swagger UI

Use the Swagger UI to access the bwagent REST API, where you can try out operations and see results using sample data.

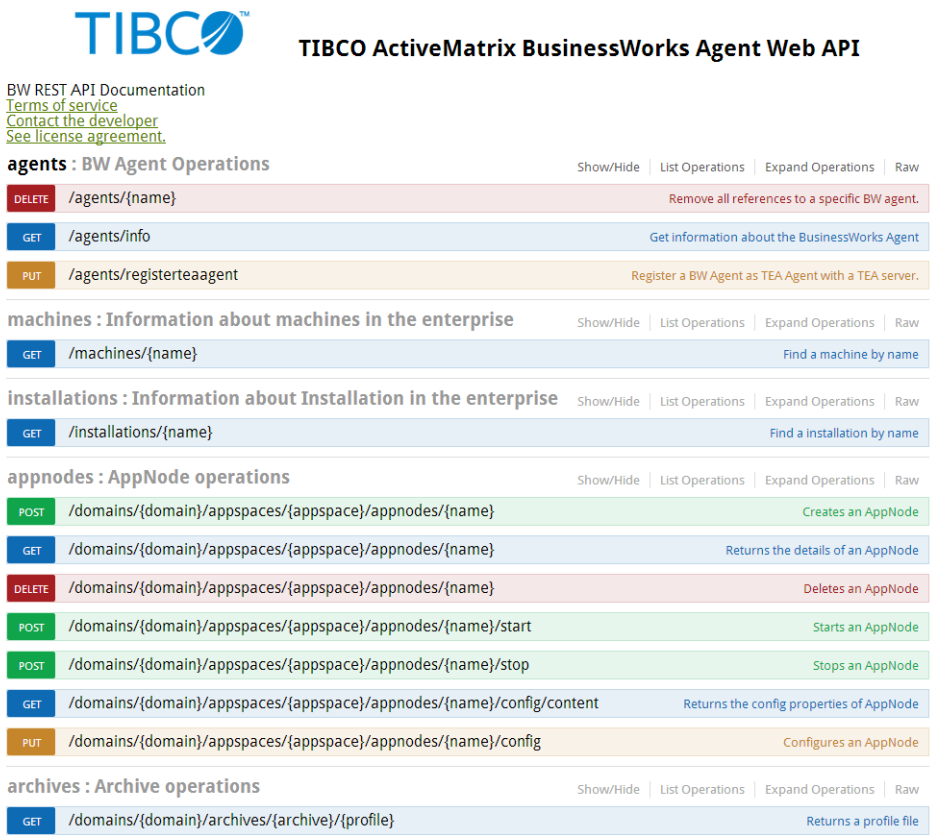
Procedure

1. Start the bwagent with the **apiserver** command:

```
BW_HOME\bin>bwagent apiserver
```

The API server is started at `http://<hostname>:5555`, where <*hostname*> is value you have set for the **bw.agent.http.host** property in the bwagent.ini file. For example, if you set **localhost** as the value for the **bw.agent.http.host** property, the API server is started at the URL `http://localhost:5555`.

2. Open a web browser and go to the URL of the API server.
The bwagent API documentation displays in the Swagger UI:



3. View sample data in the browser. To obtain the URL, go to the URL returned by an operation. For example, clicking **Try it out!** for the `GET/agents/info` operation returns the Request URL of `http://localhost:5555/api/agents/info`. Pasting this URL into the browser returns information similar to:

```
[{"name":"localhost","state":"Running","version":"6.6.0","configState":"InSync","
machineName":"bwin2k12r264b-76","description":"TIBCO ActiveMatrix BusinessWorks
version 6.6.0, build V37, 2019-10-13","adminMode":"enterprise","tibcoHome":"E:\
\BW6\\6.6.0\
\V37","pid":"5024","installationName":null,"configMap":null,"httpPort":null,"http
Host":null,"uptime":10986,"internalPort":null}]
```

To get actual data for the agent, go to the URL using the port 8079 instead of 5555. Change the `api` folder in the file path to `bw` and the version number into the path before the resource. (The bwagent must be running.) For example, the URL `http://localhost:8079/bw/v1/agents/info` returns the following information for an agent named MACHINE_1:

```
[{"name":"localhost","state":"Running","description":"TIBCO ActiveMatrix
BusinessWorks version 6.6.0, build V37, 2019-10-13","tibcoHome":"E:\\BW6\\6.6.0\
\V37","pid":"3356","configState":"InSync","machineName":"localhost","adminMode":"
enterprise","version":"6.6.0","installationName":"V37","configMap":
{"bw.agent.technology.requestTimeout":"60000","bw.agent.technology.dbftl.ftl.data
format":"bw-
format","bw.agent.technology.type":"dbftl","bw.governance.jms.ssl.trust.store.loc
ation":"","bw.agent.tea.agent.port":"9091","bw.agent.technology.dbftl.db.password
":"bwpassword","bw.agent.technology.dbftl.db.driver":"com.mysql.jdbc.Driver","bw.
monitor.ftluserpassword":"","bw.governance.jms.ssl.trust.store.type":"JKS","bw.mo
nitor.data.format":"bytestream","bw.agent.technology.dbftl.db.connectionURL":"jdb
c:mysql://localhost:3306/
V37","bw.monitor.ftlidentifier":"","bw.agent.tea.server.url":"http://%HOSTADDRESS
%:8777/tea","bw.monitor.ftlendpoint":"bwadmin-stats-
endpoint","bw.agent.technology.statsProvider.db.driver":"com.mysql.jdbc.Driver","
bw.agent.http.port":"8079","bw.monitor.ftlinbox":"bw-
inbox","bw.governance.jms.server.password":"","bw.agent.technology.statsProvider.
db.connectionURL":"jdbc:mysql://localhost:3306/
```

```
V37","bw.monitor.ftldataformat":"bw-
format","bw.agent.technology.statsProvider":"db","bw.agent.http.host":"0.0.0.0","
bw.monitor.ftlapplicationname":"bwadminstats","bw.agent.technology.dbftl.ftl.inbo
x":"bw-inbox","bw.agent.http.access.log.config":"bwagent-
access.xml","bw.admin.mode":"enterprise","bw.agent.technology.dbftl.ftl.endpoint"
:"bwadmin-endpoint","bw.governance.jms.server.url":"tcp://
localhost:7222","bw.agent.technology.dbftl.ftl.secondary":"","bw.agent.appnode.pa
ssword":"OBF:1sho1wgi1u9d1x1d1xfj1x191ua51wfg1shu","bw.agent.technology.dbftl.ftl
.username":"","bw.governance.jms.reconnect.attempt.delay":"500","bw.agent.memberN
ame":"localhost","bw.governance.jms.server.username":"admin","bw.monitor.ftlusern
ame":"","bw.agent.technology.dbftl.db.userName":"bwuser","bw.monitor.ftlsecondary
url":"","bw.agent.technology.dbftl.db.provider":"mysql","bw.agent.technology.dbft
l.ftl.realmserver":"http://
localhost:8070","bw.monitor.provider":"UDP","bw.monitor.ftlrealmserverurl":"http:
//
ip[:port]","bw.governance.jms.reconnect.attempt.timeout":"500","bw.agent.technolo
gy.statsProvider.db.userName":"bwuser","bw.governance.jms.queue.pd.receiver.name"
:"governance.de.bw.default","bw.agent.tea.agent.host":"0.0.0.0","bw.agent.technol
ogy.dbftl.ftl.password":"","bw.agent.technology.db.create.schema":"true","bw.gove
rnance.jms.reconnect.attempt.count":"120","bw.governance.jms.ssl.trust.store.pass
word":"","bw.agent.networkName":"BW6Network","bw.governance.enabled":"false","bw.
agent.technology.statsProvider.db.provider":"mysql","bw.agent.technology.statsPro
vider.db.password":"bwpassword","bw.agent.technology.dbftl.ftl.application":"bwad
min","bw.agent.technology.dbftl.ftl.identifier":""},"httpPort":8079,"httpHost":"0
.0.0.0","uptime":11850866,"internalPort":56565}]
```


To change the URL interface or port, edit the **`http.host`** or **`http.port`** settings in the `bwagent.ini` file.

Using the bwagent REST API to Return Selected Fields

You can retrieve information of only selected fields by adding query parameters to the request URL.

The following sample queries show how to retrieve selected fields:

- **Example 1**: To check the status of an Application, the REST API GET URL would be -

  http://localhost:8079/bw/v1/domains/<DomainName>/appspaces/<AppspaceName>/applications/<ApplicationName>/<ApplicationVersion>?fields=state

  The URL returns the following response:

  {"state":"Running"}

  Use comma separated fields after the question mark (?) with fields=keyword in the request query. Spaces are not permitted.

- **Example 2**: To get AppSpace details

  Normal query - http://localhost:8079/bw/v1/domains/<DomainName>/appspaces/<AppSpaceName>. It returns the total payload, that is, all fields.

- **Select Query**: http://localhost:8079/bw/v1/domains/<DomainName>/appspaces/<AppSpaceName>?select=name,status. It retrieves only 2 fields.
- **Sub-field**: http://localhost:8079/bw/v1/domains/<DomainName>/appspaces/<AppSpaceName>?select=appSpaceConfigRefs.href. It fetches href of appSpaceConfigRefs.

  The query returns an empty response when the selected field is a collection.

Securing the bwagent REST API

The bwagent REST API can be secured via authentication and roles. The bwagent REST API server can be secured with SSL access.

Enabling Authentication for the bwagent REST API Using the JAAS Property File

Authentication for the bwagent REST API is implemented using the JAAS property file login module. Different login module implementations can be used. For more information, see the Jetty documentation at eclipse.org.

Prerequisites

Stop the bwagent if it is running.

Procedure

1. To enable authentication for the bwagent REST API,
   a) Navigate to *BW_HOME*\bin.
   b) Open the `bwagent.tra` file in an editor and uncomment the following property:
      ```
      java.property.java.security.auth.login.config=%BW_HOME%/config/jaas.login.conf
      ```
      This property points to the location of the JAAS configuration file, which enables the authentication mechanism and, in turn, points to the property file. The property file identifies users and stores encrypted/obfuscated passwords.
   c) Save the file.
2. To change the name or location of the default property file, edit the *BW_HOME*`\config\jaas.login.conf` file. The default property file for the login module implementation is *BW_HOME*`\config\realm.properties`.
   ```
   bwloginmodule {
     org.eclipse.jetty.jaas.spi.PropertyFileLoginModule required
       file="../config/realm.properties";
   };
   ```
3. To change default users or passwords, edit the *BW_HOME*`\config\realm.properties` file. Two users are provided by default: `admin` and `bwappnode`. The `admin` user is the default user for access to the bwagent REST API. The `bwappnode` user is the default user for the bwappnode process.
   a) Open *BW_HOME*`\config\realm.properties` in a text editor. The default contents of the file are outlined below.

      The format of this file is: `<username>: <password>[,<rolename> ...]`

      ```
      admin: CRYPT:adpexzg3FUZAk,admin
      bwappnode: OBF:1sho1wgi1u9d1x1d1xfj1x191ua51wfg1shu,admin
      ```

      Roles specified in the `realm.properties` file are different than the TIBCO ActiveMatrix BusinessWorks™ user roles in TIBCO Enterprise Administrator™. For more information about TIBCO Enterprise Administrator™ user roles for ActiveMatrix BusinessWorks™, see Roles and Permissions.
   b) To create an encrypted or obfuscated password, navigate to the *BW_HOME*`\system\lib\tea` folder and use Java with the **`-cp`** option to call the Jetty Password utility, passing the password. In this example, the password `admin` is used. The utility returns the password in a variety of formats.
      ```
      BW_HOME\system\lib\tea> java -cp jetty-util-<version>.jar
      org.eclipse.jetty.util.security.Password admin
      password
      2019-10-10 14:19:57.505:INFO::main: Logging initialized @281ms to
      org.eclipse.jetty.util.log.StdErrLog
      password
      ```

```
OBF:1v2j1uum1xtv1zej1zer1xtn1uvk1v1v
MD5:5f4dcc3b5aa765d61d8327deb882cf99
CRYPT:advwtv/9yU5yQ
```

c) Copy the OBF, MD5 or CRYPT password from the output and paste it into the `realm.properties` file for the user name.

d) Save the file.

4. To configure the bwappnode user using the bwagent file,
   a) Open the *BW_HOME*`\config\bwagent.ini` file in a text editor and navigate to the section called "Configuration for AppNode to agent communication."
   This provides an alternate configuration mechanism.
   b) Uncomment the **`bw.agent.appnode.user`** property and change the default bwappnode user name. If you specify a different user for the **`bw.agent.appnode.user`** property, you can add the entry to the `realm.properties` file (with an obfuscated password and role). If you do not add this entry to the properties file, set the password for the user, outlined in the next step.
   c) Uncomment the property **`bw.agent.appnode.password`** and change the default password. The password must be able to be de-obfuscated. If encrypted, the bwappnode process will not be able to decrypt the password. As a result, the AppNode will not be able to communicate with the bwagent REST API and will be unable to report its status. The status for the AppNode and any applications on that AppNode will display as Stopped even though they may be running. To creating an encrypted or obfuscated password, follow the instructions.

If you specify this password, it will be used and the password set in the `realm.properties` file for the user (if that user exists in the file) will be ignored.

   d) Save the file.
5. The default authentication mechanism is set to Basic. To change the authentication to Digest,
   a) Open the *BW_HOME*`\config\bwagent.ini` file in a text editor.
   b) Navigate to the "Web server HTTP and HTTPS configuration" section of the file and uncomment the **`bw.agent.bw.auth`** property. Change the value from *BASIC* to *DIGEST* (all caps).
   c) Add a new property called **`bw.agent.bw.api.auth`**. Set the value of this property to *DIGEST* (all caps).
   d) Save the file.
6. Set the **`bw.agent.http.authorization`** property to *true* in the `bwagent.ini` file for authentication and authorization.
7. Restart the bwagent.

Enabling LDAP Authentication for the bwagent REST API

Follow these steps to enable LDAP authentication for the bwagent REST API.

Prerequisites

- Stop the bwagent
- Ensure users are assigned the admin, operator, and user roles in the LDAP sever. For more information, see the section Authorizing Access to the REST API by Role.

Procedure

1. Navigate to *BW_HOME*\bin.
2. Open the `bwagent.tra` file in an editor and uncomment the following property:
   ```
   java.property.java.security.auth.login.config=%BW_HOME%/config/jaas.login.conf
   ```
3. Replace the following lines to the *%BW_HOME%*`/config/jaas.login.conf` file.
   ```
   bwloginmodule {
     org.eclipse.jetty.jaas.spi.LdapLoginModule required
   ```

```
  debug="true"
  useLdaps="false"
  contextFactory="com.sun.jndi.ldap.LdapCtxFactory"
  hostname="10.97.106.72"
  port="10389"
  authenticationMethod="simple"
  forceBindingLogin="true"
  userBaseDn="o=tibco"
  userObjectClass="inetOrgPerson"
  userRdnAttribute="uid"
  userIdAttribute="uid"
  userPasswordAttribute="userPassword"
  roleBaseDn="ou=roles,o=tibco"
  roleNameAttribute="cn"
  roleMemberAttribute="uniqueMember"
  customRoleForLDAP=test
  roleObjectClass="groupOfUniqueNames";
};
};
```

4. In the `bwagent.ini` file, set the property **`bw.agent.http.authorization`** to *true* for LDAP authentication and authorization, and set the property to *false* to enable authorization for Custom Group.
5. Restart the bwagent.
6. Open a web browser and go to the URL of the API server. For example, enter the following URL: http://localhost:8079/bw/v1/agents/info
7. Enter the correct LDAP user name and password credentials in the dialogue box that displays.

Enabling LDAP Over SSL Authentication for the bwagent REST API

Follow these steps to enable LDAP over SSL authentication for the bwagent REST API.

Prerequisites

- Stop the bwagent
- Ensure users are assigned the admin, operator, and user roles in the LDAP sever. For more information, see the section Authorizing Access to the REST API by Role.

Procedure

1. Navigate to *BW_HOME*\bin.
2. Open the `bwagent.tra` file in an editor and uncomment the following property:
   ```
   java.property.java.security.auth.login.config=%BW_HOME%/config/jaas.login.conf
   ```
3. Replace the following lines to the *%BW_HOME%*`/config/jaas.login.conf` file.
   ```
   bwloginmodule {
      org.eclipse.jetty.jaas.spi.LdapLoginModule required
      debug="true"
      useLdaps="true"
      contextFactory="com.sun.jndi.ldap.LdapCtxFactory"
      hostname="localhost"
      port="10636"
      bindDn="cn=John Keats,ou=users,o=mojo"
      bindPassword="pass"
      authenticationMethod="simple"
      forceBindingLogin="true"
      userBaseDn="o=mojo"
      userObjectClass="inetOrgPerson"
      userRdnAttribute="uid"
      userIdAttribute="uid"
      userPasswordAttribute="userPassword"
      roleBaseDn="ou=bwgroups,o=mojo"
      roleNameAttribute="cn"
   ```

```
   roleMemberAttribute="uniqueMember"
   roleObjectClass="groupOfUniqueNames";
};
```

Ensure that the value of the attribute `useLdaps` is set to "**true**"

4. For more information about how to import the LDAP SSL certificate into the cacerts keystore file that is shipped with tibcojre, see the section Importing LDAP SSL Certificate in the cacerts keystore file.
5. Restart the bwagent.
6. Open a web browser and go to the URL of the API server. For example, enter the following URL: http://localhost:8079/bw/v1/agents/info
7. Enter the LDAP user name and password credentials.

Authorizing Access to the REST API by Role

The REST API supports the following roles: admin, operator, and user. Roles are assigned in the realm.properties file. Multiple roles can be assigned to a single user.

Prerequisites

Stop the bwagent if it is running.

Role	Description	Access to Operations
admin	Full rights: Create, Read, Update, Delete, Lifecycle (default)	All
operator	Read and Lifecycle operations	Start/Stop, GET
user	Read only	GET

Procedure

1. Navigate to *BW_HOME*`\config` and open the realm.properties file. For more information, see Enabling Authentication for the bwagent REST API.
2. For each defined user, change the default role as needed.
3. Save the file and restart the bwagent.

Securing the REST API Server

The bwagent REST API server can be secured with an SSL connection.

The SSL connection is configured in the bwagent configuration file.

Prerequisites

Stop the bwagent if it is running.

Procedure

1. Open the *BW_HOME*`\config\bwagent.ini` file in a text editor and navigate to the section called "Web server HTTP and HTTPS configuration."

2. Comment out the **bw.agent.http.port** property.
3. Uncomment the **bw.agent.https.port** property.

If the **bw.agent.http.port** property is also set, requests to the HTTP port are automatically rerouted to the HTTPS port.

4. Uncomment the following properties and add values:

Property	Value	Description
bw.agent.https.truststorepath	String	Path to the trust store file.
bw.agent.https.truststorepassword	String	The password for the trust store.
bw.agent.https.keystorepath	String	Path to the keystore file.
bw.agent.https.keystorepassword	String	Password of the keystore.

If you are using a self-signing certificate, specify **CN** to be `localhost`.

5. Add additional SSL properties to the "Web server HTTP and HTTPS configuration" section of the `bwagent.ini` file as needed.

Property	Value	Description
bw.agent.https.allowrenegotiate	true|false	Set if SSL re-negotiation is allowed.
bw.agent.https.certalias	String	Alias of SSL certificate for the connector
bw.agent.https.keymanagerpassword	Obfuscated password	The password (if any) for the specific key in the keystore.
bw.agent.https.keystoretype	String	The type of the keystore (default "JKS").
bw.agent.https.needclientauth	true|false	Set to `true` if SSL needs client authentication.
bw.agent.https.protocol	String	The SSL protocol (default "TLS") passed as the **protocol** parameter to the `SSLContext.getInstance(String, String)` method.

Property	Value	Description
bw.agent.https.provider	String	The SSL provider name, which, if set, is passed as the provider parameter to the `SSLContext.getInstance(String, String)` method.
bw.agent.https.trustall	true|false	Set to `true` if all certificates should be trusted (e.g. if there is no KeyStore or TrustStore).
bw.agent.https.trustmanagerfactoryalgorithm	String	The algorithm name (default "SunX509") passed to `TrustManagerFactory`. Use the string "TrustAll" to install a trust manager that trusts all.
bw.agent.https.truststoreprovider	String	The provider of the trust store.
bw.agent.https.truststoretype	String	The type of the trust store (default "JKS").
bw.agent.https.ocspresponderurl	String	Location of the OCSP Responder.
bw.agent.https.enablecrldp	true|false	Set to `true` to enables CRL Distribution Points support.
bw.agent.https.enableocsp	true|false	Set to `true` to enables On-Line Certificate Status Protocol support.
bw.agent.https.crlpath	String	The path to the file that contains Certificate Revocation List.
bw.agent.https.validatecerts	true|false	Set to `true` if SSL certificates have to be validated.
bw.agent.https.validatepeercerts	true|false	Set to `true` if SSL certificates of the peer have to be validated.
bw.agent.https.wantclientauth	true|false	Set to `true` if SSL wants client authentication.

Property	Value	Description
bw.agent.https.maxcertpathlength	int	The maximum number of intermediate certificates in the certification path (-1 for unlimited).
bw.agent.https.excludeciphersuites	Comma separated list of Strings	The list of cipher suite names to exclude from.
bw.agent.https.includeciphersuites	Comma separated list of Strings	The list of cipher suite names to include.
bw.agent.https.securerandomalgorithm	String	The algorithm name. If set, is passed as the **algorithm** parameter to `SecureRandom.getInstance(String)` method to obtain the `SecureRandom` instance, which is then passed to the `SSLContext` constructor.
bw.agent.https.excludeprotocols	String	Add this property to disable SSL protocols. Disabled SSL protocols are represented in a comma delimited list.
bw.agent.https.includeprotocols	String	Add this property to enable SSL protocols. Enabled SSL protocols are represented in a comma delimited list.

6. Save the file and restart the bwagent.
7. Open a web browser and go to the URL of the API server. For example, enter the following URL: https://localhost:8886/bw/v1/agents/info

Importing LDAP SSL certificate in the cacerts keystore file

To connect the bwagent to the LDAP Over SSL server, ensure that the server certificate is imported into the cacerts keystore file.

Prerequisites

Ensure that the SSL Certificate has been exported from the LDAP server.

Procedure

- Navigate to *BW_Home*\ tibcojre64\1.8.0\lib\security

```
# To List Existing Certificates use command :

BW_Home\tibcojre64\1.8.0\lib\security>keytool -list -keystore cacerts
Enter keystore password: changeit

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 5 entries

verisignclass2g2ca [jdk], Aug 25, 2016, trustedCertEntry,
Certificate fingerprint (SHA1):
B3:EA:C4:47:76:C9:C8:1C:EA:F2:9D:95:B6:CC:A0:08:1B:67:EC:9D
digicertassuredidg3 [jdk], Aug 25, 2016, trustedCertEntry,
Certificate fingerprint (SHA1):
F5:17:A2:4F:9A:48:C6:C9:F8:A2:00:26:9F:DC:0F:48:2C:AB:30:89
verisignuniversalrootca [jdk], Aug 25, 2016, trustedCertEntry,
Certificate fingerprint (SHA1):
36:79:CA:35:66:87:72:30:4D:30:A5:FB:87:3B:0F:A7:7B:B7:0D:54
digicerttrustedrootg4 [jdk], Aug 25, 2016, trustedCertEntry,
Certificate fingerprint (SHA1):
DD:FB:16:CD:49:31:C9:73:A2:03:7D:3F:C8:3A:4D:7D:77:5D:05:E4
verisignclass1g3ca [jdk], Aug 25, 2016, trustedCertEntry,
Certificate fingerprint (SHA1):
20:42:85:DC:F7:EB:76:41:95:57:8E:13:6B:D4:B7:D1:E9:8E:46:A5
identrustpublicca [jdk], Aug 25, 2016, trustedCertEntry,
E:\BW\6.4.0\V11\tibcojre64\1.8.0\lib\security>

#### Import LDAP SSL Certificates to cacerts:
BW_Home\tibcojre64\1.8.0\lib\security>keytool -import -keystore cacerts -file
<certName>.der
Enter keystore password: changeit
Owner: CN=bwin2k8r264b_52, OU=Directory, O=ASF, C=US
Issuer: CN=ApacheDS, OU=Directory, O=ASF, C=US
Serial number: 15ac649f77e
Valid from: Sun Mar 12 23:10:20 PDT 2017 until: Mon Mar 12 23:10:20 PDT 2018
Certificate fingerprints:
         MD5:  A4:25:84:6C:63:51:C5:A2:EB:D5:69:2A:74:EE:D3:31
         SHA1: F0:9D:0A:26:E3:86:61:CB:62:3F:1F:40:5A:31:F3:BC:0C:C9:C0:B0
         SHA256:
82:43:35:95:55:A6:CC:36:BB:C8:9A:6E:9D:55:FF:69:C1:7C:30:B3:EC:79:DA:3E:98:A9:F2:
B6:5C:48:B8:28
         Signature algorithm name: SHA1withRSA
         Version: 1
Trust this certificate? [no]:  yes
Certificate was added to keystore

#### Check if the certificate was imported. The number of keystore entries
should increase by 1.

BW_Home\tibcojre64\1.8.0\lib\security>keytool -list -keystore cacerts
Enter keystore password:

Keystore type: JKS
Keystore provider: SUN
Your keystore contains 6 entries


verisignclass2g2ca [jdk], Aug 25, 2016, trustedCertEntry,
Certificate fingerprint (SHA1):
B3:EA:C4:47:76:C9:C8:1C:EA:F2:9D:95:B6:CC:A0:08:1B:67:EC:9D
digicertassuredidg3 [jdk], Aug 25, 2016, trustedCertEntry,
Certificate fingerprint (SHA1):
F5:17:A2:4F:9A:48:C6:C9:F8:A2:00:26:9F:DC:0F:48:2C:AB:30:89
verisignuniversalrootca [jdk], Aug 25, 2016, trustedCertEntry,
Certificate fingerprint (SHA1):
```

```
36:79:CA:35:66:87:72:30:4D:30:A5:FB:87:3B:0F:A7:7B:B7:0D:54
digicerttrustedrootg4 [jdk], Aug 25, 2016, trustedCertEntry,
Certificate fingerprint (SHA1):
DD:FB:16:CD:49:31:C9:73:A2:03:7D:3F:C8:3A:4D:7D:77:5D:05:E4
verisignclass1g3ca [jdk], Aug 25, 2016, trustedCertEntry,
Certificate fingerprint (SHA1):
20:42:85:DC:F7:EB:76:41:95:57:8E:13:6B:D4:B7:D1:E9:8E:46:A5
identrustpublicca [jdk], Aug 25, 2016, trustedCertEntry,
Certificate fingerprint (SHA1):
BA:29:41:60:77:98:3F:F4:F3:EF:F2:31:05:3B:2E:EA:6D:4D:45:FD
utnuserfirstobjectca [jdk], Aug 25, 2016, trustedCertEntry,
BW_Home\tibcojre64\1.8.0\lib\security>
```

Viewing bwgent Information

Use the bwadmin **show** command to view information about the installation, including bwagent name, process ID, number of active threads, memory and CPU usage, and up time.

bwadmin Command Line

Make sure the bwagent is running, then issue the following command, providing the agent name in the command line.

```
BW_HOME\bin>bwadmin show agentprocessinfo M1
```

The following details are returned:

```
Agent Name:                     M1
System Process ID:              20976
Number of Active Threads:       163
Total Memory(in bytes):         1174147072
Free Memory(in bytes):          834506752
Used Memory(in bytes):          339640320
Percent Memory Used:            28.0
Percent CPU Used:               0.0
Up Time:                        0d 00:16:40
```

Restoring the File System of a bwagent

Restoring a bwagent restores the file system of the bwagent to the state of the datastore.
The file system for runtime entities in a bwagent can be restored locally or across a network, if the bwagent is part of an agent network.

Prerequisites

- The name of the bwagent must be known in order to restore.
- The bwagent must be running.

Procedure

1. To restore the file system for runtime entities in a bwagent, open a terminal and navigate to *BW_HOME*\bin.
2. Enter the **restore** command at the command line, using the **agent** argument with the name of the bwagent to restore. The bwagent can be either the local bwagent or a bwagent in the agent network. The following example restores the bwagent named Machine1.
   ```
   BW_HOME\bin>bwadmin restore agent Machine1
   ```
3. To verify the restore, check the file system. Open the *BW_HOME*\domains folder. Look for named domain folders to verify if domains for the bwagent have been restored.

Configuring the Location of the Domains Folder

Runtime entities are created in the local file system in the *BW_HOME*/domains folder. Below are several different ways to change the folder location of a specific domain.

Admin CLI

Point to a new domain home for a specific domain by executing the **-home** command. For example, run the following command to create a domain home named testDomain:

```
<BW_HOME>\ bin>bwadmin create -home /Users/testuser/domains domain testDomain
```

REST API

Use the home parameter in the REST request. For example, http://localhost:8079/bw/v1/domains/testDomain?home=/Users/testuser/domains

Domain Home Properties File

Update the domain home properties file to change the folder location of a specific domain.

Procedure

1. Stop the bwagent if it is running.
2. Open the *BW_HOME*/domains/DomainHomes.properties file in a text editor.
3. Add a the **<*domainName*>.domainHome** property, where <*domainName*> is the intended domain, to this file to point to the new domain home for a specific domain. For example, by adding the **testDomain.domainHome** property to the DomainHomes file, you are specifying a custom domain folder for the domain testDomain.
4. Save the file and restart the bwagent.
5. From the Admin UI, create the domain using the domain name you specified in the **domainHome** property. For example, if you have added **testDomain.domainHome** to the DomainHomes file, go to the Admin UI and create a domain called testDomain.

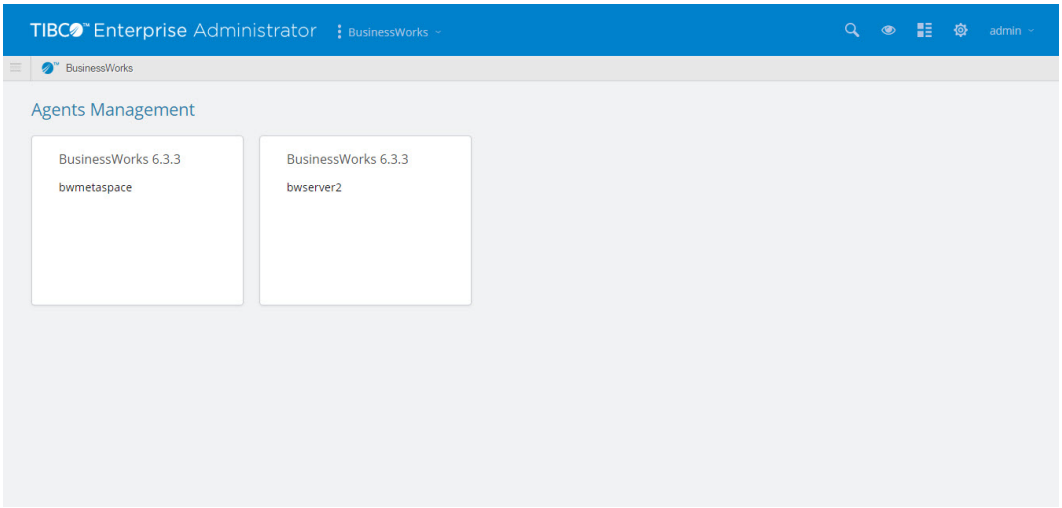
Using bwagent with TEA

The Admin UI is a web UI that runs in TIBCO® Enterprise Administrator (TEA). To enable the Admin UI, the bwagent must be registered with a running TEA server. Use the Admin UI to create, view, and monitor runtime entities.

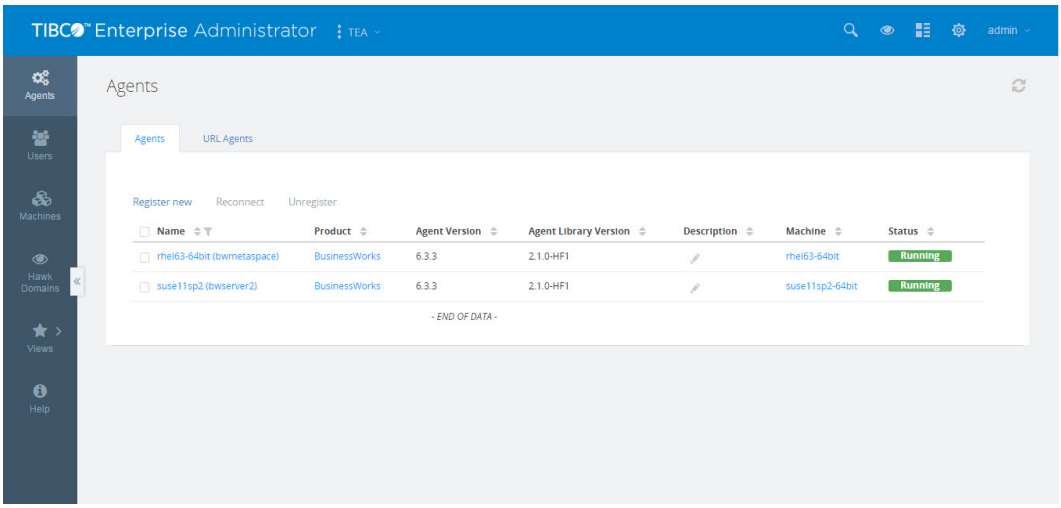
The bwagent interacts with the TIBCO Enterprise Administrator server through a TEA agent. Multiple bwagents can be registered with TEA, but only one TEA agent can be registered with a bwagent at a time.

If you register multiple bwagents with a TEA agent, ensure the bwagents are using the same version of TIBCO ActiveMatrix BusinessWorks 6.x.

Admin UI



Once the TEA agent is registered with a bwagent, the bwagent is displayed in the Admin UI, and the Admin UI can be used to manage and monitor runtime entities. .



The Admin UI allows you to perform almost all bwadmin administrative tasks. For a walk-through of the steps to start working with the Admin UI, see Running Applications in Enterprise Mode using the Admin UI

For an agent network to be managed from the Admin UI, one bwagent in the agent network must be registered with the TIBCO Enterprise Administrator server. If the registered bwagent terminates, the

connection between the server and the agent network is automatically recovered. Another bwagent in the agent network will autoregister with the server.

Registering bwagent with TIBCO Enterprise Administrator

A bwagent TEA agent must be registered with the TIBCO Enterprise Administrator server before it is available in the Admin UI.

bwadmin Command Line

Procedure

1. Start the TIBCO Enterprise Administrator server.
2. Start bwagent.
3. Register the bwagent TEA agent with the TIBCO Enterprise Administrator server. Provide the URL to the TEA server. Only one TEA agent can be registered at a time.

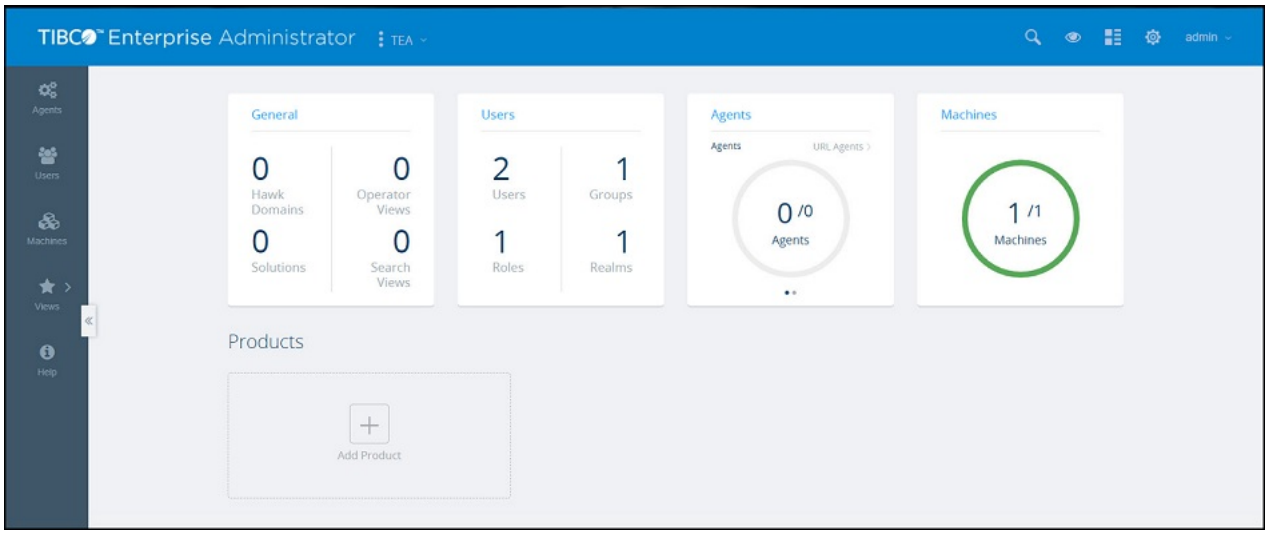
   ```
   BW_HOME\bin>bwadmin registerteaagent http://<TEA_HOST>:8777/tea/
   ```

   For more information about creating and managing runtime entities, see Administration Tasks and Reference.

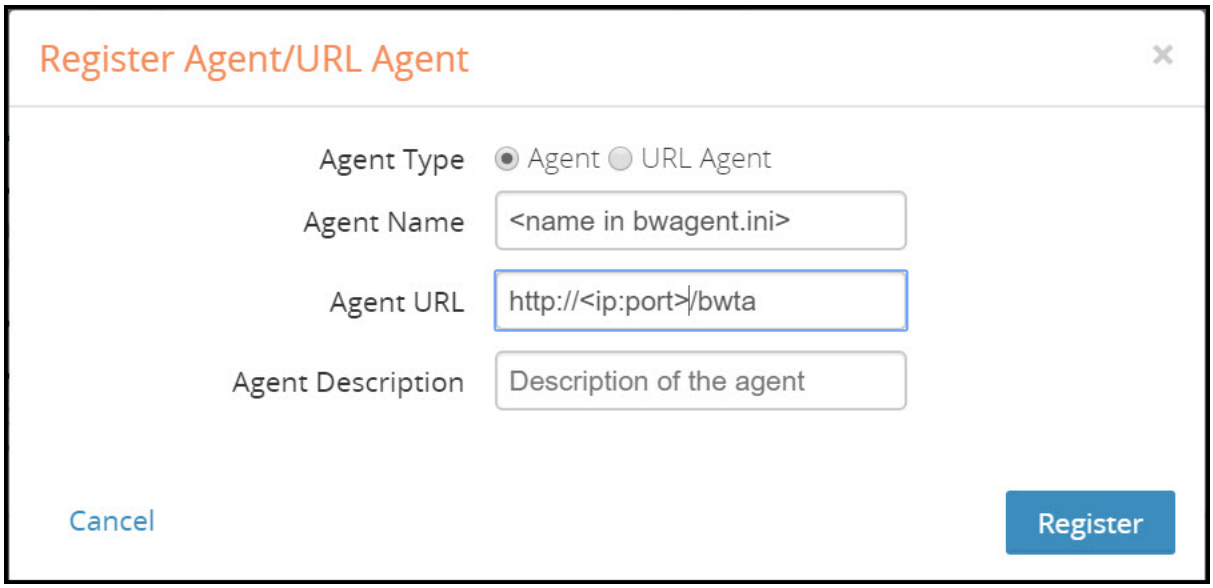
Admin UI

Procedure

1. Open Admin UI (`http://<TEA_HOST>:8777/tea`) to access the bwagent. Log in credentials are required; the default user name is `admin` and the default password is `admin`.
   Admin UI Home Page is displayed.



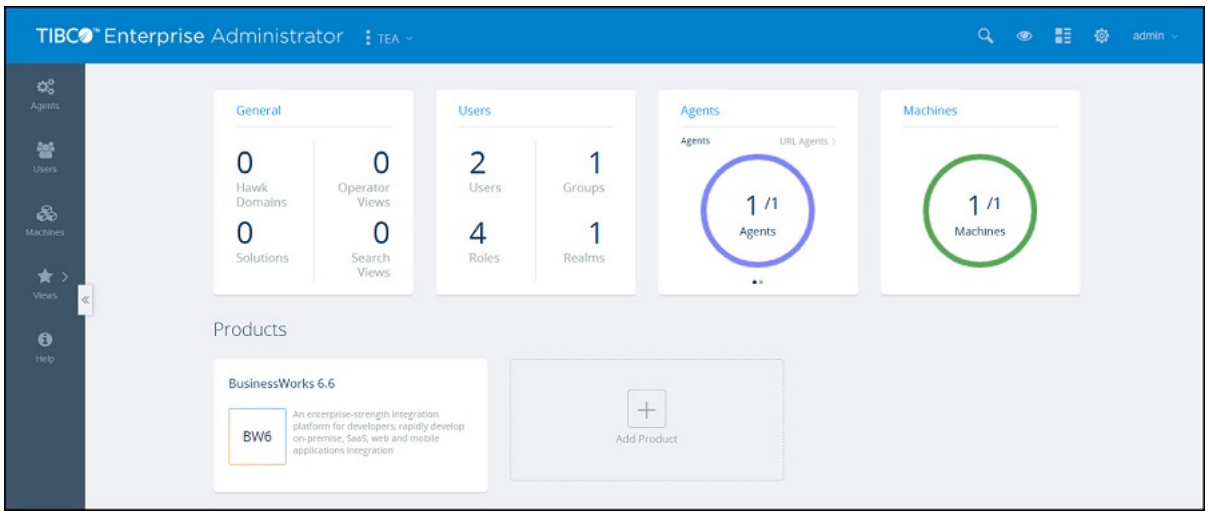
2. Click the **Add Product** icon on the Admin UI home page.
   The Register Agent/URL Agent dialog box is displayed.



3. Add the following information:

Field	Value
Agent Name	Name of the BWAgent configured in the `bwagent.ini` file.
Agent URL	URL should be in the `http://<ip:port>/bwta` format. The default port is 9091.
Agent Description	Optional. Description of the BWAgent.

4. Click **Register**.



Autoregistering bwagent with TIBCO Enterprise Administrator

A bwagent can be part of an agent network, which allows commands to be distributed to multiple bwagents simultaneously.

For an agent network to be managed from the Admin UI, one bwagent in the agent network must be registered with the TIBCO Enterprise Administrator (TEA) server. If the registered bwagent terminates,

the connection between the server and the agent network is automatically recovered. Another bwagent in the agent network will auto-register with the server.

The bwadmin **disableautoregistration** and **enableautoregistration** commands toggle the mechanism used to autoregister a bwagent. The commands can be run against any bwagent in an agent network and act on all bwagents in the agent network. If you disable autoregistration for an agent network, the members of the network are unable to communicate with the TIBCO Enterprise Administrator server. You have to manually register a bwagent in the agent network to communicate it with the server.

If the same TEA server is used for different versions of BWAgents and only one version of BWAgent is up, follow these steps to make sure TEA is working with appropriate UIs:

Procedure

1. Stop the TEA server.
2. Set the property **tea.dev.developer-mode** to `true` in the `<TEA-HOME>/tibco/cfgmgmt/tea/conf/tea.conf` file.
3. Restart the TEA server.
4. As an admin user, on the Agents page, click the **Reload** button.
5. Verify if new UI changes are picked up.
6. Stop the TEA server.
7. Reset **tea.dev.developer-mode** to `false`.
8. Restart the TEA server.
9. Clear the cache and reload Admin UI.

Unregistering bwagent with TIBCO Enterprise Administrator

You can unregister bwagent from the Admin UI, or the command line.

Unregistering bwagent Using the Admin UI

To unregister bwagent from the Admin UI, open the TEA URL and perform these steps:

1. Click **Agents** on the side bar to open the Agent page.
2. Select bwagent to unregister.
3. Click **Unregister**.
4. In the dialog window that displays, confirm that you want to unregister bwagent.

bwagent is no longer registered with the TEA server.

Unregistering bwagent Using the Command Line

Use bwadmin to execute the **unregisterteaagent** command, and enter the URL of your TEA server :

```
BW_HOME\bin>bwadmin unregisterteaagent <TEA URL>
```

Enabling and Disabling bwagent's TIBCO Enterprise Administrator Agent Port

You can disable bwagent's TEA agent port to disable registering bwagent with TIBCO Enterprise Administrator.

Execute **bwagent.exe startagent -nt** command from the Admin Console. While executing the command, you are not able to register BWAgent with Admin UI.

To enable the registering again, restart BWAgent with the command **bwagent.exe** without startagent -nt option.

Compatibility Chart for TIBCO ActiveMatrix BusinessWorks™ and TIBCO® Enterprise Administrator

The TIBCO Enterprise Administrator (TEA) server is an application administration UI that supports multiple TIBCO products, including the ActiveMatrix BusinessWorks™. Using the Admin UI you can create, view, and monitor runtime entities. Each product registers its own agent with the server and the server communicates with the products through these agents. The compatibility rules and chart help determine the minimum version of the TEA server required by a given version of ActiveMatrix BusinessWorks.

These are the compatibility rules for the TEA server and TEA agent libraries:

- The TEA server is backward compatible with earlier versions of TEA agent libraries, unless there is a known issue.
- The TEA server does not guarantee forward compatibility with newer versions of TEA agent libraries.

The following table lists the version of the TEA agent library bundled with a given version of ActiveMatrix BusinessWorks.

ActiveMatrix BusinessWorks Version	Version of TEA Agent Library Bundled
6.4.0	2.1.0 HF-001
6.4.1	2.1.0 HF-001
6.4.2	2.1.0 HF-002
6.5.0	2.1.0 HF-002
6.5.1	2.3.0 HF-005
6.6.0	2.3.0 HF-007
6.6.1	2.3.0 HF-007

Compatibility Chart

Based on compatibility rules and the version of the TEA agent library bundled in a given version of ActiveMatrix BusinessWorks, the following compatibility chart can be inferred.

ActiveMatrix BusinessWorks Version with TEA Agent Library *Version*	TEA Server Version			
	2.3.0	2.3.0 HF-002	2.3.0 HF-005	2.3.0 HF-007
ActiveMatrix BusinessWorks 6.5.1 with TEA Agent Library 2.3.0 HF-005	✗	✗	✓	✓
ActiveMatrix BusinessWorks 6.6.0 with TEA Agent Library 2.3.0 HF-007	✗	✗	✗	✓

ActiveMatrix BusinessWorks Version with TEA Agent Library *Version*	TEA Server Version			
	2.3.0	2.3.0 HF-002	2.3.0 HF-005	2.3.0 HF-007
ActiveMatrix BusinessWorks 6.6.1 with TEA Agent Library 2.3.0 HF-007	✗	✗	✗	✓

TEA Shell

A command line utility called the TEA shell is provided with TIBCO Enterprise Administrator server. It is a remote shell based on the SSH protocol that provides the command line interface for the full range of TEA operations. The scripting language is similar to that of bash from Unix.

The TEA shell has the following key features:

- Piping of commands
- Completion of commands
- Help on commands

The BusinessWorks entity structure in the TEA shell is:

```
/BusinessWorks
bwagents
domains
   apparchives
   appspaces
     applications
     appnodes
installations
machines
   bwagents
   installations
```

Change to the BusinessWorks context by typing: `admin@M1:/> cd Businessworks`

Press the `tab` key for a list of available commands for the context path.

For information about TEA shell commands, see Using TEA Shell Commands.

Using TEA Shell Commands

TEA shell commands can be used to create, monitor, and manage runtime entities.

TEA shell commands are aligned with bwadmin commands.

The steps in this section show you some simple TEA shell commands for creating a domain, AppSpace, and AppNode and starting the AppSpace. For a complete list of all supported commands, see TEA Shell Commands.

At any time in the TEA shell, press the `tab` key for a list of supported commands available for the context. To get help on a command, type the command with the `--help` option, for example: **`create --help`**

Procedure

1. Connect to the TEA shell through a terminal program, for example Putty. Connect using the following command:

   ```
   ssh -p 2222 admin@localhost
   ```

   The user name and password are both: `admin`

   On successful connection, the TEA Shell banner is displayed, illustrated below:

2. Change to the BusinessWorks context by typing: `admin@M1:/> cd Businessworks`
3. Create a domain:

```
admin@M1:/BusinessWorks> create domain TEA-D1
Executed the command 'create' successfully.
```

4. Create an AppSpace in the domain:

```
admin@M1:/BusinessWorks> create -domain TEA-D1 appspace TEA-AS1
Executed the command 'create' successfully.
```

5. Create an AppNode in the AppSpace:

```
admin@M1:/BusinessWorks> create -domain TEA-D1 -appspace TEA-AS1 appnode TEA-AN1
-httpPort 8077
Executed the command 'create' successfully.
```

6. Start the AppSpace. This starts the AppNode in the AppSpace.

```
admin@M1:/BusinessWorks> start -domain TEA-D1 appspace TEA-AS1
Executed the command 'start' successfully.
```

TEA Shell Commands

This topic lists all TEA shell commands and provides examples.

General Commands

Command	Description	Example
cd	Changes context to entity.	`cd /BusinessWorks/domains/Domain1/appspaces/AppSpace01/appnodes/AppNode01`
ls	Lists the name of each instance in the specified entity.	`cd /BusinessWorks/domains/Domain1/appspaces/AppSpace01/appnodes` `ls`

Domain Commands

Dommand	Example
Create domain	`cd /BusinessWorks/domains` `create -domain Domain1 -descr "Sanity Test Domain"`
Delete domain	`cd /BusinessWorks/domains` `delete -domain Domain1`

AppSpace Commands

Command	Example
Create an AppSpace; the **-minNodes** parameter is optional; defaults to 1	`cd /BusinessWorks/domains/Domain1/appspaces create -appspace AppSpace01 -descr"AppSpace 01"-minNodes"2"`
Delete an AppSpace	`cd /BusinessWorks/domains/Domain1/appspaces delete -appspace AppSpace01`
Start an AppSpace	`cd /BusinessWorks/domains/Domain1/appspaces start -appspace AppSpace01`
Stop an AppSpace	`cd /BusinessWorks/domains/Domain1/appspaces stop -appspace AppSpace01`

AppNode Commands

Description	Example
Create an AppNode; **-osgiPort** parameter is optional.	`cd /BusinessWorks/domains/Domain1/appspaces/AppSpace01/ appnodes create -appnode Sanity-AppNode01 -httpPort 7011 -agent localhost -osgiPort 8011`
Delete an AppNode	`cd /BusinessWorks/domains/Domain1/appspaces/AppSpace01/ appnodes delete -appnode Sanity-AppNode01`
Start an AppNode	`cd /BusinessWorks/domains/Domain1/appspaces/AppSpace01/ appnodes start -appnode Sanity-AppNode01`
Stop an AppNode	`cd /BusinessWorks/domains/Domain1/appspaces/AppSpace01/ appnodes stop -appnode Sanity-AppNode01`

Application Commands

Description	Example
Start an application	`cd /BusinessWorks/domains/Domain1/appspaces/AppSpace01/ applications start -app acme.acct.ap.application -version 1.0`
Start an application instance on the AppNode	`cd /BusinessWorks/domains/Domain1/appspaces/AppSpace01/ applications startinst -app acme.acct.ap.application -version1.0-appnode Sanity-AppNode01`
Stop an application	`cd /BusinessWorks/domains/Domain1/appspaces/AppSpace01/ applications stop -app acme.acct.ap.application -version 1.0`

Description	Example
Stop an application instance on an AppNode	`d /BusinessWorks/domains/Domain1/appspaces/AppSpace01/applications` `stopinst -app acme.acct.ap.application -version 1.0 -appnode Sanity-AppNode01`

Roles and Permissions

Privileges to perform actions and operations in Admin UI are based on the role and permissions granted to the user.

Roles

Administrators use roles to allot permissions in Admin UI. When a role is assigned to a user or a group, the user or group receives all the permissions granted to the role.

The following roles are defined in Admin UI: BW User, BW Operator and BW Administrator.

Permissions

Permissions are used to enforce access control. In Admin UI you can grant access permissions at two levels:

- Entity Based - Permission can be enforced on the complete entity such as a domain, AppSpace and so on. For example, for two domains d1 and d2, if the read permission is granted to the domain entity, you can view both the instances, d1 and d2.
- Instance Based - Permission can be enforced on a particular instance of an entity. For example, if the read permission is granted for the d1 instance of the domain entity, you have permission to view only d1.

Types of Permissions

You can assign the following types of permissions to users in Admin UI:

- Read: Read permission for the entities.
- Lifecycle: You can grant lifecycle permission to a user only if he has explicit read permission. Lifecycle permission is applicable only to AppSpaces, AppNodes, and applications, to control the lifecycle of entities such as AppSpaces, AppNodes, and applications (that is to start and stop the entities)
- Full_control: By default, this permission includes the read permission. Entities can perform the following commands with the full control access:
  - Domain: delete and backup
  - Domains: create
  - Archive: deploy and delete
  - Archives: upload
  - AppSpace: delete, create AppNode and update
  - AppSpaces: create
  - AppNode: delete and update
  - AppNodes: create
  - Application: update and undeploy
  - Agent: unregister

Role-Based Permissions

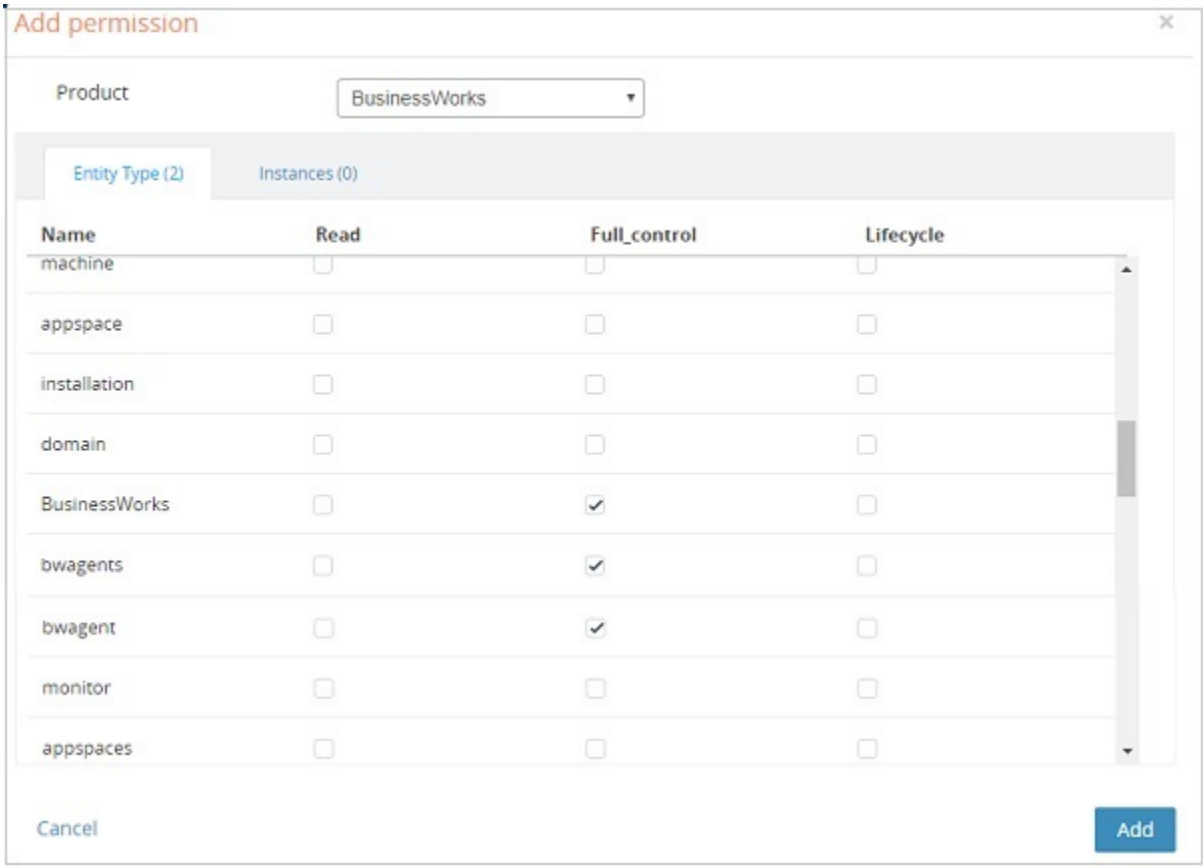
The following table shows permissions for the entities:

Roles	Permissions
BW User	Read
BW Operator	Lifecycle
BW Administrator	Full_control

Entity-Based Permissions

To add the ActiveMatrix BusinessWorks™ product under the list of **Products** on the Admin UI homepage, select the following entities on the **Add Permission** page.

- **BusinessWorks**
- **bwagent**
- **bwagents**



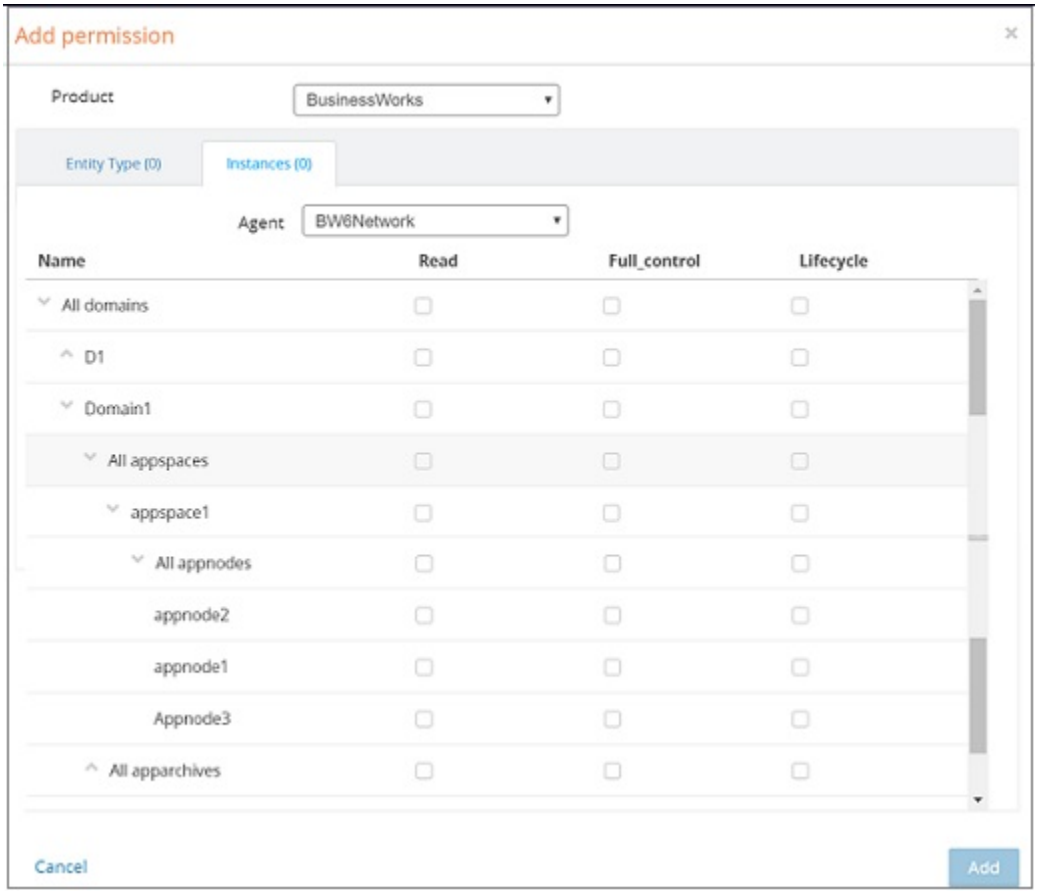
To manage permissions for an application, select the application and applications entities on the **Add Permission** page and assign the required permissions.

Grant access permissions to plural entities to access an instance of the entity. For example, to give read access to **d1** which is an instance of a domain, grant the read permission to domains (plural entity). This is applicable to all entity types.

Entity Hierarchy for Instance-Based Permission

The hierarchy of entities when granting permissions in Admin UI is illustrated in the following image. Domain is the top-level parent entity and includes AppSpaces and Archives. AppSpaces then further include AppNodes.

If an archive is uploaded to a folder, provide access for the folder first and then to the archive instances.



Granting Instance-Based Permissions

Scenario 1: Instance-based permission assigned to a child entity

When an instance-based permission is assigned to a child entity, the read permission is assigned to the parent entity if the parent does not have any permissions assigned. The administrator can, however, update the permission assigned to the parent. The updated permission is then enforced.

Scenario 2: Instance-based permission assigned to a parent entity

When an instance-based permission is assigned to a parent entity, the permission is not applied to the child entity. Permissions for the child entities can be assigned explicitly. For example, if the read permission is applied to AppSpace1, the child entities of AppSpace1 do not inherit the permission.

Example of how entity-based and instance-based permissions work

Objective

Appspace a1 contains two AppNodes, n1 and n2. AppSpace a1 is a child entity of Domain d1. Grant permissions so that you can only view AppNode n1 and start and stop AppNode n2.

1. From the entity permission page, **Add Permission**, provide read permissions to the entities BusinessWorks and bwagents. They are the top level entities and are mandatory to view the TIBCO ActiveMatrix BusinessWorks™ product.
2. Provide entity-based permission to domains, AppSpaces and AppNodes. It is mandatory to provide permission for plural entities such as domains and AppSpaces, to view the content on these pages.

3. Provide instance-based permission to AppNode n1 and lifecycle and read permission to AppNode n2.

The following section explains how the permissions granted in the example work:

- For Domain d1 to be visible, grant permissions to the entities BusinessWorks, bwagents, domains and for the instance of the domain d1.
- For AppSpace a1 to be visible, grant permission to the entities BusinessWorks, bwagents, domains, AppSpaces and for the instance a1. Explicit permissions are not required to be given to Domain d1. Parents entities are provided view permission automatically.
- For AppNode n1 to be visible grant permission to the entities BusinessWorks, bwagents and domains, AppSpaces and AppNodes and for the instance a1.

Additional Notes

- Actions taken on parent level transcends the actions taken on the child entity even if you do not have access to the child entity. For example, If you start and stop an AppSpace, all the AppNode in this AppSpace start and stop even if you do not have access over all of the AppNodes.
- Custom users cannot view any new entity they create as these users do not have instance based permission for that entity. For example, you have full control access to an Appspace and you navigate to the AppSpace page and use the **Create** button to create a new AppSpace a2. Users will not be able to view AppSpace a2 as they do not have access permissions for a2. The administrator will have to grant permissions to access this AppSpace to enable custom users to see it. This is applicable to all entity types.
- While taking a backup of a domain, all entities within this domain will be backed up irrespective of the permissions granted.
- The Appnodes, Appspaces and Application Archives count can be seen on the **Domain Management Page** irrespective of permissions granted to the user.

Administration Tasks and Reference

Administration tasks involve managing domains, AppSpaces, AppNodes, and applications.

The topics in this section show how to do administrative tasks from the bwadmin command line and the Admin UI:

- To complete tasks from the bwadmin command line, navigate to *BW_HOME*\bin. Type **bwadmin help** for a list of commands. For information, see bwadmin.
- To complete tasks from the Admin UI, register the TEA agent to the bwagent and open the TEA URL. For information see Using the Admin UI.

Managing Domains

A domain is a logical group that provides an isolated environment for applications and their resources to reside. It provides an administrative boundary for an integration project. Each domain may share machines with other domains, but does not communicate with other domains. Domains includes servers that may or may not be distributed over different machines and operating systems.

Creating a Domain

A domain comprises AppSpaces and AppNodes. Create a domain first, and then add the AppSpaces and AppNodes to the domain. The domain name is applied to all contained entities.

The following characters are allowed in the domain name:

- A-Z
- a-z
- 0-9
- - (hyphen)
- _ (underscore)

Illegal characters are stripped from the name.

The maximum length of a runtime entity name is 100 characters. If the maximum length is exceeded, the entity name is shortened to 100 characters.

BWAdmin Command Line

If the **-home** option is not specified in the **create** command (to set the path to the folder where all files related to the domain are managed), the domain is created in the default location under the *BW_HOME*\domains directory.

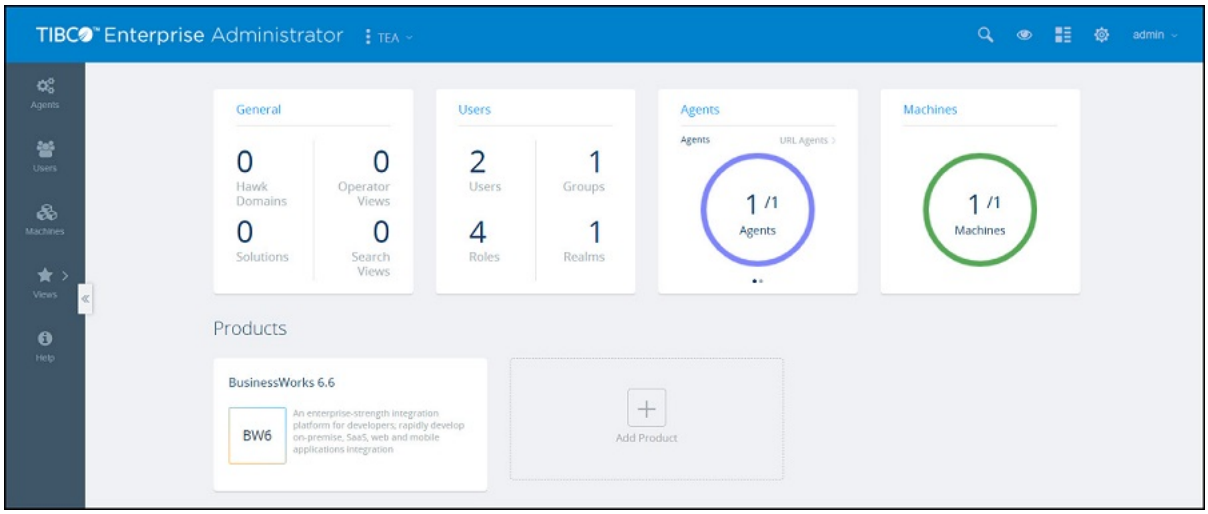
Run the following command to create a domain named MyDomain:

```
BW_HOME\bin>bwadmin create domain MyDomain
```

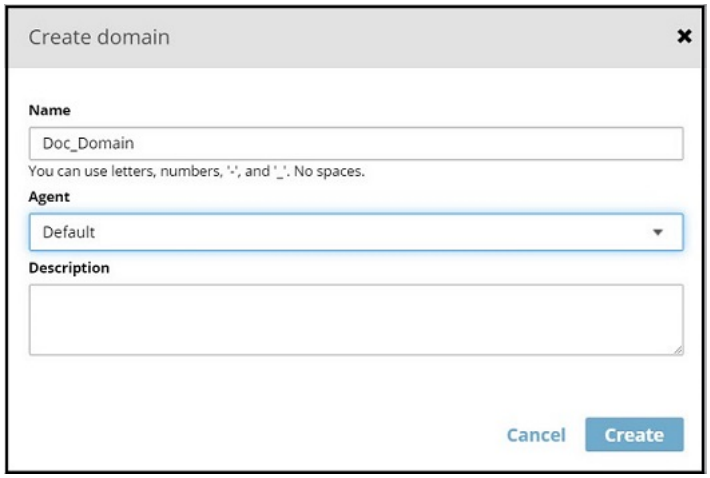
Admin UI

Procedure

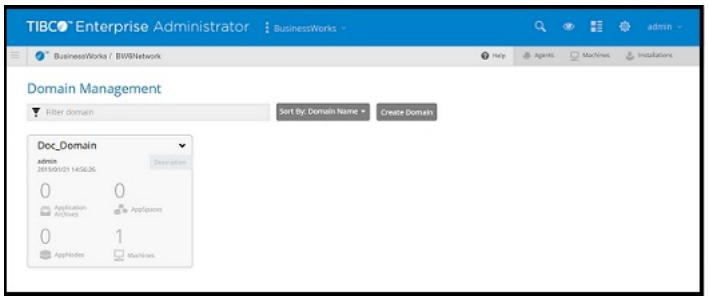
1. Click the BusinessWorks product icon on the Admin UI home page.



2. Click **Create domain**.
3. In the **Create domain** dialog box, enter domain name in the **Name** field.
4. Choose the agent registered with the TEA server from the **Agent** drop-down.



5. Click **Create**.
   The domain is displayed on the **Domain Management** page.



Deleting a Domain

Force delete a domain to remove all domain entities, including AppSpaces and AppNodes.

A domain deletion cannot be reversed. After a domain force delete, the domain and all entities inside the domain are deleted.

bwadmin Command Line

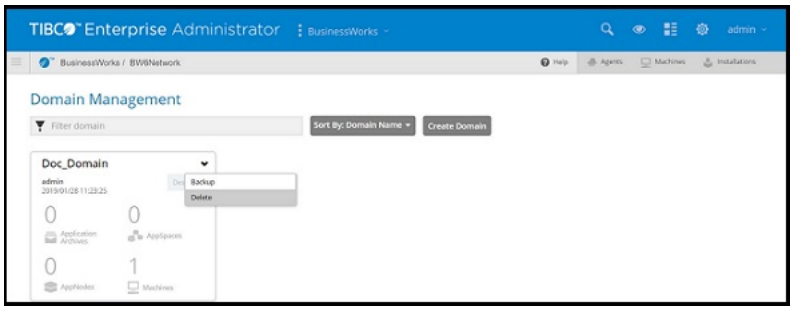
You can delete an empty domain or one that contains one or more AppSpaces.

Option	Command
To delete an empty domain	`BW_HOME\bin>bwadmin delete domain MyDomain`
To delete a domain that contains one or more AppSpaces	`BW_HOME\bin>bwadmin delete -force domain MyDomain`
To delete a domain using the timeout and force argument	`BW_HOME\bin>bwadmin delete -timeout xx(time in minutes) -force domain MyDomain` The `-timeout` argument is valid only when the AppSpace is running. For more information, see Force Shutting Down an AppNode.

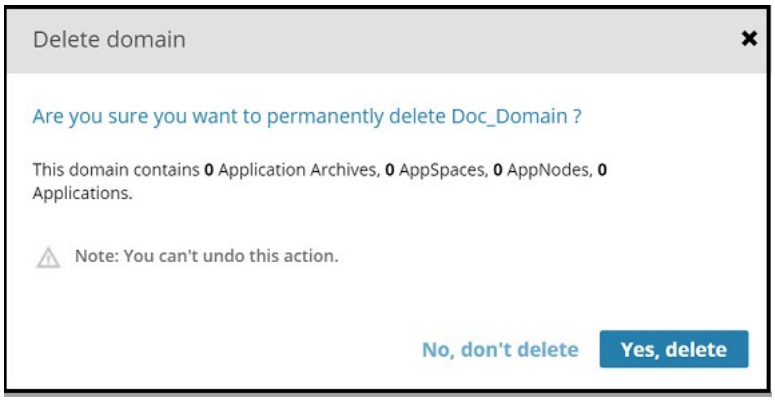
Admin UI

Procedure

1. Click the down arrow for the domain on the **Domain Management** page and choose **Delete.**



2. Click **Yes, delete** in the **Delete domain** dialog box.



Backing Up and Restoring a Domain

Backing up a domain exports the current state of the specified domain and contained runtime entities to a bwadmin command file. The entire domain is backed up, including remote bwagents, if applicable

to the specified domain. The command file can be provided to bwadmin to recreate the domain. Output can be compressed to a ZIP file with the **-zipped** option.

bwadmin Command Line

Procedure

1. To back up the current state of a domain, including profiles and archives, enter the **backup** command at the command line, using the **-s** option to identify the name of the destination file. Use the **domain** argument in the command line, with the name of the domain to back up. The domain can be either a local domain or a domain in a bwagent in the agent network. By default, destination files are written to the current working directory.
   This example backs up domain `Machine2Domain` in a networked bwagent to a command file named `machine2_domain.cmd`.

   Use the **-noarchives** option to exclude archives uploaded to the domain from the backup. (Note that references to the archives are included in the destination file. If needed, the paths in the destination file can manually be added to include archives in the restore.)

   The syntax is as follows where `-na` invokes the no archive option, and `-z` creates a zip file.

   ```
   backup -na -z -s C:/Backup/archives.zip domain Domain_Name
   ```

   ```
   backup -na -s C:/Backup/archives.cmd domain Domain_Name
   ```

   ```
   BW_HOME\bin>bwadmin backup -s machine2_domain.cmd domain Machine2Domain
   ```

   If you are restoring to a different location, you need to update the command file as follows:

   - The agent name will point to **localhost** by default; you need to change this to the name of the machine you are restoring to.
   - Update the domain home to point to the absolute path to the new location.
   - Update the path to the application archive (EAR) file to an absolute path.

2. To restore the domain,
   a) Enter the **bwadmin** command, providing the name of the backup command file. The following example recreates the domain `Machine2Domain` and the contained runtime entities.

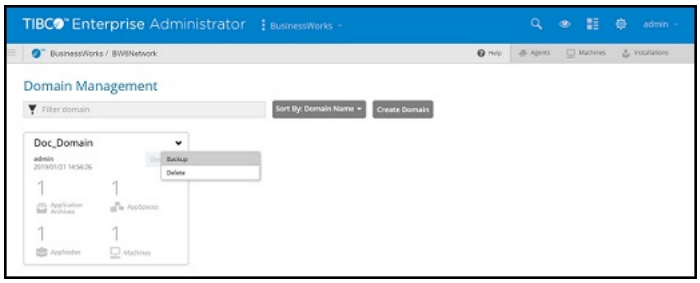
      ```
      BW_HOME\bin>bwadmin -f machine2_domain.cmd
      ```

   b) Use the **bwadmin show domains** command from the command line to verify the restore.

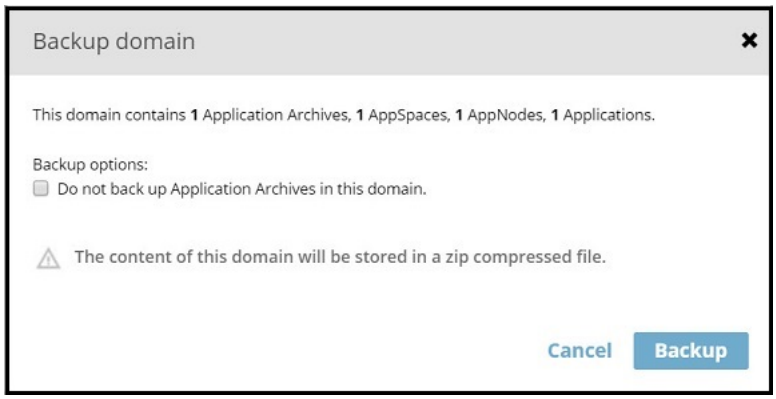
Admin UI

Procedure

1. Click the down arrow for the domain on the **Domain Management** page and choose **Backup**.

2. Click **Backup** in the **Backup domain** dialog box.
   a) To exclude archives from the backup, check the **Do not back up Application Archives in this domain** option.



The contents of the domain are written to a ZIP file that is downloaded to your computer. The filename is in the format: `Domain_backup_`*`domainName`*`.zip` The ZIP file contains a `.cmd` file that can be used to restore the environment.

Restoring the File System of a Domain

Restoring a domain restores the file system of the specified domain and all runtime entities in the domain to the state of the datastore.

Prerequisites

- The name of the domain must be known in order to restore.
- The bwagent must be running.

Procedure

1. To restore the file system for a domain and its runtime entities, enter the **`restore`** command at the command line, using the **`domain`** argument with the name of the domain to restore. The domain can be either a local domain or a domain in a bwagent in the agent network. This example restores domain `Machine2Domain` in a networked bwagent named `Machine2`.
   ```
   BW_HOME\bin>bwadmin restore -agent Machine2 domain Machine2Domain
   ```
2. To verify the restore, check the file system. Open the *`BW_HOME`*`\domains` folder. Look for a domain folder that matches the name of the domain.

Managing AppSpaces

An application is deployed to an AppSpace.

An AppSpace is a virtual pool of AppNodes where an application is deployed. When an application is deployed, the AppSpace starts the application on each of its AppNodes. More AppNodes can be added dynamically to the AppSpace to manage the load-balancing and fault tolerance needs for an application.

One or more applications can be deployed to an AppSpace.

A minimum number of AppNodes can be specified as a threshold for determining the AppSpace state. If the threshold falls below the minimum, the runtime state becomes Degraded.

When an application deployed to an AppSpace runs, and scalability is enabled, all the AppNodes in the AppSpace are started and share the load for the application. If scalability is turned off for a deployed application, the application executes on just one AppNode. For more information, see Fault Tolerance.

Creating an AppSpace

An AppSpace is created under a domain, which must exist before adding an AppSpace to it. An AppSpace contains one or more AppNodes. The domain name applies to the AppSpace.

The following characters are allowed in the AppSpace name:

- `A-Z`
- `a-z`
- `0-9`
- `-` (hyphen)
- `_` (underscore)

Illegal characters are stripped from the name.

The maximum length of a runtime entity name is 100 characters. If the maximum length is exceeded, the entity name is shortened to 100 characters.

bwadmin Command Line

To create an AppSpace named `MyAppSpace` in the domain `MyDomain`, execute the following command at the command line:

```
BW_HOME\bin>bwadmin create -d MyDomain -minNodes 1 appspace MyAppSpace
```

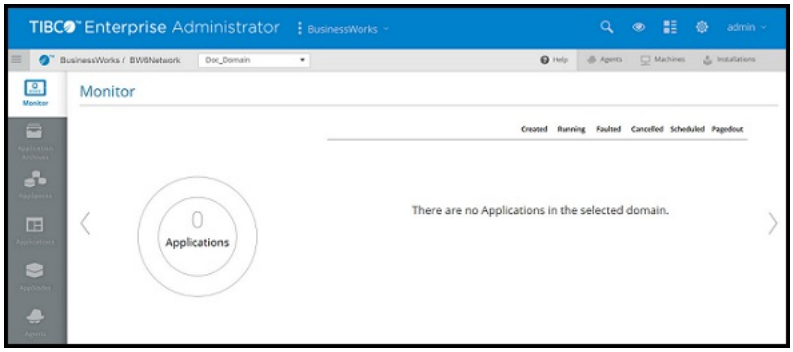
The `MyAppSpace` AppSpace is created in the domain `MyDomain` that exists on the machine where the bwagent is running. Use the **`-agent`** option to create an AppSpace running on a remote machine. Run the **`show agents`** command on the remote machine to get the agent name. If you are in local mode, the agent is not supported.

Admin UI

To create an AppSpace using the Admin UI:

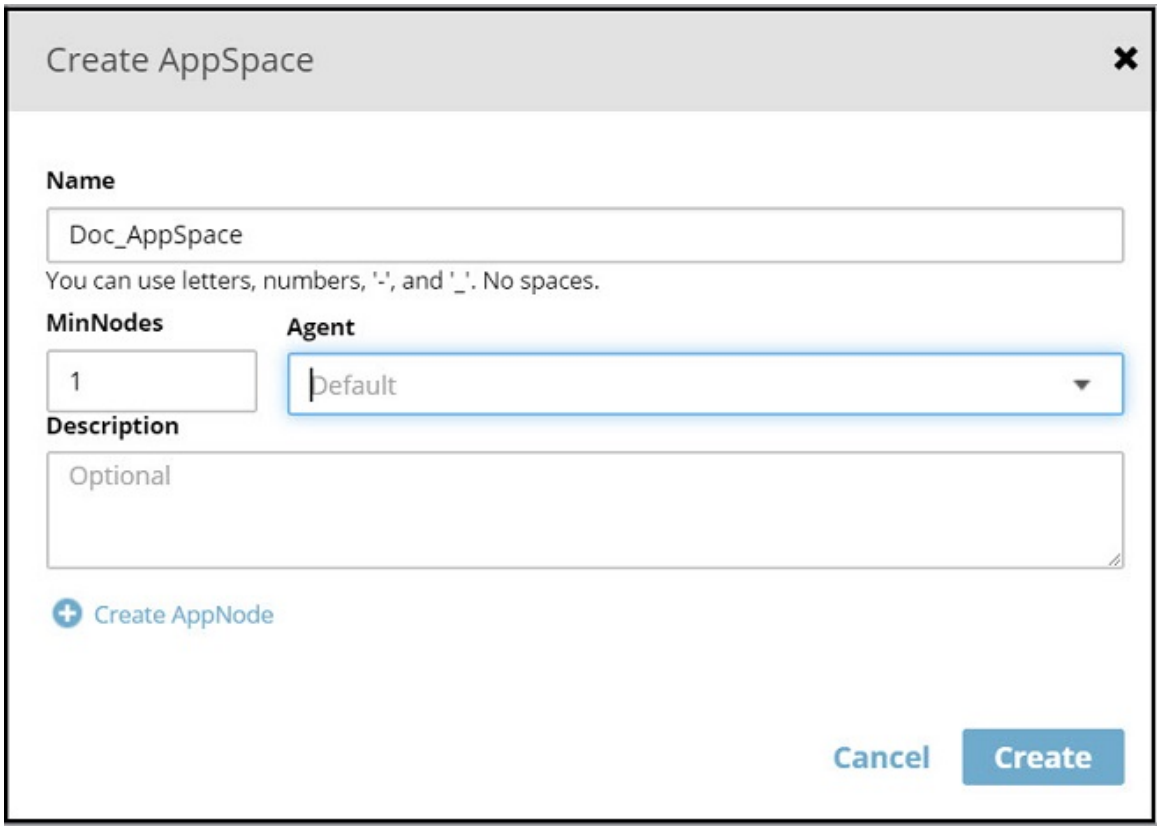
Procedure

1. On the **Domain Management** page, click the domain you want to add the AppSpace to. The **Monitor** page is displayed, showing that no runtime entities exist in the domain.



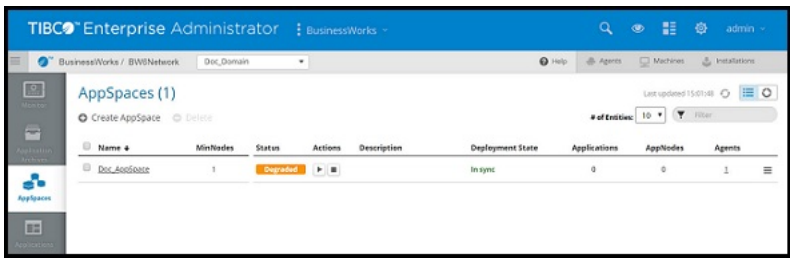
2. Click **AppSpaces** on the side bar to open the **AppSpaces** page.
3. Click **Create AppSpace** to open the **Create AppSpace** dialog box. Enter the following information.
   - **Name**: AppSpace name.
   - **MinNodes**: Minimum number of AppNodes for this AppSpace. Default is 1. The AppSpace status is set to **Degraded** if the minimum number of AppNodes is not created.

- **Agent**: The bwagent registered with the TEA server.
- **Description**: Optional description.



Click **Create AppNodes** in the **Create AppSpace** dialog box to create AppNodes. You can also create AppNodes from the **AppNodes** Admin UI page. For information, see Creating an AppNode .

4. Click **Create.**
The AppSpace is created and displayed on the **AppSpaces** page. The AppSpace status is set to Degraded as there are no AppNodes yet to satisfy the minimum requirement.



Starting an AppSpace

To run applications in an AppSpace, first start the AppSpace.

If an AppSpace does not contain any AppNodes, it will not be able to start. The AppSpace status is set to Degraded. The minimum number of nodes must be created in order for the AppSpace to start.

bwadmin Command Line

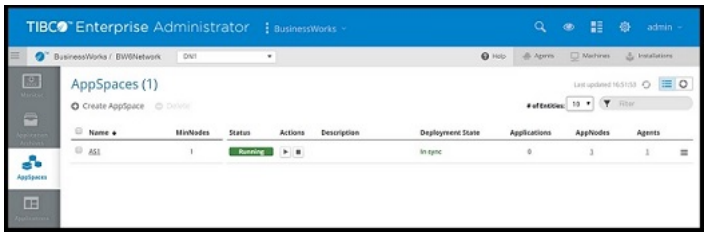
To start the AppSpace `MyAppSpace` in the domain `MyDomain`, execute the following command at the command line:

```
BW_HOME\bin>bwadmin start -d MyDomain appspace MyAppSpace
```

Admin UI

Procedure

- On the **AppSpaces** page for the domain, click the **Start** icon ▶ for the AppSpace you want to start. If the minimum number of nodes exists, the status displays as Starting, a transient state, then Running.



Editing an AppSpace Configuration

You can edit the configuration for a running AppSpace from the Admin UI.

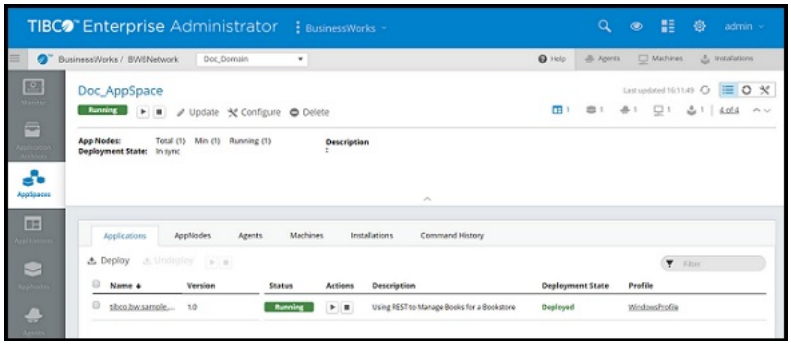
Admin UI

For information about some of the properties you can configure for an AppSpace using the Admin UI, see:

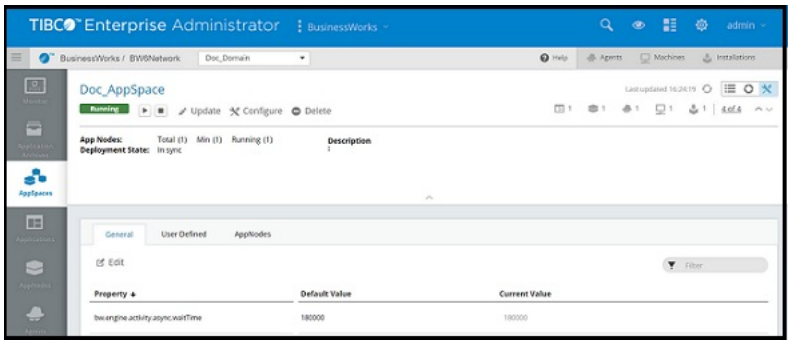
- Statistics Collection
- Engine Persistence Modes
- Tuning

Procedure

1. Select the AppSpace you want to configure on the **AppSpaces** page.



2. Click the **Edit** icon in the upper right of the **AppSpace** page. The **AppSpace Properties** page is displayed. Use the **General** tab to edit AppSpace properties.



3. Click **Edit** to open the tab for editing, and click **Submit** when you are done.

> You can also edit user defined properties on the **User Defined** tab or AppNode properties on the **AppNode** tab.

Viewing AppSpace States

An AppSpace has two states: Deployment and Runtime.

The Deployment state can have the following statuses:

AppSpace Deployment Statuses

Status	Description
In-Sync	The AppSpace is synchronized with its bwagents.
Out-of-Sync	The AppSpace is out of synchronization. The out-of-sync state may occur when: • a bwagent is not reachable due to network failure, or • the bwagent configuration may not have been applied remotely.

The Runtime state can have the following statuses:

AppSpace Runtime Statuses

Status	Operations Allowed in This Status	Description
Running	Stop	The minimum threshold of AppNodes configured for this AppSpace are running.
Stopped	Start, Delete	None of the AppNodes configured for this AppSpace are running.
Degraded	Stop	The number of AppNodes for this running AppSpace falls below the minimum specified threshold. The state also occurs when the AppSpace is not running.

AppSpaces do not have a starting state. However, AppNodes have their own lifecycle and may go from starting to stopped.

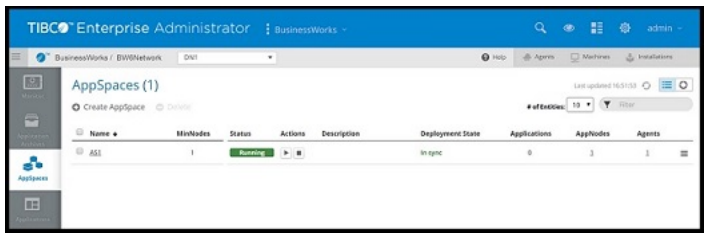
bwadmin Command Line

To view the status of the AppSpace `MyAppSpace` in the domain `MyDomain`, execute the following command at the command line:

```
BW_HOME\bin>bwadmin show -domain MyDomain appspace MyAppSpace
```

Admin UI

Navigate to the **AppSpace** page and view the **Status** column.



Stopping an AppSpace

When an AppSpace is stopped, all applications and AppNodes running in the AppSpace stop.

bwadmin Command Line

To stop the AppSpace `MyAppSpace` in the domain `MyDomain`, execute the following command at the command line:

```
BW_HOME\bin>bwadmin stop -d MyDomain appspace MyAppSpace
```

To force shut down the AppSpace `MyAppSpace` in the domain `MyDomain`, execute the following command at the command line:

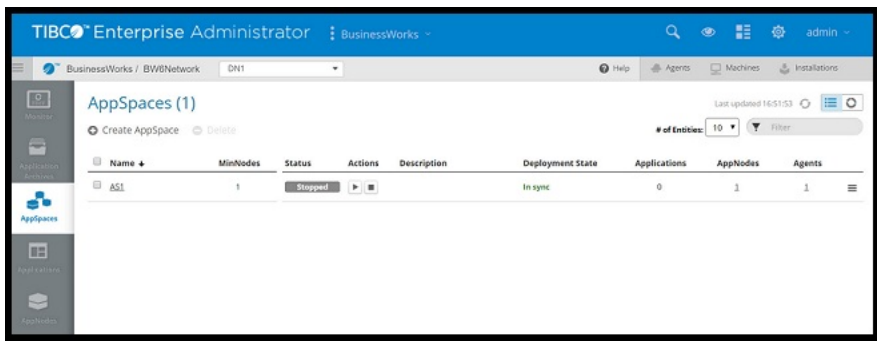
```
BW_HOME\bin>bwadmin stop -timeout xx(time in minutes) -domain MyDomain appspace
MyAppSpace
```

See Force Shutting Down an AppNode for more information.

Admin UI

On the **AppSpaces** page, click the **Stop** icon ■ for the AppSpace you want to stop.

The status for the AppSpace changes from Running to Stopping, a transient state, then Stopped.



Deleting an AppSpace

An AppSpace can be deleted if it has does not have associated AppNodes. If it contains AppNodes, you can force delete it.

bwadmin Command Line

To delete the AppSpace `MyAppSpace` in the domain `MyDomain`, execute the following command at the command line:

```
BW_HOME\bin>bwadmin delete -d MyDomain appspace MyAppSpace
```

If the AppSpace has an attached AppNode, the **`delete appspace`** command will fail. You can delete the attached AppNode and retry the **`delete appspace`** command or use the **`delete appspace`** command with the **`-force`** option.

```
BW_HOME\bin>bwadmin delete -force -domain MyDomain appspace MyAppSpace
```

To force delete the AppSpace `MyAppSpace` in the domain `MyDomain`, and forcefully shut down the running AppNodes, execute the following command at the command line:

```
BW_HOME\bin>bwadmin delete -timeout xx(time in minutes) -force -domain MyDomain
appspace MyAppSpace
```

See Force Shutting Down an AppNode for more information.

To delete all AppSpaces in the domain `MyDomain`, execute the following command at the command line:

```
BW_HOME\bin>bwadmin delete -d MyDomain -all appspace
```

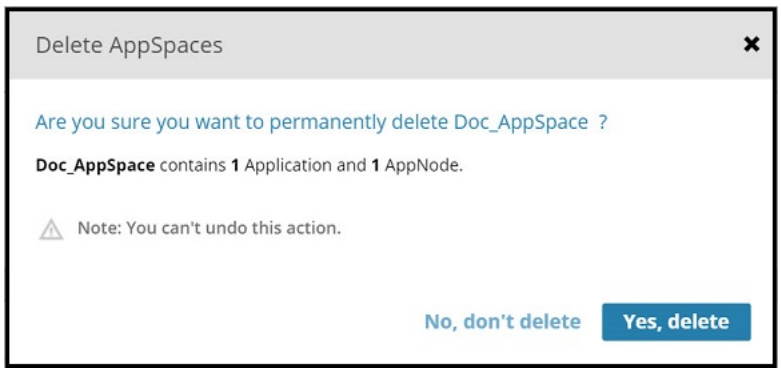
If any of the AppSpace in the domain `MyDomain` contains AppNode,

- Either first delete those AppNodes and then execute
  ```
  BW_HOME\bin>bwadmin delete -d MyDomain -all appspace
  ```
- Or If you want to forcefully delete all AppSpaces including AppNodes, execute the following command:
  ```
  BW_HOME\bin>bwadmin delete -d MyDomain -all -force appspace
  ```

Admin UI

Procedure

1. On the **AppSpaces** page, click the checkmark next to AppSpace you want to delete.
2. Click **Delete**.
3. Click **Yes, delete** in the **Delete AppSpaces** dialog box. (The dialog box message displays the number of applications and AppNodes that will be deleted.)



Backing Up and Restoring an AppSpace

Backing up an AppSpace exports the current state of the specified AppSpace to a bwadmin command file. The command file can be provided to bwadmin to recreate the AppSpace. Output can be compressed to a ZIP file with the **-zipped** option.

Procedure

1. To back up the current state of an AppSpace, enter the **backup** command at the command line, using the **-s** option to identify the name of the destination file. Use the **-domain** option with the **appspace** argument in the command line, with the name of the AppSpace to back up. The AppSpace can be either a local AppSpace or an AppSpace in a bwagent in the agent network. By default, destination files are written to the current working directory.
   This example backs up AppSpace `MyAppSpace` in Domain `MyDomain` to a command file named `myappspace.cmd`
   ```
   BW_HOME\bin>bwadmin backup -s myappspace.cmd -domain MyDomain appspace MyAppSpace
   ```
2. To restore the AppSpace:
   a) Enter the **bwadmin** command at the command line, providing the name of the backup command file. The following example recreates the AppSpace `MyAppSpace`.
      ```
      BW_HOME\bin>bwadmin -f myappspace.cmd
      ```
      If you are restoring to a different location, you need to update the command file as follows:
      - The agent name will point to **localhost** by default; you need to change this to the name of the machine you are restoring to.
      - Update the domain home to point to the absolute path to the new location.
      - Update the path to the application archive (EAR) file to an absolute path.

   b) Use the **bwadmin show appspaces** command from the command line, with the **-domain** option to verify the restore.

Restoring the File System of an AppSpace

Restoring an AppSpace restores the file system of the specified AppSpace and all runtime entities in the AppSpace to the state of the datastore.

Prerequisites

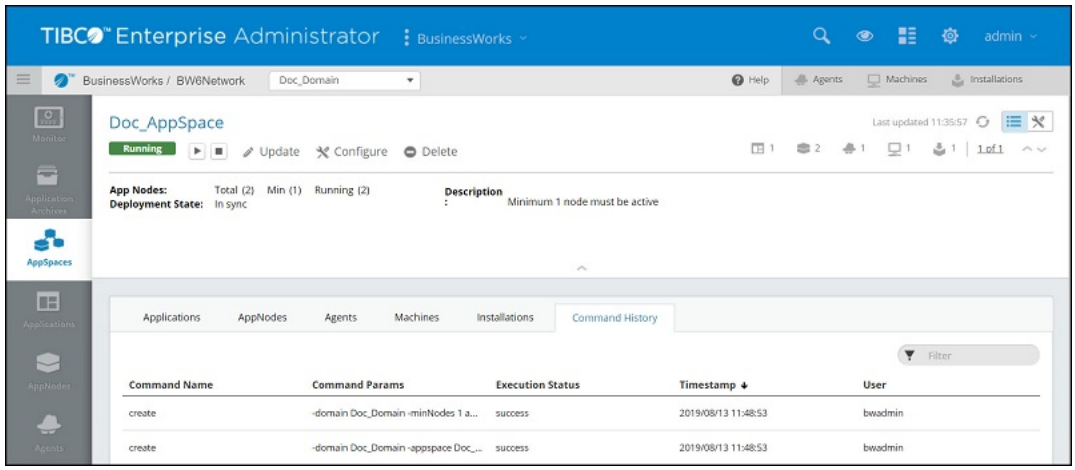
- The name of the containing domain and the name of the AppSpace must be known in order to restore.
- The bwagent must be running.

Procedure

1. To restore the file system for an AppSpace and the runtime entities in the AppSpace, open a terminal and navigate to *BW_HOME* `\bin`.
2. Enter the **restore** command from the command line, using the **-domain** option with the **appspace** argument specifying the name of the AppSpace to restore. This example restores AppSpace `MyAppSpace` in domain `MyDomain`.
   ```
   BW_HOME\bin>bwadmin restore -d MyDomain appspace MyAppSpace
   ```
3. To verify the restore, check the file system. Open the *BW_HOME*`\domains` folder. Check for the named AppSpace folder under: *BW_HOME*`\domains\`*domain_name*`\appspaces`

Command History

Open the **Command History** tab to view the commands or operations that were performed on an AppSpace.



Managing AppNodes

An AppNode is a runtime entity for hosting application modules and libraries.

An AppNode represents a physical engine process that is launched when an application starts to run.

- Install ActiveMatrix BusinessWorks on each machine hosting an AppNode.
- One or more AppNodes can be created in an AppSpace.

Creating an AppNode

An AppNode is created under an AppSpace. The domain and AppSpace name apply to the AppNode.

Multiple AppNodes can be created for an AppSpace.

When creating an AppNode that is on a remote machine, ensure that:

- the remote bwagent is a part of the network.
- the name of bwagent running on the remote machine is specified.

When an AppNode is created, do not specify the OSGi port. Only open this port for debugging when enabling the OSGi console on an AppNode. For details, see Enabling the OSGi Console for an AppNode.

The following characters are allowed in the AppNode name:

- `A-Z`
- `a-z`
- `0-9`
- `-` (hyphen)
- `_` (underscore)

Illegal characters are stripped from the name.

The maximum length of a runtime entity name is 100 characters. If the maximum length is exceeded, the entity name is shortened to 100 characters.

bwadmin Command Line

Use the **create** command to create an AppNode.

The bwagent must be running. Issue the following command to create an AppNode named `MyAppNode` in domain `MyDomain` and AppSpace `MyAppSpace`:

```
BW_HOME\bin>bwadmin create -d MyDomain -a MyAppSpace -httpPort 2222 appnode
MyAppNode
```

The **httpPort** option is required for an AppNode. If the specified port is already in use, an error is displayed and the AppNode cannot be created. To get a list of defined AppNodes for a given domain, with port numbers, with the **show** command: **show -d <DomainName> appnodes**

The following command creates an AppNode `MyAppNodeOnMac` in the domain `MyDomain` and AppSpace `MyAppSpace` on a remote machine whose agent name is `Machine2`.

```
BW_HOME\bin>bwadmin create -d MyDomain -a MyAppSpace -httpPort 2222 -agent Machine2
appnode MyAppNodeOnMac
```

- To create an AppNode on a remote machine, the member name of the bwagent on the machine where the AppNode will run must be known. Get the member name value by invoking the bwadmin **show agents** command on the remote machine.
- The validation of the HTTP ports is available by executing the command **validateport [options] port**. For example,

```
bwadmin[admin]> validateport 2233
TIBCO-BW-ADMIN-CLI-300342: HttpPort [2233] is available within BW scope.
```

Or

```
bwadmin[admin]> validateport 344566
TIBCO-BW-ADMIN-CLI-500338: HttpPort is not valid
```

To know more about validateport command, execute the command **validateport --help**.

Admin UI

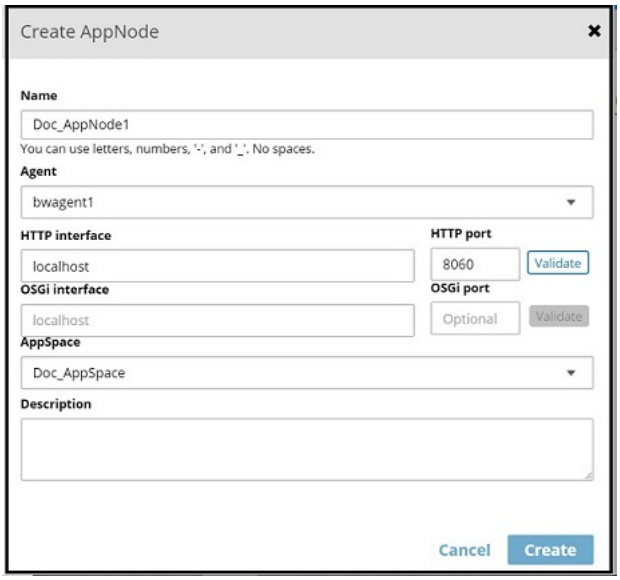
To create an AppNode using the Admin UI:

Procedure

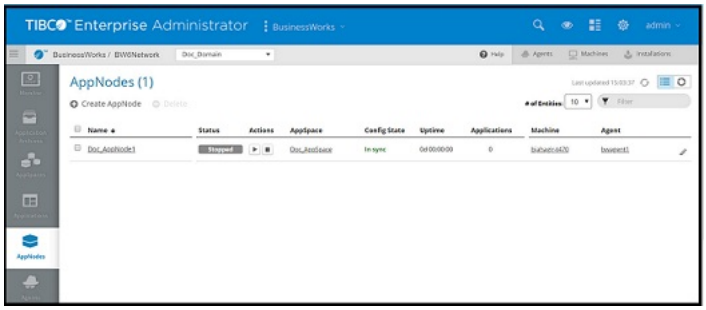
1. Click **AppNodes** on the side bar to open the **AppNodes** page.
2. Click **Create AppNode** to open the **Create AppNode** dialog box. Enter the following information:
   - **Name**: AppNode name.
   - **Agent**: The bwagent registered with the TEA server.
   - **HTTP interface**: The HTTP interface for the AppNode.
   - **HTTP port**: The HTTP port for the AppNode. Click **Validate** to see if the port is available.

     The **Validate** button validates the HTTP ports within the ActiveMatrix BusinessWorks scope only.
   - **OSGi interface**: The OSGi interface for the AppNode. Open this port only for debugging sessions.
   - **OSGi port**: The OSGi port for the AppNode. Open this port only for debugging sessions.
   - **AppSpace**: The AppSpace for this AppNode.
   - **Description**: Optional description.



3. Click **Create.**
   The AppNode is created and displayed on the **AppNode** page. The AppNode status is set to Stopped.



Starting an AppNode

Use the **`start`** command to manually start an AppNode.

When an AppSpace is started, all AppNodes associated with the AppSpace automatically start.

By default, the value for the **`bw.engine.shutdownOnFailure`** property is `true` in the AppSpace `config.ini` file. This ensures that the AppNode does not start when there are any issues when starting the bwengine. You can also configure the property at the AppNode, or the AppSpace level.

bwadmin Command Line

Execute the following command at the command line to start the `MyAppNode` AppNode:

```
BW_HOME\bin>bwadmin start -d MyDomain -a MyAppSpace appnode MyAppNode
```

If the AppNode is not gracefully shut down, it could corrupt the `/config` folder.

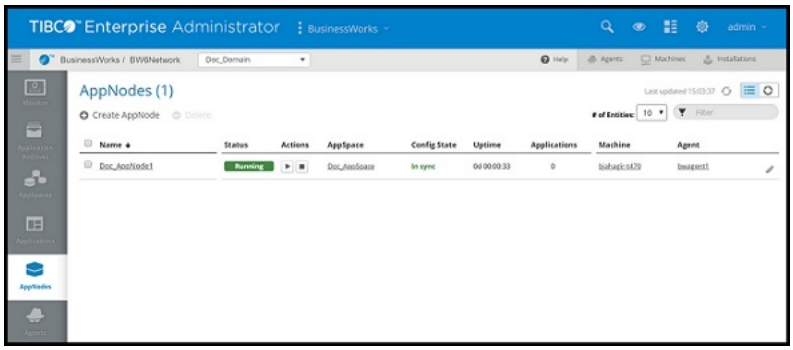
Configure the **`bw.appnode.clean.config.folder.on.startup`** property in the AppNode, or the AppSpace `config.ini` file.

To create a new `/config` folder every time the AppNode starts, set the **`bw.appnode.clean.config.folder.on.startup`** property to `true`. Setting the property to `false`, or leaving it undefined results in the `/config` folder not being deleted when the AppNode starts.

Admin UI

Procedure

- On the **AppNodes** page, click the **Start** icon ▶ for the AppNode.
  The status for an AppNode is displayed as Starting, a transient state, then Running.



Editing an AppNode Configuration

You can edit the configuration for a running AppNode from the Admin UI. Changes are applied when you restart the AppNode.

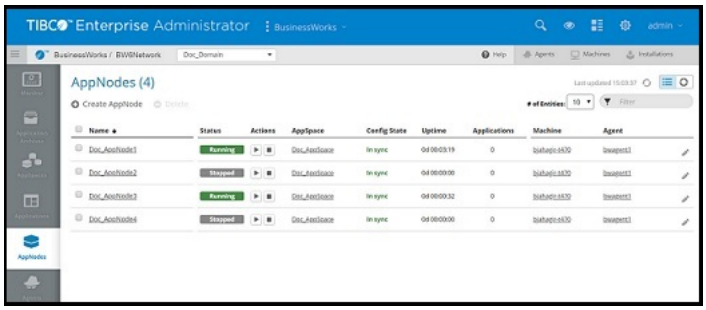
Admin UI

For information about some of the properties you can configure for an AppNode using the Admin UI, see:

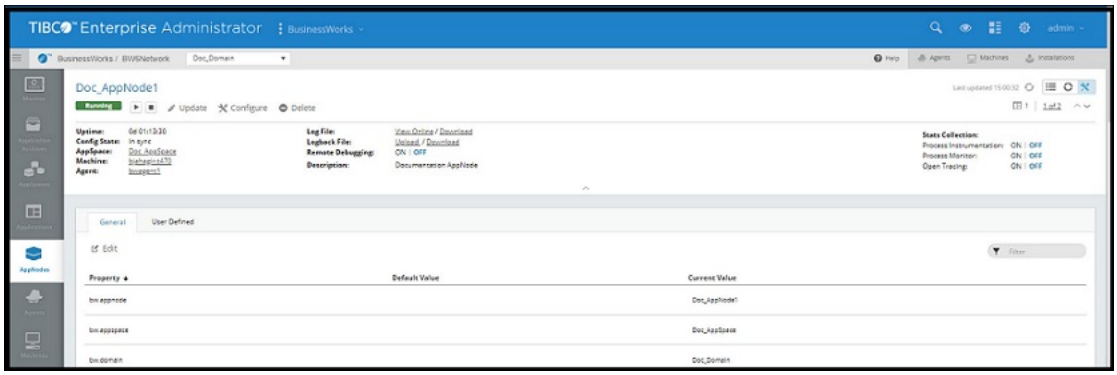
- Statistics Collection
- Engine Persistence Modes
- Tuning
- Configuring a Unified Doc URL
- AppNode Logging

Procedure

1. Select the AppNode you want to configure on the **AppNodes** page.



2. Click the **Edit** icon 🔧 in the upper right of the **AppNode** page. The **AppNode Properties** page is displayed. Use the **General** tab to edit AppNode properties.



3. Click **Edit** to open the tab for editing, and click **Submit** when you are done. You need to restart the AppNode to apply the changes. The AppNode status is set to **`Out of sync`** until the AppNode is restarted.
   You can also edit user defined properties on the **User Defined** tab.

Viewing AppNode Statuses

An AppNode has two states: Deployment and Runtime.

The Deployment state can have the following statuses:

AppNode Deployment Statuses

Status	Description
In-Sync	AppNode is in a synchronized state with deployment features, configurations, or both.
Out-of-Sync	AppNode is out of synchronization with certain deployment features, configurations, or both.

An AppNode's Runtime state relies on the timestamp of the hosting machine and on the timestamp of the machine hosting the bwagent. The timestamps on the machines must be within 20 seconds of each other for the status of the AppNode to be reported correctly. The Runtime state can have the following statuses:

AppNode Runtime Statuses

Status	Operations Allowed When in This Status	Description
Running	Stop	Reported only by the AppNode. The AppNode has successfully initialized all the product bundles and reached the target start level.
Stopped	Start, Delete	The AppNode is not running. This is determined only by the bwagent.
Deploying	<none>	This is a transitional state when an application is getting deployed on an AppSpace.

Status	Operations Allowed When in This Status	Description
Impaired	Stop	The AppNode is in an **Impaired** state after it has encountered an error during the startup in one of the product bundles. This means that the application is not yet ready to run, and may impact the functionality of the AppNode. Each application has a list of dependencies that need to be satisfied for it to run. These dependencies are listed below: • The engine must be up and running. If the engine is configured incorrectly, it might not start. • Shared resources that are defined in the application must be initialized. If shared resources are misconfigured, they might not start. • All bundles that belong to an application must be resolved. If a bundle is missing an import-package, it might not start. If any of these dependencies are not met, the application is displayed as being in the **Impaired** state. The "**la**", which stands for list applications, command of the OSGi console shows the details of the application and their resolved and unresolved dependencies. This will explicitly tell you the dependency that is not met and where to look further to investigate the problem.
Unreachable	<none>	The AppNode is unreachable when a remote bwagent and the remote machine is down, or a network failure occurs resulting in network partitioning. This state is determined only by the bwagent.
Start Failed	Start, Delete	This is determined by the bwagent when it is making an attempt to start the AppNode process when the AppNode does not start.
Starting	Stop	This is a transitional state reported by the AppNode when the Web API layer has been initialized, but the Web API is still accessible and can report the state. This happens when a bundle blocks the shutdown thread.
Stopping	<none>	This is a transitional state when an AppNode stops gracefully or is forcefully shut down from the Admin UI or from the command line.

bwadmin Command Line

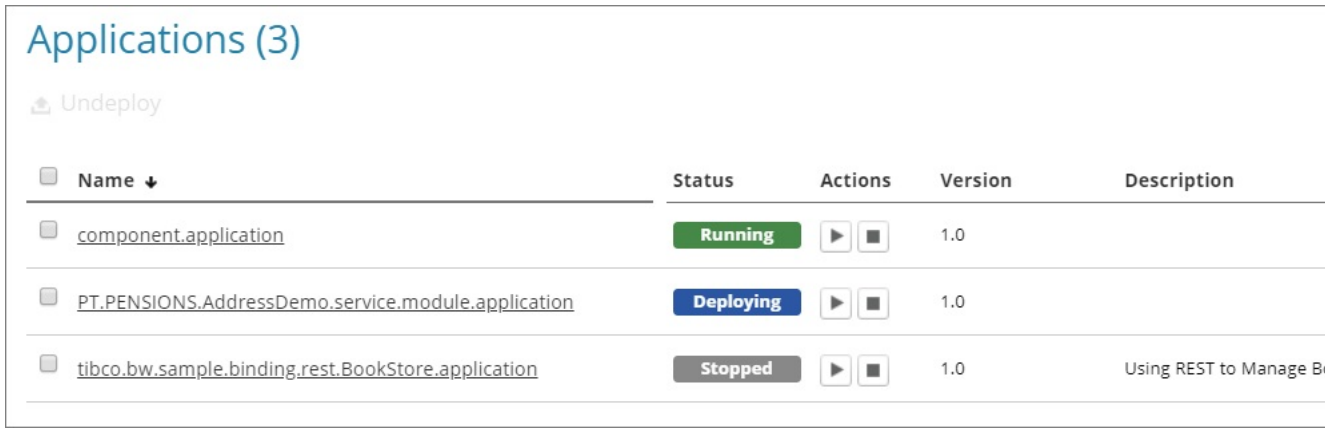
To view the status of the AppNode `MyAppNode` in the AppSpace `MyAppSpace`, execute the following command at the command line:

:

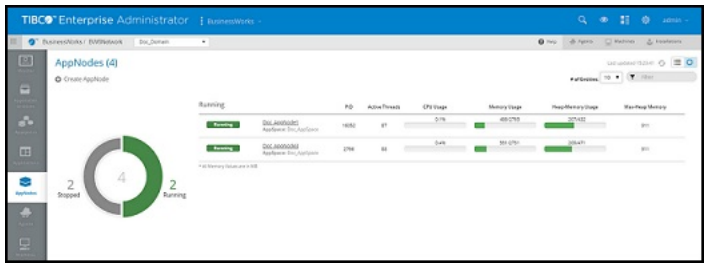
```
BW_HOME\bin>bwadmin show -domain MyDomain -appspace MyAppSpace appnode MyAppNode
```

Admin UI

Navigate to the **AppNode** page and view the **Status** column.



To view heap memory usage, click on the **graph** view.



Following fields are available:

Field	Description
PID	Process ID
Active Threads	Number of threads currently active on the AppNode
CPU Usage	CPU usage percentage
Memory Usage (MB)	Memory used versus allocated memory for a process as an AppNode in MB
Heap-Memory Usage	Shows used memory out of total memory.
MaxHeapMemory	Maximum heap memory that can be allocated to a JVM in bytes

Auto Collecting Engine Data

The collection of data require 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 or an AppNode data. Invoke the REST API as POST: `http://<host>:<port>/bw/framework.json/collect/`.

The engine data collected for an AppNode for TIBCO ActiveMatrix BusinessWorks™ is stored at *<user.dir>* `\..\debug\APPNODE_DATA_<TIME_STAMP>.zip` where *<user.dir>* is of the form

`$BW_HOME\bw\<version>\domains\<domain_name>\appnodes\<appspace_name>\<appnode_name>\bin`

bwadmin Command Line

Execute the following command at the command line to collect AppNode's data:

1. In a terminal, navigate to *BW_HOME*`\bin` and type `bwadmin`.
2. Go to `MyDomain`.

   ```
   bwadmin[admin]> cd MyDomain
   ```

3. Go to `MyAppSpace`.

   ```
   bwadmin[admin@MyDomain]> cd MyAppSpace
   ```

4. Start the AppNode, if it is not already running:

   ```
   bwadmin[admin@MyDomain/MyAppSpace]> start appnode MyAppNode
   ```

5. Go to `MyAppNode`

   ```
   bwadmin[admin@MyDomain/MyAppSpace]> cd MyAppNode
   ```

6. Run the **`collectappnodedata`** command

```
bwadmin[admin@MyDomain/MyAppSpace/MyAppNode]>collectappnodedata [options]
[operation]
```

The following options are available:

Option	Description
-o, -override	Delete all previously created data files. Generate new files as per the selected operation. It has two options true or false. The default option is true.
-i, -input	Input list of operations to be performed. Comma-separated list without space. Sample input: `"THREAD_DUMP", "HEAP_DUMP", "VM_ARGUMENTS", "ENVIRONMENT_VARIABLES", "THREAD_SNAPSHOT", "MEMORY_SNAPSHOT", "SYSTEM_PROCESS_INFORMATION", "SYSTEM_PROPERTIES", "CPU_INFORMATION", "osgiCommand1", "osgiCommand2"`
-d, -domain	Domain name

Option	Description
-p, -path	Output directory path
-n, -appnode	Name of an AppNode
-a, -appspace	AppSpace name, Applicable when an entity is an AppNode
-al, -all	Download all files from specified directory path.
-dp, -downloadpath	Download all files from the specified directory path. It has two options true or false. The default option is false.
-dd, -downloadanddelete	Delete file after download. It has two options true or false. The default option is false.
--help	Display this help message

The following operations are available:

Operation	Description
ALL Admin CLI command: `collectappnodedata -o false -p "D:/appNode/data/" ALL`	Options available: • override: [optional] Override previously created data. The default value is true. • path: [optional] Set output directory path. **Output**: All default set of operations are executed.  SYSTEM_PROCESS_INFORMATION is not executed when running in TIBCO Business Studio™ for BusinessWorks™.

Operation	Description
INCLUDE Admin CLI command: `collectappnodedata -p "D:/appNode/data/" -i "command1","command2" INCLUDE`	Only the set of operations given as an input are executed. Options available: • override: [optional] Override previously created data. The default value is true. • path: [optional] Set output directory path. • input: Set of operations to be executed. Comma-separated list for admin CLI command and JSON list for REST API. **Output**: For Example, if the input list is la,lp,"lapi *",thread_dump, then only these four operations are executed.  Input list is mandatory to execute this operation.
EXCLUDE Admin CLI command: `collectappnodedata -p "D:/appNode/data/" -i "command1","command2" EXCLUDE`	All default set operations excluding the set of operation given as input is executed. Options available: • override: [optional] Override previously created data. The default value is true. • path: [optional] Set output directory path. • input: [optional] Set of operations to be executed. Comma-separated list for admin CLI command and JSON list for REST API. **Output**: For example, if the input list is la,lp"lapi *",thread_dump, then all default set operation without these four operations are executed. The following operations are executed: ["HEAP_DUMP", "VM_ARGUMENTS", "ENVIRONMENT_VARIABLES", "SYSTEM_PROPERTIES", "THREAD_SNAPSHOT", "MEMORY_SNAPSHOT", "SYSTEM_PROCESS_INFORMATION","CPU_INFORMATION", "LMETRICS", "LCFG", "LENDPOINTS"]  Input list is expected for this operation. If the list is empty the operation works similar to ALL operation.

Operation	Description
DOWNLOAD Admin CLI command: `collectappnodedata -p "D:/appNode/data/" -dp "D:/downloads" -al true -dd true DOWNLOAD`	The operation is used to download the collected AppNode data. Options available: • `path`: [optional] the path for the directory where data is collected. OR the path for the file. • `all`: [optional] If the value of option "path" is a directory or if the value is not set, then the path is the default directory. If you set the "all" option as TRUE, all files present in that directory having names starting with keyword "APPNODE_DATA" are compressed to a single zip file with name "APPNODE_DATA" and then the file APPNODE_DATA.zip is downloaded. If the value is not set or set as false and the path value is a directory, then the last generated file is sent as output. • `downloadanddelete`: [optional] Delete the file after download. It has two options true or false. The default option is false. • `downloadpath`: [mandatory and applicable for CLI command] To provide download path directory. **Output**: The file is download at specified download path.  If an option downloadanddelete is selected and the file download operation fails because of network issue, the file is not available for download next time.

Operation	Description
LIST Admin CLI command: `collectappnodedata -p "D:/appNode/data/" LIST`	The operation is used to list the data file present at the set path. Options available: • `path`:[optional] the path for the directory where data is collected. OR the path for the file. **Output**: If the path set is a directory or is a default path, then all files present in that directory having names starting with keyword "APPNODE_DATA" are listed as output. If the path is a file, then it checks if the file exists.
DELETE Admin CLI command: `collectappnodedata -p "D:/appNode/data/" DELETE`	The operation is used to delete the data files created. Options available: • `path`:[optional] the path for the directory where data is collected. OR the path for the file. **Output**: If the path set is a directory or is a default path, then all files present in that directory having names starting with keyword "APPNODE_DATA" are deleted. If the path is a file, then the file is deleted.  The files with names starting with the keyword "APPNODE_DATA" are deleted.

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 executing 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 JSON list. Only listed operations are executed.
EXCLUDE	This API accepts a list of commands or operations as an input in the form of JSON list. All default set operations excluding the set of operation given as input is executed.
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.

Header Parameter	Description
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 name APPNODE_DATA.zip and downloaded at once. The default value is FALSE.
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 authorization of the user. This option is mandatory. It is the login id for the session.

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.

Stopping an AppNode

When an AppNode is stopped, applications running on the AppNode stop.

bwadmin Command Line

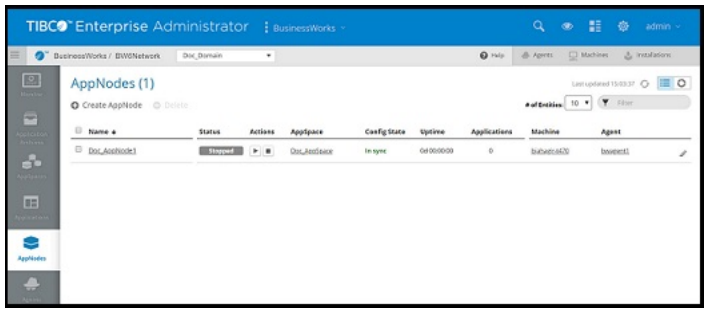
To stop the AppNode `MyAppNode` in the AppSpace `MyAppSpace`, execute the following command at the command line:

```
BW_HOME\bin>bwadmin stop -d MyDomain -appspace MyAppSpace appnode MyAppNode
```

Admin UI

On the **AppNode** page, click the **Stop** icon ■ for the AppNode.

The AppNode will change from Running to Stopping, a transient state, then Stopped.



Force Shutting Down an AppNode

Use the argument, **`-timeout`** xx (in minutes) from the command line to forcefully shut down an AppNode, after the timeout is reached. The default timeout value is zero (0) and the AppNode will stop only after the completion of all the jobs. From the Admin UI, select the **Force shutdown after wait time** check box. If the check box is not selected the AppNode will stop after the default timeout. From the Admin UI, AppNodes can also be forcefully shut down from the AppSpace level, the Application level, and from the Agent and Machine level.

When the timeout is specified, the AppNode will shut down after the timeout is reached. If the jobs are completed before the timeout value is reached, the AppNode will stop on its own, and will not wait for the timeout period that has been specified. If the jobs are not completed in the timeout period, the AppNode will shut down irrespective of the state of the running jobs.

Multiple force kill commands can be triggered one after the other, and the most recent force shut down command takes precedence over the previous commands.

bwadmin Command Line

To force shut down an AppNode, `MyAppNode` in the AppSpace `MyAppSpace` execute the following command at the command line:

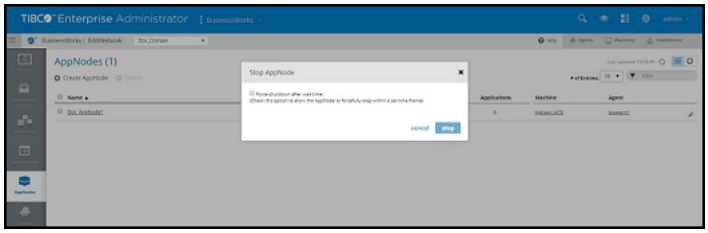
```
BW_HOME\bin>bwadmin stop -timeout xx(time in minutes) -d MyDomain -appspace
MyAppSpace appnode MyAppNode
```

Admin UI

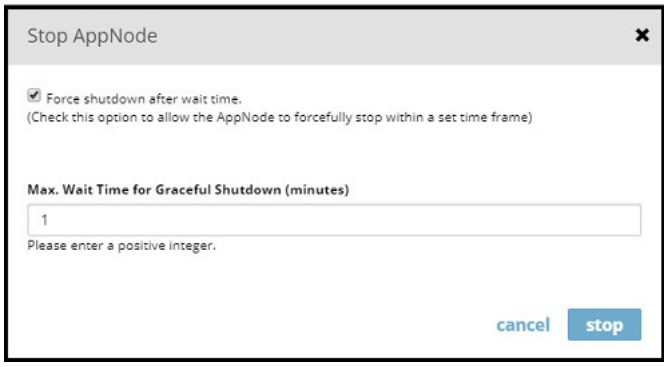
,

Procedure

1. On the **AppNode** page, click the **Stop** icon ■ for the AppNode.



2. To shut down the AppNode forcefully within the timeout period, select the **Force shutdown after wait time** check box.
3. Set the timeout value (in minutes) in the **Max Wait Time for Graceful Shutdown** field and click the **stop** button.



Deleting an AppNode

Deleting an AppNode deletes any contained applications. From the bwadmin command line, AppNodes that are running must be force deleted.

bwadmin Command Line

To delete the AppNode `MyAppNode` in the AppSpace `MyAppSpace`, execute the following command at the command line:

```
BW_HOME\bin>bwadmin delete -d MyDomain  -a MyAppSpace appnode MyAppNode
```

If the AppNode is running, the **`delete appnode`** command will fail. You can stop the AppNode and retry the **`delete appnode`** command or use the **`delete appnode`** command with the **`-force`** option.

```
BW_HOME\bin>bwadmin delete -force -d MyDomain -a MyAppSpace appnode MyAppNode
```

To force delete the AppNode `MyAppNode` in the AppSpace `MyAppSpace`, and forcefully shut down the running AppNode, execute the following command at the command line:

```
BW_HOME\bin>bwadmin delete -timeout xx(time in minutes) -force -d MyDomain -
appspace MyAppSpace appnode MyAppNode
```

See Force Shutting Down an AppNode for more information.

To delete all AppNodes in the AppSpace `MyAppSpace`, execute the following command at the command line:

```
BW_HOME\bin>bwadmin delete -d MyDomain -a MyAppSpace -all appnode
```

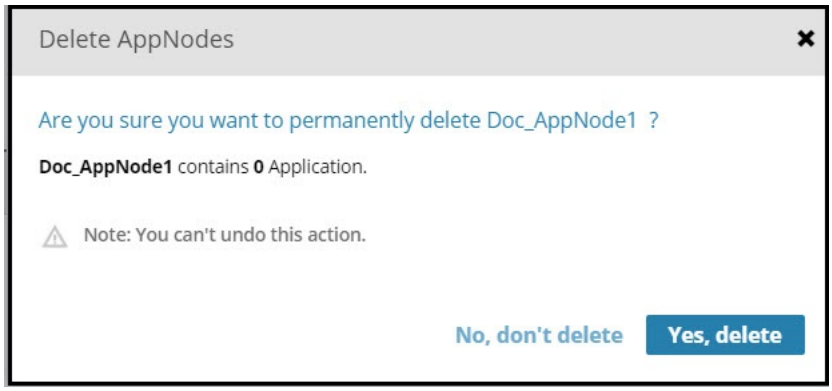
If any of the AppNode has running applications or if you want to forcefully delete all AppNodes, execute the following command:

```
BW_HOME\bin>bwadmin delete -d MyDomain -a MyAppSpace -force -all appnode
```

Admin UI

Procedure

1. On the **AppNodes** page , click the check mark next to the AppNode to delete.
2. Click **Delete**.
3. Click **Yes, delete** in the **Delete AppNodes** dialog box. (The dialog box message displays the number of applications that will be deleted.)



Debugging an AppNode

A running AppNode can be enabled for remote debugging from either bwadmin or the Admin UI. Once enabled, use TIBCO Business Studio™ for BusinessWorks™ to debug the application running on the AppNode. For more information, see "Remote Debugging" in the *TIBCO ActiveMatrix BusinessWorks™ Application Development* guide. An Appnode should be enabled for remote debugging in secure environments where only an administrator has the access rights to enable or disable ports.

bwadmin Command Line

The **`enabledebugport`** command can only be executed against a running AppNode. It should be issued from bwadmin interactive mode, not from the command line.

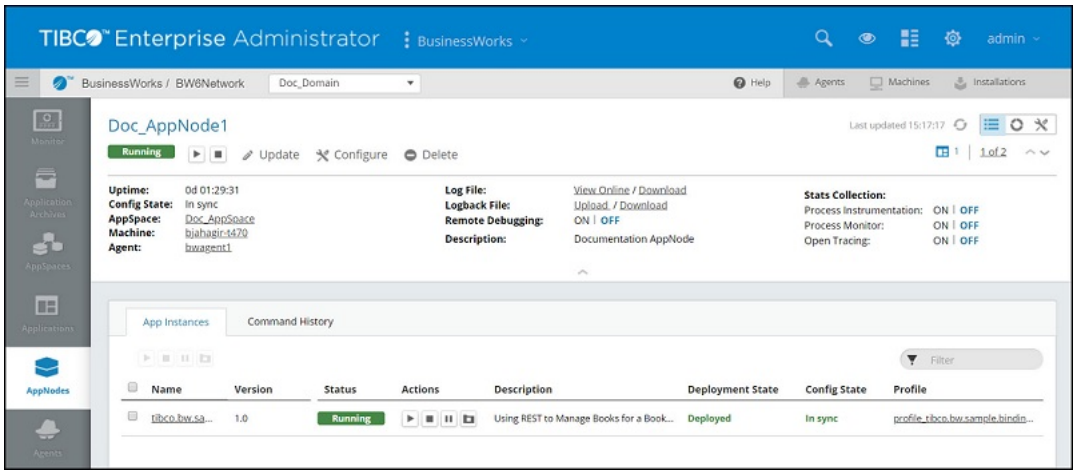
Procedure

1. In a terminal, navigate to *BW_HOME*`\bin` and type `bwadmin`.
2. Go to `MyDomain`.
   ```
   bwadmin[admin]> cd MyDomain
   ```
3. Go to `MyAppSpace`.
   ```
   bwadmin[admin@MyDomain]> cd MyAppSpace
   ```
4. Start the AppNode, if it is not already running:
   ```
   bwamdin[admin@MyDomain/MyAppSpace]> start appnode MyAppNode
   ```
5. Run the **`enabledebugport`** command, passing the host and port number. For example:
   ```
   bwamdin[admin@MyDomain/MyAppSpace]> enabledebugport -n MyAppNode JSMITH-W520 9061
   Enabled debug port on AppNode [MyAppNode] in AppSpace [MyAppSpace] in Domain
   [MyDomain]
   ```
6. **Important:** When you finish debugging, close the port to reduce security risks and reduce overhead. For example:
   ```
   bwamdin[admin@MyDomain/MyAppSpace]> disabledebugport -n MyAppNode
   Debugger disabled for AppNode [MyAppNode] in AppSpace [MyAppSpace] in Domain
   [MyDomain]
   ```

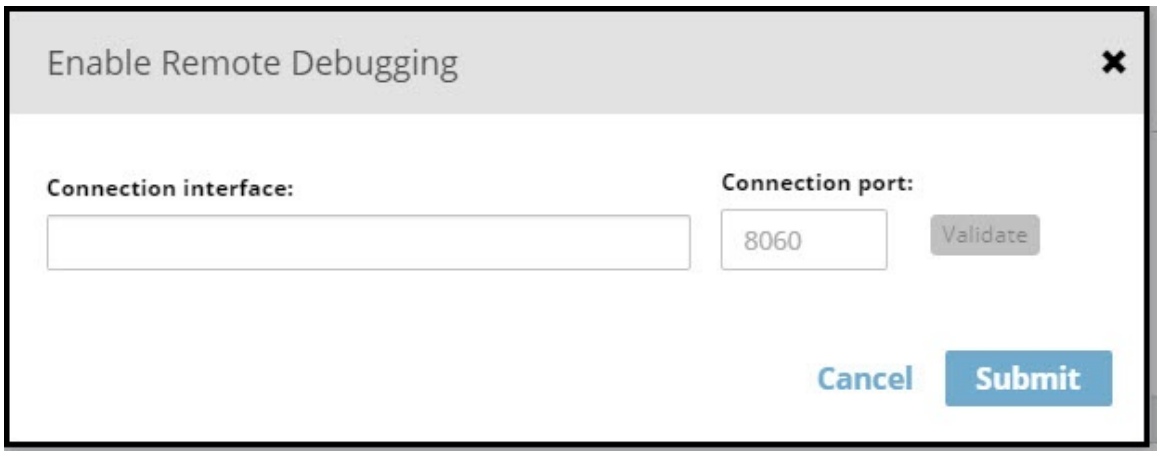
Admin UI

Procedure

1. Open the **AppNode** page for the AppNode to enable for remote debugging.



2. Click the **Remote Debugging > ON** option. The Enable Remote Debugging dialog box is displayed.



3. Enter the following information, and click **Submit** to open the port.

   - **Connection Interface**: The default connection interface is the name of the bwagent.
   - **Connection Port**: The debug port.

   The port is opened and displayed on the AppNode page:

4. When you finish debugging, close the port by clicking **OFF** to reduce security risks and reduce overhead.

Enabling the OSGi Console for an AppNode

The **`enableconsole`** command dynamically enables the OSGi console on the given port for a running AppNode. Advanced users can telnet to the port and execute native OSGi commands to get information about an AppNode's status. This is useful when collecting diagnostic data remotely. By default, the OSGi port is closed.

Although an AppNode can be created with OSGi port details specified, this is not recommended. Keeping this port open when the console is not in use poses a security risk.

The **enableconsole** command can only be executed against a running AppNode. It should be issued from bwadmin interactive mode, not from the command line.

If you are testing and running applications in TIBCO Business Studio, you can also access the OSGi commands from the **Console view**.

bwadmin Command Line

Procedure

1. In a terminal, navigate to *BW_HOME*\bin and type bwadmin.
2. Go to MyDomain.
   ```
   bwadmin[admin]> cd MyDomain
   ```
3. Go to MyAppSpace.
   ```
   bwadmin[admin@MyDomain]> cd MyAppSpace
   ```
4. Start the AppNode, if it is not already running:
   ```
   bwamdin[admin@MyDomain/MyAppSpace]> start appnode MyAppNode
   ```
5. Run the **enableconsole** command, passing the host and OSGi port number. For example:
   ```
   bwamdin[admin@MyDomain/MyAppSpace]> enableconsole -n MyAppNode JSMITH-W520 9060
   TIBCO-BW-ADMIN-CLI-300304: Console enabled for AppNode [MyAppNode] in Domain
   [MyDomain]
   ```

   You can also specify the OSGi port number using one of the following syntax:

   - *hostname*:*port number*
   - localhost:*port number*
6. Open a new terminal window and use the **telnet** command to access the OSGi console:

   telnet JSMITH-W520 9060

   The OSGi console is opened in a terminal.
7. Use OSGi commands to retrieve information about the engine, the AppNode, the running application. For a list of commands, enter: **help**. See the topic called OSGi Runtime Statistics Commands for more information.
8. When you are done, use the **disconnect** command to gracefully quit the telnet session, and leave the OSGi port open for reentry. Use the telnet **stop** command to close the connection after the debugging session is complete. Do not use the telnet **exit** command as this will shut down the AppNode.

OSGi Commands

You can run commands to gather data about running AppNodes and applications.

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<command_name>
  ```
  For example,
  ```
  curl -v http://localhost:8090/bw/framework.json/osgi?command=help%20pauseapp
  ```

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 ActiveMatrix BusinessWorks™ activity icons with a given ID and type.
bw:lais	Retrieves statistics for activities that have been executed in one of the processes for the application.
bw:lapi	Retrieves information about all process instances for the application based on the applied filters.  You can see the output of lapi command on the console. The output can be exported in the CSV format.
bw:las	Lists all instantiated activities.
bw:lat	Lists all registered activity types.
bw:lbwes	Lists all subscribers that are currently listening to ActiveMatrix BusinessWorks™ statistics events.
bw:le	Prints information about ActiveMatrix BusinessWorks engines.
bw:lec	Prints information about ActiveMatrix BusinessWorks engine configurations.
bw:lendpoints	Lists endpoints exposed by the ActiveMatrix BusinessWorks engine.
bw:les	Lists all instantiated `EventSources`.
bw:lmetrics	Prints job metrics for application(s) running on the AppNode.
bw:lpis	Prints statistics of one of the processes that have been executed for the application.
bw:lr	Lists all resource details.
bw:lrhandlers	Lists all resource handlers.
bw:lrproxies	Lists all resource proxies.
bw:startesc	Starts collection of execution statistics for a given entity (activity/process) for application(s).
bw:stopesc	Stops execution statistics collection of given entity (process/activity) for application(s).

Command	Description
bw:startpsc	Starts collection of process statistics for application(s).
bw:stoppsc	Stops collection of process statistics for application(s).
bw:lapis	Prints summary of 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 ActiveMatrix BusinessWorks processes.
frwk:ll	Print information about all libraries.
frwk:lloggers	Print all loggers currently configfured on the AppNode.
frwk:lp	Print information about all known ActiveMatrix BusinessWorks processes.
frwk:pauseapp	Stop the process starters and their bindings and pause all jobs of an ActiveMatrix BusinessWorks application.
frwk:resumeapp	Start the process starters and their bindings and resume all jobs of an ActiveMatrix BusinessWorks application.
frwk:setloglevel	Sets the log level for a given logger.
frwk:startcomps	Start all process starters and their bindings of an ActiveMatrix BusinessWorks application.
frwk:startps	Start the process starters of an ActiveMatrix BusinessWorks application.
frwk:stopps	Stop the process starters of an ActiveMatrix BusinessWorks application.
frwk:startapp	Start an ActiveMatrix BusinessWorks application gracefully.
frwk:stopapp	Stop an ActiveMatrix BusinessWorks application gracefully.

Command	Description
frwk:td	Print a full thread dump.

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

Running OSGi Commands

You can execute OSGi commands from the Admin CLI, SSH Client, and HTTP Client.

Running OSGI Commands from bwadmin Command Line

Follow these steps to execute OSGI commands from the bwadmin command line.

Prerequisites

Ensure the AppNode is running.

Procedure

1. From the CLI, navigate to the `/bin` folder.
2. Start bwadmin.
3. Enter an OSGI command. OSGI commands can be executed from the command line in the format **osgi [options] [command]** . In the following example, the option **-n** is used to specify the AppNode MyAppNode, and the **la** command is used to print information about applications on MyAppNode.

```
bwamdin[admin@MyDomain/MyAppSpace]> osgi -n MyAppNode "la"
```

OSGI Command Options

Option	Description
-descr/ -description	Description of an entity
-t/-outputfile	The OSGI command output is provided in a newly created text file. The text file is written to the `/bin` folder by default, but you can specify a different location, for example, `C:/temp/OSGI_OUTPUT.txt`.
-d/-domain	Specifies the domain name.
-a/-appspace	Specifies the name of the AppSpace.
-n/-appnode	Specifies the name of the AppNode.
--help	Lists all OSGI commands. For more information on specific commands, see OSGi Commands.

Running OSGi Commands Using SSH Client

You can run OSGi commands using SSH client.

Procedure

1. Edit the AppNode `config.ini` file.
2. Uncomment the following properties:
```
#osgi.console=null
#osgi.console.ssh=<free port>
#osgi.console.enable.builtin=false
#osgi.console.ssh.useDefaultSecureStorage=true
#java.security.auth.login.config=../sshconfig/equinox.console.jaas.login.conf
#ssh.server.keystore=../sshconfig/hostkey.ser
#org.eclipse.equinox.console.jaas.file=../sshconfig/store
```
3. Configure the property **`osgi.console`** with the host name.
4. Configure the property **`osgi.console.ssh`** with the free port. This port is used for OSGi remote access.
   Default credentials to connect via SSH are `equinox` and `equinox`
5. Restart the AppNode.

Running OSGi Commands Using HTTP Client

You can run OSGi commands using HTTP client. Preffered way is using curl.

Execute the following OSGi command using curl in the following format:

```
curl -v http://<IP address>:<port on which AppNode is running>/bw/framework.json/
osgi?command=<OSGi command>
```

For Example:

- To print information about BWEngines:
```
curl -v http://localhost:2224/bw/framework.json/osgi?command=le
```

- To pause all jobs of TIBCO ActiveMatrix BusinessWorks™ applications:
```
curl -v http://localhost:1113/bw/framework.json/osgi?command=pauseapp%20-v
%201.0%20tibco.bw.sample.binding.rest.BookStore.application
```

Backing Up and Restoring an AppNode

Backing up an AppNode exports the current state of the specified AppNode to a bwadmin command file. The command file can be provided to bwadmin to recreate the AppNode. Output can be compressed to a ZIP file with the **-zipped** option.

Procedure

1. To back up the current state of an AppNode,
   a) Open a terminal and navigate to *BW_HOME*\bin.
   b) Enter the **backup** command from the command line, using *-s* option to identify the name of the destination file. Use the **-domain** and **-appspace** options , with the **appnode** argument in the command line. The AppNode can be either a local AppNode or an AppNode in a bwagent in the agent network. By default, destination files are written to the current working directory.
   This example backs up AppNode MyAppNode in a bwagent network to a command file named MyAppNode.cmd.
   ```
   BW_HOME\bin>bwadmin backup -s MyAppnode.cmd -d Machine2Domain -a AS1 appnode
   MyAppNode
   ```
2. To restore the AppNode,
   a) Open a terminal and navigate to *BW_HOME*\bin.
   b) Enter the **bwadmin** command, providing the name of the backup command file. The following example recreates the AppNode MyAppNode.
   ```
   BW_HOME\bin>bwadmin -f MyAppnode.cmd
   ```
   If you are restoring to a different location, you need to update the command file as follows:
   - The agent name will point to **localhost** by default; you need to change this to the name of the machine you are restoring to.
   - Update the domain home to point to the absolute path to the new location.
   - Update the path to the application archive (EAR) file to an absolute path.

   c) Use the **bwadmin show appnodes** command at the command line with the **-domain** and **-appspace** options to verify the restore.

Restoring the File System of an AppNode

Restoring an AppNode restores the file system of the specified AppNode and all runtime entities in the AppNode to the state of the datastore.

Prerequisites

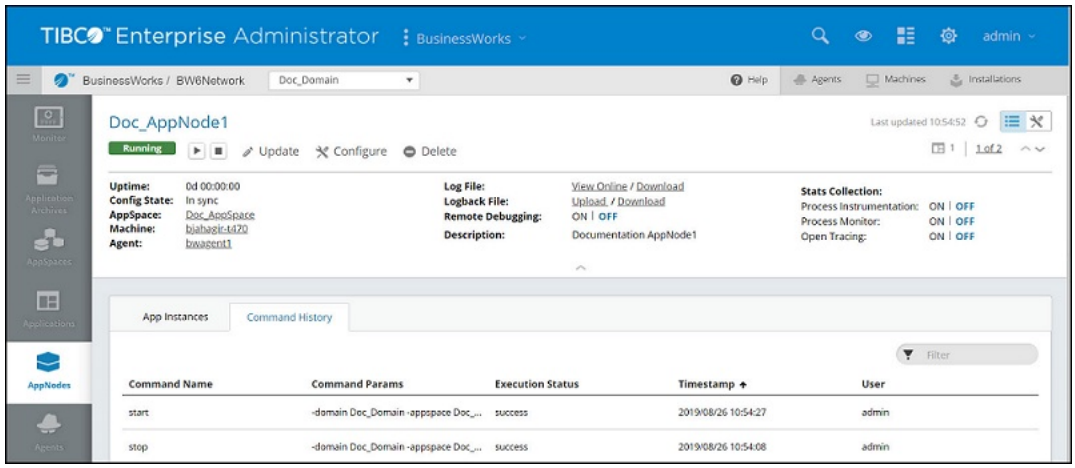
- The names of the containing domain and AppSpace and the name of the AppNode must be known in order to restore.
- The bwagent must be running.

Procedure

1. To restore the file system for an AppNode and the runtime entities in the AppNode, open a terminal and navigate to *BW_HOME*\bin.
2. Enter the **restore** command, using the **-domain** and **-appspace** options with the **appnode** argument specifying the name of the AppNode to restore. This example restores AppNode MyAppNode in domain Machine1Domain and AppSpace AS1.
   ```
   BW_HOME\bin>bwadmin restore -d Machine1Domain -a AS1 appnode MyAppNode
   ```
3. To verify the restore, check the file system. Open the *BW_HOME*\domains folder. Browse the folder and look for the named AppNode folder under: *BW_HOME*\domains*domain_name*\appnodes

Command History

Open the **Command History** tab to view the commands or operations that were performed on an AppNode.



Managing an Application

An application is an instance of a deployment archive. It has an independent lifecycle with regard to an AppSpace or AppNode. After completing the design process in TIBCO Business Studio, the application can be deployed.

An application provides the business logic to perform one or more related tasks and contains an application module that was defined in TIBCO Business Studio. The module itself can include processes, subprocesses, a process starter or a process service, and multiple activities. See *TIBCO ActiveMatrix BusinessWorks Application Development* for information about creating applications in TIBCO Business Studio.

Deployment occurs after an archive is uploaded to a domain and before an application is started. An archive is deployed to an AppSpace. One or more applications can be deployed to an AppSpace.

If an AppSpace spans multiple machines, the application is deployed onto each of the machines. If there are multiple AppNodes attached to the AppSpace either on a single machine or across multiple machines, the **`start`** command starts the application on each of the AppNodes. The runtime status of the application is reported for each AppNode and can be monitored using bwadmin.

When an application is deployed, you can choose to also start the application by giving the **`-as`** option on the command line. By default, this option is off and the application must be started explicitly after being deployed.

To configure an application, provide the desired profile that should contain the variable values for the application. This step is necessary if you want to run the application with different sets of variables and deploy it with different argument values, for example, for a Windows machine or a Mac.

Preparing for Deployment

Preparation for deployment involves the following steps:

- Creating an Application
- Creating an Application with Multiple Profiles
- Creating an Archive

After creating the application, the profile, and the archive, you are ready to deploy the application into an AppSpace. Deploying an application involves the following steps:

- Uploading an Archive
- Deploying an Archive
- Configuring an Application
- Starting an Application

Creating an Application

This section shows how to create a simple project where you design and create an application.

After creating the project, you choose activities from the palettes to design and create an application. For more information, see *TIBCO ActiveMatrix BusinessWorks™ Application Development*.

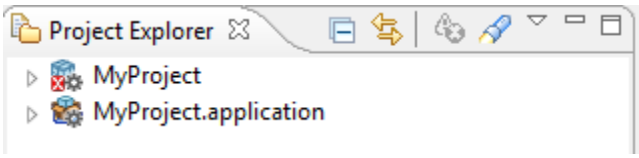
Procedure

1. Start TIBCO Business Studio™ for BusinessWorks™.
2. Launch the BusinessWorks Application Module wizard by selecting **File** > **New** > **Project** > **BusinessWorks** > **BusinessWorks Application Module** and click **Next**.
3. In the **Project name** field, provide a project name.
   Select the **Use default location**, **Create empty process**, and **Create Application** check boxes.
4. Click **Finish**.

Result

The new project is visible in the **Project Explorer**.

New Project in Project Explorer



Creating an Application with Multiple Profiles

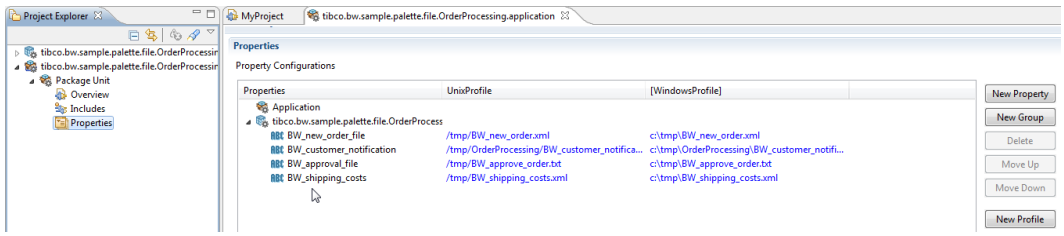
You can define multiple profiles when creating an application in TIBCO Business Studio™ for BusinessWorks™.

A *profile* is a collection of module and application properties that an application uses. When an application is deployed with different properties, different profiles are available for each deployment. For example, you can create a Windows profile for an application that runs on a Windows machine and another for the same application running on a UNIX machine.

Prerequisites

An application is created with profiles using TIBCO Business Studio for BusinessWorks. For more information about creating applications see the *TIBCO ActiveMatrix BusinessWorks™ Application Development* guide. The following screenshot shows an application with a profile for Windows and another for UNIX. Each profile has a set of defined properties and values. The values use the appropriate operating system syntax to point to the files in the file system. The files are created and maintained outside of TIBCO Business Studio for BusinessWorks.

Application Profiles



Follow these steps to create an application profile:

Procedure

1. Start TIBCO Business Studio for BusinessWorks and open an application.
2. Expand the application and double-click **Properties** under **Package Unit**.
   This displays the **Properties** pane in the **Process Editor**.
3. Click the **New Profile** button to add a new profile.
4. In the **Create New Profile** window, enter a name for the new profile. For example, enter `WindowsProfile` and click **OK**.
   The WindowsProfile gets created and available to the right of the **[default]** column in the Properties pane.
5. Double-click the field under the profile that corresponds to a property, and enter a value for the property.
6. Save the project.
   You can create multiple profiles as needed.

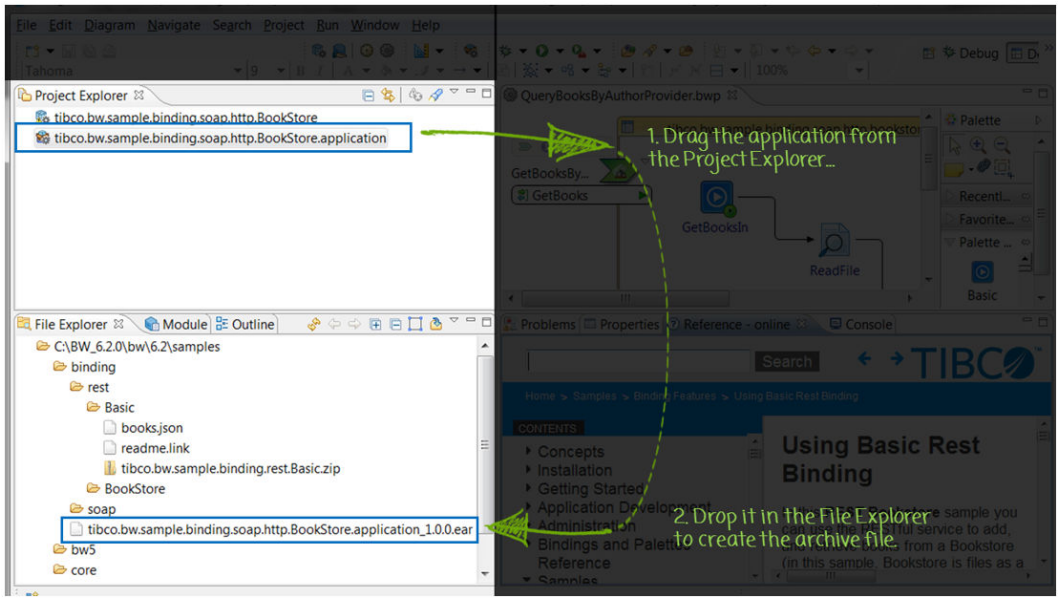
Creating an Application Archive

You can create an application archive in TIBCO Business Studio by dragging and dropping the application project from the Project Explorer to the File Explorer window.

You create an application archive after designing and testing the application.

Procedure

1. In TIBCO Business Studio, go to **File Explorer** tab and click the **Open Directory to Browse** icon.
2. Select the directory where you want to store the archive and click **OK**.
3. Drag an application from the Project Explorer to the directory in the File Explorer.



The application archive is written to the directory using the syntax `<application>_<version>.ear` where the `<version>` starts from 1.0.0 and increments as more generations occur.

Uploading an Application Archive

Uploading an application archive copies the specified file to the specified domain. The archive is copied to the *BW_HOME*`\domains\`*domain_name*`\archives` directory. If the specified archive already exists in the domain, it can be replaced on upload.

The following steps show how to upload an application archive.

bwadmin Command Line

Procedure

1. Execute the following command from the command line. Specify the fully qualified location of the application archive file. Note the use of forward slashes "/" for the Windows path. This Windows example uploads the BookStore sample that has been copied to the `C:\ear` folder.

```
BW_HOME\bin>bwadmin upload -d MyDomain C:/ear/
tibco.bw.sample.binding.rest.BookStore.application_1.0.0.ear
```

If the specified application archive already exists in the selected domain, use the **`-replace`** option to replace it.

2. By default, the **`upload`** command copies the archive file to the *BW_HOME*`\domains\`*domain_name*`\archives` folder. To upload an archive to a different location, use the **`-path`** option and specify a path relative to the *BW_HOME*`\domains\`*domain_name* folder. For example, the following example creates the *BW_HOME*`\domains\`*domain_name*`\test` folder and uploads the archive file:

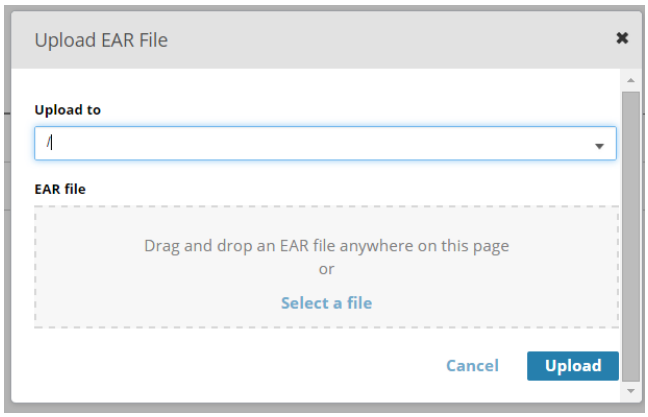
```
BW_HOME\bin>bwadmin upload -d MyDomain -path ../../test/MyArchives C:/ear/
tibco.bw.sample.binding.rest.BookStore.application_1.0.0.ear
```

Admin UI

Procedure

1. Select the domain and open the **Application Archives** page. Click **Upload**.
2. In the **Upload EAR File** dialog box, enter the following information:

- **Upload to**: EAR file upload folder. Default location is the `domains` folder.
- **EAR file**: Drag and drop the archive file. If the file already exists in that folder, select the **Replace any existing version** check box.



3. Click **Upload**, then **Done.**
   The archive is displayed on the **Application Archives** page.

Deploying an Application

You use the **`deploy`** command to deploy an application archive to an AppSpace.

You can deploy multiple applications to an AppSpace. You can deploy and run multiple versions of same application on the same AppNode at the same time.

bwadmin Command Line

When using the **`deploy`** command, the EAR filename is the relative location of the archive with respect to the *`BW_HOME`*`\domains\`*`domain_name`*`\archives` folder. For example, if `MyDomain` contains the AppSpace to deploy to and the archive file is located in the *`BW_HOME`*`\domains\MyDomain \archives`directory, do not specify a qualifier for the archive file location.

Execute the following command from the command line to deploy an uploaded application archive to `MyAppSpace`:

```
BW_HOME\bin>bwadmin deploy -d MyDomain -a MyAppSpace
tibco.bw.sample.binding.rest.BookStore.application_1.0.0.ear
```

Replacing a deployed archive file will undeploy the application. To replace an archive file that has been uploaded, but not deployed, use the **`-replace`** option with the upload command to upload the archive again. When the **`-replace`** option is used to replace an archive file that has been uploaded and deployed, the following error message is displayed,

```
Upload Operation Cancelled. As Application MyApplication from archive
MyApplication.ear has been deployed to those AppSpaces.
Please use force replace (-f)option with the upload command i.e
[upload -domain D -replace -f C:/Users/Administrator/Desktop/EAR/MyApplication.ear]]
```

The force replace option will undeploy the existing archive file and replace the old archive file with the new file.

Admin UI

Procedure

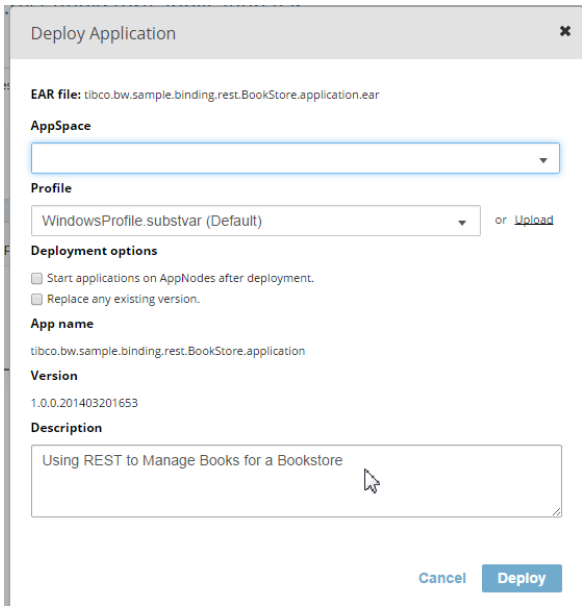
1. Select the domain and open the **Application Archives** page. Drill down into an archive. On the **Application Archive** page for the selected archive, click the **Deploy** button.

2. In the **Deploy Application** dialog box, enter the following information:

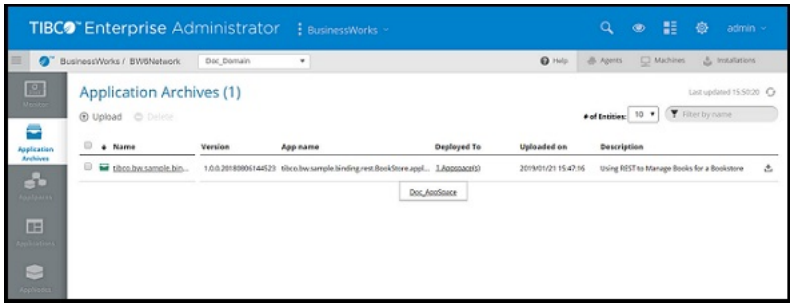
   - **AppSpace**: The AppSpace to deploy to.
   - **Profile**: The profile file to use for deployment. (An application profile can be changed after deployment. See Configuring an Application. Click **Upload** to upload a new profile.
   - **Start applications on AppNodes after deployment**: Starts the application on AppNodes after deployment.
   - **Replace any existing version**: Replaces an existing version of the same application. If the **Replace any existing version** check box is selected, a warning message is displayed, indicating that the upload action will undeploy the existing application from the Appspace.

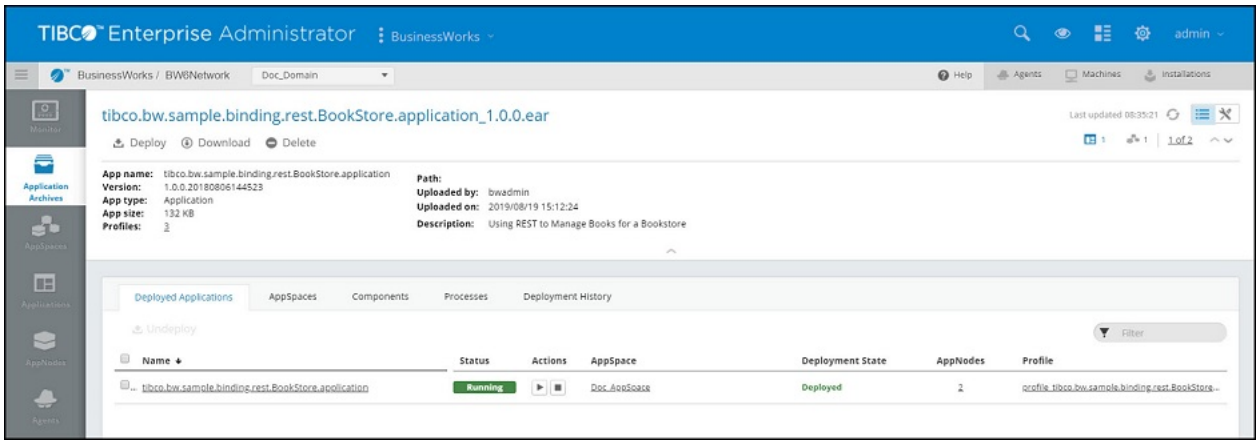
     If the EAR file is uploaded but not deployed, the warning message is not displayed and the EAR file is replaced.

   - **App name/Version:** Read-only fields that display the application name and version.
   - **Description**: Optional description; pulled from archive.

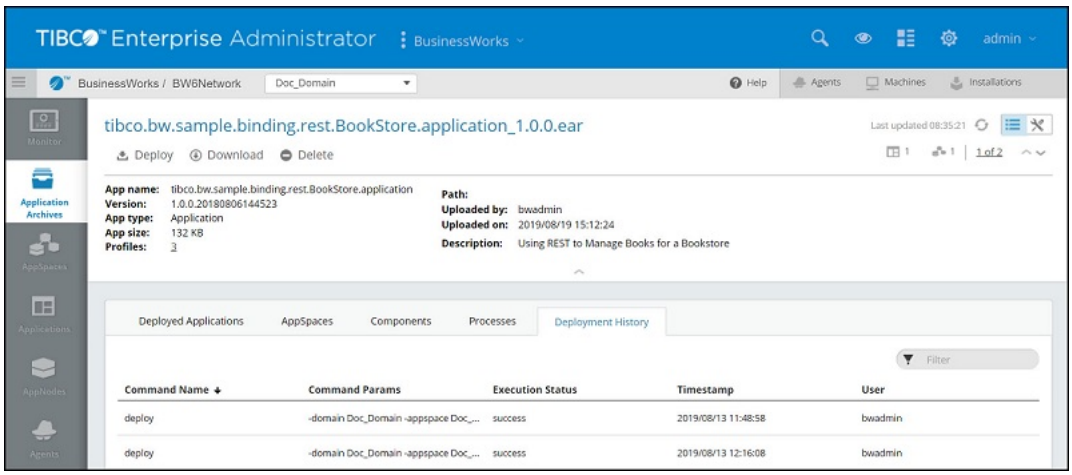


The archieved applications are displayed on the **Application Archives** page. The **Deployed To** column displays the number of AppSpaces where the application has been deployed.

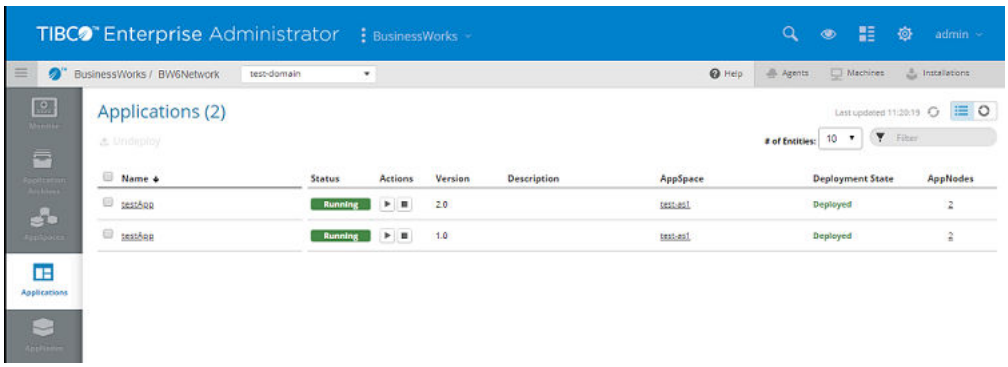


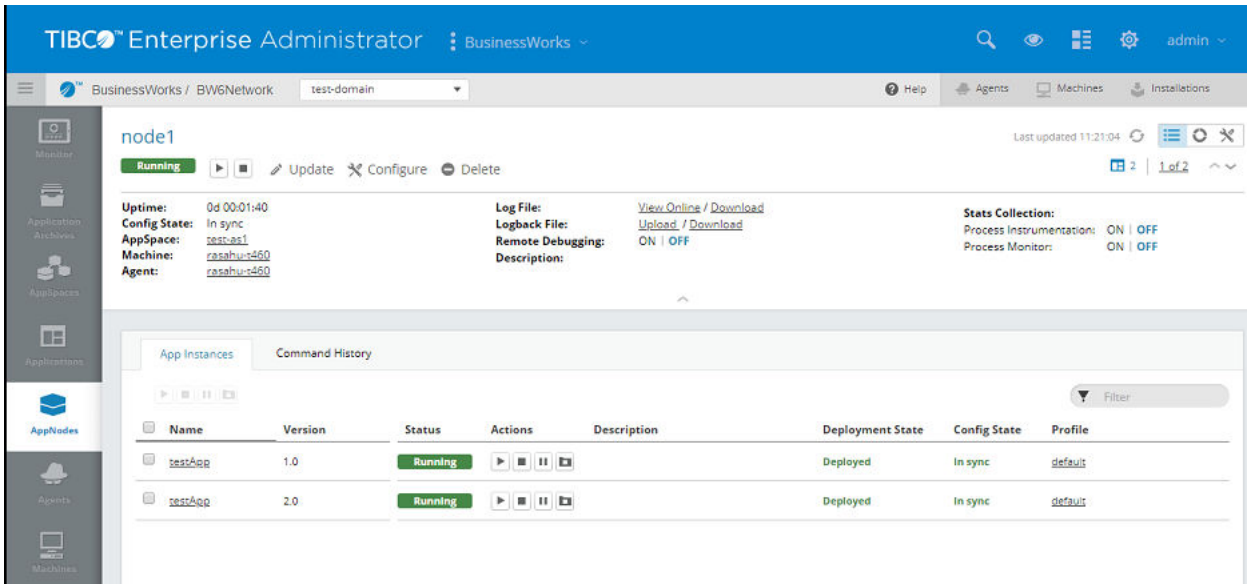


To view deployment history of the application archive, open the **Deployment History** tab. You can view the commands that were issued on an application archive, the execution status of the commands and the timestamp.



To view multiple versions of the same application, open the **Applications** page.





Downloading an Application Archive

You can download an application archive on your local system.
The following steps show how to download an application archive.

bwadmin Command Line

Procedure

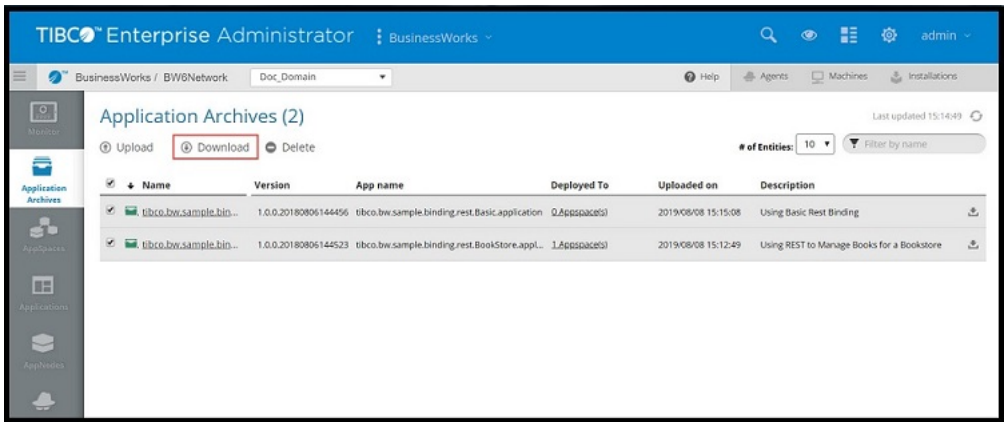
1. Execute the following command from the command line to download all application archive files in a specified domain. Note the use of forward slashes "/" for the Windows path.
   ```
   BW_HOME\bin>bwadmin download -d MyDomain -s C:/Tmp/Archives/
   ```
2. To download specific application archive file:
   ```
   BW_HOME\bin>bwadmin download -d MyDomain -s C:/Temp/Archives -a
   Application_Name.ear
   ```

   For more information about all supported options, use the following command:
   ```
   BW_HOME\bin>bwadmin download --help
   ```

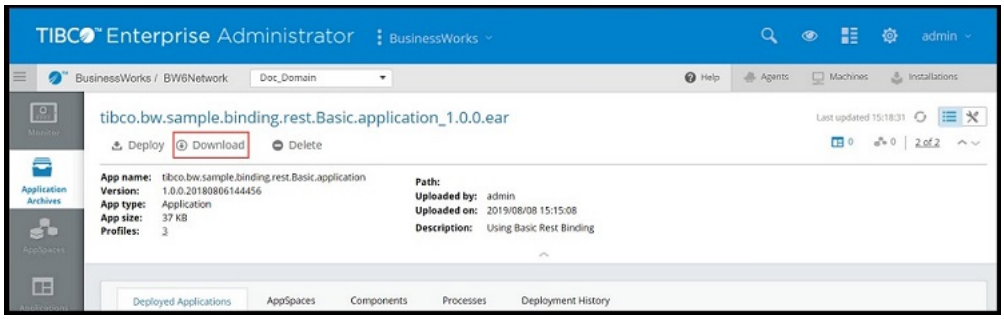
Admin UI

Procedure

1. Select the domain and open the **Application Archives** page.
2. Select one or more application archives and select **Download**.



Select any application archive and download it from Page 2.



Editing Application and Application Instance Properties

Application or application instance properties can be changed after deployment. You can then export a configuration and apply it to another application or application instance.

bwadmin Command Line

The profile for an application is located under the `META-INF` folder in the application's archive. The profile is the file with the extension `.substvar`. For details on how to generate more than the default profile for an application, see Creating an Application with Multiple Profiles.

When the **`config`** command is applied to an application, the profile changes are applied to all application instances by default. To apply a profile change to a specific application instance, use the **`config -appnode`** option to identify the specific AppNode.

If the archive contains the `WindowsProfile.substvar` file, use the following command to update the profile:

```
BW_HOME\bin>bwadmin config -d MyDomain -a MyAppSpace
-n MyAppNode -p WindowsProfile.substvar application
tibco.bw.sample.binding.rest.BookStore.application 1.0
```

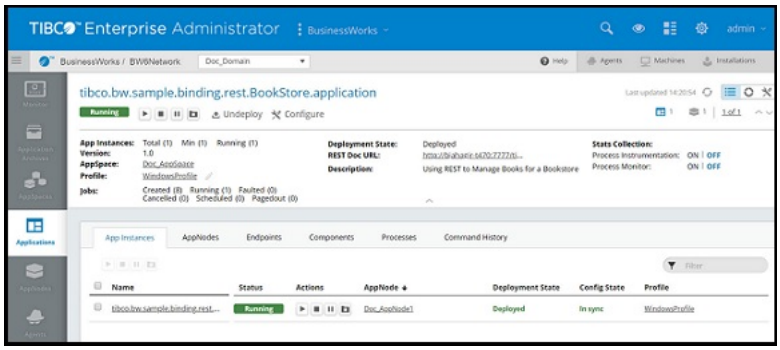
Admin UI

From the Admin UI you can change the profile for all instances of the application, or for a single application instance.

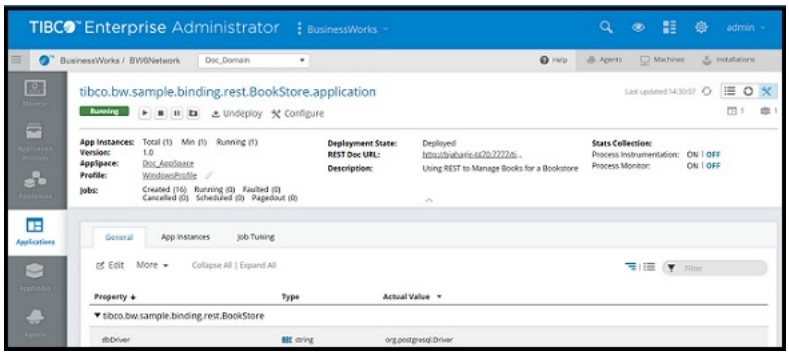
Procedure

1. Select the application you want to configure on the **Applications** page.

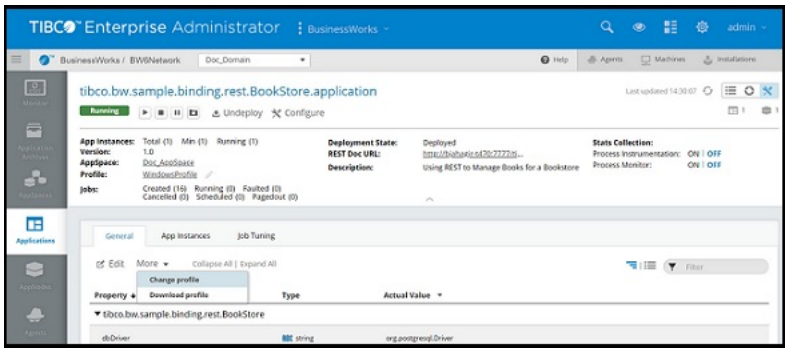
The **Application** page is opened, where you can view the application instances, the AppNodes each instance is deployed to, the deployment state, and the applied profile file.



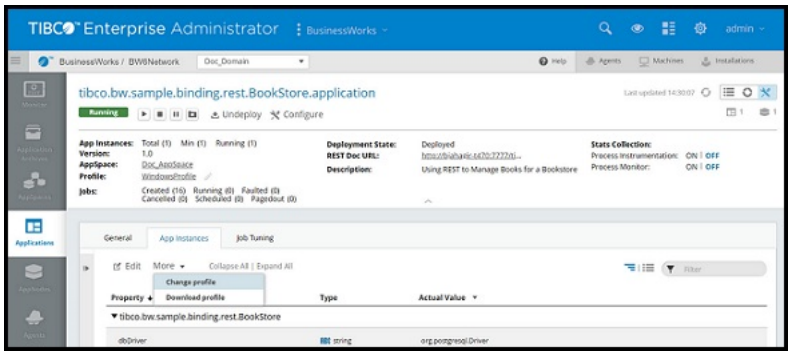
2. To edit application or application instance properties, click the **Edit** icon in the upper right of the **Applications** page.
   The **Application Properties** page is displayed.



3. Use the **General** tab to edit application properties and the **AppInstances** tab to edit application instance properties.
4. Click **Edit** on the **General** tab to edit application properties.
   a) Click **Submit** to save the property changes and apply them to the application.
   b) Click **More > Change Profile** to open the **Change Profile** dialog box where you can select a new profile to apply to the application or upload a new profile. Restart the application to apply the new profile.



5. Click **Edit** on the **AppInstances** tab to edit application instances properties. Select the instance (by AppNode)you want to edit.
   a) Click **Submit** to save the property changes and apply them to the application instance.
   b) Click **More > Change Profile** to open the **Change Profile** dialog box where you can select a new profile to apply to the application instance or upload a new profile. Restart the application to apply the new profile.



Exporting an Application Profile

An application profile can be exported from the application archive with the **`export`** command or from the Admin UI. After an application is configured with a profile, it becomes part of the application archive. An application configuration can be used to configure another application. If property changes are required after deployment, export the profile, edit, and deploy with the edited profile file.

bwadmin Command Line

Configurations are exported to the file system in the working directory.

The configuration is saved to: *`application_name_profile_name`*`.substvar`

To export a profile, execute the following command at the command line:

```
BW_HOME\bin>bwadmin export -d MyDomain -a MyAppSpace application
tibco.bw.sample.binding.rest.BookStore.application 1.0
```

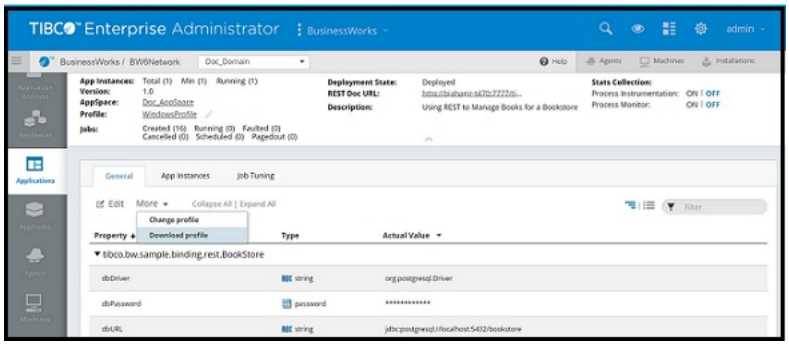
The application configuration file `tibco.bw.sample.binding.rest.BookStore.application_WindowsProfile.substvar` is written to the working directory.

Admin UI

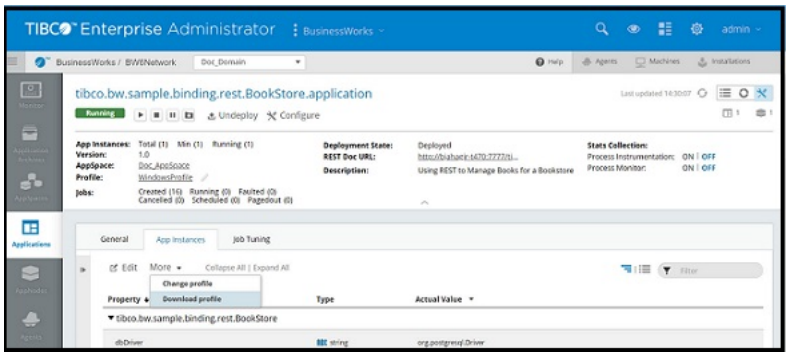
From the Admin UI you can export the profile for an application, or for a single application instance.

Procedure

1. Open the **Application** page and drill down into the application.
2. Click the **Edit** icon in the upper right of the page. The **Application Properties** page is displayed. The **General** tab displays application properties and the **AppInstances** tab displays application instance properties.
3. Click **More > Download Profile** on the **General** tab to export the application profile.



4. Click **More > Download Profile** on the **AppInstances** tab to export the application instance profile.



Starting an Application

To start an application after deployment, run the **`start`** command or click the **Start** icon in the Admin UI. If you stop an AppNode for a running application, the application state is persisted when you restart the AppNode.

bwadmin Command Line

When an application archive is deployed, the default action starts the application on each AppNode defined in the AppSpace. However, an archive file can be deployed with the **`-startondeploy`** option set to *false* so it is not started after deployment. Then, the **`start`** command can be used with the **`-appnode`** option to start the application on a specific AppNode.

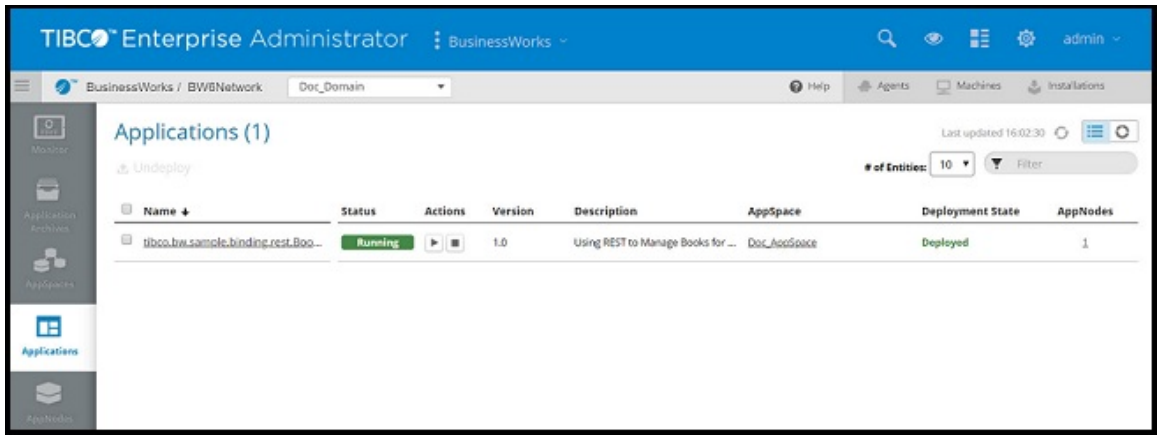
Procedure

1. Start the AppSpace.
2. Execute the **`start`** command for the application. For example:

```
BW_HOME\bin>bwadmin start -d MyDomain -a MyAppSpace -n MyAppNode application
tibco.bw.sample.binding.rest.BookStore.application 1.0
```

Admin UI

To start the application, click the **Start** icon on the **Applications** page. The AppSpace and AppNodes must be running.



Viewing Running Applications

Use the bwadmin **show** command to verify a running application, or view the application in the Admin UI.

bwadmin Command Line

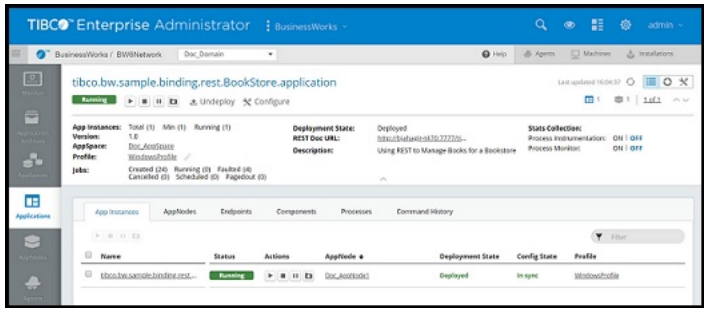
Execute the **show** command to see application and configuration status.

```
BW_HOME\bin>bwadmin show -domain MyDomain -appspace MyAppSpace applications
```

Admin UI

View the application's status on the **Application** page. (To open this page, drill into the application from the **Applications** page.) The Admin UI displays the following information:

- Total number of application instances, the minimum number of instances (AppNodes), and running number of instances.
- Application version.
- AppSpace
- Number of created jobs, running jobs, and faulted jobs.
- The applied profile.
- The deployment state.
- The REST Doc URL for applications using REST services. Click the link to open the REST UI page where you can test out operations. (The application must be running.)
- The application description.



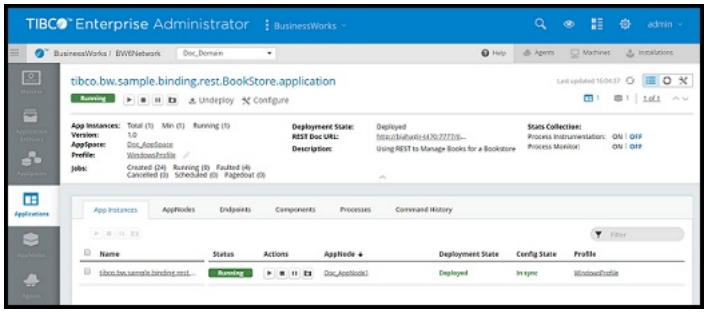
Viewing Endpoints, Components, Processes and Command History

You can view endpoints, components, processes and the command history for a running application from the Admin UI.

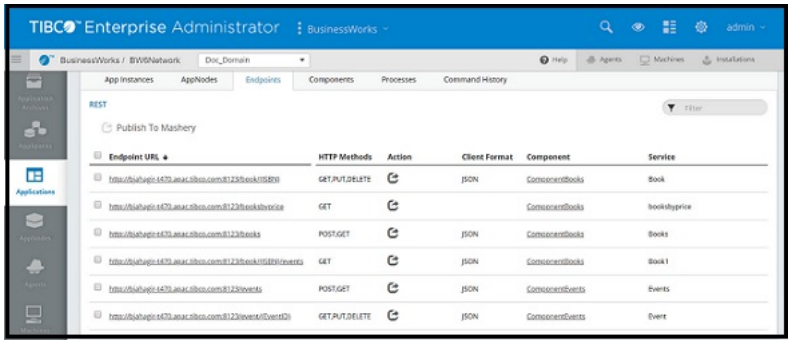
Admin UI

Procedure

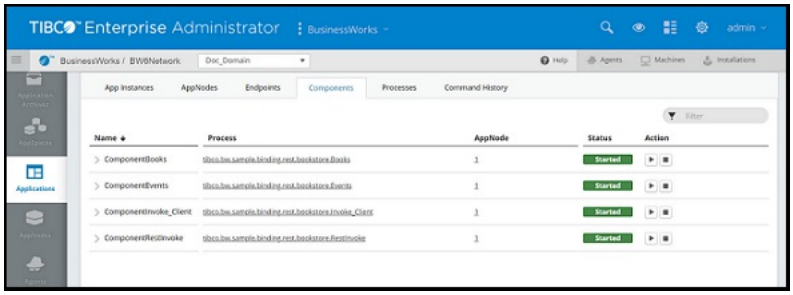
1. Select the running application you want to view details for configure on the **Applications** page.



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



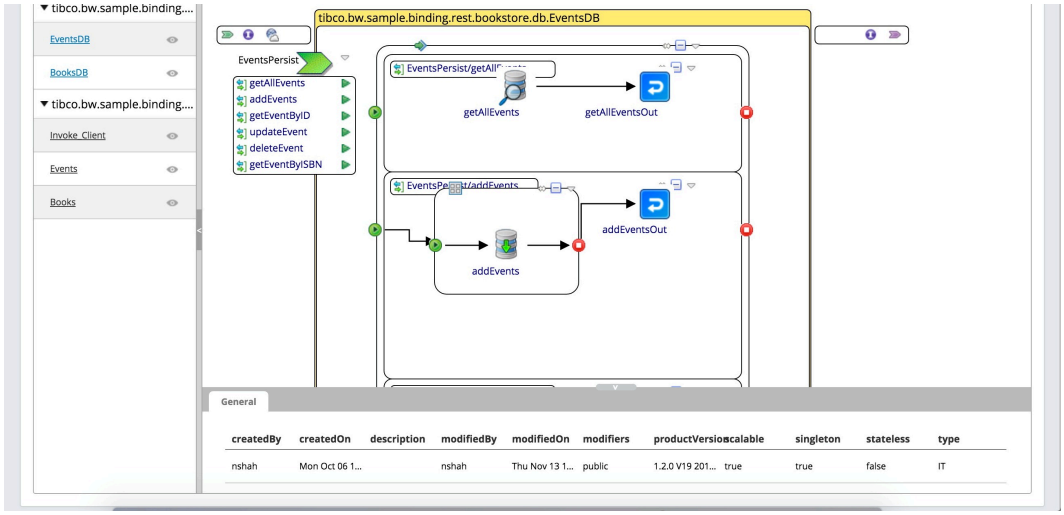
3. Open the **Components** tab to view components in the application.



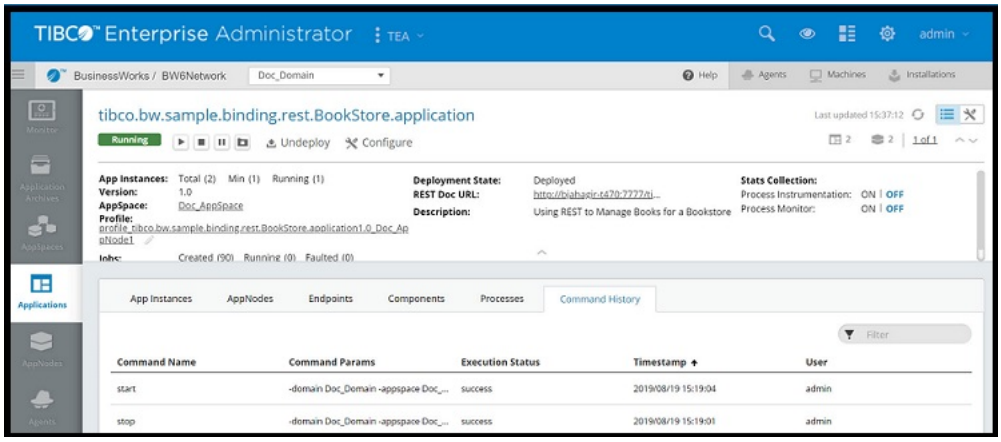
4. Open the **Processes** tab to view application processes.

   If the application archive file was generated using TIBCO ActiveMatrix BusinessWorks 6.2.x or higher, you can view expand a process and click it to view the SVG process diagram.

   When viewing process diagrams through the Admin UI, the optimal resolution is 1920 x 1080 pixels or higher.



5. Open the **Command History** tab to view the commands or operations that were performed on an Application.



Configuring a Unified Doc URL

You can configure a unified documentation URL for all applications that use REST services that are running in a single AppSpace. Alternatively, the documentation URL can be configured for each AppNode in an AppSpace.

Documentation endpoint configuration properties can be specified at the AppSpace or the AppNode level. The details specified at the AppSpace level apply to all applications running on all AppNodes within the AppSpace. Properties set at the AppNode level only apply to applications running on that AppNode.

Properties are configured in the AppSpace or AppNode configuration file or configuration template file.

Procedure

1. To set documentation endpoint properties at the AppSpace level:
   a) Copy the existing AppSpace configuration template file `appspace_config.ini_template` (located in *`BW_HOME`*`/config/`) to a temporary location.
   b) Uncommment and configure the following properties in the **BW REST Swagger Configuration** section of the file:

```
#
--------------------------------------------------------------------------------
# Section:  BW REST Swagger Configuration.  The properties in this section
```

```
# are applicable to the Swagger framework that is utilized by the BW REST
# Binding.
#
# Note: There are additional BW REST Swagger configuration properties that
# can be specified in the BW AppNode configuration file "config.ini".  Refer to
# the BW AppNode configuration file's section "BW REST Swagger configuration"
# for details.
#
--------------------------------------------------------------------------------
# Swagger framework reverse proxy host name.  This property is optional and
# it specifies the reverse proxy host name on which Swagger framework serves
# the API's, documentation  endpoint, api-docs, etc..
#bw.rest.docApi.reverseProxy.hostName=localhost

# Swagger framework port.  This property is optional and it specifies the
# reverse proxy port on which Swagger framework serves the API's, documentation
# endpoint, api-docs, etc.
#bw.rest.docApi.reverseProxy.port=0000
```

2. To set documentation endpoint properties at the AppNode level:
   a) Copy the existing AppNode `config.ini` file (located in the root of the AppNode folder), or the `appnode_config.ini_template` (located in *BW_HOME*`/config/`) file, to a temporary location.
   b) Configure the following properties in the **BW REST Swagger Configuration** section of the file (note that the port property is uncommented by default):

```
#
--------------------------------------------------------------------------------
# Section:  BW REST Swagger configuration.  The properties in this section
# are applicable to the Swagger framework that is utilized by the BW REST
# Binding.
#
# Note: There are additional BW REST Swagger configuration properties that
# can be specified in the BW AppSpace configuration file "config.ini".
# Refer to the BW AppSpace configuration file's section
# "BW REST Swagger configuration" for details.
#
--------------------------------------------------------------------------------
# Swagger framework host name.  This property is optional and it specifies the
# host name on which Swagger framework serves the API's, documentation
endpoint,
# api-docs, etc..  The default value is the host name on which the BW AppNode
# is executed.
#bw.rest.docApi.hostName=localhost

# Swagger framework port.  This property is required and it specifies the port
# on which Swagger framework serves the API's, documentation endpoint,
# api-docs, etc..
bw.rest.docApi.port=7777
```

3. Use one of the following **`config`** admin commands to push the configuration to the AppSpace or the AppNode:

   - AppSpace:

```
bwadmin[admin] > config -d myDomain -a myAppSpace -cf
                 <temporaryLocation>/config.ini
```

   - AppNode:

```
bwadmin[admin]> config –d myDomain –a myAppSpace –n myAppNode –cf
<temporaryLocation>/config.ini
```

Result

Documentation for all applications in the AppSpace that use REST services is available at the given URL. You can open the documentation URL by clicking the REST Doc URL link for the running application in the Admin UI or by opening the URL at the specified host name and port. The application must contain REST services and must be running. If you configured the documentation URL just for the AppNode, the documentation for applications running in the specified AppSpace will be available at the given URL.

Stopping an Application

To stop a running application after deployment, run the **`stop`** command or click the **Stop** icon in the Admin UI. Applications or application instances can be stopped. Stop an application before undeploying.

bwadmin Command Line

Execute the following command at the command line to stop an application:

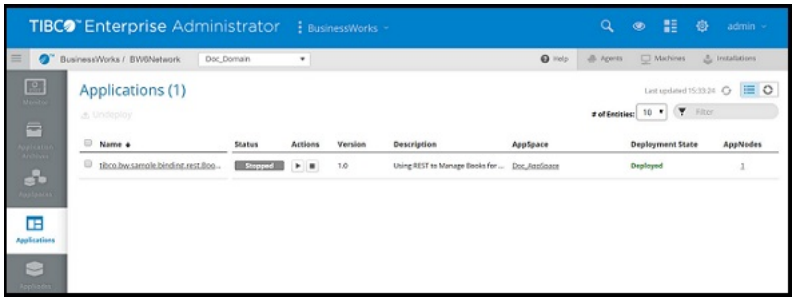
```
BW_HOME\bin>bwadmin stop -d MyDomain -a MyAppSpace application
tibco.bw.sample.binding.rest.BookStore.application 1.0
```

To stop an application instance, stop the AppNode the instance is running in.

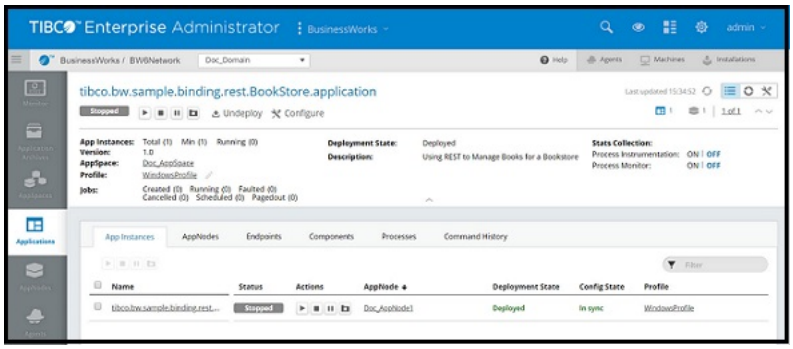
Admin UI

Procedure

1. To stop an application, click the **Stop** icon ■ for the application on the **Applications** page. The status of the application instance will change from Running to Stopping, a transient state, then Stopped:



2. To stop an application instance, click the **Stop** icon ■ for the instance on the **App Instances** tab. The status of the application instance will change from Running to Stopping, a transient state, then Stopped:



Undeploying an Application

Undeploying an application removes the deployed application from the AppSpace.

bwadmin Command Line

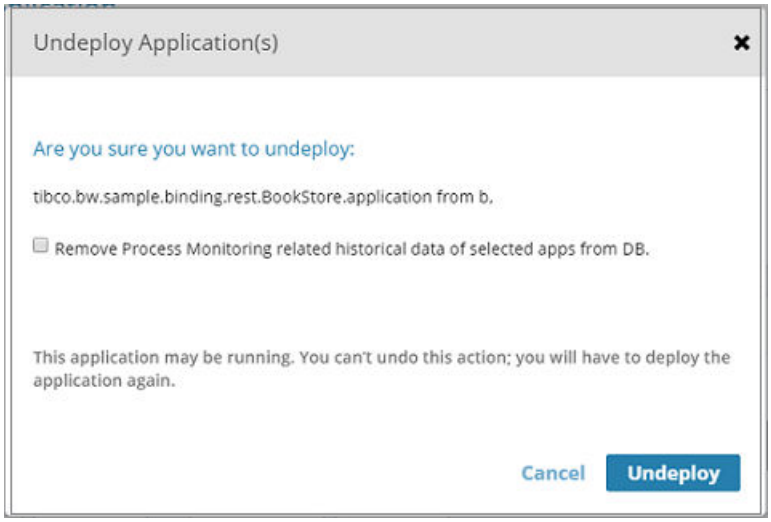
Execute the following command to undeploy:

```
BW_HOME\bin>bwadmin undeploy -d MyDomain -a MyAppSpace application
tibco.bw.sample.binding.rest.BookStore.application 1.0
```

Admin UI

Procedure

1. On the **Applications** page, select the application check box and click **Undeploy**.
2. The **Undeploy Application(s)** dialog box is displayed. To undeploy all process monitoring historical data from the database, select the **Remove Process Monitoring related historical data of selected apps from DB** check box. Click the **Undeploy** button to undeploy the application. A running application can be undeployed.



Starting a component in an Application

bwdmin Command Line

To start a component in an application, run the **`startcomponent`** command or click the **Stop** icon in the Admin UI.

When an application is started, the default action starts all the components of the application on each AppNode, defined in the AppSpace. However, a component can be stopped using the **`stopcomponent`** command. The **`startcomponent`** command can then be used with the **`-appnode`** option to start a component of an application on a specific AppNode.

Procedure

1. Start the AppSpace.

2. Execute the **`startcomponent`** command for the component of an application. For example:

```
BW_HOME\bin>bwadmin startcomponent -d MyDomain -a MyAppSpace -n MyAppNode
MycomponentName tibco.bw.sample.MyApp.application 1.0
```

To enable auto start of a component with process starter activity, use **`enablecomponentautostart`** command. For example:

```
BW_HOME\bin>bwadmin enablecomponentautostart -d MyDomain -a MyAppSpace -
n MyAppNode ComponentReceiver jmsSenderReceiver.application 1.0
```

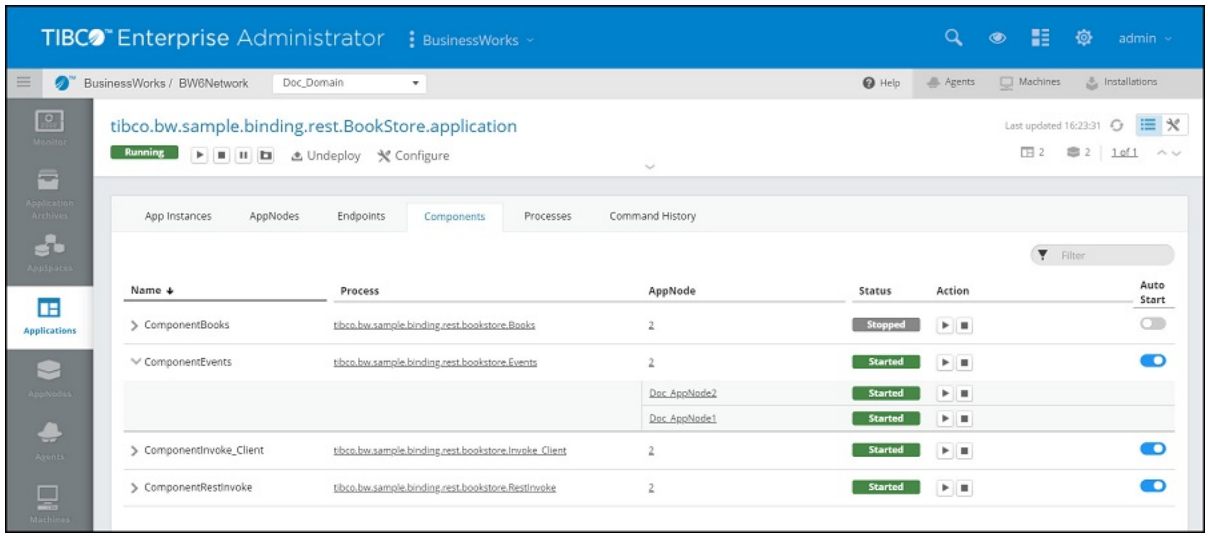
This functionality is applicable at AppSpace level and not at AppNode level.

Admin UI

When an application is started, the default action starts all the components of the application on each AppNode defined in the AppSpace. Click the **Components** tab to view the components in the application. Click the **Start** icon ▶ to start all the components in the Appnodes. If all the components are running, the **Status** changes to started. The status bar displays the number of components running on the AppNodes. On hovering over the status bar, you can see the number of running components. To start a component on a specific AppNode, click the › icon on the left to view the list of AppNodes the component is running on. Click the **Start** icon ▶ against the component you want to start.

The › icon to collapse the list of AppNodes the component is running on, is enabled only after the application has started.

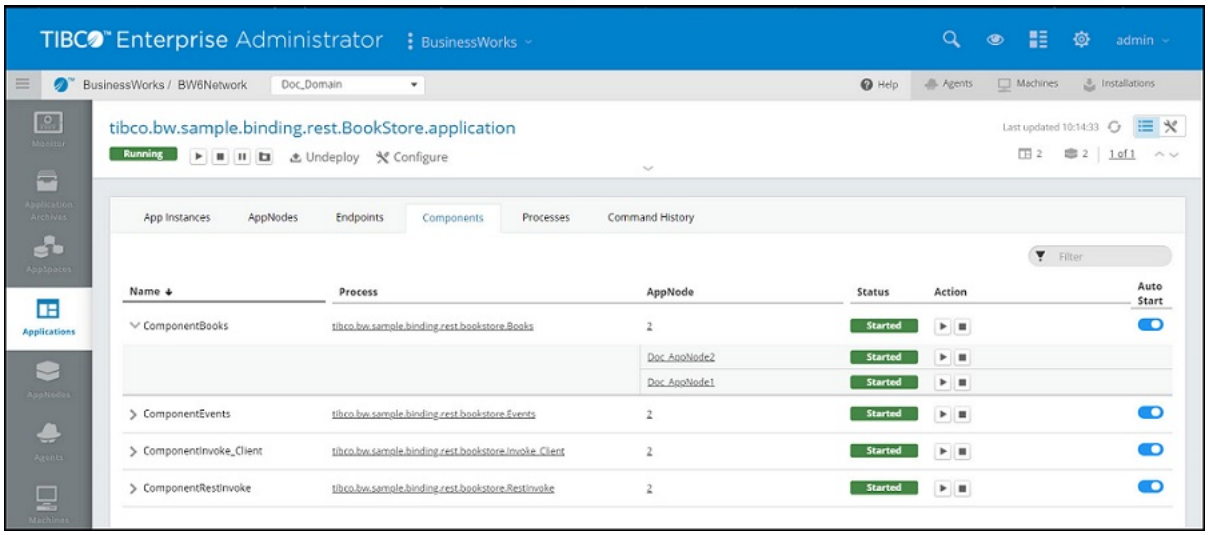
Starting the top level **Start** icon ▶ starts all the components with auto start toggle button on for all the AppNodes.



To enable auto start of a component:

Procedure

1. Select Application Level 2 page and select the **Component** tab.
2. Use the toggle button in the **Auto Start** column for each component process to decide whether to enable auto start of a component process or not during execution.



3. Restart an application to take effect the changes.

Stopping a component in an Application

To stop a component of an application , run the **`stopcomponent`** command or click **Stop** icon in the Admin UI.

bwdmin Command Line

When an application is started, the default action starts all the components of the application on each AppNode, defined in the AppSpace. However, the **`stopcomponent`** command can be used with the **`-appnode`** option to stop a component of the application on a specific AppNode.

Procedure

1. Start the AppSpace.
2. Execute the **`stopcomponent`** command for the component of an application. For example:

```
BW_HOME\bin>bwadmin stopcomponent -d MyDomain -a MyAppSpace -n MyAppNode
MycomponentName tibco.bw.sample.MyApp.application 1.0
```

To disable auto start of a component with process starter activity, use the **`disablecomponentautostart`** command. For example:

```
BW_HOME\bin>bwadmin disablecomponentautostart -d D -a AS -n AN
ComponentReceiver jmsSenderReceiver.application 1.0
```

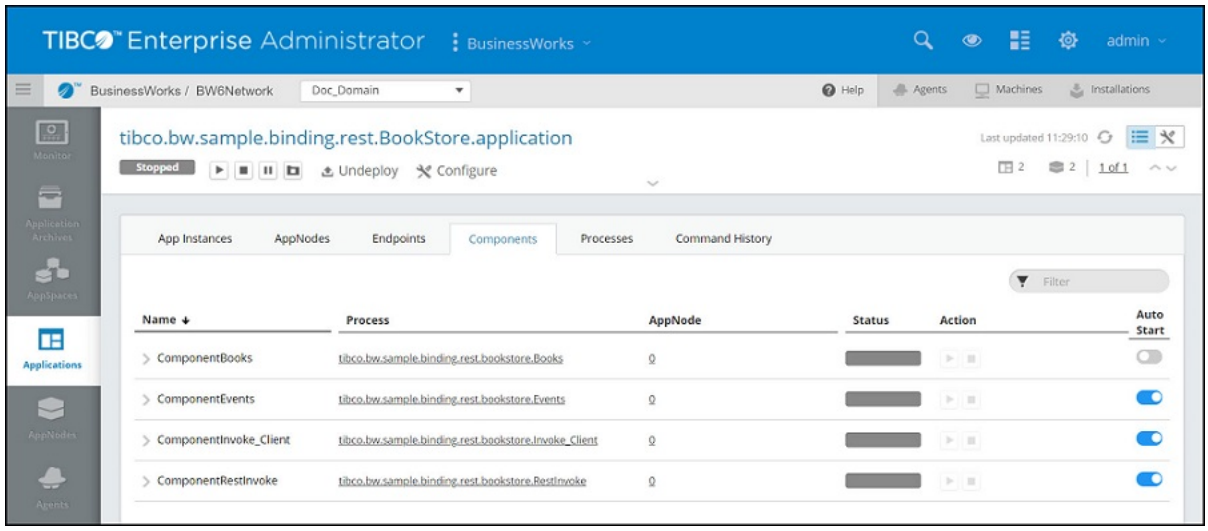
This functionality is applicable at AppSpace level and not at AppNode level.

Admin UI

When an application is started, the default action starts all the components of an application on each AppNode defined in the AppSpace. Click the **Components** tab to view the components in the application, and select the running application you want to stop the components for. To stop a component on a specific AppNode, click the > icon on the left to view the list of AppNodes the component is running on, and click the **Stop** icon ■ against the component you want to stop. The status bar displays the number of components that have stopped running on the AppNodes. On hovering over the status bar, in the primary row table, you can see the number of components that are running and the number of components that have stopped running on the Appnode.

The collapse icon > is enabled only after an application has started.

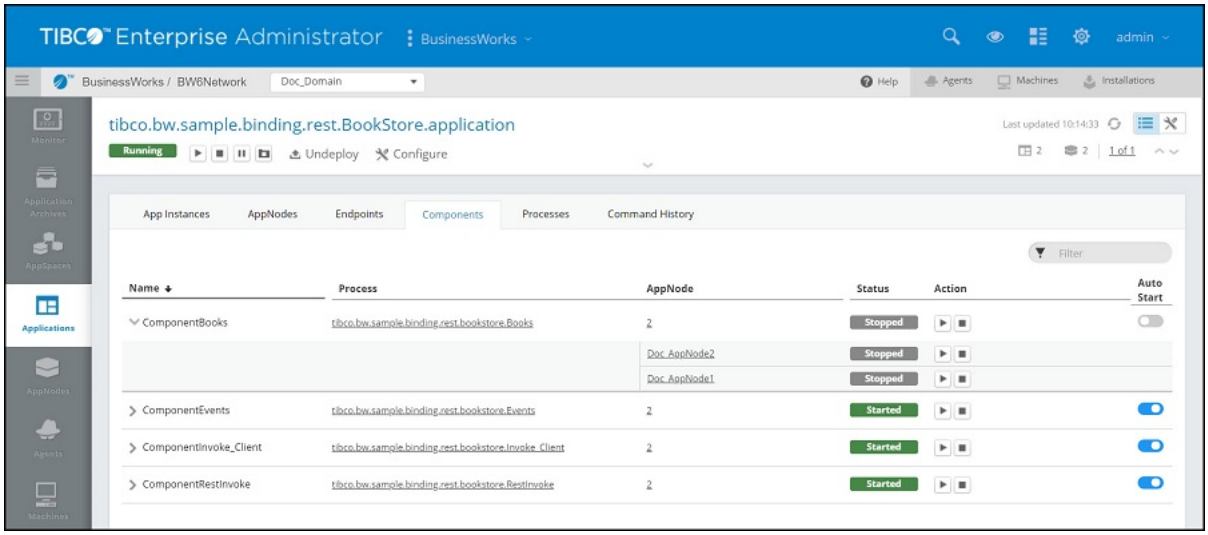
Stopping the top level **Stop** icon ■ stops all the running components on all the AppNodes.



To disable auto start of a component:

Procedure

1. Select Application Level 2 page and select the **Component** tab.
2. Use the toggle button in the **Auto Start** column for each component process to decide whether to enable auto start of a component process or not during execution.



3. Restart an application to take effect the changes.

Retrieving list of components in an Application

To get the list of components of an application , run the **`getcomponents`** command or click the **Components** tab in the Admin UI.

bwdmin Command Line

Procedure

1. Start the AppSpace.
2. Execute the **`getcomponents`** command for an application. For example:

```
BW_HOME\bin>bwadmin getcomponents -d MyDomain -a MyAppSpace -n MyAppNode
tibco.bw.sample.MyApp.application 1.0
```

Admin UI

On the **Applications** page, select the running application you want to view the components for. Open the **Components** tab to view the components in the application.

Retrieving details of a component in an Application

To get the details of a particular component of an application , run the **`getcomponentdetail`** command or view the details under the **Status** label, in the Admin UI.

bwdmin Command Line

Procedure

1. Start the AppSpace.
2. Execute the **`getcomponentdetail`** command for a component of an application. For example:

```
BW_HOME\bin>bwadmin getcomponentdetail -d MyDomain -a MyAppSpace -n MyAppNode
MycomponentName tibco.bw.sample.MyApp.application 1.0
```

Admin UI

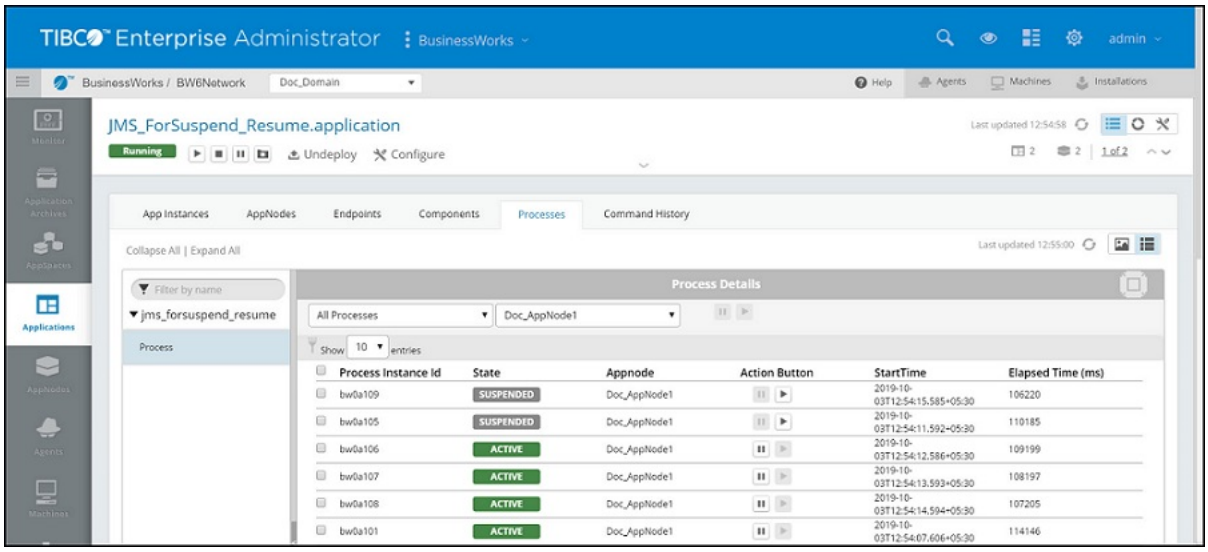
On the **Applications** page, the **Status** label displays the details of the component.

Suspending and Resuming Process Instances

Admin UI

Procedure

1. On the Application level 2 page select **Processes** tab.
   Process Details view is opened.
2. To perform bulk operation on process instances, use filter based on process state (Active Processes, Suspended processes, All processes) as well as for AppNodes.



3. Use Select Columns filter to display additional columns such as Parent Process Name, Parent Process Instance Id, Main Process Name.

Backing Up and Restoring an Application

Backing up an application exports the current state of the specified application to a bwadmin command file. The command file can be provided to bwadmin to recreate the application state. Output can be compressed to a ZIP file with the **`-zipped`** option.

The default application profile is backed up with the application. Additional `*.substvar` files that contain application configurations can be created in the output folder.

A specific application version can be backed up by providing the **`version`** argument. If this argument is not specified, all application versions are backed up.

Procedure

1. To back up the current state of an application,
   a) Open a terminal and navigate to *BW_HOME*`\bin`.
   b) Enter the **`backup`** command at the command line, using **`-s`** option to identify the name of the destination file. Use the **`-domain`** and **`appspace`** options with the **`application`** argument. Provide the application version number to back up a specific version. By default, destination files are written to the current working directory.
   This example backs the current state of an application to a command file named `app_backup.cmd`.

```
BW_HOME\bin>bwadmin backup -s app_backup.cmd -d Machine2Domain -a AS1
application acme.acct.ap.application 1.0
```

2. To restore the application,
   a) Open a terminal and navigate to `BW_HOME\bin`.
   b) Enter the **`bwadmin`** command, providing the name of the backup command file. The following example recreates the state of the application *acme.acct.ap.application*. Because the application was deployed on backup, it is restored to the deployed state.

```
BW_HOME\bin>bwadmin -f app_backup.cmd
```

   If you are restoring to a different location, you need to update the command file as follows:

   - The agent name will point to **`localhost`** by default; you need to change this to the name of the machine you are restoring to.
   - Update the domain home to point to the absolute path to the new location.

- Update the path to the application archive (EAR) file to an absolute path.

c) Use the **`show applications`** command with the **`-domain`** and **`-appspace`** options to verify the restore. If the bwagent is not running, its status will be listed as Unreachable.

Restoring the File System of an Archive

Restoring an archive restores the file system of the specified archive to the state of the datastore.

Prerequisites

- The bwagent must be running.

Wildcards can be used to restore archives if archive name(s) are not known.

Procedure

1. To restore the file system for an archive, open a terminal and navigate to `BW_HOME\bin`.
2. Enter the **`restore`** command, with the **`-domain`**, **`-appspace`**, and **`appnode`** options. Provide the **`archive`** argument specifying the name of the archive to restore or wildcards to restore all archives uploaded to domains in this agent.

```
BW_HOME\bin>bwadmin restore -d Machine1Domain -a AS1 -n AN1 archive *.*
```

3. To verify the restore, check the file system. Open the *`BW_HOME`*`\domains` folder. Browse the folder and look for the archive in the *`BW_HOME`*`\domains\`*`domain_name`*`\archives` folder.

Restoring the File System of an Application

Restoring an application restores the file system of the specified application to the state of the datastore.

Prerequisites

- The bwagent must be running.

Wildcards can be used to restore applications if application name(s) are not known.

Procedure

1. To restore the file system of an application to the state of the datastore, open a terminal and navigate to *BW_HOME*`\bin`.
2. Enter the **`restore`** command at the command line, with the **`-domain`**, **`-appspace`**, and **`-appnode`** options. Provide the **`application`** argument with the name of the application or wildcards to restore all applications for this domain.

```
BW_HOME\bin>bwadmin restore -d Machine1Domain -a AS1 -n AN1 application *.*
```

3. To verify the restore, check the file system. Open the *`BW_HOME`*`\domains` folder. Look for the application in the *`BW_HOME`*`\domains\`*`domain_name`*`\appspaces\`*`appspace_name`*`\apps` folder.

Publishing APIs to TIBCO Mashery®

Follow these steps to publish application endpoints, or API endpoints, from the Admin UI to Mashery.

Prerequisites

Complete the following tasks:

- Set up a Mashery account.

- Register your application endpoint domain with Mashery, and ensure Mashery can access it.
- Create the mashery.ini file:

  1. Open a text editor, and add the following properties with the correct values.

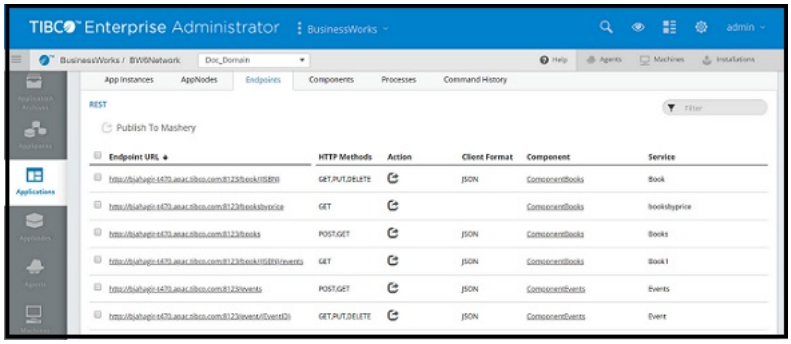
     ```
     bw.mashery.clientId=<clientId>
     bw.mashery.clientSecret=<clientSecret>
     bw.mashery.areaUuid=<areaUuid>
     bw.mashery.username=<username>
     bw.mashery.password=<password>
     bw.mashery.trafficManagerDomain=<trafficManagerDomain>
     bw.mashery.apiUrl=<https://api.HOST>
     ```

  2. Save the file as mashery.ini
  3. Add the mashery.ini file to *BW_HOME*`/config`.
- Edit the bwagent.ini file at *BW_HOME*`/config/bwagent.ini`, by adding the following line to the file:

  `bw.mashery.config.file=../config/mashery.ini`

Procedure

1. Start the application from the Admin UI.
2. Click on the **Application** tab, and click the **Endpoints** tab.
3. From the **Endpoints** tab, select all, or individual, application endpoints.



Select the **Select All** check box, located to the left of Endpoint URL, to select all application endpoints.

4. Click the Publish To Mashery icon, or the icon, to publish the selected endpoints.

Result

Your API endpoints are published to Mashery.

For additional details about managing your API's from Mashery, see Mashery documentation.

Backing Up and Restoring from the Backup

The **backup** command backs up a specified runtime entity to a command file. You provide the command file as input to bwadmin to recreate the environment.

The following runtime entities can be backed up and restored from a command file:

- Domain
- AppSpace
- AppNode
- Application

Runtime entities can be local or part of an agent network. Run this command frequently on all runtime entities, so that you always have a backup of your environment. Runtime entities created in local mode can only be restored when bwadmin is in local mode. Runtime entities created in enterprise mode can only be restored when the bwagent is running.

If you provide the same command file name on a subsequent backup, the existing command file is overwritten. Output can be compressed to a ZIP file with the **-zipped** option.

The bwadmin **backup** command and the bwadmin **restore** command are not complimentary. The **restore** command requires a bwagent and restores the file system to the state of the datastore. For information on the **restore** command, see Restoring the File System of Runtime Entities.

For more information about backing up and restoring from the backup, see:

- Backing Up and Restoring a Domain
- Backing Up and Restoring an Application
- Backing Up and Restoring an AppNode
- Backing Up and Restoring an AppSpace

Restoring the File System of Runtime Entities

The **restore** command restores the file system of a bwagent or a runtime entity to the state of the datastore. If a machines goes offline and cannot be restarted, that environment can be recreated given the name of the bwagent or the runtime entities.

The **restore** command can restore a bwagent or a specified domain, AppSpace, AppNode, archive, or application. The command is only available when the bwagent is running.

- When a bwagent is restored, the file system for all runtime entities in the bwagent datastore are restored.
- When a domain is restored, the file system for the specified domain and all contained runtime entities are restored.
- When an AppSpace is restored, the containing domain must exist. The file system for the specified AppSpace and all contained runtime entities are restored.

This pattern also applies to AppNodes, archives, and applications.

The bwadmin **backup** command and the bwadmin **restore** command are not complimentary. The **backup** command backs up a runtime entity to a command file. For information on the **backup** command, see Backing Up and Restoring from the Backup.

For more information about restoring the file system, see:

- Restoring the File System of a bwagent
- Restoring the File System of a Domain
- Restoring the File System of an AppSpace
- Restoring the File System of an AppNode
- Restoring the File System of an Archive
- Restoring the File System of an Application

Debugging

In the event of an error, the software provides detailed messages that can help you trace through to the cause of an issue. You can adjust logging levels as needed to capture different granularity of messages for different loggers.

Messages returned by TIBCO ActiveMatrix BusinessWorks™ are categorized by component and by error code within component. The following table shows the components of the software that return messages:

Component ID	Description
BX	ActiveMatrix BusinessWorks™ Engine Layer
PVM	ActiveMatrix BusinessWorks Engine Layer
TIBCO-BW-ADMIN	ActiveMatrix BusinessWorks Administrator
TIBCO-BW-ADMIN-CLI	ActiveMatrix BusinessWorks Administrator Command Line Interface
TIBCO-BW-BINDING-REST	ActiveMatrix BusinessWorks REST Binding
TIBCO-BW-BINDING-SOAP	ActiveMatrix BusinessWorks SOAP Binding
TIBCO-BW-CORE	ActiveMatrix BusinessWorks Engine Layer
TIBCO-BW-FRWK	ActiveMatrix BusinessWorks Framework
TIBCO-BW-PALETTE	ActiveMatrix BusinessWorks Palette Layer
TIBCO-BW-PALETTE-*<PaletteName>*	ActiveMatrix BusinessWorks Palette specific activity implementation
TIBCO-BW-SR	ActiveMatrix BusinessWorks Shared Resource API Layer
TIBCO-BW-SR-*<UniqueName>*	ActiveMatrix BusinessWorks specific Shared Resource implementation
TIBCO-BW-STATS	ActiveMatrix BusinessWorks Stats Collector
TIBCO-THOR-FRWK	Thor Framework

The Engine layers with component IDs BX and PVM do not follow this convention.

The following tables identifies the log levels used by the software and the corresponding error code range:

Log Level	Error Code
TRACE	100001 - 109999
DEBUG	200001 - 209999
INFO	300001 - 309999
WARN	400001 - 409999
ERROR	Errors can be indicated by one of the following error code ranges: • 500001 - 509999 • 600001 and higher • 0xxxxx  • Error codes 600001 and higher indicate exceptions in the execution and are always associated with a 5xxxxx error code, which can be traced in the log file. • Error codes starting with 0xxxxx indicate internal errors. Contact TIBCO Support for possible resolution or a workaround.

If you encounter an error that does not start with a 0, change the logging level of the AppNode and bwagent loggers to DEBUG. Try to recreate the scenario and examine the log file to try and trace the issue. The error messages are detailed and can help you understand the chain of events that led up to the issue. For more information on log files and log file configuration see the section on Logging in the *TIBCO ActiveMatrix BusinessWorks™ Administration* guide.

The following sections in the *TIBCO ActiveMatrix BusinessWorks™ Concepts* guide list issues you might encounter, along with possible resolutions:

- Troubleshooting bwagent Issues
- Troubleshooting Runtime Entities Issues
- Troubleshooting Archive Issues
- Troubleshooting Application Issues
- Troubleshooting TIBCO Enterprise Administrator Integration Issues

Troubleshooting bwagent Issues

Some bwagent issues and possible resolutions are listed below. This list is not complete but provides examples of messages that might be returned.

For a complete list of error codes, see the *TIBCO ActiveMatrix BusinessWorks™ Error Codes* guide.

Issue	Message	Resolution
When registering a bwagent to a domain or AppSpace, or unregistering the AppSpace the bwagent cannot be registered.	TIBCO-BW-ADMIN-CLI-500132: Failed to unregister BW Agent [*bwagent* from the AppSpace [*AppSpace*]. <CausedBy> TIBCO-BW-ADMIN-500004: Error invoking [*methodName*] method on the agent [*Agent*], The BW Agent [*Agent*] on the remote machine is not running.	Verify the bwagent name. Check that the bwagent on the remote machine is running.
If you are unable to enable or disable the autoregistration feature from the command line after executing the **`enableautoregistration`** utility or the **`disableautoregistration`** utility.	TIBCO-BW-AGENT-500004: Error invoking [method] method on the agent [bwagent]. The BW Agent [bwagent] on the remote machine is not running.	Check if bwagent is running, and if it is, verify its name.
The bwagent could not start because the mode is not set to enterprise.	TIBCO-BW-AGENT-500006: Cannot start the agent. The admin mode in bwagent.ini is not configured for enterprise mode. Check your configuration and restart.	Open a terminal and type the following command at the command line: `bwadmin mode enterprise` Restart the bwagent.
A bwadmin command could not be completed.	TIBCO-BW-ADMIN-500008: Error in initializing data manager, TIBCO-BW-ADMIN-PRSTNC-500001: Connection to BW Agent failed. or TIBCO-BW-ADMIN-CLI-500006: Failed to initialize transport, TIBCO-BW-ADMIN-PRSTNC-500001: Connection to BW Agent failed.	The first message indicates that the bwagent is not running. Start the bwagent. The second message is displayed when bwadmin is configured for enterprise mode and the command could not be completed due to a failed bwagent connection.

Issue	Message	Resolution
The bwagent could not start due to an error with datastore initialization.	TIBCO-BW-AGENT-500009: Failed to start agent due to an error in initializing data store, reason: *Reason*	The datastore is written to the *BW_HOME*`\domains\.datastore` folder. Verify the following: * The specified folder exists. * The **`bw.agent.technology.as.dataStoreLocation`** property in the *BW_HOME*`\config\bwagent.ini` file points to the datastore folder name. * The **`bw.agent.technology.as.role`** property is set to `server`. You might also see this message if there is an issue with the **`bw.agent.technology.as.dataStoreLocation`** property setting in the *BW_HOME*`\config\bwagent.ini` file. Make sure that the TCP protocol is specified only for the first URL in the string; not for subsequent URLs.
When starting a bwagent that is configured as part of an agent network, the URL of another agent in the network could not be found.	unable to resolve network specification ('*bwagent*')	Check the setting of the **`bw.agent.technology.as.dataStoreLocation`** property in the *BW_HOME*`\config\bwagent.ini` file. Verify that: • The URL is specified as either IP address and port or host name and port, in the format: *IP_address*/*hostname*:*port* • There are no typos in the URL or port number. • A semicolon separator is used between URLs.
When starting a bwagent that is configured as part of an agent network, the bwagent starts but does not indicate that the agent has joined the network.	N/A	Check that the setting of the **`bw.agent.network.name`** property in the *BW_HOME*`\config\bwagent.ini` file is the same as the setting in other the configuration file for other bwagents in the network.

Issue	Message	Resolution
When starting a bwagent that is configured as part of a bwagent network, and the bwagent is not configured for the network, warnings are displayed.	There are [2] agents in the BW Agent group that store data. However, the property "minSeederCount" in the bwagent.ini file is set to [1], refer to the bwagent.ini file or the documentation and choose an appropriate value. There are [2] agents in the BW Agent group that store data. However, the property "quorumSize" in the bwagent.ini file is set to [1], refer to the bwagent.ini file or the documentation and choose an appropriate value.	See comments in the *BW_HOME* `\config\bwagent.ini` file for information.
The TEA agent could not be registered.	TIBCO-BW-ADMIN-500504: Failed to register TEA Agent [*teaagent*] with TEA server [http://*host*: *port*], TEA Agent registration failed, TIBCO-BW-TEAAGENT-500300: Failed to register BW TEA agent [*teaagent*] with TEA server, <CausedBy> Unable to register agent with name [*teaagent*]'	Verify the URL to the TEA agent. Check that the TEA agent is running.

Troubleshooting Runtime Entity Issues

Some runtime entity issues and possible resolutions are listed below. This list is not complete but provides examples of messages that might be returned.

For a complete list of error codes, see the *TIBCO ActiveMatrix BusinessWorks™ Error Codes* guide.

Issue	Message	Resolution
The specified runtime entity could not be created as it contains invalid characters or contains over 100 characters.	TIBCO-BW-ADMIN-CLI-500501: Name contains invalid characters and does not comply with naming conventions. Valid characters are upper and lower case characters of the alphabet as well as digits, '.' and '-'. or TIBCO-BW-ADMIN-CLI-500502: Name length exceeds 100 characters and it will be truncated to satisfy the length limit	Create the runtime entity with valid characters: • `A-Z` • `a-z` • `0-9` • `-` (hyphen) • _ (underscore) Illegal characters are stripped from the name. The maximum length of a runtime entity name is 100 characters. If the maximum length is exceeded, the entity name is shortened to 100 characters.
The specified domain could not be created; it already exists.	TIBCO-BW-ADMIN-CLI-500102: Failed to create Domain [*Domain*], TIBCO-BW-ADMIN-500101: Domain [*Domain*] already exists, check this and re-try.	The specified domain already exists. Domain names must be unique; enter a different name. To view existing domains, use the **`show domains`** command.
The specified command could not be completed; the domain home folder specified could not be found.	TIBCO-BW-ADMIN-CLI-500103: Domain home folder [*Path*] does not exist. Verify if the folder is present and use forward slash in the folder path.	Check that the *`BW_HOME`*`\domains` folder exists. If it does exist, make sure forward slashes are specified (both Windows and Unix).
The specified domain could not be deleted.	TIBCO-BW-ADMIN-CLI-500104: Failed to delete Domain [*Domain*], *Reason*	The specified domain may have AppSpaces associated with it. If this is the case, the following message is displayed: TIBCO-BW-ADMIN-500109: The Domain [*Domain*] has AppSpaces associated with it. Use -force option to override Use the **`-force`** option with the **`delete`** command to delete the domain and all contained runtime entities.
The minimum number of AppNodes for an AppSpace has to be at least 1.	TIBCO-BW-ADMIN-CLI-500218: The minNodes configuration value for an AppSpace has to be at least 1.	The `minNodes` value for an AppSpace has to be set to an integer value greater than 0. To create an AppSpace with 1 AppNode, the `minNodes` option is not required.

Issue	Message	Resolution
The minimum number of AppNodes is invalid. It has to be an integer value greater than 0.	TIBCO-BW-ADMIN-CLI-500219: The minNodes configuration argument is invalid. Valid argument are Integer values greater than 0.	The `minNodes` value for an AppSpace has to be set to an integer value greater than 0.
The AppSpace could not be found in the specified domain.	TIBCO-BW-ADMIN-CLI-500201: AppSpace [*AppSpace*] not found in Domain [*Domain*]	The specified AppSpace does not exist in the specified domain. Check the value for typos. Use the **`show appspaces`** command with the **`-domain`** option to view AppSpaces in the domain.
The specified AppSpace in the specified domain could not be created; it already exists.	TIBCO-BW-ADMIN-CLI-500203: Failed to create AppSpace [*AppSpace*] in Domain [*Domain*]. TIBCO-BW-ADMIN-500202: The AppSpace [*AppSpace*] is already present in the Domain [*Domain*].	The specified AppSpace has already been created. Use the **`show appspaces`** command with the **`-domain`** option to view AppSpaces in the domain.
The specified AppSpace already exists for the specified bwagent.	TIBCO-BW-ADMIN-CLI-500204: AppSpace [*AppSpace*] already exists with BW Agent [*bwagent*]	The specified AppSpace has already been expanded to the specified bwagent.

Issue	Message	Resolution
The specified AppSpace in the specified domain could not be started.	TIBCO-BW-ADMIN-CLI-500210: AppSpace [*AppSpace*] in Domain [*Domain*] could not be started, *Reason*	The specified AppSpace may not have associated AppNodes or it might be on a bwagent that is not reachable. The following message is displayed if there are no contained AppNodes: TIBCO-BW-ADMIN-500210: AppSpace [*AppSpace*] in Domain [*Domain*] did not start completely, status is Degraded. An AppSpace can only be started if it contains at least one AppNode. Use the **`show appnodes`** command with the **`-appspace`** and **`-domain`** options to check for AppNodes. If no AppNodes exist, create at least one and try to start the AppSpace again. If the minimum number of AppNodes was specified when the AppSpace was created, that minimum number of AppNodes must exist. A message similar to the following will be displayed if the AppSpace is on an unreachable bwagent: TIBCO-BW-ADMIN-500210: AppSpace [*AppSpace*] in Domain [*Domain*] did not start completely, status is Stopped. The machine might be down or the bwagent might not be running.
The specified AppSpace in the specified domain could not be deleted.	TIBCO-BW-ADMIN-CLI-500205: Failed to delete AppSpace [*AppSpace*] from Domain [*Domain*], *Reason*	The specified AppSpace may have AppNodes associated with it or it may be scaled across bwagents. If it has associated AppNodes, the following message is displayed: TIBCO-BW-ADMIN-500216: AppSpace [*AppSpace*] has AppNodes associated with it. Delete the AppNodes first and re-try or use the **`-force`** option to override. If the AppSpace is scaled across machines, the following message is displayed: TIBCO-BW-ADMIN-500220: AppSpace [*AppSpace*] is scaled across multiple BW Agents. Cannot be deleted. Use -force option to override. In both cases, either delete the contained AppNodes or use the **`-force`** option with the **`delete`** command to delete the AppSpace and all contained runtime entities.

Issue	Message	Resolution
When creating an AppNode, the HTTP Port value is required.	httpPort is a mandatory argument for creating an AppNode.	Create the AppNode again with the **-httpPort** option. The port must be unique for each AppNode on the machine.
The specified AppNode in the specified AppSpace and domain could not be created; it already exists.	TIBCO-BW-ADMIN-CLI-500302: Failed to create AppNode [*AppNode*] in AppSpace [*AppSpace*] in Domain [*Domain*], TIBCO-BW-ADMIN-500301: The AppNode [*AppNode*] already exists in the AppSpace [*AppSpace*] Domain [*Domain*].	The specified AppNode has already been created. Use the **show appnodes** command with the **-appspace** and **-domain** options to view AppNodes.
The specified AppNode in the specified AppSpace and domain could not be started.	TIBCO-BW-ADMIN-CLI-500304: AppNode [*AppNode*] in Domain [*Domain*] did not start, *Reason*	The specified AppNode may not exist. In this case, the following error is displayed: TIBCO-BW-ADMIN-500300: The AppNode [*AppNode*] does not exist in AppSpace [*AppSpace*] and Domain [*Domain*]. If the bwagent is in a network, the AppSpace might be on a bwagent that is not reachable. The machine might be down or the bwagent might not be running. In this case, the following error is displayed: TIBCO-BW-ADMIN-500004: Error invoking [startappnode] method on the agent [*agent*], The BW Agent [*agent*] on the remote machine is not running. Start the bwagent on the remote machine and start the AppNode again.

Issue	Message	Resolution
The specified AppNode in the specified AppSpace and domain could not be deleted.	TIBCO-BW-ADMIN-CLI-500306: Failed to delete AppNode [*AppNode*] in AppSpace [*AppSpace*] in Domain [*Domain*]	The specified AppNode may be running. If this is the case, the following message is displayed: TIBCO-BW-ADMIN-500314: The AppNode [*AppNode*] is still in [Running] state. Please stop the AppNode first or use the -force option. Stop the AppNode and delete it or use the **`-force`** option with the **`delete`** command. If the bwagent is in a network, the AppNode might be on a bwagent that is not reachable. The machine might be down or the bwagent might not be running. In this case, the following error is displayed: TIBCO-BW-ADMIN-500306: Failed to delete AppNode [*AppNode*] in AppSpace [*AppSpace*] in Domain [*Domain*], The BW Agent [*agent*] on the remote machine is not running.
When the AppNode is started, the specified HTTP port cannot be allocated.	TIBCO-THOR-FRWK-500300: Eclipse Jetty server bundle has reported an error that it cannot allocate the HTTP management port [*port*]. Shutting down the AppNode.	An AppNode (or a JVM) is already running with the specified port. Stop that AppNode or process and restart the AppNode.
The OSGi console could not be enabled on the specified AppNode.	TIBCO-BW-ADMIN-CLI-500314: Failed to enable console on AppNode [*AppNode*] in AppSpace [*AppSpace*] in Domain [*Domain*], *Reason*	The OSGi console could not be enabled for the specified AppNode. The AppNode is not running or is not reachable. In this case, the following additional message is displayed: TIBCO-BW-ADMIN-500309: Failed to enable console on AppNode [*AppNode*] in AppSpace [*AppSpace*] in Domain [*Domain*] Make sure the AppNode is running and try again.

Issue	Message	Resolution
The OSGi console could not be disabled on the specified AppNode.	TIBCO-BW-ADMIN-CLI-500315: Failed to disable console on AppNode [*AppNode*] in AppSpace [*AppSpace*] in Domain [*Domain*], *Reason*	The OSGi console could not be disabled for the specified AppNode. The AppNode is not running or is not reachable. In this case, the following additional message is displayed: TIBCO-BW-ADMIN-500310: Failed to disable console on AppNode [*AppNode*] in AppSpace [*AppSpace*] in Domain [*Domain*] Make sure the AppNode is running and try again.
The debug port on the AppNode could not be enabled.	TIBCO-BW-ADMIN-CLI-500440: Failed to enable debug port on AppNode[*AppNode*] in [*AppSpace*] in Domain[*Domain*], *Reason*	The debug port on the AppNode could not be enabled. The AppNode is not running or is not reachable. In this case, the following additional message is displayed: TIBCO-BW-ADMIN-500313: AppNode [*AppNode*] is not running or could not be contacted. Make sure the AppNode is running and try again.
The debug port on the AppNode could not be disabled.	TIBCO-BW-ADMIN-CLI-500317: Failed to disable debugger on AppNode[*AppNode*] in [*AppSpace*] in Domain[*Domain*], *Reason*	The debug port on the AppNode could not be disabled. The AppNode is not running or is not reachable. Make sure the AppNode is running and try again.

Troubleshooting Archive Issues

Some archive issues and possible resolutions are listed below. This list is not complete but provides examples of messages that might be returned.

For a complete list of error codes, see the *TIBCO ActiveMatrix BusinessWorks™ Error Codes* guide.

Issue	Message	Resolution
The specified archive could not be uploaded; it could not be located.	TIBCO-BW-ADMIN-CLI-500433: Failed to upload archive [*Archive*], <CausedBy> TIBCO-THOR-FRWK-CMN-500101: Ear file [*Archive*] is not found.	The specified archive could not be found. Check the path and the archive filename and issue the command again.

Issue	Message	Resolution
The specified archive has already been uploaded to the specified domain.	TIBCO-BW-ADMIN-CLI-500433: Failed to upload archive [*archive*], <CausedBy> TIBCO-BW-ADMIN-500447: Archive [*Archive*] is already present in the domain, use -replace option to replace the existing archive.	The specified archive has already been uploaded to the specified domain. To replace the archive, upload the archive again, using the **`-replace`** option with the **`upload`** command. Use the **`show archives`** command to view archives, versions, time uploaded, size, and path.
The specified archive could not be uploaded to all machines in the domain.	TIBCO-BW-ADMIN-CLI-500432: Failed to upload ear file [*Archive*] to some machines in the domain.	Not all machines that the domain has been expanded to are reachable. Check that machines are running.
The specified archive has already been uploaded; the application archive has already uploaded and deployed.	TIBCO-BW-ADMIN-CLI-500438: The application [*Application*] from archive [*Archive*] has been deployed to these AppSpaces: *AppSpace*	You already uploaded the archive and deployed the specified application from the archive. The message displays the AppSpaces the application has been deployed to. Use the **`show applications`** command to view applications, versions, and statuses.
The specified archive could not be deleted.	TIBCO-BW-ADMIN-CLI-500434: Failed to delete archive [*Archive*] <CausedBy> *Reason*	The specified archive may not exist in the current domain context. If this is the case, the following message is appended: TIBCO-BW-ADMIN-500418: Application archive file [*Archive*] not found in the domain [*Domain*]. Use the **`show archives`** command to view archives, versions, time uploaded, size, and path. The specified archive may be deployed and cannot be deleted. In this case, the following message is displayed: TIBCO-BW-ADMIN-500450: Archive [*Archive*] has been deployed to the AppSpaces: [*AppSpaces*].
The specified application is not in sync with the specified archive.	TIBCO-BW-ADMIN-CLI-500439: Applications are out of sync with the archive they were deployed from, They have to be re-deployed to keep them in sync.	The application is no longer in sync with the archive. Use the **`deploy -replace`** command to replace the existing version of the application with the version from the archive. This message is displayed with other messages, for example, if the archive could not be uploaded.

Troubleshooting Application Issues

Some application issues and possible resolutions are listed below. This list is not complete but provides examples of messages that might be returned.

For a complete list of error codes, see the *TIBCO ActiveMatrix BusinessWorks™ Error Codes* guide.

Issue	Message	Resolution
The specified application could not be deployed; no AppNodes exist.	TIBCO-BW-ADMIN-CLI-300432: Deployed application [*Application:Version*], The AppSpace [*AppSpace*] does not have any AppNodes.	No AppNodes have been created in the specified AppSpace. AppNodes are required for deployment. Create one or more AppNodes and issue the **`deploy`** command again.
The specified application could not be deployed; an application has already been deployed for that archive.	TIBCO-BW-ADMIN-CLI-300433: An application is already deployed with archive [*Archive*].	The specified application has already been deployed for the specified archive. Use the **`show applications`** command to view it.
The specified application version format is not supported.	TIBCO-BW-ADMIN-CLI-500407: Version [*Version*] is not valid. Only <major>.<minor> version format is supported.].	An application version must be specified as a major.minor version within an AppSpace. Check the formatting of the version number.
The specified application could not be started; application not found in domain.	TIBCO-BW-ADMIN-CLI-500409: Failed to start application [*Application:Version*]. <CausedBy> TIBCO-BW-ADMIN-500401: Application [*Application*] not found in the Domain [*Domain*]	The specified application is not found in the specified domain. Use the **`show applications`** command to verify the application.

Issue	Message	Resolution
The specified application could not be started.	TIBCO-BW-ADMIN-CLI-500409: Failed to start application [*Application:Version*]. <CausedBy> TIBCO-BW-ADMIN-500444: Failed to start Application in AppNode [*AppNode*]. Check the AppNode log files for messages starting with TIBCO-THOR-FRWK, TIBCO-BW-FRWK, or TIBCO-BW-SR-FRWK for details. Application State [Start failed], reason: [*Reason*]	The specified application failed to start. This could be caused by unresolved shared resources, missing constraints, or missing components. If you see this message, open the log file for the AppNode (in the *BW_HOME*\domains\<domain>\<AppSpace>\<AppNode>\log folder) and check for messages starting with: • TIBCO-THOR-FRWK • TIBCO-BW-FRWK • TIBCO-BW-SR-FRWK These messages should help you identify the source of the issue. You may need to adjust the logging level for the log file. See AppNode Logging for information. An application might not start if the specified AppNode is not running or is not reachable. In this case, the following message will be displayed: TIBCO-BW-ADMIN-CLI-500409: Failed to start application [*Application:Version*]. <CausedBy> TIBCO-BW-ADMIN-500313: AppNode [*AppNode*] is not running or cannot be contacted.
The specified application has been deployed but could not be started.	TIBCO-BW-ADMIN-C*Reason*LI-500414: Deployed application from the archive [*Archive*], however not all application instances started. <CausedBy> TIBCO-BW-ADMIN-500313: AppNode [*AppNode*] is not running or cannot be contacted.	The applicatoin instance couldn't be started on the specified node, as it is not running or is unreachable. Check that the bwagent the AppNode is registered to is reachable. Check that the Appnode is running.
The specified application is not in sync with the archive.	TIBCO-BW-ADMIN-CLI-500439: Applications are out of sync with the archive they were deployed from, They have to be re-deployed to keep them in sync.	The EAR file has been replaced with a new version or a new file of the same version. Redeploy the application using the -replace option: **`deploy -replace <EAR>`**

Troubleshooting Admin UI Issues

Some Admin UI issues and possible resolutions are listed below. This list is not complete but provides examples of messages that might be returned.

For a complete list of error codes, see the *TIBCO ActiveMatrix BusinessWorks™ Error Codes* guide. When a new version of the product is installed, you may need to open the Admin UI **Agents** page and reconnect to the bwagent.

Issue	Message	Resolution
The specified bwagent could not be registered in the Admin UI.	Failure registering agent with [*URL*].	This error can be caused by several issues: • The specified bwagent at the URL provided in the Register Agent dialog box could not be located. Verify the URL and register the bwagent again. • A bwagent with the specified name already exists. Verify the name of the bwagent and register the bwagent again. • The specified bwagent might not be running. Make sure the bwagent is running and register the bwagent again.
The specified bwagent is unreachable.	The status of `Unreachable` is displayed on the **Agent Management** page for the specified bwagent.	The specified bwagent might not be running. Make sure the bwagent is running and register the bwagent again.
The details for a selected runtime entity cannot be displayed.	Error executing operation 'getStates'; status '0'; reason 'Connection refused: no further information'.	Since the Admin UI page for the selected runtime entity was displayed, either the runtime entity was deleted or the bwagent was stopped. Use bwadmin commands for the registered bwagent to verify the state of runtime entities and check the status of the bwagent.
The specified domain already exists.	TIBCO-BW-TEAAGENT-500309: Failed to created Domain [*Domain*] TIBCO-BW-ADMIN-500101: Domain [*Domain*] already exists, check this and re-try.	The specified domain already exists. Domain names must be unique; enter a different name. View existing domains on the **Domain Management** page.

Issue	Message	Resolution
The specified domain could not be created; the bwagent on the remote machine is not running.	TIBCO-BW-TEAAGENT-500309: Failed to created Domain [*Domain*] TIBCO-BW-ADMIN-500004: Error invoking [joinmachine] method on the agent [*bwagent*], The BW Agent [*bwagent*] on the remote machine is not running.	Verify the bwagent name. Check that the bwagent on the remote machine is running.
The specified AppSpace in the selected domain already exists.	TIBCO-BW-TEAAGENT-500425: AppSpace [*AppSpace*] already exists in domain [*Domain*].	The specified AppSpace already exists. AppSpace names must be unique; enter a different name. View existing AppSpaces on the **AppSpaces** page.
The AppSpace status is **`Degraded`** and cannot be started.	The status of **`Degraded`** is displayed on the **AppSpaces** page for the specified AppSpace.	An AppSpace can only be started if it contains at least one AppNode. Check the **`minNodes`** value on the **AppSpaces** page and then pivot to the **AppNodes** page. Verify that the minimum number of AppNodes has been created. When the **`minNodes`** value is reached, the status of the AppSpace is changed to **`Stopped`**.
The specified AppNode in the selected AppSpace and domain already exists.	TIBCO-BW-TEAAGENT-500313: Failed to create AppNode [*AppNode*] in AppSpace [*AppSpace*] in Domain [*Domain*], TIBCO-BW-ADMIN-500301: The AppNode [*AppNode*] already exists in the AppSpace [*AppSpace*] Domain [*Domain*].	The specified AppNode already exists. AppNode names must be unique; enter a different name. View existing AppNodes on the **AppNodes** page.
The specified archive could not be uploaded.	Failed to Upload. TIBCO-BW-ADMIN-500447: Archive [*Archive*] is already present in the domain, use -replace option to replace the existing archive.	The specified archive has already been uploaded to the specified domain. To replace the archive, select the **Replace any version** check box in the Upload EAR File dialog box, and upload the archive again.

Issue	Message	Resolution
The specified archive could not be deployed.	TIBCO-BW-TEAAGENT-300016: Deployed Application [*Application*] in AppSpace [*AppSpace*] of Domain [*Domain*]. The AppSpace [*AppSpace*] does not have any AppNodes.	The specified AppSpace does not have any associated AppNodes. Click **Create AppNodes** on the **AppNodes** page. Select the specified AppSpace in the **Create AppNode** dialog.
The specified application could not be started.	TIBCO-BW-TEAAGENT-500315: Failed to deploy Application from archive [*Archive*] TIBCO-BW-ADMIN-500444: Failed to start Application in AppNode [*AppNode*]. Check the AppNode log files for messages starting with TIBCO-THOR-FRWK, TIBCO-BW-FRWK, or TIBCO-BW-SR-FRWK for details. Application State [Start failed], reason: *Reason*	The specified application failed to start. This could be caused by unresolved shared resources, missing constraints, or missing components. If you see this message, open the log file for the AppNode (in the *BW_HOME*`\domains\<domain>\<AppSpace>\<AppNode>\log` folder) and check for messages starting with: • TIBCO-THOR-FRWK • TIBCO-BW-FRWK • TIBCO-BW-SR-FRWK These messages should help you identify the source of the issue. You may need to adjust the logging level for the log file. For more information, see AppNode Logging.

Issue	Message	Resolution
The specified application could not be started.	TIBCO-BW-TEAAGENT-500410: Failed to start Application TIBCO-BW-ADMIN-500313: AppNode [*AppNode*] is not running or cannot be contacted.	The specified AppSpace may not have associated AppNodes or it might be on a bwagent that is not reachable. An application can only be started if the specified AppNode is running. Pivot to the AppNodes page and verify the status of AppNodes in the AppSpace. If no AppNodes exist, create at least one and try to start the application again. If the minimum number of AppNodes was specified when the AppSpace was created, that minimum number of AppNodes need to exist. If the application is on a remote machine, the machine might be down or the bwagent might not be running.
The specified archive could not be deleted.	Archive could not be deleted. TIBCO-BW-ADMIN-500450: Archive [*Archive*] has been deployed to the AppSpaces: [*AppSpaces*].	The specified archive has been deployed. Undeploy the archive from the **Application Archives** page, then click **Delete** to delete the archive.
The application is not in sync with the archive.	The **`Out of Sync`** state is displayed for the application **Deployment state** on the **Applications** page.	Click the **Details** link to view the reason for the state. The **`Out of Sync`** state is displayed when the EAR file is replaced with a new version or a new file of the same version. The software can detect that the application for this archive was already deployed. Redeploy the application to resolve the **`Out of Sync`** state. The following error message can also be displayed for an out of sync situation: TIBCO-BWTEAAGENT-300015: Uploaded Archive [*Archive*] in Domain [*Domain*]. Application deployed in the following AppSpace(s) [*AppSpace*] is out of sync. Please re-deploy.

Logging

Log files are generated for bwadmin, bwagent, AppNodes, and applications. Log files capture all executed commands and, depending on the logging level, the corresponding activities.

Log file configuration follows the Logback standard. (Refer to the Logback Project at `http://logback.qos.ch/` for detailed information on configuration parameters.)

The log files created by bwadmin and bwagent are written to the *BW_HOME*`/logs` folder. (Log files created by other utilities, such as bwdesign, are also written to this folder.) AppNode log files are written to the `/log` folder for the AppNode. If you contact TIBCO support, your support representative will most likely ask you to send the appropriate log file.

Logging configurations are customized in the following files:

- For bwadmin: *BW_HOME*`/bin/bwadmin-logback.xml`
- For bwagent: *BW_HOME*`/bin/bwagent-logback.xml`
- For an AppNode: *BW_HOME*`/domains/domain/<AppSpace>/<AppNode>/logback.xml` (created after the AppNode has been started)

Logs can be output to the bwadmin console and a log file. Log files can be retrieved by bwadmin and displayed in the console.

To upload or download a logback file, click the **Upload** or **Download** link, from the Admin UI.

Logging Levels

The global logging level for each type of log is set to INFO by default. Five levels are supported. Specifying a level includes all higher levels. Levels are listed below in order from lowest (least restrictive) to highest (most restrictive):

- TRACE: Records fine-grained informational events.
- DEBUG: Records fine-grained informational events that help in debugging. Useful for diagnostics.
- INFO: Records informational messages that highlight the progress of the application. Useful for production mode.
- WARN: Records potentially harmful situations.
- ERROR: Records error events that are harmful enough to prevent the application from running.

Setting the logging level to DEBUG can adversely affect the performance, especially when logging SOAP messages with attachments or mail with attachments. In such cases, we recommend fine tuning the loggers to log at ERROR level instead of DEBUG.

See the Debugging topic for information on log file components and error code ranges.

Log File Encoding

Log files are generated according to the configuration specified in the `logback.xml` file. If the encoding is not specified in the `logback.xml` file, the system default encoding is used when generating the log files.

To save the log files in a specific encoding like UTF-8, add the `charset` element to the File Appender Encoder configuration in the `logback.xml` file.

```
<appender name="FILE" class="ch.qos.logback.core.FileAppender">
    <file>test.log</file>
      <append>true</append>
        <encoder>
          <charset>UTF-8</charset>
            <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</
pattern>
```

```
        </encoder>
  </appender>
```

Application Logging

You can generate separate log files for an application either by configuring the **Log** activity in TIBCO Business Studio™ for BusinessWorks™ or by modifying the `logback.xml` file of the AppNode.

To support application logging, the following prerequisites must be satisfied.

Prerequisites

- A sifting appender `<appender-ref ref="APPLICATION-FILE"/>` must be present in the `logback.xml` file of the AppNode.
- To support Debugging a Specific Application on the AppNode, the logger BWApp must be present in the the `logback.xml` file of the AppNode.

```
<!-- Do not modify this logger-->
<logger name="BWApp">
       <level value="ERROR"/>
 </logger>
```

- To avoid having hash (#) as part of the logger names in the log files, the appenders present in the `logback.xml` file must use the following encoder:
  - For AppNode:

```
<encoder class="ch.qos.logback.core.encoder.LayoutWrappingEncoder">
          <layout
class="com.tibco.bw.extensions.logback.BWLoggerPatternLayout"/>
</encoder>
```

  - For TIBCO Business Studio for BusinessWorks:

```
<encoder class="ch.qos.logback.core.encoder.LayoutWrappingEncoder">
          <layout
class="com.tibco.bw.extensions.logback.BWLoggerPatternLayoutStudio"/>
</encoder>
```

By default, new log files are created under `{BW.HOME}/bw/6.x/logs` directory for TIBCO Business Studio for BusinessWorks, and for the AppNode, new log files are created under `{APPNODE.HOME}/log` directory.

You can specify a custom location for creation of the log file. However you must use the variable **`${fileName}`** to define the name of the file.

Controlling Output of the Log Activity

You can configure **Log** activity messages to be logged into separate files in one of the following ways:

- **By configuring the Log Activity in TIBCO Business Studio for BusinessWorks**

  You can configure the **Log** activity in TIBCO Business Studio for BusinessWorks to separate logs by application, process, or event type, depending upon the options selected in the **Log** activity.

  For more information about options in the **Log** activity, see the "Log" section in the *TIBCO ActiveMatrix BusinessWorks™ Binding and Palettes Reference*.

- **By configuring the Logger in the AppNode's logback.xml file**

  You can separate logs from the **Log** activity based on your application, without changing the **Log** activity just by modifying `logback.xml` file of the AppNode.

  - For one application, add a new logger in the `logback.xml` file of the AppNode as follows:

```
<logger name="BWApp.#APPNAME#.com.tibco.bw.palette.generalactivities.Log"
additivity="false">
```

```
    <level value="DEBUG"/>
    <appender-ref ref="APPLICATION-FILE"/>
</logger>
```

APPNAME is the name of the application.

– For multiple or all aplications, add the previous logger for each application or follow the steps specified in the Creating Separate Log Files for Each Application on the AppNode section.

Displaying Log Activity Messages on TIBCO Business Studio for BusinessWorks Console

To display the **Log** activity messages in the TIBCO Business Studio for BusinessWorks console, the additivity attribute can be omitted from the logger if the root logger is using STDOUT as its appender.

By modifying the `logback.xml` file, the logs can only be separated by application and not by process or eventType.

Creating Separate Log Files for Each Application on the AppNode

There are two steps to generate separate log files for each application running on an AppNode:

1. Change the value of the property **`bw.engine.separate.logs.by.app`** to `true` from the default value `false` in the `config.ini` file of the AppSpace or the AppNode.

   To enable this property through TIBCO Business Studio™ for BusinessWorks™, pass it as a VM argument using the `-D` option.

   If the **`bw.engine.separate.logs.by.app`** property is set at the AppNode level, this setting takes precedence over the property set at the AppSpace level. Restart the AppNode when the property is updated.

2. Add or modify the loggers in the `logback.xml` file of AppNode to use `<appender-ref ref="APPLICATION-FILE" />`.

If there are any logs that are not specific to an application, and logs generated from all loggers other than supported loggers, are written to the defaulf `bwappnode.log` file.

To create separate log files for all applications in the AppNode, without modifying the **Log** activity and without adding multiple loggers to the `logback.xml` file, set the property **`bw.engine.separate.logs.by.app`** to `true` and if not already present add the `<appender-ref ref="APPLICATION-FILE" />` to the logger **`com.tibco.bw.generalactivities.palette`** as follows:

```
<logger name="com.tibco.bw.palette.generalactivities.Log"  additivity="false">
    <level value="DEBUG"/>
     <appender-ref ref="APPLICATION-FILE"/>
 </logger>
```

The following examples demonstrate some common use cases:

1. To separate supported logs by application name, modify the root logger's appender to use the new sifting appender.

   ```
   <root level="ERROR">
      <appender-ref ref="APPLICATION-FILE" />
   </root>
   ```

   This configuration causes all ERROR logs from all the supported loggers to be separated by application name. The level of individual loggers can be set to a desired value such as DEBUG, ERROR, or INFO.

2. To separate logs of only **`com.tibco.bw.core`** by application name, modify the logger as follows:

   ```
   <logger name="com.tibco.bw.core" additivity="false">
   <level value="ERROR"/>
   <appender-ref ref="APPLICATION-FILE"/>
   </logger>
   ```

3. To separate all palette logs by application name, modify the **com.tibco.bw.palette** logger as follows:

```
<logger name="com.tibco.bw.palette" additivity="false">
<level value="DEBUG"/>
<appender-ref ref="APPLICATION-FILE"/>
</logger>
```

   This configuration causes all DEBUG logs from all the palettes in the AppNode to be separated by application name.

   A similar configuration can be extended to **com.tibco.bw.sharedresource** and **com.tibco.bx**.

```
<logger name="com.tibco.bw.sharedresource" additivity="false">
<level value="DEBUG"/>
<appender-ref ref="APPLICATION-FILE"/>
</logger>
<!-- ---For bx--->
<logger name="com.tibco.bx" additivity="false">
<level value="DEBUG"/>
<appender-ref ref="APPLICATION-FILE"/>
</logger>
```

4. To separate logs of only JMS Connection shared resource by application name, a new logger must be added to the `logback.xml` file.

```
<logger name="com.tibco.bw.sharedresource.jms" additivity="false">
<level value="DEBUG"/>
<appender-ref ref="APPLICATION-FILE"/>
</logger>
```

   This configuration causes all DEBUG logs from all JMS Connection shared resources in the AppNode to be separated by application name. All other shared resources follow the behavior of the logger **com.tibco.bw.sharedresource**.

For TIBCO Business Studio for BusinessWorks, the `bwappnode.log` file is created in the same location where the application logs are generated.

If the **additivity** attribute is not set to `false`, the logger continues to use the appenders of the parent logger all the way up to the root logger until it finds a logger whose additivity is set to false.

If not configured correctly, this can lead to duplicate logging.

Debugging a Specific Application on the AppNode

- To enable DEBUG logging on a specific application of the supported loggers, deployed on the AppNode, where multiple applications are running on the AppNode, you can add a new logger to the existing `logback.xml` file of the AppNode.

  In this scenario the property **bw.engine.separate.logs.by.app** need not be set to `true`.

```
<logger name="BWApp.#APPNAME#">
    <level value="DEBUG"/>
</logger>
```

  Where 'APPNAME' is the name of the application whose debug logs are desired.

  For example, if there are three applications running on the AppNode, App1.application, App2.application, and App3.application, the debug logs can be turned on only for App2.application by adding a logger `BWApp.#App2.application#` to the `logback.xml` file.

```
<logger name="BWApp.#App2.application#">
    <level value="DEBUG"/>
</logger>
```

  The new logger can be appended by any of the supported loggers.

- If you want debug logs only for the **HTTP** palette of the App2.application, add a new logger to the `logback.xml` as follows:

```
<logger name="BWApp.#App2.application#.com.tibco.bw.palette.http">
    <level value="DEBUG"/>
</logger>
```

- Use `<appender-ref ref="APPLICATION-FILE"/>` to separate log files for each supported logger.

```
<logger name="BWApp.#APPNAME#" additivity="false">
      <level value="DEBUG"/>
      <appender-ref ref="APPLICATION-FILE"/>
</logger>
```

Supported log level value for these appenders is DEBUG and TRACE only.

Supported Loggers

You can generate separate logs for each application for the following loggers only:

- **`com.tibco.bw.core`**
- **`com.tibco.bw.palette`** and the hierarchical children. E.g **`com.tibco.bw.palette.http`**, **`com.tibco.bw.palette.file`**
- **`com.tibco.bw.sharedresource`** and the hierarchical children. E.g. **`com.tibco.bw.sharedresource.jdbc`**, **`com.tibco.bw.sharedresource.jms`**
- **`com.tibco.bx`**

Backward Compatibility for Application Logging

You can execute applications created in the versions prior to the TIBCO ActiveMatrix BusinessWorks™ 6.5.0 version having **Log** activities, in the new version with the new or old `logback.xml` file. There is no change in the behavior of an application, and it continues to work as is.

If the property `bw.engine.separate.logs.by.app` is set to `false`, there is no change in the behavior of an application, and it continues to work as is.

AppNode Logging

AppNode logging is enabled for every AppNode.

The AppNode log file is named `bwappnode.log`. It is written to the *`BW_HOME`*`/domains/<domain>/<AppSpace>/<AppNode>/log` folder.

An AppNode log file is created when an AppNode is started. By default, AppNode logs are written to a file appender only. The logs can be viewed in a text editor or displayed in the bwadmin console. The default logging level is ERROR.

In addition to AppNode logging, execution statistics are collected through logback. For information, see Integrating Process Statistics Collection Using Logback.

To view and change the logging level for a single AppNode:

Procedure

1. Create and start an AppNode.
2. Open the `logback.xml` file in the AppNode root folder in a text editor: *`BW_HOME`*`/domains/<domain>/<AppSpace>/<AppNode>`
3. Change the ROOT level at the end of the file as needed.

4. Save the `logback.xml` file.

   By default, the new configuration is reloaded in 60 seconds. There is no need to restart the AppNode for the logging level to take effect.

   To change the scan period, add the `scanPeriod` attribute to the `configuration` element in the `logback.xml` file. For example:

   ```
   <configuration scan="true" scanPeriod="10 seconds">
   ```

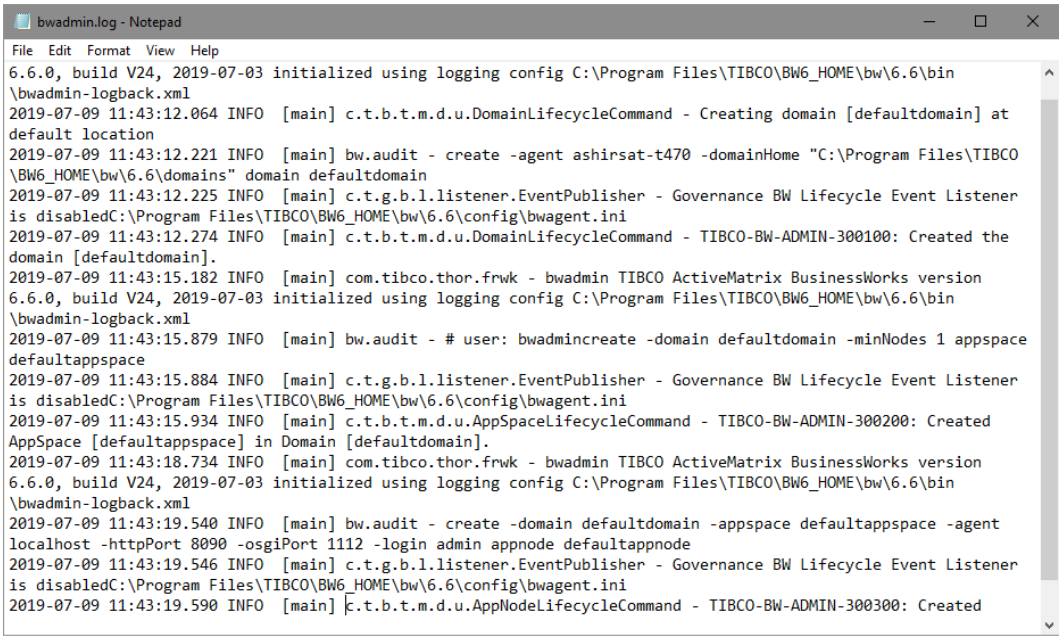
5. Save the file, start the AppNode, and the application.
6. Open the `bwappnode.log` file in the root folder of the AppNode to see what has been logged. The log file can be opened in the bwadmin console with the following command from the containing AppSpace: **`getlogfile appnode`** *AppNode*

bwadmin Logging

bwadmin creates a log file called `bwadmin.log` that is written to the *BW_HOME*`/logs` folder. The default log is configured as a daily roller appender and is automatically compressed as a ZIP file. The default logging level is INFO. The logback configuration file is *BW_HOME*`/bin/bwadmin-logback.xml`.

A bwadmin log file is created on installation, showing the runtime entities that are created by default (domain, AppSpace, and AppNode). The contents of the default log file will look similar to the following image.

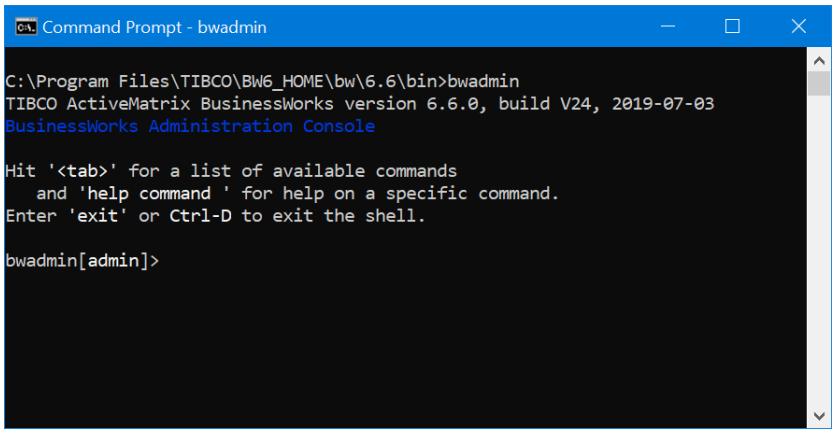
bwadmin Log File on Install



To view and change the logging level for bwadmin, follow these steps.

Procedure

1. Start bwadmin. Notice that no INFO messages are displayed:

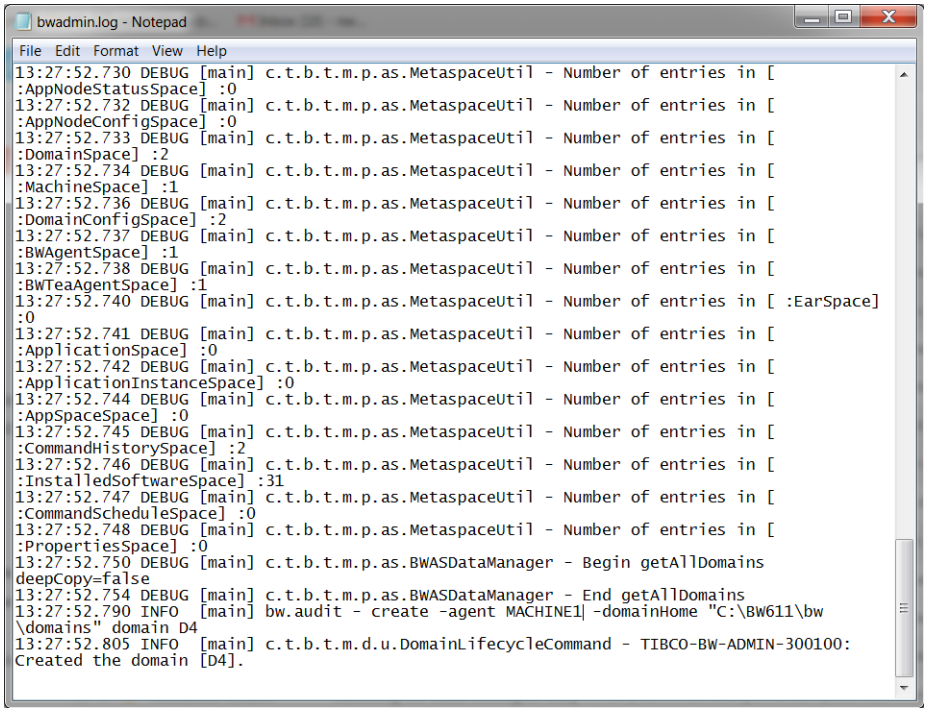


If bwadmin could not be started, informational messages will be displayed to help you track down the cause.

2. Exit bwadmin.
3. Open *BW_HOME*`/bin/bwadmin-logback.xml` in a text editor. Change the ROOT level setting (indicated below in bold font) at the end of the file as needed. This changes the level for the file appender.

```
<root level="INFO">
    <appender-ref ref="STDOUT" />
    <appender-ref ref="FILE" />
```

4. Save the logback configuration file and start bwadmin. The configuration file is reloaded in 30 seconds by default. Create a domain.
5. Open the log file at *BW_HOME*`/logs/bwadmin.log`. More detail is captured in the log file.



bwagent Logging

The bwagent creates a log file called `bwagent.log` that is saved in the *BW_HOME*`/logs` folder. The default log is configured as a daily roller appender and is automatically compressed as a ZIP file. The default logging level is INFO. The logback configuration file is *BW_HOME*`/bin/bwagent-`

`logback.xml`. When the bwagent is started, the `bwagent.log` log file is created. If the bwagent could not be started, informational messages will be displayed to help track down the cause. For the default logging level, the bwagent displays messages similar to the following.

```
C:\work\BW-v25\bw\6.6\bin>bwagent
TIBCO ActiveMatrix BusinessWorks version 6.6.0, build V25, 2019-07-09
13:45:39.405 WARN  [main] org.eclipse.jetty.server.Server - ErrorPageMapper not
supported for Server level Error Handling
TIBCO-BW-AGENT-300002: BusinessWorks Agent started successfully.
```

To view and change the logging level for bwagent, follow these steps. (The bwagent does not have to be restarted for the logging level change to take effect.)

Procedure

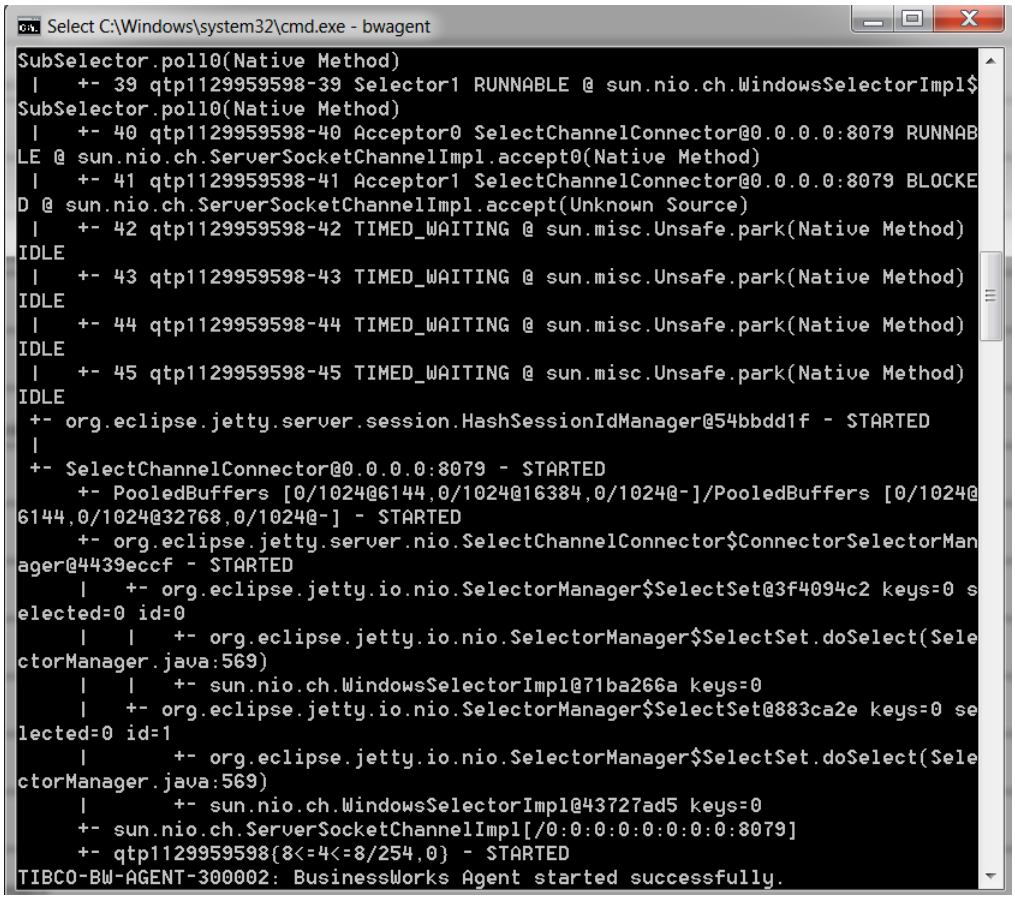
1. Open *BW_HOME*`/bin/bwagent-logback.xml` in a text editor. Change the ROOT level setting (indicated below in bold font) at the end of the file as needed. This changes the level for the file appender.

   ```
   <root level="INFO">
       <appender-ref ref="STDOUT" />
       <appender-ref ref="FILE" />
   ```

2. Save the logback configuration file. The configuration file is reloaded in 30 seconds by default.

   You can also set the logging level using the **`bw.agent.technology.as.logleve`** property in the *BW_HOME*`/config/bwagent.ini` file.

3. To get the bwagent log file at the bwadmin console, open a terminal and start bwadmin if it is not already started. Type: **`getlogfile agent`**
   The log file contains messages for each activity. The following illustration displays messages bwagent startup messages.



You can also view the log file in a text editor.

HTTP Logging

When Jetty servers used in shared resources receive requests, logs are created to capture all attempts to access the servers.

By default, logging is disabled. You can enable HTTP logging from TIBCO Business Studio for BusinessWorks and from a deployed application.

To enable and test logging from TIBCO Business Studio for BusinessWorks in the **HTTP Connector Resource** shared resource:

1. In the **Advanced** tab, select the **Enable Access Logs** check box.
2. In the **Advanced** tab, select **Logging Configuration**.
3. Navigate to `bw\<Release_Number>\config\design\logback` and select `logback_leveldebug.xml`.
4. Check the console logs to make sure they display content.

If you use REST service binding to create the HTTP Connector resource at runtime, set the `bw.engine.http.jetty.accesslogs.enable` system property to `true` to enable Jetty logs.

To enable and test logging from a deployed application:

1. Deploy the application on a TEA server.
2. Enable the debug logging for `appnode`.
3. Check the `appnode.log` file.
4. Check for log contents similar to the following:

```
++++++++++++++++++++
2017-01-06 16:26:17.609 INFO [bwResourceHTTPConnector.qtp-112]
com.tibco.bw.http.jetty.accesslogger - [HTTP Connector: mpandav-t450:6565]
mpandav-t450
192.168.56.1 - - [06/Jan/2017:16:26:17 +0530] "GET /?hello HTTP/1.1" 200 41 "-"
"Jakarta Commons-HttpClient/3.1" - 32 32
++++++++++++++++++++
```

To populate the HTTP logs, enable debug logging for your application.

For more information on enabling HTTP access logging, see "HTTP Connector" in the *TIBCO ActiveMatrix BusinessWorks™ Bindings and Palette Reference* guide.

The check box, **Enable Access Logs,** is supported only from ActiveMatrix BusinessWorks™ 6.3.2 version onwards. To enable HTTP access logging for applications created on previous versions of ActiveMatrix BusinessWorks, set the property **`bw.plugin.http.jetty.accesslogs`** to `true`.

Viewing Log Files from the Admin UI

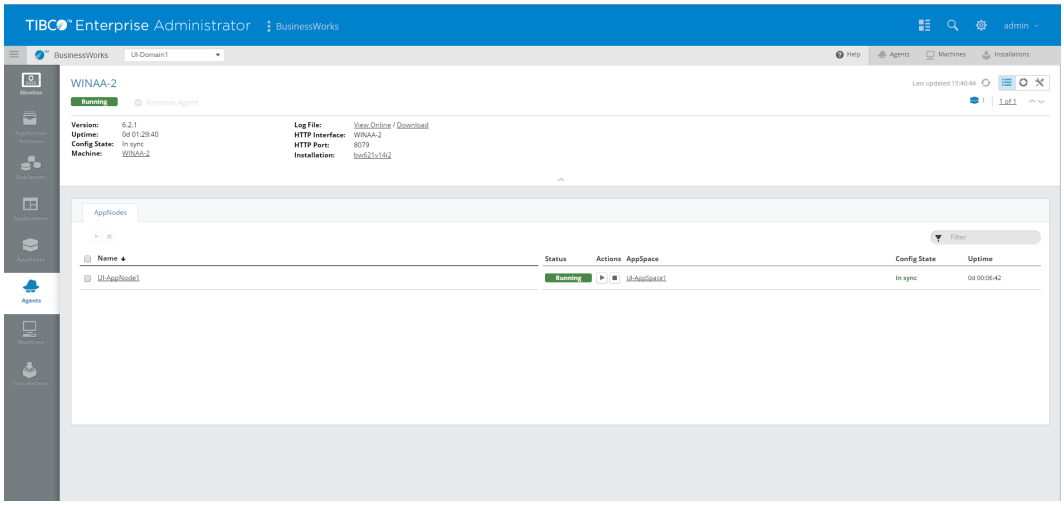
AppNode and bwagent log files are available from the Admin UI.

Prerequisites

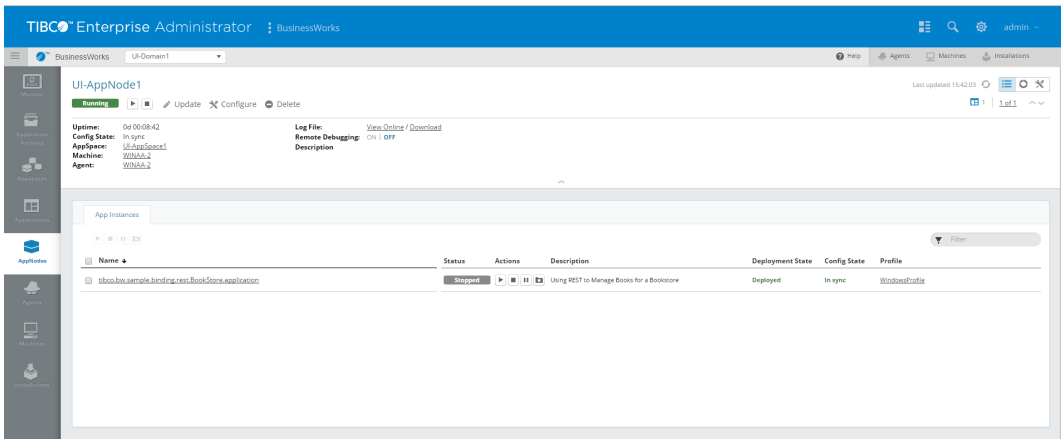
- The Admin UI is running.
- At least one AppNode has been started.

Procedure

1. Open the Admin UI.
2. Select the BusinessWorks product icon on the home page and click a domain.
3. To view the log file for the bwagent, click the Agents icon on the left, then drill into the bwagent by clicking the bwagent name on the Agents page. The page for the selected bwagent is displayed:



4. Click the Log File **View Online** or **Download** link. View Online displays the bwagent log file in a new browser window. Download downloads the file to your machine. This log file is created by the bwagent using the logging configuration provided in the `logback.xml` file. See bwagent Logging for more information.
5. To view the log file for an AppNode, click the AppNodes icon on the left, then choose an AppNode by clicking the AppNode name on the AppNode page. The page for the selected AppNode is displayed:



6. Click the Log File **View Online** or **Download** link. View Online displays the AppNode log file in a new browser window. Download downloads the file to your machine. For more information about AppNode logging, see AppNode Logging.

Fault Tolerance

Fault tolerance is the ability of the system to continue processing requests when an unexpected failure occurs on one of the AppNodes in the AppSpace.

Fault tolerance is supported only at the AppNode level. When an unexpected failure occurs on one of the AppNodes in an AppSpace, the application will no longer be available on that AppNode. However, the fault tolerance configuration enables the application to continue to provide service and process requests through the other AppNodes in the AppSpace. Depending on the activation mode selected for the application (See Activation Modes), the fault tolerance configuration can behave in the following ways:

- Distributes the incoming request load among other AppNodes in the AppSpace.
- If an AppNode that has an application in active state fails, another AppNode that has an application in the passive (stand-by) state takes over and starts processing requests.
- The check-pointed job data from an application in the failed AppNode can be recovered by another AppNode.
- If an application is in the standby or disabled mode, the status in the **Components** tab in Admin UI changes to Stopped, and the starter state displayed in the command line changes to Not Active. For more information on retrieving the list of components, see Retrieving list of components in an Application.

ActiveMatrix BusinessWorks fault tolerance feature can be classified into two types: Managed Fault Tolerance and Non-Managed Fault Tolerance.

Managed Fault Tolerance

In managed fault tolerance, when an AppNode fails, the application on another AppNode takes over automatically. The AppNodes in an AppSpace are aware of each other's existence and the engines collaborate to provide fault tolerance.

The managed fault tolerance requires:

- The engine persistence mode (**`bw.engine.persistenceMode`**) to be set to type `group`. The persistence mode of type `group` requires both database and group provider configurations. See Engine Persistence Modes for details.
- A minimum of two AppNodes in an AppSpace.

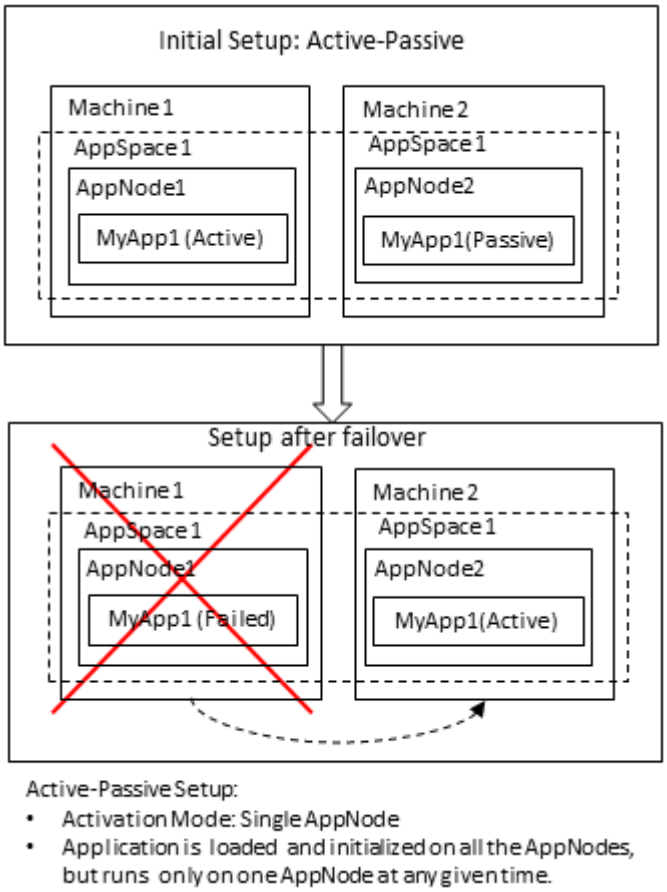
The managed fault tolerance configuration supports both the application activation modes - **Single AppNode** and **Multiple AppNodes**. See Application Activation Modes for details. The following table lists the managed fault tolerance features available for each of the activation modes.

Managed Fault Tolerance Features for Application Activation Modes

Single AppNode (Active-Passive)	Multiple AppNode (Active-Active)
The incoming requests are only processed by an AppNode where the application is in an active state.	The incoming requests can be processed by any AppNode since the application is active in all AppNodes.
On failure of an AppNode that has the application in an active state will automatically enable the application in another AppNode to take over and start processing requests.	On failure of an AppNode, other AppNodes will continue to process new requests.

Single AppNode (Active-Passive)	Multiple AppNode (Active-Active)
The check pointed data from an application in the failed AppNode can be recovered by the application that is automatically enabled in another AppNode.	The check pointed data from an application in the failed AppNode can be automatically recovered by another AppNode.

Fault-tolerant Fail-over



Non-managed Fault Tolerance

In non-managed fault tolerance, the AppNodes in an AppSpace are not aware of each other's existence and there is no collaboration between the engines. Consequently, if an AppNode fails, then another AppNode in the AppSpace will not take over.

The non-managed fault tolerance requires:

- The engine persistence mode (**`bw.engine.persistenceMode`**) to be set to type `datastore`. The persistence mode of type `datastore` requires database configurations. See Engine Persistence Modes for details.
- If there are multiple AppNodes in the AppSpace, then each AppNode must be configured with a unique database configuration. An AppNode specific database configuration is stated through the AppNode `config.ini` file.

The application activation mode is not applicable in non-managed fault tolerance configuration. That is, the application activation modes `Single AppNode` or `Multiple AppNodes` are not supported in the non-managed fault tolerance. See Activation Modes for details. The application is activated in all AppNodes, however unlike the managed fault tolerance, the other AppNodes in the AppSpace are not aware of each other. The following features are available for non-managed fault tolerance:

- The incoming requests can be processed by any AppNode since the application is active in all AppNodes.
- On failure on an AppNode, other AppNodes will continue to process new requests.
- An application can have checkpoint; however on failure of an AppNode; other AppNode will not recover the check-pointed data.

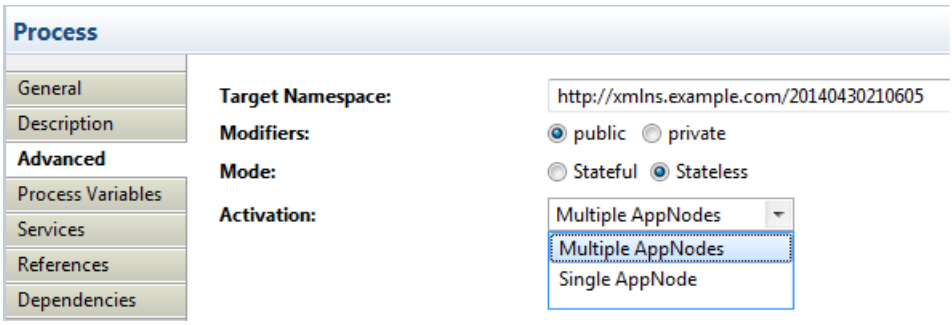
Application Activation Modes

Activation mode for an application defines the way an application is loaded, initiated and started when deployed to an AppSpace.

By choosing the appropriate activation mode, the component in an application module can be configured to be active in multiple AppNodes or in a single AppNode. When configured to be active in a single AppNode, the component is active in a single AppNode and in a passive state in all the other AppNodes of the AppSpace.

A component in an active state is loaded, initialized and enabled to process new events. A component in a passive state is loaded, initialized; however it is not enabled to process new events, instead it is in a stand-by mode. When a component is active in multiple AppNodes, it is considered to be Active-Active. And when a component is active in a single AppNode and passive in other AppNodes, it is considered to be Active-Passive.

The component's activation mode is determined by the process that implements the component. At design-time, the process that is implemented by a component in an application module can be configured to be active in a single AppNode (Active-Passive) or in multiple AppNodes (Active-Active). To do so, set the **Activation** field on the **Advanced** tab of the Properties view as shown:



If the activation mode for a process is set to Multiple AppNodes, then the activation mode for all the sub-processes and child processes (other processes that are called by the process or one of its child processes) must be set to Multiple AppNodes as well. Failure to do so can result in a design-time validation error.

Enabling the activation mode for an application requires both design-time and runtime configurations as described in the following sections.

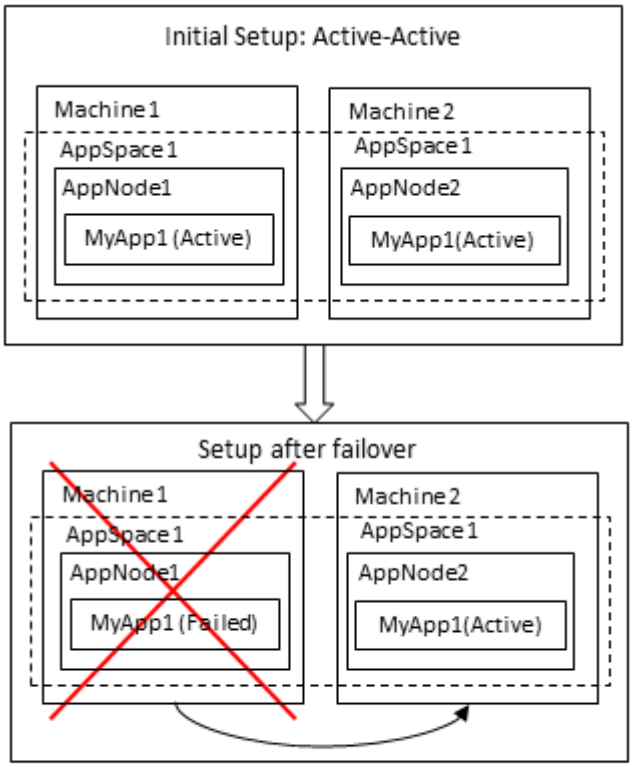
Active-Active (Multiple AppNodes)

In active-active mode, an application is loaded, initiated, and ready to run on all the AppNodes in the AppSpace. Enabling an application for active-active mode requires both design-time and runtime configurations. Perform the following steps to configure an application for active-active mode:

- **Design-time**: Set the activation mode of the component process to `Multiple AppNodes`. The **`Activation`** field is configurable from the **Advanced** tab of the Properties view for the process.
- **Runtime**: The engine persistence mode must be set to `group` to enable the engines running in the AppNodes to be aware of the existence of other engines in the AppSpace and provide managed

fault tolerance feature. See Engine Persistence Modes for details about persistence modes. See Managed Fault Tolerance for details on how to configure the runtime.

Active-Active Mode



Active-Active Setup:

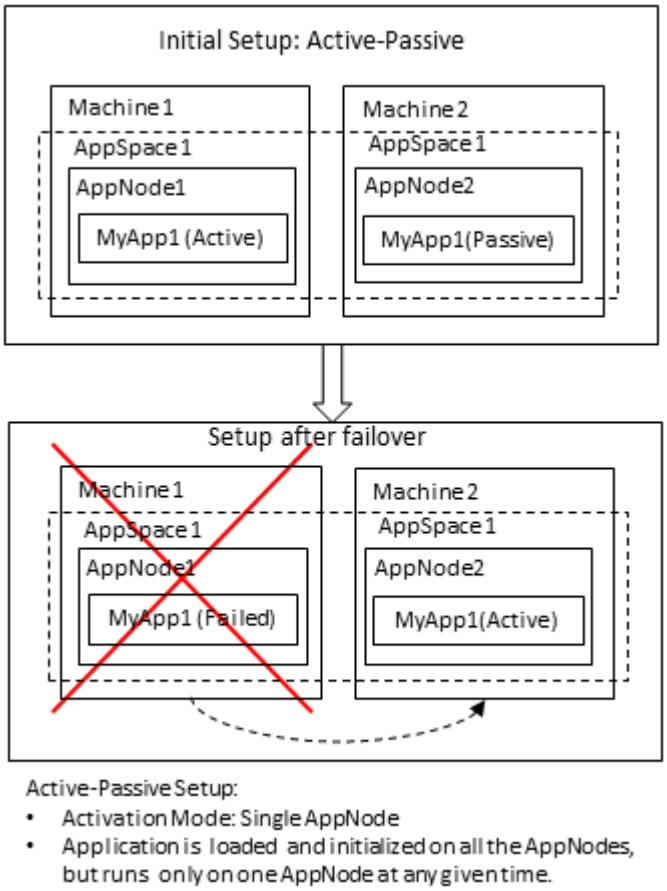
- Activation Mode: Multiple AppNodes
- Application is loaded and is running on all the AppNodes

Active-Passive (Single AppNode)

In active-passive mode, an application is loaded on all the AppNodes in the AppSpace, and is activated to run on a single AppNode at any given time. Enabling an application for active-passive mode requires both design-time and runtime configurations. Perform the following steps to configure an application for active-passive mode:

- **Design-time**: Set the activation mode of the component process to `Single AppNode`. The **`Activation`** field is configurable from the **Advanced** tab of the Properties view for the process.
- **Runtime**: The engine persistence mode must be set to `group` to enable the engines running in the AppNodes to be aware of the existence of other engines in the AppSpace and provide managed fault tolerance feature. See Engine Persistence Modes for details about persistence modes. See Managed Fault Tolerance for details on how to configure the runtime.

Active-Passive Mode



Active-Passive Setup:

- Activation Mode: Single AppNode
- Application is loaded and initialized on all the AppNodes, but runs only on one AppNode at any given time.

When an application is running in active-passive mode on multiple AppNodes, stopping an application instance that is active in an AppNode does not trigger the passive application instances in other AppNodes to become active. The only way a passive application instance in an another AppNode becomes active is when the AppNode that contains the active application is terminated.

Engine Persistence Modes

Engine persistence mode defines whether engines located on one or more physical machines collaborate with each other or work independently. There are four modes of engine persistence: `memory`, `datastore`, `group`, and `ftgroup`.

Engine persistence is defined at the AppSpace level and is set to `memory` by default. To change the engine persistence mode, run the utility to set the persistence mode property **`bw.engine.persistenceMode`** to `datastore`,`group` or `ftgroup`.

Engine Persistence Mode: Memory

`bw.engine.persistenceMode`=`memory`: By default, the engine persistence mode is set to `memory`. In the `memory` mode, there is no persistence and the engines are unaware of the existence of each other. As a result, there is no collaboration between engines.

Engine Persistence Mode: Datastore

`bw.engine.persistenceMode`=`datastore`: The `datastore` mode uses a database to provide persistence and requires a database configuration. The database connection configuration can be specified at the AppNode level. For more information, see Configuring Database for the Engine. The `datastore` engine persistence mode is required for persistence features such as checkpointing and module shared variables.

Engine Persistence Mode: Group

bw.engine.persistenceMode=group: The group mode uses a database and a group provider to provide persistence and collaboration between the AppNodes. In the group mode, the engines are aware of each other's existence and they can collaborate and work together to enable features such as checkpointing and managed fault tolerance. The configuration details must be specified at the AppSpace level. For more information, see Configuring Database for the Engine and Configuring the Engine for Group Persistence Mode.

In group mode, the engine requires both DB and TIBCO Enterprise Message Service™ (EMS) or TIBCO FTL® as mandatory infrastructure requirement. Along with the data persistence and collaboration, AppNode uses the database for internal functions too. So ensure that the database is always available.

Engine Persistence Mode: FTGroup

bw.engine.persistenceMode=ftgroup:

The ftgroup mode does not use a database, and disregards the application activation mode, but does use a group provider to provide minimal collaboration between the AppNodes. Only one AppNode will run the applications, while the other AppNodes will be standing by, to take over in the event of a failure of the active AppNode. When configured for the ftgroup persistence mode, the engine requires a group provider such as TIBCO Enterprise Message Service™ (EMS) or TIBCO FTL®, to be configured. Also, note that, since the ftgroup mode does not use a database, this mode does not support checkpointing. For more information, see Configuring the Engine for FTGroup Persistence Mode.

Configuring Database for the Engine

Checkpoint activity and other persistence features require the engine persistence mode (**bw.engine.persistenceMode**) to be configured for a **datastore** or **group** mode. When the engine persistence mode property is configured for **datastore** or **group** mode, the engine requires a database configuration.

Procedure

1. Scripts for creating the engine database for various database types are located at *BW_HOME*/config/dbscripts/engine. Based on whether the engine persistence mode property is configured for **datastore** mode or **group** mode, complete one of the following steps:
   a) If the engine persistance mode property is set to **datastore** mode, run the bundled scripts create.sql and create-scp.sql to create the engine database.
   b) If the engine persistance mode property is set to **group** mode, run the bundled scripts create.sql and create-dcp.sql to create the engine database.

   The create.sql, create-scp.sql, and create-dcp.sql scripts are available for each vendor directory in the *{BW_HOME}*\config\dbscripts\engine directory.
2. To change the engine persistence mode, run the utility to set the persistence mode property **bw.engine.persistenceMode** to datastore or group, and then configure the engine database connection details.

   ```
   bw.engine.persistenceMode=[datastore | group]
   ```

   Before updating the AppSpace configuration, you must stop the AppSpace if it is running.

   The database connection configuration can be specified at the AppSpace or the AppNode level. The database connection details specified at the AppSpace level will apply to all AppNodes within the AppSpace. The configuration specified at the AppNode level takes precedence over the configuration specified at the AppSpace level.

When the engine persistence mode property is set to `group`, the database connection configuration must be specified only at the AppSpace level.

When the engine persistence mode property is set to `datastore`, the database connection configuration cannot be shared by two or more AppNodes in the same AppSpace. As a result, the database connection configuration can be specified at the AppSpace level only if the AppSpace contains a single AppNode. For an AppSpace that contains two or more AppNodes, each AppNode requires a unique database and the database connection configuration must be specified at the AppNode level.

3. To set database configuration properties at the AppSpace level, follow these steps:

Ensure you are using a different database instance for each AppSpace. To do this with a single database, create a tablespace or schema for each AppSpace.

   a) Copy the existing AppSpace `config.ini` file (located in the root of the AppSpace folder), or the AppSpace `config.ini` template file `appspace_config.ini_template` (located in *BW_HOME*`/config/`) to a temporary location.
   b) Edit the engine persistence mode property, **`bw.engine.persistenceMode`**, and set it to `datastore` or `group`.

```
bw.engine.persistenceMode=[datastore | group]
```

   c) Configure the following database connection properties in the **BW Engine Database Configuration** section of the `config.ini` file:

```
#--------------------------------------------------------------------------------
# Section:  BW Engine Database Configuration.
#
# The properties in this section are applicable to the BW Engine database.
# All properties in this section are required when the BW Engine
# property "bw.engine.persistenceMode" set to "datastore" or "group".
#
--------------------------------------------------------------------------------
# BW Engine Database Driver.
bw.engine.db.jdbcDriver=org.postgresql.Driver

# BW Engine Database URL.
bw.engine.db.url=jdbc:postgresql://<servername>:<portnumber>/<dbname>

# BW Engine Database User Name.
bw.engine.db.userName=user1

# BW Engine Database User Password.
bw.engine.db.password=

# BW Engine Database Connection Pool Size.
bw.engine.db.maxConnections=15
```

   When setting the password property (**`bw.engine.db.password`**), the default format is plain text. Execute the command **`bwadmin obfuscate`**, or the command **`bwobfuscator`**, from the command line to encrypt the password; use the generated encrypted text as the password.

The bwadmin **`bwenginedb`** command will display BW engine datastore configuration settings.

4. To set the database for datastore mode at the AppNode level, follow these steps:
   a) Copy the existing AppNode `config.ini` file (located in the root of the AppNode folder) to a temporary location.
   b) Set engine persistence mode property **`bw.engine.persistenceMode`** to `datastore` and configure engine database connection details.

```
bw.engine.persistenceMode=[datastore]
```

   c) Configure the engine database connection properties in the **BW Engine datastore configuration** section of the `config.ini` file. By default, the AppNode `config.ini` file does not contain these properties. Copy these properties from the AppSpace `config.ini` template file , `appspace_config.ini_template`, located in *BW_HOME*`/config` to the AppNode `config.ini` file and provide the database connection details.

5. Use one of the following **config** admin commands to push the configuration to the AppSpace or the AppNode:
   - AppSpace:
     ```
     bwadmin[admin] > config -d myDomain -a myAppSpace -cf <temporaryLocation>/
     config.ini
     ```
   - AppNode:
     ```
     bwadmin[admin]> config -d myDomain -a myAppSpace -n myAppnode -cf
     <temporaryLocation>/config.ini
     ```
6. Restart the AppSpace.
7.  Before you clean the engine database, ensure that you have backed up all important data.

   To clean the engine database that is configured for **datastore mode**, run the `drop.sql` and `drop-scp.sql` scripts. If the engine database is configured for **group mode**, run the `drop.sql` and `drop-dcp.sql` scripts.

Result

You used the bwadmin command line to set the database configuration property. You can also use the Admin UI to set this property. See the following topics from the *TIBCO ActiveMatrix BusinessWorks™ Administration* guide.

- Editing an AppSpace Configuration
- Editing an AppNode Configuration

Configuring the Engine for Group Persistence Mode

The managed fault tolerance feature requires the engine persistence mode to be configured for the `group` mode. The `group` mode also supports the Checkpoint activity and other persistence features. When configured for the `group` persistence mode, the engine requires both a database and a group provider, such as TIBCO Enterprise Message Service™ (EMS) or TIBCO FTL®, to be configured.

Refer to the following topics for instructions about setting TIBCO EMS or TIBCO FTL as the group provider technology for the engine:

- Configuring EMS as the Group Provider for Engine
- Configuring FTL as the Group Provider for Engine

Configuring EMS as the Group Provider for Engine

Follow these steps to configure the engine for `group` persistence mode, and to set TIBCO EMS as the group provider technology.

Procedure

1. Create the engine database by executing the bundled scripts `create.sql`,`create-scp.sql` and `create-dcp.sql`. Scripts for creating the engine database for various database types are located in `BW/Home/config/dbscripts/engine`. The engine directory contains folders for the supported database types, and scripts for each database can be found in the respective folders.
2. Set engine persistence mode property (**`bw.engine.persistenceMode`**) to `group` and configure the engine group configuration.
   a) Copy the existing AppSpace `config.ini` template file `appspace_config.ini_template` (located in *`BW_HOME`*`/config`) to the root of the AppSpace folder, or a temporary location, and rename the file as `config.ini`.

b) Edit the ActiveMatrix BusinessWorks engine persistence mode property, **`bw.engine.persistenceMode`**, and set it to `group`.

Follow these steps to configure the engine for `group` persistence mode, and to set TIBCO Enterprise Message Service™ (EMS) as the group provider technology.

```
bw.engine.persistenceMode=group
```

c) Specify the group name and group provider technology in the `config.ini` file. The group name is optional and it defaults to domain and AppSpace names separated by an underscore (_). Only TIBCO Enterprise Message Service (EMS) is supported by the group provider technology.

You can use a different database instance for each AppSpace. Alternatively, you can use a single database instance for multiple AppSpaces if you create a tablespace or schema for each one.

```
#
--------------------------------------------------------------------------------
# Section:  BW Engine Group Configuration.
#
# The properties in this section are applicable to the BW Engine group.
# Some of the properties in this section are required when the BW Engine
# property "bw.engine.persistenceMode" is set to "group".
#
--------------------------------------------------------------------------------
# BW Engine Group Name.  This is an optional property and it specifies name of
# the BW engine group.  If this property is not specified, then the group name
# defaults to "Group_<DomainName>_<AppSpaceName>".
#bw.engine.groupName=mytestgroup

# BW Engine Group Connection Provider Technology.  This is a required
property
# when the persistenceMode is set to "group"
(bw.engine.persistenceMode=group)
# and it specifies the BW Engine group communication technology.  The only
# supported values are "ems" and "ftl".  The group connection provider
technology property
# requires additional configuration.  See section "Configuring the Engine for
Group Persistence Mode"
# for additional configuration.
bw.engine.groupProvider.technology=ems
```

d) Specify the group provider configuration:

```
#
--------------------------------------------------------------------------------
# Section:  BW Engine Group Connection Provider EMS Configuration.
#
# Some of the properties in this section are required when the BW Engine Group
# Connection Provider Technology property
"bw.engine.groupProvider.technology"
# value is set to "ems".
#
--------------------------------------------------------------------------------
# BW Engine Group Connection Provider EMS URL.  This property is required if
# the group provider technology is "ems".
bw.engine.groupProvider.qin.EMSServerUrl=tcp://localhost:7222

# BW Engine Group Connection Provider EMS User Name.  This property is
required
# if the group provider technology is "ems".
bw.engine.groupProvider.qin.EMSUserName=admin

# BW Engine Group Connection Provider EMS User Password.  This property is
# required if the group provider technology is "ems".
bw.engine.groupProvider.qin.EMSPassword=

# BW Engine Group Connection Provider EMS Member Prefix.  This property is
# optional and the default value is "EMSGMS".
#bw.engine.groupProvider.qin.EMSPrefix=EMSGMS

# BW Engine Group Connection Provider EMS Recovery Timeout in ms.  This
# property is optional and the default value is "5000" ms.
```

```
#bw.engine.groupProvider.qin.EMSRecoveryTimeout=5000

# BW Engine Group Connection Provider EMS Recovery Attempt Delay in ms.  This
# property is optional and the default value is "500" ms.
#bw.engine.groupProvider.qin.EMSRecoveryAttemptDelay=500

# BW Engine Group Connection Provider EMS Recovery AttemptCount.  This
# property is optional.
#bw.engine.groupProvider.qin.EMSRecoveryAttemptCount=

# BW Engine Group Connection Provider EMS Connect Attempt Count.  This property
# is optional.
#bw.engine.groupProvider.qin.EMSConnectAttemptCount=

# BW Engine Group Connection Provider EMS Connect Attempt Delay in ms.  This
# property is optional.
#bw.engine.groupProvider.qin.EMSConnectAttemptDelay=
```

When setting the password property (**`bw.engine.groupProvider.qin.EMSPassword`**), the default format is plain text. Execute the command **`bwadmin obfuscate`**, or the command **`bwobfuscator`**, from the command line to encrypt the password; use the generated encrypted text as the password.

3. **Optional**. The following properties are available for EMS SSL configuration.

```
EMS SSL Configuration
#client identity consisting of the certificate,
#private key and optionally extra issuer certificates can be included into a
single data block using PKCS12.
#KeyStore or Entrust Store encodings
#bw.engine.groupProvider.ems.ssl.trust.identity=

#The set of Trusted Certificates represents all trusted issuers of the server
certificate.
#It must be specified by the client application unless the host certificate
verification is completely disabled.
#bw.engine.groupProvider.ems.ssl.trust.certlocation=

#EMS SSL connection trust password. This
#property is required if the JMS server protocol is ssl. The password may
#be clear text or supplied as an obfuscated string.
#bw.engine.groupProvider.ems.ssl.trust.password=

#trusted certificate commonname must match the ems server hostname if set to
false
#bw.engine.groupProvider.ems.ssl.disable.verifyHostName=

#client and server certificates must match if set to false
#bw.engine.groupProvider.ems.ssl.trust.disable.verifyHost=
```

4. **Optional.** If you have saved the `config.ini` file to a temporary location, ensure you copy it to the AppSpace root folder located in *`BW_HOME`*`/domains/defaultdomain/appspaces/defaultappspace`.
5. Use the **`config`** admin command to push the configuration to the AppSpace: `bwadmin[admin] > config -d myDomain -a myAppSpace -cf <temporaryLocation>/config.ini`.

Configuring TIBCO FTL® as the Group Provider for Engine

Follow these steps to configure the engine for `group` persistence mode, and to set TIBCO FTL as the group provider technology.

Use of TIBCO FTL with TIBCO ActiveMatrix BusinessWorks™ for configuring bwagent and for configuring group provider for engine does not require TIBCO FTL licenses.

Prerequisites

- See the ActiveMatrix BusinessWorks™ readme for the version of TIBCO FTL that is supported with the version of ActiveMatrix BusinessWorks 6.x you are using.
- Ensure you have installed FTL client libraries. For more information, see Integrating with TIBCO FTL in the*TIBCO ActiveMatrix BusinessWorks™ Installation* guide.
- These steps are only applicable if you are not using TIBCO FTL as the bwagent transport.
- If you are installing TIBCO FTL after you have already installed ActiveMatrix BusinessWorks, set the *tibco.env.FTL_HOME* variable in the `bwcommon.tra` file. You can find this file in the bin folder at *BW_HOME*\bin for Windows, or *${BW_HOME}*/bin for Unix.

1. Install TIBCO FTL. For instructions, see the *TIBCO FTL® Installation* guide.
2. Start the FTL Realm server by executing the `./tibrealmserver -ht <hostIP>:<port>` FTL command.
   ```
   ./tibrealmserver -ht <hostIP>:<port>
   ```
3. Execute the following FTL command to populate data in the bwadmin_ftlrealmserver.json template file, located in the config folder at *BW_HOME*`/config`:
   ```
   ./tibrealmadmin -rs <realmserverurl> -ur <PATH of bwadmin_ftl_realmserver.json>
   ```

For instructions about how to configure an FTL backup server for high availability, see "Configuring Backup Realm Servers for Fault Tolerance" in the *TIBCO FTL® Administration* guide.

Procedure

1. Create the engine database by executing the bundled scripts `create.sql`,`create-scp.sql` and `create-dcp.sql`. Scripts for creating the engine database for various database types are located at *BW_HOME*`/config/dbscripts/engine`. The engine directory contains folders for the supported database types, and scripts for each database can be found in the respective folders.
2. Set engine persistence mode property (**`bw.engine.persistenceMode`**) to `group` and configure the engine group configuration.
   a) Copy the existing AppSpace `config.ini` template file `appspace_config.ini_template` (located in *BW_HOME*`/config`) to the root of the AppSpace folder, or a temporary location, and rename the file as `config.ini`.
   b) Edit the ActiveMatrix BusinessWorks engine persistence mode property, **`bw.engine.persistenceMode`**, and set it to `group`.
   ```
   bw.engine.persistenceMode=group
   ```
   c) Specify the group name and group provider technology as `ftl` in the `config.ini` file. The group name is optional and it defaults to domain and AppSpace names separated by an underscore (_).

Ensure you are using a different database instance for each AppSpace.

```
#
--------------------------------------------------------------------------------
# Section:  BW Engine Group Configuration.
#
# The properties in this section are applicable to the BW Engine group.
# Some of the properties in this section are required when the BW Engine
# property "bw.engine.persistenceMode" is set to "group".
#
--------------------------------------------------------------------------------
# BW Engine Group Name.  This is an optional property and it specifies name of
# the BW engine group.  If this property is not specified, then the group name
# defaults to "Group_<DomainName>_<AppSpaceName>".
#bw.engine.groupName=mytestgroup
```

```
# BW Engine Group Connection Provider Technology.  This is a required
property
# when the persistenceMode is set to "group"
(bw.engine.persistenceMode=group)
# and it specifies the BW Engine group communication technology.  The only
# supported values are "ems" and "ftl".  The group connection provider
technology property
# requires additional configuration.  See section "Configuring the Engine for
Group Persistence Mode"
# for additional configuration.
bw.engine.groupProvider.technology=ftl
```

d) Specify the group provider configuration:

```
#
--------------------------------------------------------------------------------
# Section:  BW Engine Group Connection Provider FTL Configuration.
#
# Some of the properties in this section are required when the BW Engine Group
# Connection Provider Technology property
"bw.engine.groupProvider.technology"
# value is set to "ftl"
#
--------------------------------------------------------------------------------
# BW Engine Group Connection Provider FTL Realm Server.  This property is
required if
# the group provider technology is "ftl".
bw.engine.groupProvider.ftl.realmserver=http://localhost:8080

# BW Engine Group Connection Provider FTL Realm client user name
# This property is required if the group provider technology is "ftl".
bw.engine.groupProvider.ftl.username=

# BW Engine Group Connection Provider FTL Realm client password
# This property is required if the group provider technology is "ftl".
bw.engine.groupProvider.ftl.password=

# BW Engine Group Connection Provider FTL application identifier
# This property is required if the group provider technology is "ftl".
bw.engine.groupProvider.ftl.appinstance.id=bwadmin-endpoint

# BW Engine Group Connection Provider FTL secondary realm server
# This property is optional.
#bw.engine.groupProvider.ftl.secondaryserver=

# BW Engine Group Connection Provider FTL group name
# This property is required if the group provider technology is "ftl".
bw.engine.groupProvider.ftl.groupname=

# BW Engine Group Connection Provider FTL application name
# This property is required if the group provider technology is "ftl".
bw.engine.groupProvider.ftl.appname=bwadmin

# BW Engine Group Connection Provider FTL publish endpoint
# This property is required if the group provider technology is "ftl".
bw.engine.groupProvider.ftl.publish.endpoint=bwadmin-endpoint

# BW Engine Group Connection Provider FTL application name
# This property is required if the group provider technology is "ftl".
bw.engine.groupProvider.ftl.subscribe.endpoint=bwadmin-endpoint
```

When setting the password property (**`bw.engine.groupProvider.ftl.password`**), the default format is plain text. Execute the command **`bwadmin obfuscate`**, or the command **`bwobfuscator`**, from the command line to encrypt the password; use the generated encrypted text as the password.

3. **Optional.** If you have saved the `config.ini` file to a temporary location, ensure you copy it to the AppSpace root folder located in *`BW_HOME`*`/domains/defaultdomain/appspaces/defaultappspace`.

4. Use the **config** admin command to push the configuration to the AppSpace: `bwadmin[admin] > config -d myDomain -a myAppSpace -cf <temporaryLocation>/config.ini`.

Configuring the Engine for FTGroup Persistence Mode

In the managed fault tolerance ftgroup mode, only one AppNode runs the application, and the other AppNodes are standing by.

When the **bw.engine.ftgroup.lbmode** property is set to `true` in the `config.ini` file at the AppSpace level, all processes having **Activation mode** set as `Multiple AppNodes` in TIBCO Business Studio™, run on all bwengines in the group.

By default, **bw.engine.ftgroup.lbmode** is set to `false`.

When you want to elect an AppNode as a leader AppNode, then set the **bw.engine.use.weighted.node** property to `true` at an AppSpace level. For more information about the **bw.engine.use.weighted.node** property, see BW Engine ftgroup Properties.

By default, the **bw.engine.use.weighted.node** property is set to `false`.

The `ftgroup` mode does not support checkpointing. When configured for the `ftgroup` persistence mode, the bwengine requires the group provider, such as TIBCO Enterprise Message Service™ (EMS) or TIBCO FTL®, to be configured.

For instructions about setting TIBCO EMS or TIBCO FTL as the group provider technology for the engine, see the following topics:

- Configuring EMS as the FTGroup Provider for Engine
- Configuring TIBCO FTL® as the FTGroup Provider for Engine

Configuring EMS as the FTGroup Provider for Engine

Follow these steps to configure the engine for ftgroup persistence mode, and to set TIBCO EMS as the group provider technology.

Procedure

1. Set engine persistence mode property (**bw.engine.persistenceMode**) to `ftgroup` and configure the engine group configuration.
   a) Copy the existing AppSpace `config.ini` template file `appspace_config.ini_template` (located in *BW_HOME*`/config`) to the root of the AppSpace folder, or a temporary location, and rename the file as `config.ini`.
   b) Edit the ActiveMatrix BusinessWorks™ engine persistence mode property, **bw.engine.persistenceMode**, and set it to `ftgroup`.

      Follow these steps to configure the engine for `group` persistence mode, and to set TIBCO Enterprise Message Service™ (EMS) as the group provider technology.

      ```
      bw.engine.persistenceMode=ftgroup
      ```

   c) Specify the group name and group provider technology in the `config.ini` file. The group name is mandatory.

      ```
      #
      ------------------------------------------------------------------------
      # Section:  BW Engine Group Configuration.
      #
      # The properties in this section are applicable to the BW Engine group.
      # Some of the properties in this section are required when the BW Engine
      # property "bw.engine.persistenceMode" is set to "group" or "ftgroup".
      #
      ------------------------------------------------------------------------
      # BW Engine Group Name.  This is a required property and it specifies name of
      ```

```
# the BW engine group.  If this property is not specified, then the group name
# defaults to "Group_<DomainName>_<AppSpaceName>".
#bw.engine.groupName=mytestgroup

# BW Engine Group Connection Provider Technology.  This is a required
property
# when the persistenceMode is set to "group" (bw.engine.persistenceMode=group)
or
# "ftgroup" (bw.engine.persistenceMode=ftgroup) and it specifies the BW Engine
group
# communication technology. The only supported values are "ems" and "ftl".
# The group connection provider technology property requires additional
configuration.
# See section "Configuring the Engine for Group Persistence Mode"
# for additional configuration.
bw.engine.groupProvider.technology=ems
```

d) Specify the group provider configuration:

```
#
--------------------------------------------------------------------------------
# Section:  BW Engine Group Connection Provider EMS Configuration.
#
# Some of the properties in this section are required when the BW Engine Group
# Connection Provider Technology property
"bw.engine.groupProvider.technology"
# value is set to "ems".
#
--------------------------------------------------------------------------------
# BW Engine Group Connection Provider EMS URL.  This property is required if
# the group provider technology is "ems".
bw.engine.groupProvider.qin.EMSServerUrl=tcp://localhost:7222

# BW Engine Group Connection Provider EMS User Name.  This property is
required
# if the group provider technology is "ems".
bw.engine.groupProvider.qin.EMSUserName=admin

# BW Engine Group Connection Provider EMS User Password.  This property is
# required if the group provider technology is "ems".
bw.engine.groupProvider.qin.EMSPassword=

# BW Engine Group Connection Provider EMS Member Prefix.  This property is
# optional and the default value is "EMSGMS".
#bw.engine.groupProvider.qin.EMSPrefix=EMSGMS

# BW Engine Group Connection Provider EMS Recovery Timeout in ms.  This
# property is optional and the default value is "5000" ms.
#bw.engine.groupProvider.qin.EMSRecoveryTimeout=5000

# BW Engine Group Connection Provider EMS Recovery Attempt Delay in ms.  This
# property is optional and the default value is "500" ms.
#bw.engine.groupProvider.qin.EMSRecoveryAttemptDelay=500

# BW Engine Group Connection Provider EMS Recovery AttemptCount.  This
# property is optional.
#bw.engine.groupProvider.qin.EMSRecoveryAttemptCount=

# BW Engine Group Connection Provider EMS Connect Attempt Count.  This property
# is optional.
#bw.engine.groupProvider.qin.EMSConnectAttemptCount=

# BW Engine Group Connection Provider EMS Connect Attempt Delay in ms.  This
# property is optional.
#bw.engine.groupProvider.qin.EMSConnectAttemptDelay=
```

When setting the password property (**bw.engine.groupProvider.qin.EMSPassword**), the default format is plain text. Execute the command **bwadmin obfuscate**, or the command **bwobfuscator**, from the command line to encrypt the password; use the generated encrypted text as the password.

2. **Optional**. The following properties are available for EMS SSL configuration.

```
EMS SSL Configuration
#client identity consisting of the certificate,
#private key and optionally extra issuer certificates can be included into a
single data block using PKCS12.
#KeyStore or Entrust Store encodings
#bw.engine.groupProvider.ems.ssl.trust.identity=

#The set of Trusted Certificates represents all trusted issuers of the server
certificate.
#It must be specified by the client application unless the host certificate
verification is completely disabled.
#bw.engine.groupProvider.ems.ssl.trust.certlocation=

#EMS SSL connection trust password. This
#property is required if the JMS server protocol is ssl. The password may
#be clear text or supplied as an obfuscated string.
#bw.engine.groupProvider.ems.ssl.trust.password=

#trusted certificate commonname must match the ems server hostname if set to
false
#bw.engine.groupProvider.ems.ssl.disable.verifyHostName=

#client and server certificates must match if set to false
#bw.engine.groupProvider.ems.ssl.trust.disable.verifyHost=
```

3. **Optional.** If you have saved the `config.ini` file to a temporary location, ensure you copy it to the AppSpace root folder located in *BW_HOME*`/domains/defaultdomain/appspaces/defaultappspace`.
4. Use the **`config`** admin command to push the configuration to the AppSpace: `bwadmin[admin] > config -d myDomain -a myAppSpace -cf <temporaryLocation>/config.ini`.

Configuring TIBCO FTL® as the FTGroup Provider for Engine

Follow these steps to configure the engine for `ftgroup` persistence mode, and to set TIBCO FTL as the group provider technology.

Use of TIBCO FTL with TIBCO ActiveMatrix BusinessWorks™ for configuring bwagent and for configuring group provider for engine does not require TIBCO FTL licenses.

Prerequisites

- See the ActiveMatrix BusinessWorks™ readme for the version of TIBCO FTL that is supported with the version of ActiveMatrix BusinessWorks 6.x you are using.
- Ensure you have installed FTL client libraries. For more information, see "Integrating with TIBCO FTL" in the *TIBCO ActiveMatrix BusinessWorks™ Installation* guide.
- These steps are only applicable if you are not using TIBCO FTL as the bwagent transport.
- If you are installing TIBCO FTL after you have already installed ActiveMatrix BusinessWorks, set the *tibco.env.FTL_HOME* variable in the `bwcommon.tra` file. You can find this file in the bin folder at *BW_HOME*\bin for Windows, or *${BW_HOME}*/bin for Unix.

1. Install TIBCO FTL. For instructions, see the *TIBCO FTL® Installation* guide.
2. Start the FTL Realm server by executing the `./tibrealmserver -ht <hostIP>:<port>` FTL command.

```
./tibrealmserver -ht <hostIP>:<port>
```

3. Execute the following FTL command to populate data in the bwadmin_ftlrealmserver.json template file, located in the config folder at *BW_HOME*`/config`:

```
 ./tibrealmadmin -rs <realmserverurl> -ur <PATH of bwadmin_ftl_realmserver.json>
```

For instructions about how to configure an FTL backup server for high availability, see "Configuring Backup Realm Servers for Fault Tolerance" in the *TIBCO FTL® Administration* guide .

Procedure

1. Set engine persistence mode property (**`bw.engine.persistenceMode`**) to `ftgroup` and configure the engine group configuration.
   a) Copy the existing AppSpace `config.ini` template file `appspace_config.ini_template` (located in *`BW_HOME`*`/config`) to the root of the AppSpace folder, or a temporary location, and rename the file as `config.ini`.
   b) Edit the ActiveMatrix BusinessWorks™ engine persistence mode property, **`bw.engine.persistenceMode`**, and set it to `ftgroup`.

```
bw.engine.persistenceMode=ftgroup
```

   c) Specify the group name and group provider technology as `ftl` in the `config.ini` file. The group name is mandatory.

```
#
--------------------------------------------------------------------------------
# Section:  BW Engine Group Configuration.
#
# The properties in this section are applicable to the BW Engine group.
# Some of the properties in this section are required when the BW Engine
# property "bw.engine.persistenceMode" is set to "ftgroup".
#
--------------------------------------------------------------------------------
# BW Engine Group Name.  This is a reqired property and it specifies name of
# the BW engine group.  If this property is not specified, then the ftgroup
name
# defaults to "Group_<DomainName>_<AppSpaceName>".
#bw.engine.groupName=mytestgroup

# BW Engine Group Connection Provider Technology.  This is a required
property
# when the persistenceMode is set to "ftgroup"
(bw.engine.persistenceMode=group)
# and it specifies the BW Engine group communication technology.  The only
# supported values are "ems" and "ftl". The group connection provider
technology property
# requires additional configuration. See section "Configuring the Engine for
Group Persistence Mode"
# for additional configuration.
bw.engine.groupProvider.technology=ftl
```

   d) Specify the group provider configuration:

```
#
--------------------------------------------------------------------------------
# Section:  BW Engine Group Connection Provider FTL Configuration.
#
# Some of the properties in this section are required when the BW Engine Group
# Connection Provider Technology property
"bw.engine.groupProvider.technology"
# value is set to "ftl"
#
--------------------------------------------------------------------------------
# BW Engine Group Connection Provider FTL Realm Server.  This property is
required if
# the group provider technology is "ftl".
bw.engine.groupProvider.ftl.realmserver=http://localhost:8080

# BW Engine Group Connection Provider FTL Realm client user name
# This property is required if the group provider technology is "ftl".
bw.engine.groupProvider.ftl.username=

# BW Engine Group Connection Provider FTL Realm client password
# This property is required if the group provider technology is "ftl".
bw.engine.groupProvider.ftl.password=

# BW Engine Group Connection Provider FTL application identifier
```

```
# This property is required if the group provider technology is "ftl".
bw.engine.groupProvider.ftl.appinstance.id=

# BW Engine Group Connection Provider FTL secondary realm server
# This property is optional.
#bw.engine.groupProvider.ftl.secondaryserver=

# BW Engine Group Connection Provider FTL group name
# This property is required if the group provider technology is "ftl".
bw.engine.groupProvider.ftl.groupname=

# BW Engine Group Connection Provider FTL application name
# This property is required if the group provider technology is "ftl".
bw.engine.groupProvider.ftl.appname=

# BW Engine Group Connection Provider FTL publish endpoint
# This property is required if the group provider technology is "ftl".
bw.engine.groupProvider.ftl.publish.endpoint=

# BW Engine Group Connection Provider FTL application name
# This property is required if the group provider technology is "ftl".
bw.engine.groupProvider.ftl.subscribe.endpoint=
```

When setting the password property (**`bw.engine.groupProvider.ftl.password`**), the default format is plain text. Execute the command **`bwadmin obfuscate`**, or the command **`bwobfuscator`**, from the command line to encrypt the password; use the generated encrypted text as the password.

2. **Optional.** If you have saved the `config.ini` file to a temporary location, ensure you copy it to the AppSpace root folder located in *`BW_HOME`*`/domains/defaultdomain/appspaces/defaultappspace`.
3. Use the **`config`** admin command to push the configuration to the AppSpace: `bwadmin[admin] > config -d myDomain -a myAppSpace -cf <temporaryLocation>/config.ini`.

Engine and Job Tuning

The engine, job tuning and checkpointing properties are specified in the `config.ini` file for each AppNode and alternatively, at the AppSpace level. The properties specified in the AppSpace `config.ini` file apply to all AppNodes associated with the AppSpace; however the properties specified in the AppNode `config.ini` file only apply to a specific AppNode and furthermore, they overwrite any property specified in the AppSpace `config.ini` file.

The TIBCO ActiveMatrix BusinessWorks™ engine is a multi-threaded engine. When events that trigger the execution of a process occur concurrently, the engine executes the same process multiple times, concurrently, once for each event. Each process execution, referred to as a *process instance*, provides an execution scope for the activities that are a part of the process.

Execution of a component process is called a *job*. When the business logic is spread across multiple processes, a process instance is created for each of these processes and executed in conjunction with a particular event. Even though these are separate process instances they are all working together and can be executed as part of the same job. A job can spawn multiple process instances and can provide the execution context for activities that are part of multiple processes. The engine always executes a job in one engine thread; however, it is not guaranteed that the same engine thread will be used for the entirety of the job.

Engine Tuning

The rate at which the ActiveMatrix BusinessWorks™ engine can execute and complete processes depends on the **`ThreadCount`** and **`StepCount`** engine properties.

`ThreadCount (bw.engine.threadCount)`: The process instances in memory are executed by the engine. The number of process instances that can be executed concurrently by the engine is limited by the maximum number of threads, indicated by the **`ThreadCount`** property. This property specifies the size of the job thread pool and is applied to all the AppNodes in the AppSpace. Threads execute a finite number of tasks or activities uninterrupted and then yield to the next process instance that is ready. Engine threads are shared by all the applications deployed on the same AppNode. The CPU and memory resources should be measured under a typical processing load to determine if the default **`ThreadCount`** is suitable for your environment. By default, the thread count is eight. See Setting Engine and Job Tuning Properties for instructions on how to change the default value.

If the engine throughput has reached a plateau, but the CPU and memory are not fully utilized, you can increase the thread count to have a positive effect on the throughput.

If the engine thread count value is too high, it can cause CPU thrashing, or an increase in latency caused by a large number of messages in the queue.If the engine thread count value is too low, it can cause higher memory use and lower engine throughput as some CPU resources remain unutilized.

The process instances created by the engine are typically held in memory. However, this may not be the case if the **`FlowLimit`** and **`PageThreshold`** application properties are set. The number of process instances that can be created in memory is also limited by the memory available on the machine and the memory allocated to the JVM on which the engine executes.

`StepCount (bw.engine.stepCount)`: The engine **`StepCount`** property determines the number of activities that are executed by an engine thread, without any interruption, before yielding the engine thread to another job that is ready in the job pool. This value is applied to all the AppNodes in the AppSpace. Exceptions to **`StepCount`** can occur when the job is in a transaction, is blocked, or is waiting for an asynchronous activity to complete. When a job is in a transaction, the thread is not released until the transaction is complete, even when the **`StepCount`** is exceeded. However, if a job is blocked or waiting for an asynchronous activity to complete, the thread can be yielded even when the **`StepCount`** has not been reached.

The default value of this property is `-1`. When the values is set to `-1`, the engine can determine the necessary StepCount value. A low **`StepCount`** value can degrade engine performance due to frequent

thread switches. A high **StepCount** value may cause less concurrency in executing jobs and hence, result in an inefficient use of CPU.

Job Tuning

Job tuning is done at the application level. Tuning can be narrowed to a specific application version, and a specific component within the application. Job tuning is set by the **FlowLimit**, **PageThreshold**, and **Priority** application properties. When setting these properties, specify the application name. The application version and component name are optional. If the version or component name is not specified, then the property value applies to all versions or components in the application. To dynamically push any of these properties to runtime, stop the application, update the property in the AppNode or AppSpace `config.ini` file, and restart the application.

Flow Limit (bw.application.job.flowlimit): The **FlowLimit** property specifies the flow limit value for an application's process starters or service bindings and is applicable to all the AppNodes in an AppSpace. Flow limit is useful when the engine needs to be throttled as it specifies the maximum number of jobs that can be started before suspending the process starter. Thus ensuring that the incoming requests do not overwhelm the engine performance and the CPU and memory is preserved. If the number of jobs being created exceeds the **FlowLimit**, the engine suspends the creation of new jobs but continues executing the jobs in memory. The engine resumes creating new jobs when sufficient resources are available. There is no default **FlowLimit** value and it is not enforced by the engine unless the **FlowLimit** property is specified for an application.

Only set the **bw.application.job.flowlimit** property for applications using non-HTTP-based transports, for example, JMS. If applications are using HTTP-based transports, ensure you set the **Max QTP Threads** value of the **HTTP Connector** shared resource to apply Flow Limit.

Get the logs for the component states such as Start, Stop, and Resume based on whether the **FlowLimit** is breached or complied by enabling the core runtime logger `com.tibco.bw.core` at INFO level

PageThreshold (bw.application.job.pageThreshold): The **PageThreshold** property specifies the job page threshold value for an application's process starters or service bindings and is applicable to all the AppNodes in an AppSpace. It specifies the maximum number of jobs that can concurrently be loaded into memory, thus limiting the number of running jobs in the memory. Jobs are paged out of memory on the basis of paging strategy selected after the **PageThreshold** value is reached. The default value of strategy is `PageOnDelete`.

There is no default **PageThreshold** value and it is not enforced by the engine unless the **PageThreshold** property is specified for an application. The **PageThreshold** feature requires that the engine persistent mode property (**bw.engine.persistenceMode**) is set to `datastore` or `group`. For more information, see Engine Persistence Modes.

Paging Strategy (bw.application.job.paging.strategy): The Paging Strategy property specifies how paging should take place.

It supports two options: `PageOnIdle` and `PageOnDelete`.

When you set the value of the property to `PageOnIdle`, and when the **PageThreshold** is reached, jobs that are idle are moved out of memory, and are paged-out to the engine database. The new or old jobs are loaded back into the memory in place of idle jobs when they are created or scheduled.

When you set the value of the property to `PageOnDelete`, all new jobs created after the **PageThreshold** value is reached are temporarily paged out to the engine database. These jobs are moved back into the memory when the number of jobs in the memory is less than the **PageThreshold** value.

Priority (bw.application.job.priority): The **Priority** property specifies the application job's priority. The option for this property is one of `low`, `normal`, or `high`. The default value is `normal`. Engine threads process lower priority jobs only when higher priority jobs are all blocked from continuing. Lower priority jobs are not preempted while in execution.

To dynamically push any of these properties to runtime, stop the application, update the property in the AppNode or AppSpace config.ini file, and restart the application.

`Retain Faulted Job (bw.application.checkpoint.retainFaultedJob)`  : This is an optional property and specifies whether to enable a failed process recovery. The supported values are `true` and `false`. The default value is `false`. The **Application Name** must be a included as a part of this property, however , the **Application Version** is optional. After setting these properties in the AppNode `config.ini` file, they can be modified and pushed to runtime by restarting the application.

`Recover On Restart (bw.application.checkpoint.recoverOnRestart)`  :This is an optional property and specifies whether the checkpointed process instances should automatically restart when a process engine restarts. The supported values are `true` and `false`. The default value is `true`. The **Application Name** must be included as part of this property, and the **Application Version** is optional.

`Duplicate key Timeout (bw.application.checkpoint.dupKeyTimeout)`: This is an optional property and specifies the amount of time in minutes to keep duplicate keys stored after the chekpointed job finished executing. By default, the timeout is `0` minutes, and indicates that the duplicate key is deleted as soon as the checkpointed job completes execution.

The ActiveMatrix BusinessWorks application name must be included as part of this property. However the application version and component name are optional. If the component name is not specified, the value is applied to all components in the ActiveMatrix BusinessWorks application.

```
bw.application.checkpoint.dupKeyTimeout.<UsersBWApplicationName>
[.<UsersBWApplicationVersion>][.<UsersBWComponentName>]=0
```

When the property **`bw.application.checkpoint.retainFaultedJob`** is set to `true` for an application, the job is not automatically removed after a failure. So the duplicateKey remains as long as the job remains. Such a job can be restarted or purged later.

You can also configure the periodic interval in which expired dupKeys should be purged from the database by configuring **`bw.engine.checkpoint.expired.dupkey.purge.interval`** property. It specifies the default interval for the background thread to poll for expired duplicate keys.

The numerical value can be preceded either by "P" or "D". A value "P60" indicates the background thread polls after every 60 minutes whereas a value "D2" indicates the background thread polls after every two days.

The default value is "P30".

Setting Engine and Job Tuning Properties

Engine and job tuning properties are specified at the **Application Level 2** > **Config** > **Job Tuning** from Admin UI and are set in the Appspace `config.ini` file. These properties are configured from the bwadmin command line and from the Admin UI at the AppNode level in the AppNode's `config.ini` file. These properties can also be specified at the AppSpace level, but the AppNode property setting takes precedence, and the App Instances go out of sync. However, the AppNode remains in a sync. If you want to apply engine and job tuning property to an application having component in a shared module, use the component name displayed in Admin UI or on CLI.

bwadmin Command Line

Execute the following commands at the command line to set the engine and job tuning properties at the AppNode level or the AppSpace level.

AppNode level

Procedure

1. Copy the existing AppNode `config.ini` file (located in the root of the AppNode folder) to a temporary location.
2. Uncomment the engine **`ThreadCount`** property **`bw.engine.threadCount`** and change the default value as needed.

   ```
   bw.engine.threadCount=8
   ```
3. Uncomment the engine **`StepCount`** property **`bw.engine.stepCount`** and change the default value as needed. The default value of -1 allows the engine to determine the necessary value.

   ```
   bw.engine.stepCount=-1
   ```
4. Uncomment the application **`FlowLimit`** property **`bw.application.job.flowlimit`** and change the default value as needed. The application name must be included in the property. The application version and component name are optional. If the component name is not specified, the **`FlowLimit`** value will apply to all the components in the application.

   ```
   bw.application.job.flowlimit.<UsersBWApplicationName>[.<UsersBWApplicationVersion
   >][.<UsersBWComponentName>]=8
   ```
5. Uncomment the application **`PageThreshold`** property **`bw.application.job.pageThreshold`** and change the default value as needed. The application name must be included in the property. The application version and component name are optional. If the component name is not specified, the **`PageThreshold`** value will apply to all the components in the application.

   ```
   bw.application.job.pageThreshold.<UsersBWApplicationName>[.<UsersBWApplicationVer
   sion>][.<UsersBWComponentName>]=10
   ```
6. Uncomment the application paging strategy property **`bw.application.job.paging.strategy`** and change the default value as needed. The application name must be included in the property. The application version and component name are optional. If the component name is not specified, the **`PageThreshold`** value will apply to all the components in the application.

   ```
   bw.application.job.paging.strategy.<UsersBWApplicationName>[.<UsersBWApplicationV
   ersion>][.<UsersBWComponentName>]=PageOnIdle
   ```
7. Uncomment the application **`Priority`** property **`bw.application.job.priority`** and change the default value as needed. The application name must be included in the property. The application version and component name are optional. If the component name is not specified, the **`Priority`** value will apply to all the components in the application.

   ```
   bw.application.job.priority.<UsersBWApplicationName>[.<UsersBWApplicationVersion>
   ][.<UsersBWComponentName>]=normal
   ```
8. Uncomment the application **`Retain Faulted Job`** property **`bw.application.checkpoint.retainFaultedJob`** and change the default value as needed. The application name must be included in the property. The application version and component name are optional. If the component name is not specified, the **`Retain Faulted Job`** value will apply to all the components in the application.

   ```
   bw.application.checkpoint.retainFaultedJob.<UsersBWApplicationName>[.<UsersBWAppl
   icationVersion>][.<UsersBWComponentName>]=false
   ```
9. Uncomment the application **`Recover On Restart`** property **`bw.application.checkpoint.recoverOnRestart`** and change the default value as needed.The application name must be included in the property. The application version and component name

are optional. If the component name is not specified, the **`Recover On Restart`** value will apply to all the components in the application.

```
bw.application.checkpoint.recoverOnRestart.<UsersBWApplicationName>[.<UsersBWAppl
icationVersion>][.<UsersBWComponentName>]=true
```

10. Save the edited file(s) and use the**`config`** admin command to push the configuration to the AppNode:

```
bwadmin[admin]> config –d myDomain -a myAppSpace –n myAppNode
    –cf <temporaryLocation>/config.ini
```

AppSpace level

Procedure

1. Copy the existing AppSpace `config.ini` file (located in the root of the AppSpace folder), or the AppSpace `config.ini` template file `appspace_config.ini_template` (located in *`BW_HOME/config/`*) to a temporary location .
2. Uncomment the engine **`ThreadCount`** property **`bw.engine.threadCount`** and change the default value as needed.

```
bw.engine.threadCount=8
```

3. Uncomment the engine **`StepCount`** property **`bw.engine.stepCount`** and change the default value as needed. The default value of -1 allows the engine to determine the necessary value.

```
bw.engine.stepCount=-1
```

4. Uncomment the application **`FlowLimit`** property **`bw.application.job.flowlimit`** and change the default value as needed. Provide the application name. The application name must be included in the property. The application version and component name are optional. If the component name is not specified, the **`FlowLimit`** value will apply to all the components in the application.

```
bw.application.job.flowlimit.<UsersBWApplicationName>[.<UsersBWApplicationVersion
>][.<UsersBWComponentName>]=8
```

5. Uncomment the application **`PageThreshold`** property **`bw.application.job.pageThreshold`** and change the default value as needed. The application name must be included in the property. The application version and component name are optional. If the component name is not specified, the **`PageThreshold`** value will apply to all the components in the application.

```
bw.application.job.pageThreshold.<UsersBWApplicationName>[.<UsersBWApplicationVer
sion>][.<UsersBWComponentName>]=10
```

6. Uncomment the application paging strategy property **`bw.application.job.paging.strategy`** and change the default value as needed. The application name must be included in the property. The application version and component name are optional. If the component name is not specified, the **`PageThreshold`** value will apply to all the components in the application.

```
bw.application.job.paging.strategy.<UsersBWApplicationName>[.<UsersBWApplicationV
ersion>][.<UsersBWComponentName>]=PageOnIdle
```

7. Uncomment the application **`Priority`** property **`bw.application.job.priority`** and change the default value as needed. The application name must be included in the property. The application version and component name are optional. If the component name is not specified, the **`Priority`** value will apply to all the components in the application.

```
bw.application.job.priority.<UsersBWApplicationName>[.<UsersBWApplicationVersion>
][.<UsersBWComponentName>]=normal
```

8. Uncomment the application **`Retain Faulted Job`** property **`bw.application.checkpoint.retainFaultedJob`** and change the default value as needed. The application name must be included in the property. The application version and component name are optional. If the component name is not specified, the **`Retain Faulted Job`** value will apply to all the components in the application.

```
bw.application.checkpoint.retainFaultedJob.<UsersBWApplicationName>[.<UsersBWAppl
icationVersion>][.<UsersBWComponentName>]=false
```

9. Uncommon the application **`Recover On Restart`** property **`bw.application.checkpoint.recoverOnRestart`** and change the default value as needed. The application name must be included in the property. The application version and component name are optional. If the component name is not specified, the **`Recover On Restart`** value will apply to all the components in the application.

```
bw.application.checkpoint.recoverOnRestart.<UsersBWApplicationName>[.<UsersBWApplicationVersion>][.<UsersBWComponentName>]=true
```

10. Save the edited file(s) and use the**`config`** admin command to push the configuration to the AppSpace:

```
bwadmin[admin] > config -d myDomain -a myAppSpace -cf
                 <temporaryLocation>/config.ini
```

Admin UI

The application properties, **`pageThreshold`** and **`FlowLimit`** can be passed dynamically from the Admin UI at the Application, AppNode and AppSpace level.

- To set the job tuning properties at the Application level, open the **Application** page, click **Configure** and click the **Job Tuning** tab and edit the **`pageThreshold`** or the **`FlowLimit`** property as required.
- To set the job tuning properties at the AppSpaces or the AppNodes level, open the **AppSpaces** or the **AppNodes** page and select the AppSpace or AppNode. Click **Configure** and then click the **General** tab to change the value under the **Current Value** column of the required property.

To delete the **`pageThreshold`** and **`FlowLimit`** properties, set the property value to zero.

Viewing Engine Properties

View engine properties, including name, step count, thread count, persistence mode, and engine state, from the bwadmin console.

To view properties, the engine must be running on an AppNode. Open the bwadmin console and enter the following command to view properties for the engine running on AppNode `AN2` in Domain `D2` and AppSpace `AS2`:

```
bwadmin show -d D2 -a AS2 -n AN2 bwengine
```

The following details are displayed:

Property	Description	More Information
Engine Name	The name of the engine running on the specified AppNode.	N/A
Engine Step Count 	The number of activities executed by an engine thread uninterruptedly before yielding the thread to another job.	Engine and Job Tuning
Engine Thread Count 	The maximum number of threads that can be executed concurrently.	Engine and Job Tuning
Engine Persistence Mode	The type of collaboration between machines.	Engine Persistence Modes
Engine State	The state of the engine.	N/A

Engine Properties

TIBCO ActiveMatrix BusinessWorks™ allows you to set engine properties at AppNode or Appspace `config.ini` file. Configure the engine by changing or assigning appropriate values to the properties. For more information, see `appspace_config.ini_template` or `appnode_config.ini_template` files at `{TIBCO_HOME}\bw\6.x\config` directory.

BW Engine General Configuration

The properties in this section are applicable to the bwengine.

Property	Description
bw.engine.threadCount	Engine thread count. It specifies the engine thread pool size. Value of this property must always be greater than 0. The default value is 8.
bw.engine.stepCount	Engine step count. It specifies the number of activities to execute for a process instance, before the bwengine yields the thread. The default value is -1.
bw.engine.separate.logs.by.app	Application level logging. It enables separate log files at the application level. The default value is false.
bw.engine.node.weight	Set the value between 1 to 99. The AppNode with highest weight is the primary node.
bw.engine.name	Specify the name of the engine. The default value is Main.
bw.engine.persistenceMode	Specifies engine execution mode. The default value is memory.
bw.engine.shutdownOnFailure	BW Engine Shutdown Option. The default value is true.
bw.engine.activity.async.waitTime	BW Asynchronous Activity Timeout. It specifies the default timeout or wait time value in milliseconds for the asynchronous activities executed by the BW Engine. The default value is 180000 (3 minutes).
bw.engine.activity.signalin.eventTimeout.purge.interval	Specifies the default interval for the background thread that looks for expired messages. The value must be specified in minutes. The default value is 30 minutes.

Property	Description
bw.engine.show.all.errors.while.application.startup	Lists all errors within an application during an application startup. The default value is false. This is a two phase process: 1. **Deserialization phase**: In this phase, all deserialization errors of processes are shown in logs. Once all of the deserilization errors are resolved, restart an application to see initialization errors. 2. **Initialization phase**: In this phase, all initialization errors related to activities are shown in logs.
bw.engine.checkpoint.expired.dupkey.purge.interval	It specifies the default interval for the background thread to poll for expired duplicate keys. The default value is "P30".
bw.engine.inline.subprocess.multiLogging.disable	Set the property to true if you do not want to list ERRORs generated by inline sub-process on console or in AppNode's log files.
bw.engine.enable.memory.saving.mode	This property, when set to true, enables usage of memory saving mode, which frees activity output variables once they are no longer needed. The default value is false. This property is applicable only when you select the **Window** > **Preferences** > **BusinessWorks** > **Process Diagram** > **Save information to support memory saving mode** check box when building a process for new projects in TIBCO Business Studio™ for BusinessWorks™. To update existing projects, use repair tool.

BW Engine Database Configuration

The properties in this section are applicable to the BW Engine database. All properties in this section are mandatory, when the BW Engine property **`bw.engine.persistenceMode`** is set to `datastore` or `group`.

Property	Description
bw.engine.db.jdbcDriver	The bwengine database driver.
bw.engine.db.url	The bwengine database URL.
bw.engine.db.userName	The bwengine database user name.

Property	Description
bw.engine.db.password	The bwengine database password.
bw.engine.db.maxConnections	The number of connections that can be made to the bwengine database.

BW Engine Group Configuration

The properties in this section are applicable to the BW Engine group. Some of the properties in this section are mandatory when the BW Engine property **`bw.engine.persistenceMode`** is set to `group` or `ftgroup`.

Property	Description
bw.engine.groupName	It specifies name of the BW engine group. If this property is not specified, then the group name defaults to Group_<DomainName>_<AppSpaceName>"
bw.engine.group.ats.timeout	BW Engine Active to Standby Timeout property specifies the time to wait (in seconds) before force stopping an application on an AppNode that is transitioning from active to standby state. The default value is 60 secs. If the value is set to 0, it indicates the appnode waits till the application is gracefully stopped.
bw.engine.groupProvider.technology	BW Engine Group Connection Provider Technology. This is a required property when the **`bw.engine.persistenceMode`** property is set to `group` or `ftgroup`. The supported values for the **`bw.engine.groupProvider.technology`** property are `ems` and `ftl`.

BW Engine ftgroup Properties

Property	Description
bw.engine.ftgroup.lbmode	When this property is `false`, all processes run on a single engine in the group. One of the engines in the group takes over in the event the primary engine fails. When this property is `true`, all Multiple AppNodes processes run on all the engines in the group. This property only applies when the BW Engine property **`bw.engine.persistenceMode`** is set to `ftgroup`. The default value is `false`.

Property	Description
bw.engine.use.weighted.node	Indicates whether or not node weights should be used. The property is applied at the AppSpace level. When the property is set to `true`, the node with the highest weight is chosen as the primary node in the group. Use the property **`bw.engine.node.weight`** in the node config to specify that node's weight. This property only applies when the BW Engine property **`bw.engine.persistenceMode`** is set to `ftgroup`. When you want to elect an AppNode as a leader AppNode, then set the **`bw.engine.use.weighted.node`** property to `true` at an AppSpace level. The default value is `false`.

BW Event Configuration Properties

The properties in this section are applicable to the BW Event Publisher and various BW Event Subscribers that consume the generated events.

Property	Description
bw.engine.event.publisher.enabled	Enable or disable the BW Engine Event Publisher property specifies whether BW Engine Event Publisher should be enabled or disabled in the BW Engine. The default value is `true`.

BW Engine Group Connection Provider EMS Configuration

Some of the properties in this section are mandatory when the BW Engine Group Connection Provider Technology property **`bw.engine.groupProvider.technology`** value is set to `ems`

Property	Description
bw.engine.groupProvider.qin.EMSServerUrl	Mandatory. The bwengine Group Connection Provider EMS URL.
bw.engine.groupProvider.qin.EMSUserName	Mandatory. The bwengine Group Connection Provider EMS User Name.
bw.engine.groupProvider.qin.EMSPassword	Mandatory. The bwengine Group Connection Provider EMS User Password.

Property	Description
bw.engine.groupProvider.qin.EMSPrefix	The bwengine Group Connection Provider EMS Member Prefix. The default value is EMSGMS.
bw.engine.groupProvider.qin.EMSRecoveryTimeout	The bwengine Group Connection Provider EMS Recovery Timeout in milliseconds. The default value is 5000.
bw.engine.groupProvider.qin.EMSRecoveryAttemptDelay	The bwengine Group Connection Provider EMS Recovery Attempt Delay in milliseconds. The default value is 500.
bw.engine.groupProvider.qin.EMSRecoveryAttemptCount	The bwengine Group Connection Provider EMS Recovery AttemptCount.
bw.engine.groupProvider.qin.EMSConnectAttemptCount	The bwengine Group Connection Provider EMS Connect Attempt Count.
bw.engine.groupProvider.qin.EMSConnectAttemptDelay	The bwengine Group Connection Provider EMS Connect Attempt Delay in milliseconds.
bw.engine.groupProvider.ems.ssl.trust.identity	EMS ssl configuration client identity consisting of the certificate, private key and optionally extra issuer certificates can be included into a single data block using PKCS12, KeyStore or Entrust Store encodings.
bw.engine.groupProvider.ems.ssl.trust.certLocation	The set of Trusted Certificates represents all trusted issuers of the server certificate.
bw.engine.groupProvider.ems.ssl.trust.password	EMS SSL connection trust password.
bw.engine.groupProvider.ems.ssl.disable.verifyHostName	Trusted certificate common name must match the ems server hostname if set to false.
bw.engine.groupProvider.ems.ssl.trust.disable.verifyHost	The client and server certificates must match if set to false.

BW Engine Group Connection Provider FTL Configuration

Some of the properties in this section are mandatory when the BW Engine Group Connection Provider Technology property **`bw.engine.groupProvider.technology`** value is set to `ftl`

Property	Description
bw.engine.groupProvider.ftl.realmserver	Mandatory. BW Engine Group Connection Provider FTL Realm Server. The default value is http://localhost:8080

Property	Description
bw.engine.groupProvider.ftl.username	Mandatory. The bwengine Group Connection Provider FTL Realm client user name.
bw.engine.groupProvider.ftl.password	Mandatory. The bwngine Group Connection Provider FTL Realm client password.
bw.engine.groupProvider.ftl.appinstance.id	Mandatory. The bwengine Group Connection Provider FTL application identifier.
bw.engine.groupProvider.ftl.secondaryserver	The bwengine Group Connection Provider FTL secondary realm server.
bw.engine.groupProvider.ftl.groupname	Mandatory. The bwengine Group Connection Provider FTL group name.
bw.engine.groupProvider.ftl.appname	Mandatory. The bwengine Group Connection Provider FTL application name.
bw.engine.groupProvider.ftl.publish.endpoint	Mandatory. The bwengine Group Connection Provider FTL publish endpoint.
bw.engine.groupProvider.ftl.subscribe.endpoint	Mandatory. The bwengine Group Connection Provider FTL application name.
bw.engine.groupProvider.ftl.client.retries	Use the property to set the FTL property `TIB_REALM_PROPERTY_LONG_CONNECT_RETRIES`. The default value is 5. To retry forever, set the value to 0. If the connect call cannot connect to the FTL server after the maximum number of connection attempts, an exception is displayed.

Governance and Monitoring

Enable the agents in the AppNode to monitor applications, to enforce policies, to view statistics, and monitor process instances for an environment in ActiveMatrix BusinessWorks.

Monitoring Processes

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

All the process instances in the application are grouped by packages, and you can monitor the status of the process instances and subprocesses that were sucessfully executed, cancelled 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 from the `bwagent_as.json`, `bwagent_db.json`, or `bwagent_ftl.json` files, set the property `statsprovider` to `true` to enable process monitoring.

You can use same or different databases for process monitoring.

Process Monitoring

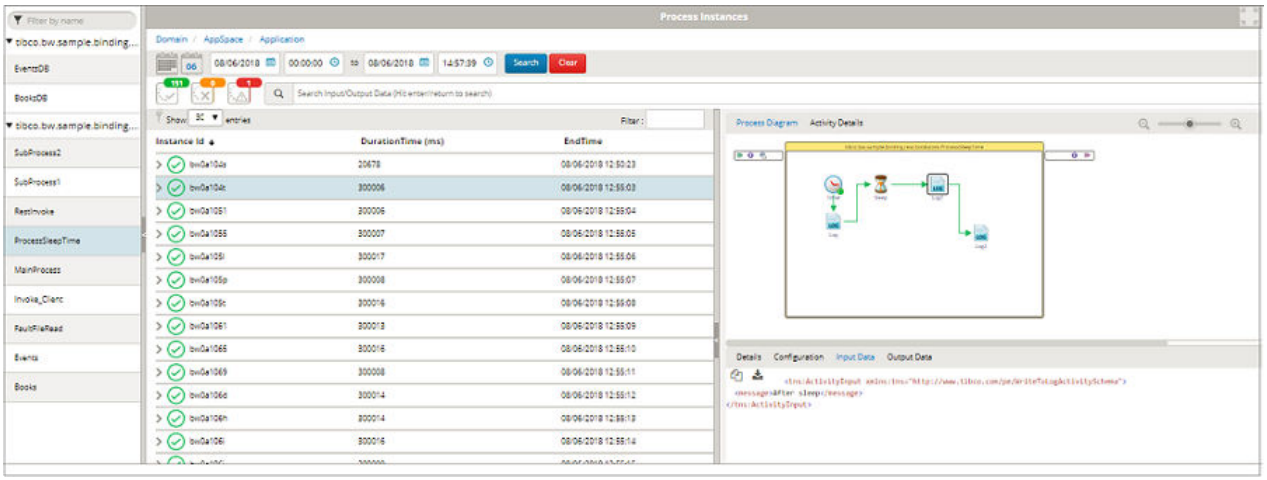
The transport layers communicate between the bwagent and the AppNode, and the supported transport layers are UDP and FTL. Set the **`statstransport`** property to UDP. This is the default setting in the agent and the AppNode.

The **`statstransport`** property is also added to the AppNode config file when the AppNode is created. For more information about UDP and FTL configurations, see Configuring using UDP and Configuring using FTL respectively.

To access the process monitoring landing page, go to the **Application Level 2** page, navigate to the **Process** tab, click the **Graph View** icon  and click the **Process Instance** icon  . You can also use the shortcut key Shift + P to navigate to the process monitoring landing page directly.

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 Instance icon after deploying the application.



By default, all the instances in the selected process are displayed. Subprocesses, if any, can be viewed by expanding the process.

Double click navigation is supported up to 9 subprocesses, and will only work for direct subprocesses. Service subprocesses are not supported.

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

Additional Features

In the Admin UI the following details are displayed in the default view.

- **Instance Id** - displays all the (instance ids of the) process instances.
- **DurationTime (ms)** - displays the total time taken to execute the process instance (in milliseconds)
- **EndTime**- 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, **AppNodes**, **StartTime**, and **DurationTime (ms)**.

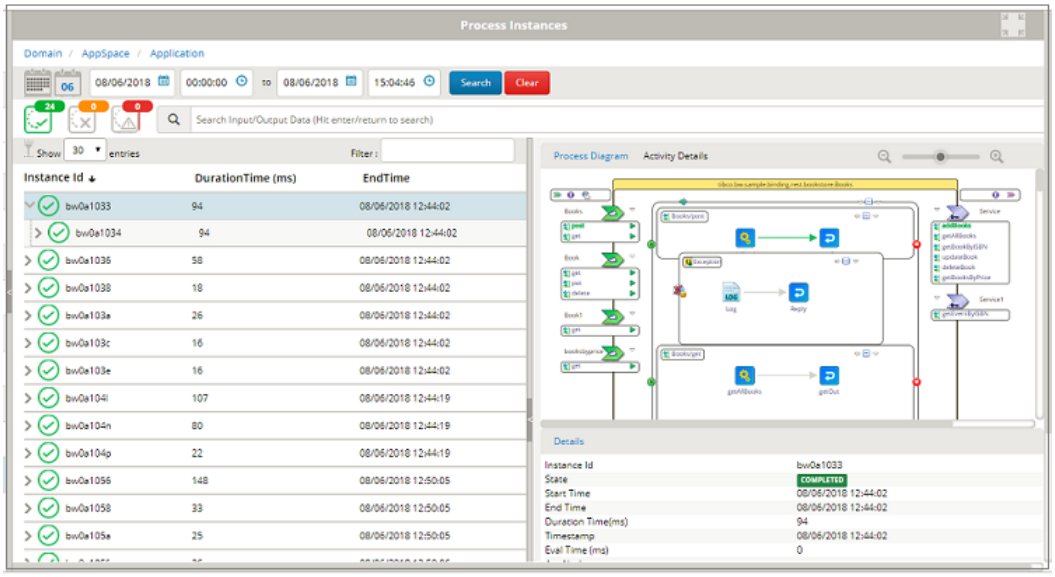
The other filters provided in Admin UI are:

- Date filters - Job data can be filtered for a particular date or a time range. Alternatively, the Calendar icon filters and displays job data for Last Month and Today.
- 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 were cancelled and  filters the jobs that faulted.
- Search Input/Output Data - This filter searches thorough the input and output data and displays the required information.
- Filter - This filter searches through the instances that are displayed for any specific value provided in this filter text box.

Navigating through the UI

To navigate through the Domain quickly, the Admin UI also provides breadcrumb navigation. The **Domain** link in the breadcrumb navigation, **Domain/ AppSpaces/ Application** navigates to the page where all the applications within the domain are dispalyed. The **AppSpace** link navigates to the **AppSpace level 2** > **Application** tab, and the **Application** link navigates to the page where all the application instances for all the processes are displayed.

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

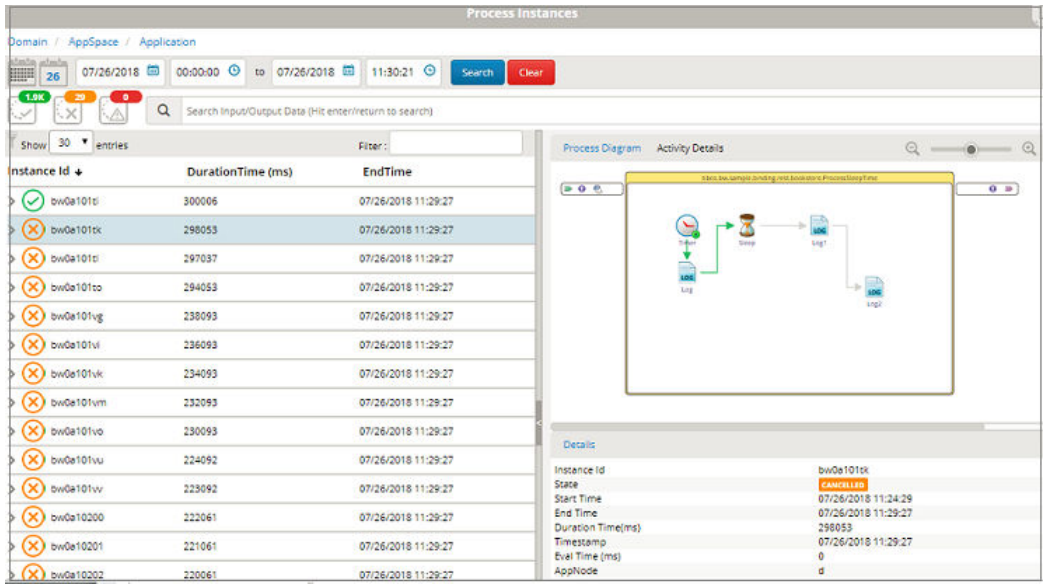


Selecting any instance from the table highlights the reference or service, if any, and also highlights in green, the activity transitions that were successfully executed.

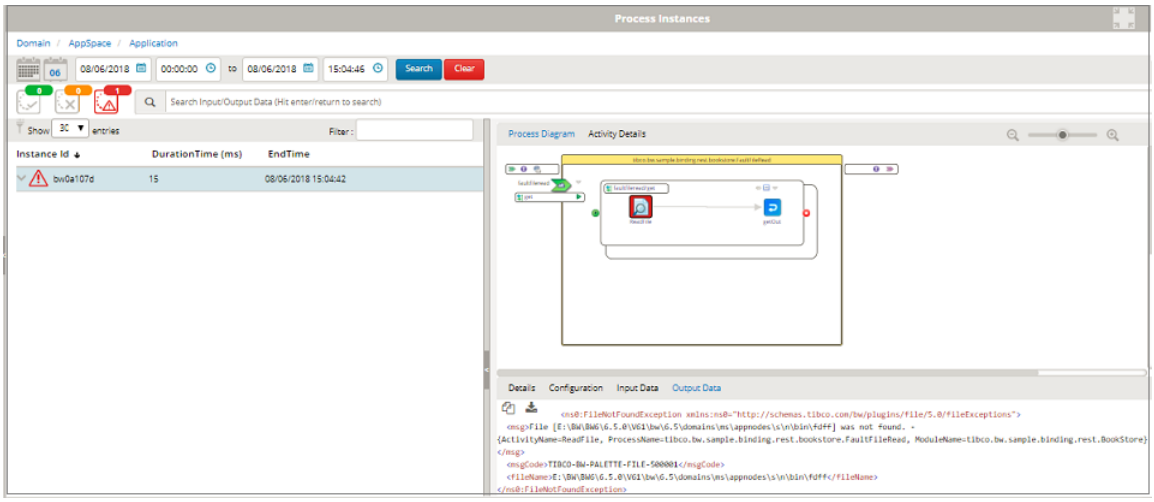
When a process contains multiple constructors, the activities and transitions in the constructor are not visible in the Admin UI when the constructor is minimized while creating the EAR file. Expand the constructors and regenerate the EAR file to view the transitions inside constructors.

Fix any ActivityID related warnings displayed in Studio , and then create the EAR file to ensure the plotting and Input and Output data is displayed correctly.

The process monitoring for cancelled jobs displays the successful transctions in green up until the point where the process was successfully executed.



Processes that faulted due to errors are highlighted in red.



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

Configuring using UDP

You can configure process monitoring using UDP.

Procedure

1. Update the following properties in the bwagent configuration files, `bwagent_as.json`, `bwagent_db.json` or `bwagent_ftl.json` based on your bwagent configuration.

   - Set **`statstransport`** to `UDP`
   - Set **`statsprovider`** to `true`
   - Set **`dbprovidertype`** to *<db type>*
   - Set **`dbproviderdriver`** to *<db provider>*
   - Set **`dbproviderconnectionurl`** to *<db connection url>*
   - Set **`dbprovideruser`** to *<db user>*
   - Set **`dbproviderpassword`** to *<db password>*

   Run the bwadmin config command with the **`-cf`** option to push the changes from the bwagent configuration JSON file to the `bwagent.ini` file.

   - To start the bwagent in the dbems mode

     ```
     BW_HOME\bin>bwadmin config -cf ../config/bwagent_db.json agent
     ```

   - To start the bwagent in the dbftl mode

     ```
     BW_HOME\bin>bwadmin config -cf ../config/bwagent_ftl.json agent
     ```

2. Start the bwagent and create the AppNode(s).
3. To ensure minimum data loss, set the values of the following two AppNode properties on the lower side: For example, you can set **`bw.monitor.publishtimer= 1500`** and **`bw.monitor.batchsize= 1`**.

   batchsize: Process Monitoring data is published in batches. This property specifies batch size for the data.

   publishtimer: This property specifies the time interval for publishing Process Monitoring data.

   The criteria that is fulfilled first will take effect.

You can configure the AppNode properties using the Configure icon from the Admin UI AppNode level 2 page or from the BWAppNode's `config.ini` file.

Ensure the monitor provider property (**`bw.monitor.provider=UDP`**) is present in the the AppNode `config.ini` file and in the `bwagent.ini` file.

4. Start the Appnode(s).
5. Upload and deploy the Application. Start the Application.
6. You can enable pocess monitoring for any particular application by navigating to the Application Level 2 page, turning the **Process Monitor** button **ON** and restarting the Application.
7. Navigate to **Application Level 2** > **Processes** > **Graph View** > **Process Instance**. Alternatively you can also use the shortcut key Shift + P from the Application Level 2 Page to directly open the Process Monitoring landing Page.

   To enable process monitoring form CLI, run the following commands:

   ```
   bwadmin[admin]> enablestats -d D -a AS processinstance b1.application 1.0
   TIBCO-BW-ADMIN-300413: Enabled statistics collection for Application
   [b1.application:1.0].

   bwadmin[admin]> enablestats -d D -a AS activityinstance b1.application 1.0
   TIBCO-BW-ADMIN-300413: Enabled statistics collection for Application
   [b1.application:1.0].
   ```

   Alternatively, you can run the single command as follows:

   ```
   bwadmin[admin]> enablestats -d D -a AS processmonitor b1.application 1.0
   TIBCO-BW-ADMIN-300413: Enabled statistics collection for Application
   [b1.application:1.0].
   ```

8. Select any activity in the process diagram and verify the details in the Details, Configurations, Input data or the Output data tabs.

   In your production environment, it is recommended to use TIBCO FTL®.

Configuring using FTL

You can configure process monitoring using TIBCO FTL ®.

Procedure

1. Download the latest FTL driver and install TIBCO FTL®. Refer to the TIBCO FTL® *Installation* guide for installation instructions. Set the property in `bwcommon.tra -tibco.env.FTL_HOME=<FTL_HOME>` and install the FTL driver using the bwinstall utility.
   The *FTL_HOME* path provided should be till the version folder. For example, `tibco.env.FTL_HOME=/opt/tibco/ftl/5.4`
2. Start the FTL realm server using ./tibrealmserver -ht <*IP address*>:8080.
3. Execute the FTL command, `-./tibrealmadmin -rs http://<IP address>:8080 -ur <PATH of bwadmin_ftlrealmserver.json>`.
   Two applications are created on the FTL server.
4. Update the following properties in the `bwagent_db.json, bwagent_as.json or bwagent_ftl.json` file, based on your bwagent configuration.

   - Set **`statstransport`** to FTL
   - Set **`statsdataformat`** to bytestream
   - Set **`statsprovider`** to `true`
   - Set **`dbprovidertype`** to <*db type*>
   - Set **`dbproviderdriver`** to <*db provider*>
   - Set **`dbproviderconnectionurl`** *to <db connection url>*

- Set **dbprovideruser** to *<db user>*
- Set **dbproviderpassword** to *<db password>*
- Set **statsftlrealmserverurl** to http://*<IP Address>*[:port]

  In case of FTL 6.x server in FT mode, set multiple realmserver values separated by pipe. (|).

  For example: `bw.agent.technology.dbftl.ftl.realmserver= http://10.97.240.76:8050 | http://10.97.240.76:8051 | http://10.97.240.76:8052`

  If any of the configuration settings are different from the default settings, update the following additional properties as applicable.

- **statsftlapplicationname**
- **statsftlsecondaryurl**

  This property is only applicable for FTL 5.x. To use this property for FTL 6.x, set the **statsftlsecondary** to `true`.

  By default, the **statsftlsecondary** property is set to `false`.

- **statsftlusername**
- **statsftluserpassword**
- **statsftlendpoint**
- **statsftldataformat**
- **statsftlinbox**

Run the bwadmin config command with the **-cf** option to push the changes from the bwagent configuration JSON file to the `bwagent.ini` file.

- To start the bwagent in the dbems mode

  BW_HOME\bin>bwadmin config -cf ../config/bwagent_db.json agent

- To start the bwagent in the dbftl mode

  BW_HOME\bin>bwadmin config -cf ../config/bwagent_ftl.json agent

5. Start the bwagent and create the AppNode(s).

   Ensure the monitor data format property (**bw.monitor.data.format=bytestream**) and the monitor provider property (**bw.monitor.provider=FTL**) are present in the the AppNode `config.ini` file and in the `bwagent.ini` file.

6. Upload and deploy the Application. Start the Application.
7. You can enable pocess monitoring for any particular application by navigating to the Application Level 2 page, turning the **Process Monitor** button **ON** and restarting the Application.

   To enable process monitoring form CLI, run the following commands:

```
bwadmin[admin]> enablestats -d D -a AS processinstance b1.application 1.0
TIBCO-BW-ADMIN-300413: Enabled statistics collection for Application
[b1.application:1.0].

bwadmin[admin]> enablestats -d D -a AS activityinstance b1.application 1.0
TIBCO-BW-ADMIN-300413: Enabled statistics collection for Application
[b1.application:1.0].
```

   Alternatively, you can run the single command as follows:

```
bwadmin[admin]> enablestats -d D -a AS processmonitor b1.application 1.0
TIBCO-BW-ADMIN-300413: Enabled statistics collection for Application
[b1.application:1.0].
```

8. Navigate to **Application Level 2** > **Processes** > **Graph View** > **Process Instance**.

   Alternatively, you can also use the shortcut key Shift + P from the Application Level 2 Page to directly open the Process Monitoring landing Page.

Configuring with CSV

You can now get activity and process statistics in .csv files.

Procedure

1. Set the property `bw.monitor.provider=csv` in the `AppNode.config` file
2. Set the property **`statsprovider:`** to `true` in `bwagent_db.json` and `bwagent_ftl.json` files. By default, the property value is false.

   The generated .csv file does not contain input and output data. It contains limited data set as follows:

   `processstats.csv` contains the following information:

   - Application Name
   - Application Version
   - Module Name
   - Module Version
   - Component Process Name
   - JobId
   - Parent Process Name
   - ParentProcessInstanceId
   - Process Name
   - ProcessInstanceId
   - Start Time (Milliseconds)
   - End Time (Milliseconds)
   - Elapsed Time (Milliseconds)
   - Evaluation Time (Milliseconds)
   - Status

   `activitystats.csv` contains the following information:

   - Application Name
   - Application Version
   - Module Name
   - Module Version
   - Activity Name
   - Process Name
   - ProcessInstanceId
   - Start Time (Milliseconds)
   - Elapsed Time (Milliseconds)
   - Evaluation Time (Milliseconds)
   - Status

Application Statistics Collection

You can collect three types of statistics for an application: application job metrics, process statistics, and execution statistics.

For more information, see the following sections:

- Application job metrics - Application Metrics
- Process instrumentation statistics for ActiveMatrix BusinessWorks 6.x processes and activities - Process Statistics
- Execution statistics for ActiveMatrix BusinessWorks 6.x processes and activities - Process Execution Statistics

Application Metrics

Application metrics provides the job statistics of an application. You can view application metrics from the command line, or from the Admin UI.

Command Line

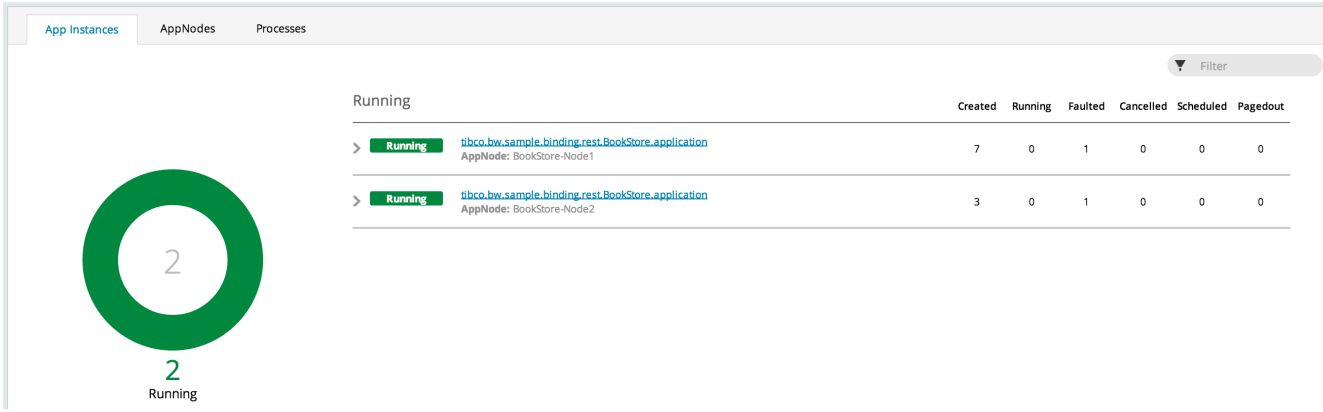
To view application metrics from the command line execute the Admin CLI command **`show metrics`**.

The application metrics collection functionality is enabled by default, but can be disabled by setting the property `,bw.frwk.event.subscriber.metrics.enabled`, to `false` in the AppSpace `config.ini` file. Use the bwadmin **`config`** command to push this configuration to the AppSpace.

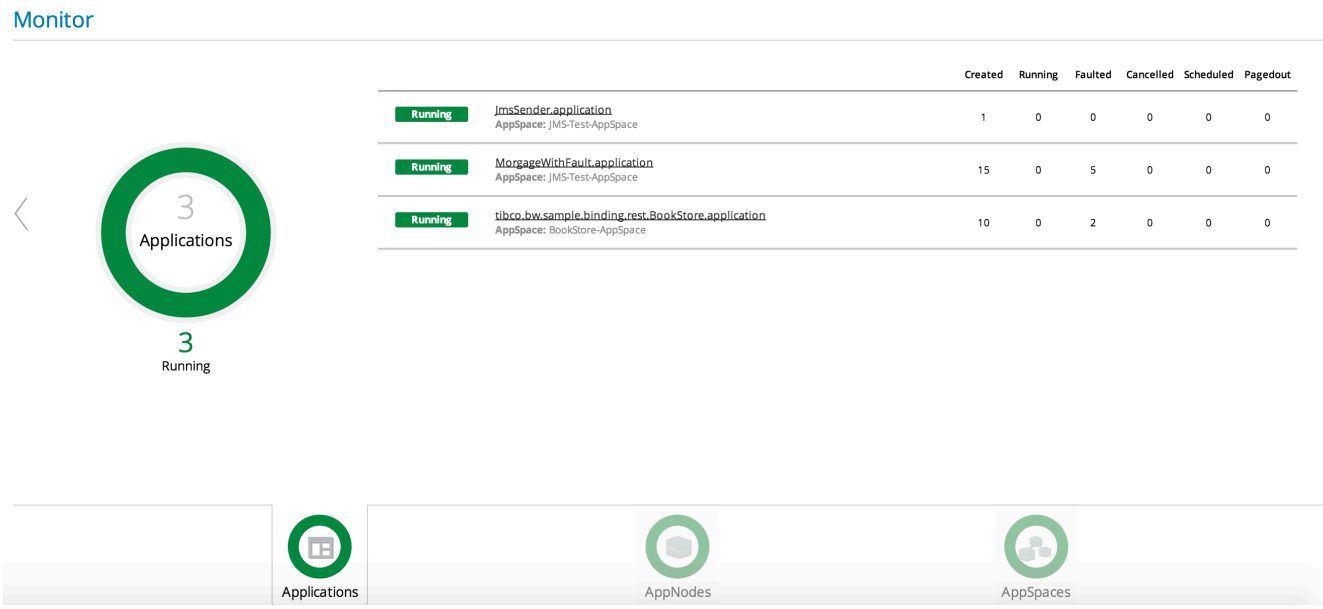
Admin UI

You can view application metrics in the Admin UI from the **Monitoring** tab, or the **Applications** tab.

To view job count statistics for multiple applications, click the **Monitoring** tab.



To view job count statistics for a specific application, click the **Applications** tab, select the application you want to view, and switch to the monitoring view by clicking the  icon.



Process Statistics

Process statistics collection can be enabled or disabled from the command line, or from the Admin UI.

Enabling and Disabling Process Statistics

Process statistics can be enabled and disabled from the command line or from the Admin UI.

Command Line

Enable or disable the collection of process statistics for all applications in an AppNode by using the **`enablestats`** and **`disablestats`** commands respectively.

To view command syntax, use the help option on the bwadmin console. For example, enter **`help enablestats`** or **`help disablestats`** to view the syntax for the **`enablestats`** command or the **`disablestats`** command.

To enable collection of statistical data for all processes running in an AppNode at the time of an application startup, set the **`bw.frwk.event.subscriber.instrumentation.enabled`** property to `TRUE` in the AppSpace `config.ini` file.

If you set the property to `FALSE` the process instrumentation statistics is disabled at the time of an application startup.

If the property is not set, the previous state of the process instrumentation persists.

- **Enable Process Statistics**
  - To enable process statistics for all applications on a single AppNode, execute the following command:
    ```
    enablestats -d defaultdomain -a MyAppSpace -n MyAppNode process
    ```
  - To enable process statistics for a single application on an AppNode, execute the **`enablestats`** command, and specify the application name and version. In the following example, the application `testApp` and version `1.0` are provided in the command syntax:
    ```
    enablestats -d defaultdomain -a MyAppSpace process testApp 1.0
    ```
- **Disable Process Statistics**

- To disable process statistics for all applications on a single AppNode, execute the following command:

```
disablestats -d defaultdomain -a MyAppSpace -n MyAppNode process
```

- To disable process statistics for a single application on an AppNode, execute the **`disablestats`** command, and specify the application name and version. In the following example, the application `testApp` and version `1.0` are provided in the command syntax:

```
disablestats -d defaultdomain -a MyAppSpace process testApp 1.0
```

Admin UI

Application statistics collection can be enabled or disabled from the Admin UI by setting the following properties.

Property	Description
Process Instrumentation	Click **ON** to enable process instrumentation data collection. To enable data collection of an application that is running on multiple Appnodes, click on the **Application** tab, and enable the **Process Instrumentation** property. Process instrumentation statistics will be collected for all application instances. To enable data collection of all applications running on an AppNode, click on the AppNodes tab, and enable the **Process Instrumentation** property. Process instrumentation statistics will be collected for all applications running on the specified AppNode.
Process Monitor	Click **ON** to enable process monitoring to view the process instances. To enable process monitoring of an application that is running on multiple Appnodes, click on the **Application** level 2 page, and click the Process Monitor button. Process monitoring will be enabled for all application instances. To enable process monitoring of all applications running on an AppNode, click on the AppNodes tab and click the Process Monitor button. Process monitoring will be enabled for all applications running on the specified AppNode.

Viewing Collected Statistics

You can view process statistics through the command line, or the Admin UI.

Prerequisites

Ensure that you have enabled process statistics collection. For more information about how to do this, see Enabling and Disabling Process Statistics .

Command Line

You can use the following commands on the OSGi console to view collected statistics:

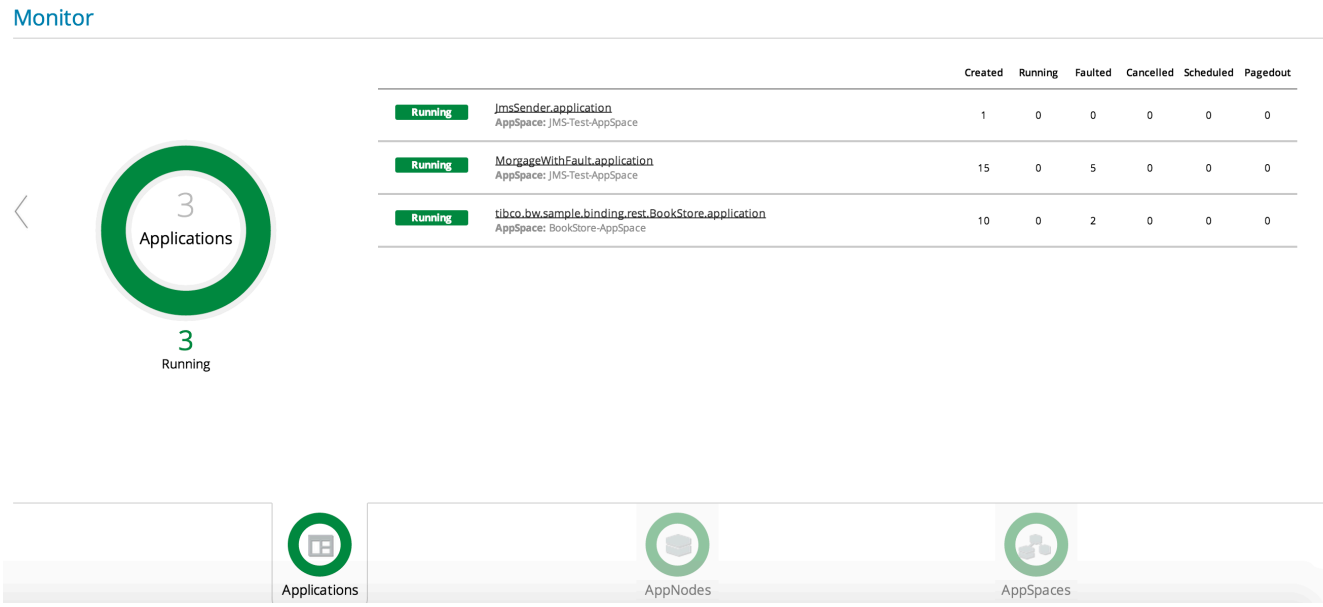
- bw:lpis to print statistics of processes that have been executed for the application.
- bw:lais to retrieve statistics for activities that have been executed in processes for the application.

Admin UI

For details about how to view application, process, and activity data from the Admin UI, see the following sections.

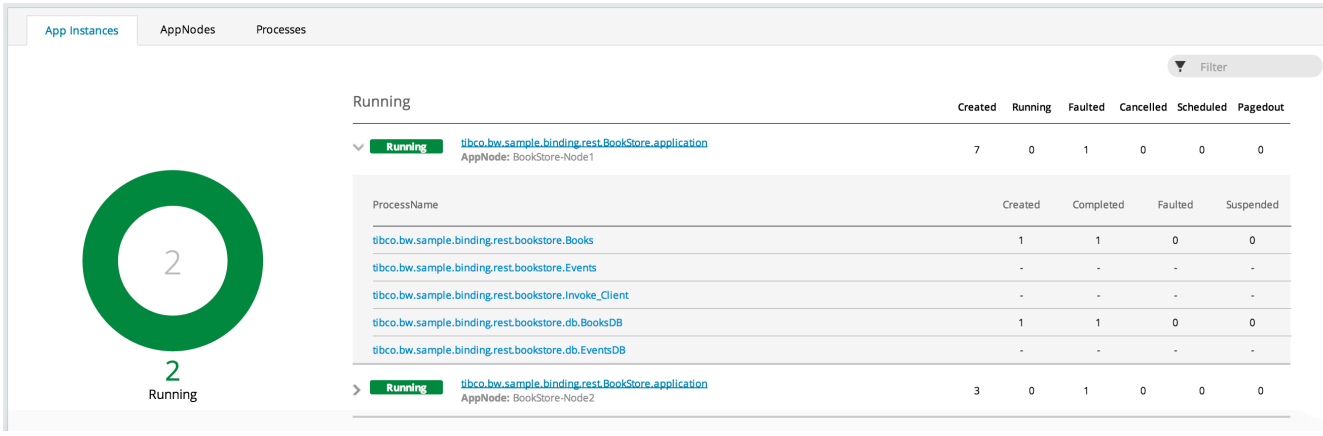
View Application Data

To view application job counts on each AppNode, select the **Application** tab, and click the icon to switch to the Monitoring View.

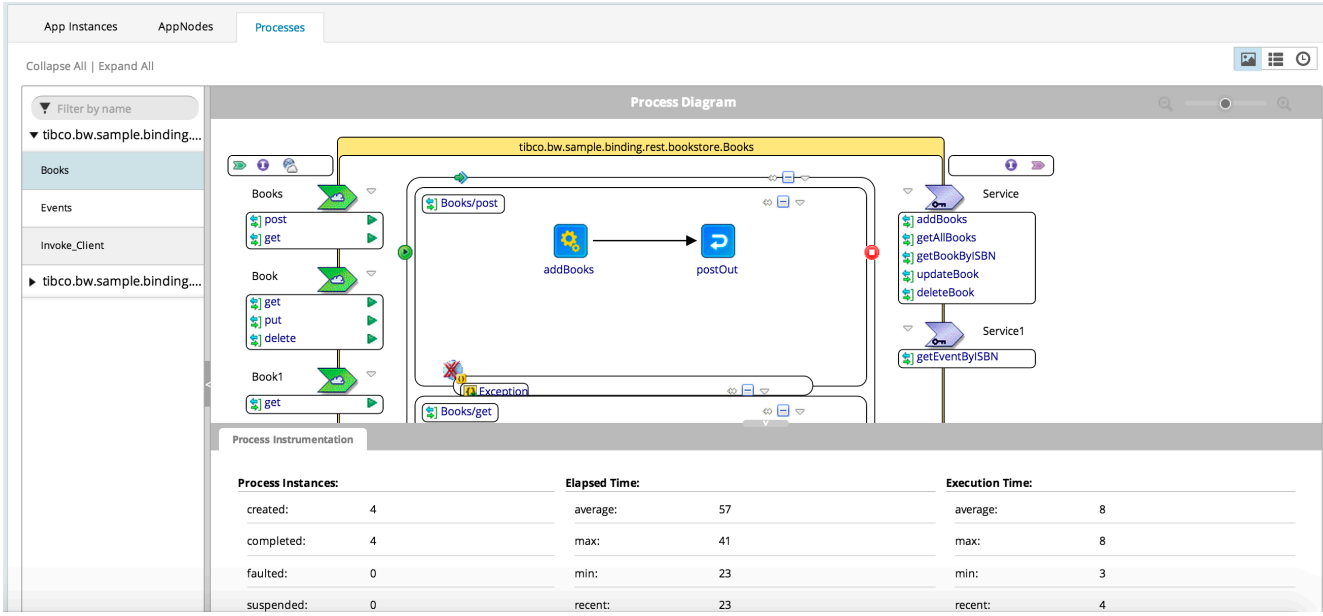


View Process Data

Ensure you are on the **Application** tab, and click the icon to switch to the Monitoring View. Next, select an application and expand it to view job process counts.

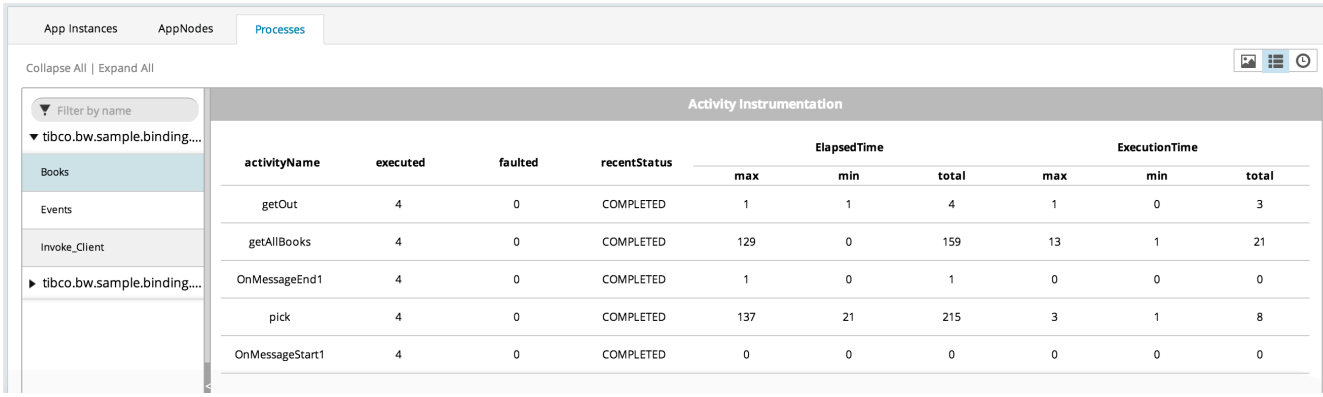


To view process instrumentation data, click on an individual process. The Admin UI switches to the **Processes** tab, and the process diagram, along with process instrumentation data, displays.



View Activity Data

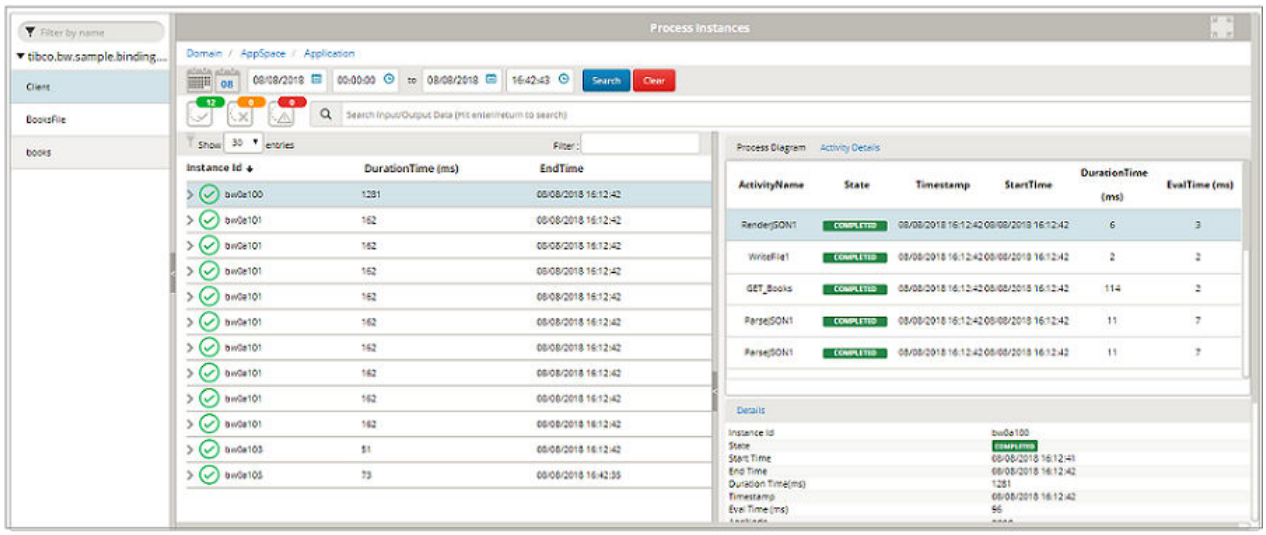
Select the **Application** tab, click the icon to switch to Monitoring View, and select the **Processes** tab to view the process diagram. From this point you can view activity instrumentation data by clicking on an activity in the process diagram, or clicking the icon to the top left corner of the **Processes** tab.



View Process and Activity Instance Data Logged to an External Database

Before you can view process and activity instance data that has been logged to an external database, complete the steps outlined in the section Writing Process Statistic Data to an External Database, and enable process and activity instance statistics collection from the Admin UI.

Ensure you are on the **Application** tab, and click the icon to switch to Monitoring View. Expand an application to view the individual processes, and select one. Next, click the **Processes** tab, and click the icon, located at the top left corner of the **Processes** tab, to view process instance details. Click on a process instance ID to view its activity instances.



Process Execution Statistics

The following process execution statistics are collected by logback.

Activity Instance Statistics

Activity Execution Statistics

Statistic	Description
Application Name	Name of application.
Application Version	Version of application.

Statistic	Description
Module Name	Name of BW module.
Module Version	Version of BW module.
Activity Name	Name of the activity.
Process Name	Name of the process.
Process Instance ID	Instance ID of the process.
Start Time	Activity start time (in milliseconds).
Duration Time	Total time (in milliseconds) taken by activity.
Eval Time	Total evaluation time (in milliseconds) taken by the activity.
Status	Status of activity, for example: Completed/Faulted/Canceled
Domain	Name of the domain.
AppSpace	Name of the AppSpace.
AppNode	Name of the AppNode.

Process Instance Statistics

Process Instance Execution Statistics

Statistic	Description
Application Name	Name of application.
Application Version	Version of application.
Module Name	Name of BW module.
Module Version	Version of BW module.
Component Process Name	Name of process configured to a component. If the process is a non in-lined subprocess, this could be empty.
Job ID	Job ID of the process.
Parent Process Name	If the process is an in-lined subprocess, the name of the parent process.
Parent Process Instance ID	If the process is an in-lined subprocess, the instance ID of the parent process.

Statistic	Description
Process Name	Name of process.
Process Instance ID	Instance ID of the process.
Start Time	Process instance start time.
End Time	Process instance end time.
Duration Time	Total time (in milliseconds) taken by the process instance to finish.
Eval Time	Total evaluation time (in milliseconds) for all activities executed for this process instance.
Status	Status of process instance, for example: Completed or Faulted

Data is written as comma separated values.

Integrating Execution Statistics Collection Using Logback

Edit the `logback.xml` to integrate execution statistics collection.

Procedure

1. Upload and deploy the application to an AppNode.
2. To enable activity execution statistics and process execution statistics, run the following commands from the Admin CLI:

   ```
   enablestats activityinstance appname appversion
   ```

   ```
   enablestats processinstance appname appversion
   ```
3. To disable activity execution statistics and process execution statistics, run the following commands from the Admin CLI:

   ```
   disablestats activityinstance appname appversion
   ```

   ```
   disablestats processinstance appname appversion
   ```
4. To retrieve execution statistics for a specific process, run the following command from the Admin CLI:

   ```
   enablestats -bp processname processinstance appname appversion
   ```
5. To retrieve execution statistics for a specific activity in a specific process, run the following command from the Admin CLI:

   ```
   enablestats -bp processname -ba activityname activityinstance appname appversion
   ```
6. By default, statistics are collected in the following files:
   - Activity statistics: *AppNode_root*`/stats/activitystats.csv`
   - Process statistics: *AppNode_root*`/stats/processstats.csv`
7. To customize activity statistics collection, go to *AppNode_root*`/logback.xml` and configure the following appender:

   ```
   <appender name="activityStatsFileAppender"
   class="ch.qos.logback.core.rolling.RollingFileAppender">
      <File>../log/activitystats.log</File>
        <encoder>
          <Pattern>%msg%n</Pattern>
   ```

```
     </encoder>
   <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
      <FileNamePattern>../log/activitystats.%d{yyyy-MM-dd}.log</FileNamePattern>
   </rollingPolicy>
</appender>

<logger name="com.tibco.bw.statistics.activity" additivity="false">
  <level value="INFO"/>
  <appender-ref ref="activityStatsFileAppender" />
</logger>
```

a) To write the log as a formatted HTML file, add the following file appender to the APPENDER: File Appender section of the `logback.xml` file.

```
<appender name="activityStatsFileAppender"
class="ch.qos.logback.core.FileAppender">
<File>../log/activitystats.html</File>
<encoder class="ch.qos.logback.core.encoder.LayoutWrappingEncoder">
<layout class="com.tibco.bw.logback.layout.ActivityExcecutionStatsHTMLLayout"/>
</encoder>
</appender>

<logger name="com.tibco.bw.statistics.activity" additivity="false">
<level value="INFO"/>
<appender-ref ref="activityStatsFileAppender" />
</logger>
```

8. To customize process statistics collection, go to *AppNode_root*`/logback.xml` and configure the following appender:

```
<appender name="processinstanceStatsFileAppender"
class="ch.qos.logback.core.rolling.RollingFileAppender">
   <File>../log/processinstancestats.log</File>
     <encoder>
       <Pattern>%msg%n</Pattern>
     </encoder>
   <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
      <FileNamePattern>../log/processinstancestats.%d{yyyy-MM-dd}.log</
FileNamePattern>
   </rollingPolicy>
</appender>

<logger name="com.tibco.bw.statistics.processinstance" additivity="false">
  <level value="INFO"/>
  <appender-ref ref="processinstanceStatsFileAppender" />
</logger>
```

Results look similar to this:

```
bw0a104,paging.Process,Paging.application,1.0,2014-07-28 14:14:43:354,2014-07-28 14:14:53:357,10003,1,COMPLETED
bw0a105,paging.Process,Paging.application,1.0,2014-07-28 14:14:48:353,2014-07-28 14:14:58:357,10004,0,COMPLETED
bw0a106,paging.Process,Paging.application,1.0,2014-07-28 14:14:53:354,2014-07-28 14:15:03:355,10001,0,COMPLETED
bw0a107,paging.Process,Paging.application,1.0,2014-07-28 14:14:58:353,2014-07-28 14:15:08:356,10003,1,COMPLETED
bw0a108,paging.Process,Paging.application,1.0,2014-07-28 14:15:03:354,2014-07-28 14:15:13:356,10002,0,COMPLETED
bw0a109,paging.Process,Paging.application,1.0,2014-07-28 14:15:08:353,2014-07-28 14:15:18:354,10001,0,COMPLETED
bw0a10a,paging.Process,Paging.application,1.0,2014-07-28 14:15:13:353,2014-07-28 14:15:23:355,10002,1,COMPLETED
bw0a10b,paging.Process,Paging.application,1.0,2014-07-28 14:15:18:353,2014-07-28 14:15:28:354,10001,1,COMPLETED
bw0a10c,paging.Process,Paging.application,1.0,2014-07-28 14:15:23:354,2014-07-28 14:15:33:355,10001,0,COMPLETED
bw0a10d,paging.Process,Paging.application,1.0,2014-07-28 14:15:28:353,2014-07-28 14:15:38:356,10003,0,COMPLETED
bw0a10e,paging.Process,Paging.application,1.0,2014-07-28 14:15:33:354,2014-07-28 14:15:43:359,10005,1,COMPLETED
bw0a10f,paging.Process,Paging.application,1.0,2014-07-28 14:15:38:353,2014-07-28 14:15:48:354,10001,0,COMPLETED
bw0a10g,paging.Process,Paging.application,1.0,2014-07-28 14:15:43:352,2014-07-28 14:15:53:356,10004,0,COMPLETED
bw0a10h,paging.Process,Paging.application,1.0,2014-07-28 14:15:48:354,2014-07-28 14:15:58:355,10001,0,COMPLETED
bw0a10i,paging.Process,Paging.application,1.0,2014-07-28 14:15:53:354,2014-07-28 14:16:03:357,10003,0,COMPLETED
bw0a10j,paging.Process,Paging.application,1.0,2014-07-28 14:15:58:354,2014-07-28 14:16:08:355,10001,0,COMPLETED
bw0a10k,paging.Process,Paging.application,1.0,2014-07-28 14:16:03:354,2014-07-28 14:16:13:356,10002,0,COMPLETED
bw0a10l,paging.Process,Paging.application,1.0,2014-07-28 14:16:08:353,2014-07-28 14:16:18:355,10002,2,COMPLETED
bw0a10m,paging.Process,Paging.application,1.0,2014-07-28 14:16:13:353,2014-07-28 14:16:23:355,10002,0,COMPLETED
bw0a10n,paging.Process,Paging.application,1.0,2014-07-28 14:16:18:354,2014-07-28 14:16:28:358,10004,1,COMPLETED
bw0a10o,paging.Process,Paging.application,1.0,2014-07-28 14:16:23:353,2014-07-28 14:16:33:356,10003,0,COMPLETED
bw0a10p,paging.Process,Paging.application,1.0,2014-07-28 14:16:28:354,2014-07-28 14:16:38:355,10001,0,COMPLETED
bw0a10q,paging.Process,Paging.application,1.0,2014-07-28 14:16:33:353,2014-07-28 14:16:43:354,10001,0,COMPLETED
bw0a10r,paging.Process,Paging.application,1.0,2014-07-28 14:16:38:353,2014-07-28 14:16:48:354,10001,1,COMPLETED
bw0a10s,paging.Process,Paging.application,1.0,2014-07-28 14:16:43:353,2014-07-28 14:16:53:356,10003,0,COMPLETED
bw0a10t,paging.Process,Paging.application,1.0,2014-07-28 14:16:48:353,2014-07-28 14:16:58:354,10001,0,COMPLETED
bw0a10u,paging.Process,Paging.application,1.0,2014-07-28 14:16:53:353,2014-07-28 14:17:03:354,10001,0,COMPLETED
bw0a10v,paging.Process,Paging.application,1.0,2014-07-28 14:16:58:353,2014-07-28 14:17:08:354,10001,1,COMPLETED
bw0a1010,paging.Process,Paging.application,1.0,2014-07-28 14:17:03:353,2014-07-28 14:17:13:354,10001,0,COMPLETED
bw0a1011,paging.Process,Paging.application,1.0,2014-07-28 14:17:08:353,2014-07-28 14:17:18:357,10004,1,COMPLETED
bw0a1012,paging.Process,Paging.application,1.0,2014-07-28 14:17:13:353,2014-07-28 14:17:23:355,10002,1,COMPLETED
bw0a1013,paging.Process,Paging.application,1.0,2014-07-28 14:17:18:353,2014-07-28 14:17:28:355,10002,0,COMPLETED
bw0a1014,paging.Process,Paging.application,1.0,2014-07-28 14:17:23:352,2014-07-28 14:17:33:354,10002,0,COMPLETED
```

a) To write the log as a formatted HTML file, add the following file appender to the `APPENDER: File Appender` section of the `logback.xml` file.

```
<appender name="processinstanceStatsFileAppender"
class="ch.qos.logback.core.FileAppender">
   <File>../log/processinstancestats.html</File>
     <encoder class="ch.qos.logback.core.encoder.LayoutWrappingEncoder">
       <layout
class="com.tibco.bw.logback.layout.ProcessInstanceStatsHTMLLayout"/>
     </encoder>
</appender>

<logger name="com.tibco.bw.statistics.activity" additivity="false">
<level value="INFO"/>
<appender-ref ref="processinstanceStatsFileAppender" />
</logger>
```

Results look similar to this:

Application Name	Application Version	Module Name	Module Version	Component Process Name	JobId	Parent Process Name	Parent ProcessInstanceId	Process Name	ProcessInstanceId	Start Time	End Time	Elapsed Time(Milliseconds)
Paging.application	1.0	Paging	1.0.0.qualifier	paging.Process	bw0a101	-	-	paging.Process	bw0a101	2014-07-31 17:03:23:353	2014-07-31 17:03:33:354	10001
Paging.application	1.0	Paging	1.0.0.qualifier	paging.Process	bw0a102	-	-	paging.Process	bw0a102	2014-07-31 17:03:28:353	2014-07-31 17:03:38:354	10001
Paging.application	1.0	Paging	1.0.0.qualifier	paging.Process	bw0a103	-	-	paging.Process	bw0a103	2014-07-31 17:03:33:353	2014-07-31 17:03:43:355	10002
Paging.application	1.0	Paging	1.0.0.qualifier	paging.Process	bw0a104	-	-	paging.Process	bw0a104	2014-07-31 17:03:38:353	2014-07-31 17:03:48:353	10000
Paging.application	1.0	Paging	1.0.0.qualifier	paging.Process	bw0a105	-	-	paging.Process	bw0a105	2014-07-31 17:03:43:352	2014-07-31 17:03:53:354	10002
Paging.application	1.0	Paging	1.0.0.qualifier	paging.Process	bw0a106	-	-	paging.Process	bw0a106	2014-07-31 17:03:48:353	2014-07-31 17:03:58:354	10001
Paging.application	1.0	Paging	1.0.0.qualifier	paging.Process	bw0a107	-	-	paging.Process	bw0a107	2014-07-31 17:03:53:353	2014-07-31 17:04:03:353	10000
Paging.application	1.0	Paging	1.0.0.qualifier	paging.Process	bw0a108	-	-	paging.Process	bw0a108	2014-07-31 17:03:58:353	2014-07-31 17:04:08:355	10002
Paging.application	1.0	Paging	1.0.0.qualifier	paging.Process	bw0a109	-	-	paging.Process	bw0a109	2014-07-31 17:04:03:353	2014-07-31 17:04:13:353	10000
Paging.application	1.0	Paging	1.0.0.qualifier	paging.Process	bw0a10a	-	-	paging.Process	bw0a10a	2014-07-31 17:04:08:353	2014-07-31 17:04:18:353	10000
Paging.application	1.0	Paging	1.0.0.qualifier	paging.Process	bw0a10b	-	-	paging.Process	bw0a10b	2014-07-31 17:04:13:352	2014-07-31 17:04:23:354	10002
Paging.application	1.0	Paging	1.0.0.qualifier	paging.Process	bw0a10c	-	-	paging.Process	bw0a10c	2014-07-31 17:04:18:352	2014-07-31 17:04:28:354	10002
Paging.application	1.0	Paging	1.0.0.qualifier	paging.Process	bw0a10d	-	-	paging.Process	bw0a10d	2014-07-31 17:04:23:353	2014-07-31 17:04:33:354	10001

9. Create a table in the database using following script or use an existing one.

```
create table PROCESS_INSTANCE_STAT_TABLE (APPLICATION_NAME
varchar(255),APPLICATION_VERSION varchar(255),MODULE_NAME
varchar(255),MODULE_VERSION varchar(255),COMPONENT_PROCESS_NAME
varchar(255),JOBID varchar(255),PARENT_PROCESS_NAME
varchar(255),PARENT_PROCESS_INSTANCEID varchar(255),PROCESS_NAME
varchar(255),PROCESS_INSTACEID varchar(255),START_TIME varchar(255),END_TIME
varchar(255),ELASPED_TIME varchar(255),EVAL_TIME varchar(255),STATUS
varchar(255));
```

Statistics is written either in the database or in the .csv file, but not both. If statistic is to be written in .csv, you need to put `statsprovider=false` in the `bwagent_xxx.json` file, and re-push the configuration to the `bwaget.ini` file.

Writing Process Statistic Data to an External Database

A customized database appender can also be used to collect statistics. Once statistics are in a database, reporting tools like TIBCO Jaspersoft or TIBCO Spotfire can be used to create graphs or reports. Add the following appenders to `APPENDER: File Appender` section of the `logback.xml` file to log data to an external database.
You can log process statistics data from applications running on existing or new AppNodes:

The supported databases are Oracle Database 12c, MySQL 5.7.x, Microsoft SQL Server 2012, 2014, PostgreSQL 9.2.x.

- To log process statistic data from applications running on existing AppNodes, edit the logback.xml file located at *BW_HOME*`/domains/`*<domain_name>*`/appnodes/`*<appspace_name>*`/`*<appnode_name>*. Restart the AppNode after you have finished editing the file.

- To log process statistic data from applications that will run on new AppNodes, edit the logback.xml file at *<BW_HOME>*/config/ before you create the new AppNodes.

Procedure

1. Update the property, **bw.agent.technology.logbackappender=true** in the bwagent.ini file.
2. Complete the following sub-steps to configure the external database properties in your bwagent configuration file, located at *<BW_HOME>*/config/. For example, follow these steps to edit these database settings in the bwagent_db.json file.

```
statsprovider: false,
providertechnology: "db",
dbprovidertype: postgresql,
dbproviderdriver: "org.postgresql.Driver",
dbproviderconnectionurl: "jdbc:postgresql://localhost:5432/bwadmindb",
dbprovideruser: bwuser,
dbproviderpassword: bwuser,
```

   a) Set **statsprovider** to `true`.
   b) Do not change the value for the **providertechnology** property.
   c) Use the same database details specified in the bwagent.ini file to set the **dbprovidertype**, **dbproviderdriver**, **dbproviderconnectionurl** , **dbprovideruser**, and **dbproviderpassword** properties.
   d) Run the bwadmin **config** command with the **-cf** option to push the changes from the JSON file to the `bwagent.ini` file.

```
BW_HOME\bin>bwadmin config -cf ../config/bwagent_db.json agent
```

3. Edit the `logback.xml` file, located at *BW_HOME*/domains/*<domain_name>*/appnodes/*<appspace_name>*/*<appnode_name>*, and add the following appender:

```
<appender name="activityStatsDBAppender"
class="com.tibco.bw.thor.management.stats.logback.db.BWActivityStatsDBAppender">
        <connectionSource
class="ch.qos.logback.core.db.DriverManagerConnectionSource">
             <driverClass>org.postgresql.Driver</driverClass>
             <url>jdbc:postgresql://localhost:5432/bwadmindb</url>
             <user>bwuser</user>
             <password>bwuser</password>
    </connectionSource>
 </appender>

 <appender name="processInstanceStatsDBAppender"
class="com.tibco.bw.thor.management.stats.logback.db.BWProcessInstanceStatsDBAppe
nder">
    <connectionSource
class="ch.qos.logback.core.db.DriverManagerConnectionSource">
              <driverClass>org.postgresql.Driver</driverClass>
              <url>jdbc:postgresql://localhost:5432/bwadmindb</url>
              <user>bwuser</user>
              <password>bwuser</password>
    </connectionSource>
 </appender>
```

Ensure the values you set for the **driverClass**, **url**, **user**, and **password** properties match the values set in the bwagent.ini file.

4. Locate the "LOGGER: BusinessWorks statistics collection loggers" section in the `logback.xml` file, and add the following appender to the appender-ref for activity logger and processinstance logger.

```
 <logger name="com.tibco.bw.statistics.activity" additivity="false">
    <level value="INFO"/>
    <appender-ref ref="activityStatsDBAppender"/>
  </logger>

 <logger name="com.tibco.bw.statistics.processinstance" additivity="false">
    <level value="INFO"/>
    <appender-ref ref="processInstanceStatsDBAppender"/>
  </logger>
```

5. Enable process statistics collection. For more information about enabling statistics collection from the command line or the Admin, see Enabling and Disabling Process Statistics.

Enabling and Disabling Auditing Events

The auditing is used to build metrics and statistics for an application.

Configure the following bwengine property in the `config.ini` file to enable and disable auditing at AppSpace and AppNode level.

- **`bw.engine.disable.auditevent`**: By default, the value of the property is `false`.

Restart the AppNode after configuring the property.

You can perform auditing without restarting an AppNode or an application with the help of following CLI commands.

Command	Description
`enableauditevents`	Enable Audit Events for an AppNode
`disableauditevents`	Disable Audit Events for an AppNode  Once you disable audit events, metrics and statistics related features of the TIBCO ActiveMatrix BusinessWorks™ Administrator will not work.

Enabling and Disabling Input and Output Data for Audit Events

Configure the following bwengine property in the `config.ini` file to enable and disable input and output data for audit events at an AppSpace and an AppNode level

- **`bw.engine.enable.activity.input.output.data.for.audit.events`**: By default, the value of the property is `false`.

Restart the AppNode after configuring these properties.

You can perform enable or diable input and output data for audit events without restarting an AppNode or an application by using the following CLI commands.

Command	Description
`enableinputoutputdataforauditevents`	Enable input output data for audit events for an AppNode
`disableinputoutputdataforauditevents`	Disable input output data for audit events for an AppNode

Applying Security Policies

TIBCO ActiveMatrix BusinessWorks™ includes a governance agent that enforces policies for ActiveMatrix BusinessWorks™ applications. Every installation of ActiveMatrix BusinessWorks includes the governance agent that facilitates the enforcement of cross-functional requirements such as security and compliance for your applications. The governance agent is disabled by default. In order to apply security policies, you must enable and configure the governance agent.

For information about enabling the governance agent with TEA, see Enabling the Governance Agent with TEA.

For information about enabling the governance agent using the AppSpace configuration file, see Enabling the Governance Agent Using the AppSpace Configuration File.

Backwards compatiblity is disabled by default for ActiveMatrix BusinessWorks 6.x applications using TIBCO ActiveMatrix Policy Director 2.0 to enforce security policies. To enable backwards compatibility, add `bw.governance.pd.compatibility.mode=true` to the existing AppSpace configuration file `appspace_config.ini` (located in the root of the AppSpace folder), or the AppSpace configuration template file, `appspace_config.ini_ template`, located in `<BW_HOME>\config\`.

Enabling the Governance Agent Using the Admin UI

The governance agent is disabled by default. In order to apply security policies, enable the `bw.governance.enabled` property.

Prerequisites

Complete the following tasks:

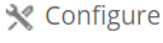
- The bwadmin mode must be set to enterprise
- The TIBCO Enterprise Administrator server must be running
- The bwagent TEA agent must be registered with the TEA server

Follow these steps to enable the governance agent using the Admin UI:

Procedure

1. Open a web browser and go to the TEA URL. Sign in, by typing **admin** for the user name and **admin** for the password.

   BusinessWorks is displayed in the Products list.
2. Click the BusinessWorks icon to go to ActiveMatrix BusinessWorks.
   The Domain Management page displays.
3. Click the **AppSpace** 

 icon to open the AppSpace page.
4. Click the AppSpace hosting your application.
5. On the AppSpaces page, click the **Configure** 

 icon to view a list of AppSpace properties you can modify.You can also click **Configure**  icon in the upper right of the AppSpace page.
6. Scroll down to find the `bw.governance.enabled` property. By default, the value is **false**.
7. Type **true** and click the **Check** icon to enable the governance agent. Ensure you enter the value under the **Current Value** column.

   Ensure the property `bw.governance.jms.server.url` does not have a value. The property is used to specify the JMS server URL used to communicate with the TIBCO Policy Director Administrator.
8. Stop and restart the AppSpace to apply the changes.

Result

The governance agent is enabled.

Enabling the Governance Agent Using an AppSpace Configuration File

The governance agent within each AppNode is disabled by default. You must enable it by setting properties within their respective `config.ini` files.

Enabling the Governance Agents in the AppNodes of an AppSpace

Each AppNode in TIBCO ActiveMatrix BusinessWorks includes a governance agent that enforce policies for TIBCO ActiveMatrix BusinessWorks applications. The governance agents are disabled by default. In order to apply security policies, you must enable these governance agents and configure the environment as described below.

To enable governance on an AppSpace, configure the governance agent property on the AppSpace by following these steps:

1. Copy the existing AppSpace configuration file `appspace_config.ini` that is located in the root of the AppSpace folder, or the AppSpace configuration template file, `appspace_config.ini_template`, that is located in *<BW_HOME>*`\config\` to a temporary location.

Do not modify the original AppSpace configuration file, `config.ini` located in the root of the AppSpace folder, or the AppSpace configuration template file, `appspace_config.ini_template` file. Instead, make changes to the copy of the file that is in the temporary location.

2. Edit the configuration file in the temporary location to set the following properties.

- Set the value for **`bw.governance.enabled`** to `true` to enable the governance agent. If no ActiveMatrix BusinessWorks applications are using TIBCO ActiveMatrix Policy Director to enforce security policies, comment out the property **`bw.governance.jms.server.url`**.
- If TIBCO ActiveMatrix Policy Director is already setup, ensure that the JMS server properties specified in the AppSpace configuration file match the JMS server configured in the TIBCO ActiveMatrix Policy Director server. For more information, see Applying Security Policies to TIBCO ActiveMatrix BusinessWorks 6.2 Applications.

```
#
--------------------------------------------------------------------------
# Section: BW Governance Agent & SPM Configuration. The properties in
# this section are applicable to Governance Agent and the Governance SPM
# EventSubscriber that is executed within a BW AppNode.
#
--------------------------------------------------------------------------
# Enable or disable the governance agent. This property is optional and
# it specifies whether the governance agent should be enabled or disabled
# in the AppNode. The supported values are: true or false. The default
# value is “false”.
bw.governance.enabled=true

# BW Governance Agent JMS URL. This property is optional and it is used
# to specify the JMS server URL used to communicate with the
# TIBCO Policy Director Administrator. If this property is not set, then
# the BW Governance agent will not attempt to connect to the JMS server.
# The URL is expected to start with 'tcp://' or 'ssl://' and the failover
# URLs can be specified as a ',' or '+' separated list.
bw.governance.jms.server.url=tcp://localhost:7222

# BW Governance Agent JMS User Name. This property is required if the
# Governance Agent JMS URL is specified.
bw.governance.jms.server.username=admin

# BW Governance Agent JMS User Password. This property is required if the
# Governance Agent JMS URL is specified.
bw.governance.jms.server.password=
```

```
# BW Governance Agent JMS SSL connection trust store type. This property
# is required if the JMS server protocol is ssl. The supported values are
# 'JKS'and 'JCEKS'. The default value is 'JKS'
bw.governance.jms.ssl.trust.store.type=JKS

# BW Governance Agent JMS SSL connection trust store location. This
# property is required if the JMS server protocol is ssl.
bw.governance.jms.ssl.trust.store.location=

# BW Governance Agent JMS SSL connection trust store password. This
# property is required if the JMS server protocol is ssl. The password
# may be clear text or supplied as an obfuscated string.
bw.governance.jms.ssl.trust.store.password=

# BW Governance Agent JMS Connection attempt count. This property is
# required if the Governance Agent JMS URL is specified and it specifies
# the number of JMS connection attempts the Governance Agent will make.
# The default value is '120'.
bw.governance.jms.reconnect.attempt.count=120

# BW Governance Agent JMS Connection attempt timeout. This property is
# required if the Governance Agent JMS URL is specified and it specifies
# the timeout between the attempt to reestablish connection to the JMS
# server. The default value is '500'.
bw.governance.jms.reconnect.attempt.timeout=500

# BW Governance Agent JMS Connection attempt delay. This property is
# required if the Governance Agent JMS URL is specified and it specifies
# the delay in milliseconds between attempts to establish reestablish
# connection to the JMS server. The default value is '500'.
bw.governance.jms.reconnect.attempt.delay=500

# BW Governance Agent JMS receiver queue name. This property is required
# if the Governance Agent JMS URL is specified and it specifies receiver
# queue name for the governance agent and administrator communication.
# The default value is 'queue.bw.governance.agent.bw.default'.
bw.governance.jms.queue.receiver.name=queue.governance.agent.bw.default

# BW Governance Agent JMS sender queue name. This property is required
# if the Governance Agent JMS URL is specified and it specifies the
# sender queue name for the governance agent and administrator
# communication. It must match the value specified in the Policy Director
# Administrator configuration.
# The default value is 'governance.de.bw.default'.
bw.governance.jms.queue.sender.name=governance.de.bw.default

# BW Governance Agent JMS JNDI custom property. This property is optional
# and it provides the ability to specify custom property for the
# JMS JNDI Initial Context. For example to provide a custom property
# called "myProperty" for the JNDI Initial Context, then specify
# a property "bw.governance.jms.application.property.myProperty=".
#bw.governance.jms.application.property.<UserCustomProperty>=<userValue>
# BW Governance Agent Shared Resource lookup. This property is optional
# and it provides ability for the Governance Agent to lookup shared
# resources.
# bw.governance.sr.WSSConfiguration=com.tibco.trinity.runtime.core.
# provider.authn.wss
```

3. Restart the AppSpace from the TIBCO ActiveMatrix BusinessWorks agent user interface in TEA.

OpenTracing

OpenTracing is an open, vendor neutral standard for distributed systems that can be used to keep track of the current state of the job.

TIBCO ActiveMatrix BusinessWorks™ supports Jaeger libraries to display 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 etc. For every process instance, root span is created and all the activity instances are child spans of it.

OpenTracing does not support checkpointing. Activities recovered after restarting are not seen on Jaeger UI.

Trace represents multiple related process instance spans.

In case of HTTP palette, JMS palette, REST binding, and SOAP binding, client and server process instances are shown in one trace, whereas for all other palettes, every process instance is a trace.

For more information about Jaeger, see Jaeger documentation.

Configure the following BWEngine property in the `BW_JAVA_OPTS` environment variable while running the application to enable and disable OpenTracing at an AppNode level:

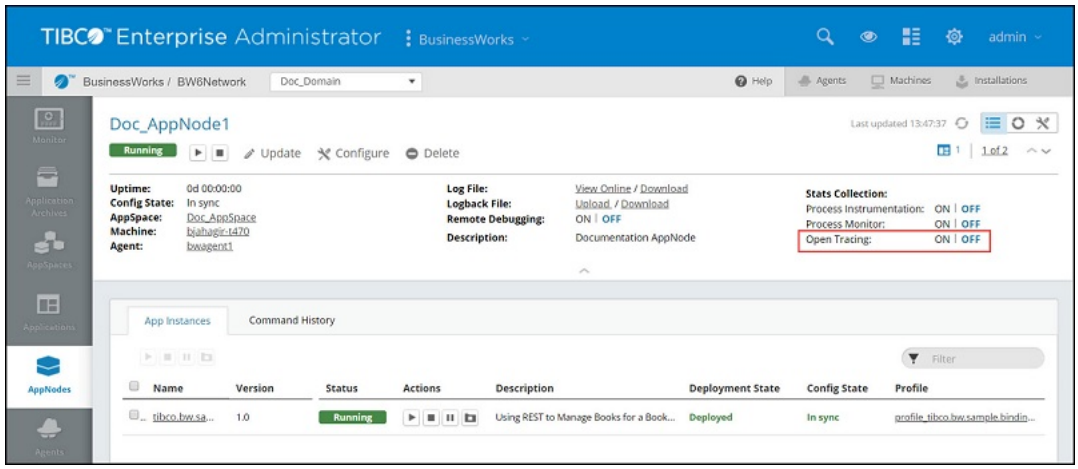
- **`bw.engine.opentracing.enable=true`**.

  By default, the property is false.

Additionally you can configure a few more properties specific to Jaeger.

Admin UI

Enable or disable OpenTracing at an AppNode Page 2 level.



Open Tracing via UDP connection

For obtaining Open Tracing via UDP connection, configure the following Jaeger properties:

Properties	Description
JAEGER_SAMPLER_MANAGER_HOST_PORT	The host name and port when using the remote controlled sampler.

Properties	Description
JAEGER_AGENT_HOST	The hostname for communicating with agent via UDP.
JAEGER_AGENT_PORT	The port for communicating with agent via UDP

Open Tracing via HTTP connection

For obtaining Open Tracing via HTTP connection, configure the following Jaeger properties:

Properties	Description
JAEGER_SAMPLER_MANAGER_HOST_PORT	The host name and port when using the remote controlled sampler.
JAEGER_ENDPOINT	The traces endpoint, in case the client should connect directly to the Collector. For example, `http://<jaeger-collector>:<port>/api/traces`

For more information about the properties, see Configuration via Environment section at github.com.

Supported tags for querying on Jaeger UI

Currently, the following tags are supported for querying on Jaeger UI.

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.
IP	IP address
ActivityName	Name of an activity in a process
ActivityID	Id of an activity
ProcessInstanceId	Process instance ID
JobId	Job ID of the process.

Tag	Description
ProcessName	Name of the process displayed for starter activities.

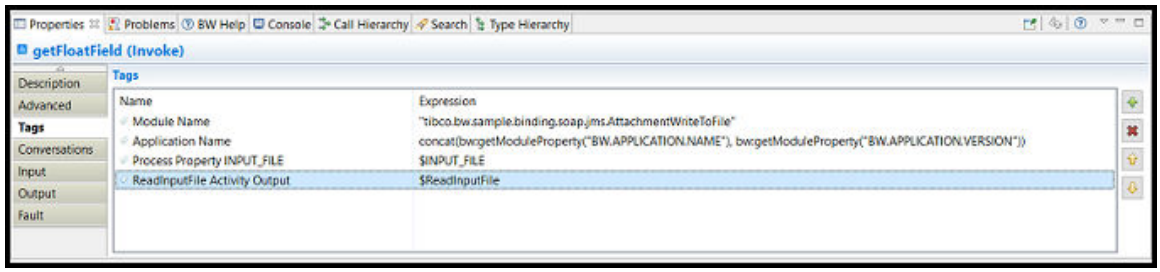
Dynamically Enabling and Disabling Open Tracing

You can enable OpenTracing without restarting an AppNode or an application with the help of following CLI commands.

Command	Description
`enableopentracing`	Enable open tracing for an AppNode
`disableopentracing`	Disable open tracing for an AppNode

Custom Tags For OpenTracing

For OpenTracing, 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, asterick (*) prefix is added for names of the custom tags. It avoids overriding of pre-defined engine tags.

OpenTracing 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 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
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

Activity Name	Supported Tags
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 • peer.port
FTP Get	• peer.hostname • peer.port
FTP Get Default Directory	• peer.hostname • peer.port
FTP Make Remote Directory	• peer.hostname • peer.port
FTP Put	• peer.hostname • peer.port
FTP Quote	• peer.hostname • peer.port
FTP Remove Remote Directory	• peer.hostname • peer.port

Activity Name	Supported Tags
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 • http.url • span.kind • error • ErrorMessage
Send HTTP Request	• span.kind • http.url • HTTPRequestQuery • HTTPPostDataType • HTTPCookiePolicy • http.method • IsSecureHTTP • error • ErrorMessage • ErrorCode • ErrorStatus

Activity Name	Supported Tags
Send HTTP Response	• span.kind • http.status_code • peer.hostname • peer.port • http.method • peer.ipv4 • HttpServerProtocol • ContentType • IsSecureHTTP • error • 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

Activity Name	Supported Tags
Java To XML	• Class Name • Constructor Declared • Cache Declared
XML To Java	Class Name

JDBC Palette

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

JMS Palette

Activity Name	Suported Tags
Get JMS Queue Message	• message_bus.destination • MessagingStyle • MessageType • AcknoledgementMode
JMS Receive Message	• message_bus.destination • MessagingStyle • MessageType • span.kind • ReplyQueue
JMS Request Reply	• message_bus.destination • MessagingStyle • MessageType • span.kind
JMS Send Message	• message_bus.destination • MessagingStyle • MessageType • span.kind
Reply to JMS Message	• MessagingStyle • MessageType • span.kind • ReplyQueue
Wait for JMS Request	• message_bus.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 • StringValue or FileName - Depending upon 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 • peer.hostname • peer.port • http.method • error • ErrorType • ErrorMessage
Parse JSON	• SchemaType • OutputRootElementName • IsBadgerfishEnabled • error • ErrorType • ErrorMessage
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 • peer.hostname • peer.port
TCP Open Connection	• peer.hostname • peer.port
TCP Receiver	• peer.hostname • peer.port
Wait For TCP Request	• peer.hostname • peer.port
Write TCP Data	• Data Type • peer.hostname • peer.port

XML Palette

Activity Name	Supported Tags
Parse XML	• IsOutputValidationEnabled • Input Style • error • ErrorType • ErrorMessage

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

OpenTracing Tags From SOAP Bindings

The follwoing 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 • message_bus.destination • MessagingStyle • MessageType • Operation
Client	• TransportType • EndpointReference • ReplyTo • MessagingStyle • Service Name • Operation Name • message_bus.destination • span.kind • MessageType

OpenTracing Tags From REST Binding

The follwoing 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 • peer.port • http.status_code • span.kind • peer.hostname • clientResponseFormat • http.method

Side	Supported tags
Client	• http.url • isUsingSSL • error • errorMessage • errorStatus • peer.port • http.status_code • span.kind • peer.hostname • isRequestBuffered • contentType • http.method

List of Ports

This is a list of ports that are used.

List of Ports

Port Description	Default Value	Configuration
External database port. Applies to bwagent technology type of Database/EMS.	5432	**`bw.agent.technology.dbems.db.connectionURL`** property in *`BW_HOME`*`\config\bwagent.ini`
The internal HTTP communication port the Thor engine uses to communicate with bwagent to send the status of AppNodes and applications. Update this property to specify a port to start the internal server on.	56565	**`bw.appnode.agent.http.communication.port`** property in *`BW_HOME`*`\config\bwagent.ini`
AppNode HTTP management port. Must be unique across all defined AppNodes on the same machine.	User-specified	**`-httpPort`** option in bwadmin **`create`** command for AppNode creation or **HTTP Port** option in `Create AppNode` dialog box in Admin UI.
AppNode remote debug port.	User-specified	**`-port`** option in bwadmin **`enabledebugport`** command for remote debugging of an AppNode.
OSGi console port. Must be unique.	User-specified	**`port`** argument in bwadmin **`enableconsole`** command for enabling OSGi port for AppNode. Can also be specified with **`-osgiPort`** option when AppNode created with **`create`** command.
• EMS server port for group configuration. • EMS server port for Database/EMS bwagent technology type.	7222	Engine: **`bw.engine.groupProvider.qin.EMSServerUrl`** property in *`BW_HOME`*`\config\appspace_config-ini_template` Technology type: **`bw.agent.technology.dbems.db.ems.serverURL`** property in *`BW_HOME`*`\config\bwagent.ini`
Web server HTTP port for Swagger UI.	5555	Not configurable
Web server HTTP port.	8079	**`bw.agent.http.port`** property in *`BW_HOME`*`\config\bwagent.ini`
Web server HTTPs port.	8886	**`bw.agent.https.port`** property in *`BW_HOME`*`\config\bwagent.ini`

Port Description	Default Value	Configuration
bwagent TEA agent port.	9091	**bw.agent.tea.agent.port** property in *BW_HOME*\config\bwagent.ini
bwagent TEA agent shutdown port.	5678	**bw.agent.tea.agent.shutdown.port** property in *BW_HOME*\config\bwagent.ini
TEA listen port.	8777	**tea.http.port** property in *TEA_CONFIG_HOME*\tibco\cfgmgmt\conf\tea.conf
TEA SSH port.	2222	**tea.shell.port** property in *TEA_CONFIG_HOME*\tibco\cfgmgmt\conf\tea.conf