|  |
| --- |
| Tibco Software Inc |
| BusinessEvents TEA Agent |
| Functional Specifications Document |

|  |
| --- |
| Bhalchandra Gokhale  11/2/2015 |

Revision History

|  |  |  |
| --- | --- | --- |
| Author | Date | Summary of Change |
| Bhalchandra Gokhale | 2nd November 2, 2015 | Initial Document |
|  | 18th January 2016 | Added section on User/Roles/Permissions/ Log levels |
|  | 2nd Feb 2016 | Project specific TRA  Hot-deploy classes |
|  | 10th Feb 2016 | Added Chapter 5: Authentication and SSL Configuration |
|  | 25th Feb 2016 | Added Chapter 6 and 7: BE Properties and Global Variables |
|  | 1st March 2016 | Added Chapters 8 and 9: Charts Rules and Alerts |
|  | 29th July 2016 | Added Chapters 10 , 11,12: Export/Import,Deployment Profile and Upload External Jars, Download logs ,Updated Add Machine,Create Instance |
|  | 25th August 2016 | Added Chapters 13 , 14: i18n Support, Alert Notifications |

Table of Contents

[Table of Contents 3](#_Toc457555435)

[1 Introduction 7](#_Toc457555436)

[1.1 Purpose and Motivation 7](#_Toc457555437)

[1.2 Audience 7](#_Toc457555438)

[1.3 Document Scope 7](#_Toc457555439)

[1.4 Assumptions 7](#_Toc457555440)

[1.5 Overview 7](#_Toc457555441)

[2 User Interactions Functional Specifications 8](#_Toc457555442)

[2.1 BE application management use cases 8](#_Toc457555443)

[2.1.1 Add Machine 8](#_Toc457555444)

[2.1.2 Machine Repository 9](#_Toc457555445)

[2.1.3 Create New Application Deployment 9](#_Toc457555446)

[2.1.4 Delete Deployment 9](#_Toc457555447)

[2.1.5 Create Instance 9](#_Toc457555448)

[2.1.6 Copy Instance 10](#_Toc457555449)

[2.1.7 Update Instance 10](#_Toc457555450)

[2.1.8 Delete Instance 10](#_Toc457555451)

[2.1.9 Deploy Instance 11](#_Toc457555452)

[2.1.10 Un-deploy Instance 11](#_Toc457555453)

[2.1.11 Start Instance 11](#_Toc457555454)

[2.1.12 Stop Instance 11](#_Toc457555455)

[2.1.13 Kill Instance 11](#_Toc457555456)

[2.1.14 Hot-Deploy Instance 11](#_Toc457555457)

[2.1.15 Hot-Deploy Application 11](#_Toc457555458)

[2.1.16 Group Operation Support 11](#_Toc457555459)

[2.1.17 Import Site Topology “ST” file 11](#_Toc457555460)

[2.2 Configuration Management Use cases 12](#_Toc457555461)

[2.2.1 Edit Application Deployment 12](#_Toc457555462)

[2.2.2 Edit Machine Details 12](#_Toc457555463)

[2.2.3 JVM Properties 12](#_Toc457555464)

[2.2.4 System Properties 12](#_Toc457555465)

[2.2.5 Group Operation Support 13](#_Toc457555466)

[2.2.6 Instance Operations 13](#_Toc457555467)

[2.3 BE application monitoring use cases 13](#_Toc457555468)

[2.3.1 Instance Status 13](#_Toc457555469)

[2.3.2 CPU/Memory and other charts 13](#_Toc457555470)

[2.3.3 Download Logs 13](#_Toc457555471)

[2.3.4 Thread-dumps 14](#_Toc457555472)

[2.3.5 Log Levels 14](#_Toc457555473)

[2.3.6 Group Operation Support 14](#_Toc457555474)

[2.4 UI Views 14](#_Toc457555475)

[2.4.1 All Deployments View 14](#_Toc457555476)

[2.4.2 All Instances View 14](#_Toc457555477)

[2.4.3 Machines View 14](#_Toc457555478)

[2.4.4 Processing Units View 14](#_Toc457555479)

[2.4.5 Agent View 14](#_Toc457555480)

[2.4.6 Navigability and Bread Crumbs 14](#_Toc457555481)

[2.5 User Management 15](#_Toc457555482)

[2.5.1 Roles and Permissions 15](#_Toc457555483)

[2.6 Command-line Interface 16](#_Toc457555484)

[2.6.1 ConfigurationMgmt.py 16](#_Toc457555485)

[2.6.2 ApplicationsMgmt.py 16](#_Toc457555486)

[2.7 TEA Agent Self-monitoring View 18](#_Toc457555487)

[3 TEA Agent functional Specification 18](#_Toc457555488)

[3.1 TEA Agent as an Installer Option 18](#_Toc457555489)

[3.2 “Wrapper” based Command Line 19](#_Toc457555490)

[3.3 TEA Server Installation 19](#_Toc457555491)

[3.4 TEA Agent Configuration 19](#_Toc457555492)

[3.5 TEA Server and TEA Agent 19](#_Toc457555493)

[3.6 TEA Agent State persistence 19](#_Toc457555494)

[3.7 TEA Agent Transport 19](#_Toc457555495)

[3.8 TEA Agent Fault Tolerance 19](#_Toc457555496)

[4 Additions 19](#_Toc457555497)

[4.1 Project specific Master TRA file 19](#_Toc457555498)

[4.2 Upload and Load - Hot-deploy Classes and Rule Template Instances 20](#_Toc457555499)

[5 Authentication and SSL Configuration 20](#_Toc457555500)

[5.1 Securing BE-TEA Agent / BE PU Instance JMX channel 20](#_Toc457555501)

[5.1.1 JMX Authentication 20](#_Toc457555502)

[5.1.2 One way SSL between BE PU Instances (JMX Server) and BE TEA Agent (JMX client) 21](#_Toc457555503)

[5.1.3 Two way SSL between BE PU Instances (JMX Server) and BE TEA Agent (JMX client) 21](#_Toc457555504)

[5.2 SSL Configuration for BE TEA Agent / TEA Server connection 21](#_Toc457555505)

[5.3 SSL Configuration for Browser / TEA Server connection 21](#_Toc457555506)

[6 BE Properties 22](#_Toc457555507)

[6.1 PU Instance configuration 22](#_Toc457555508)

[6.2 Group Operation 22](#_Toc457555509)

[7 Global Variables 22](#_Toc457555510)

[7.1 PU Instance configuration 22](#_Toc457555511)

[7.2 Group Operation 23](#_Toc457555512)

[8 Charts 23](#_Toc457555513)

[8.1 Five Minute Statistics 23](#_Toc457555514)

[8.1.1 Average Used Memory 23](#_Toc457555515)

[8.1.2 Average CPU 23](#_Toc457555516)

[8.1.3 Average RTC Transaction Latency 23](#_Toc457555517)

[8.1.4 Average RTC Transaction Throughput 23](#_Toc457555518)

[8.1.5 Total Locks Held 23](#_Toc457555519)

[8.1.6 Even Throughput 24](#_Toc457555520)

[8.1.6.1 Event Throughput Configuration 24](#_Toc457555521)

[8.2 Hourly Statistics 24](#_Toc457555522)

[8.3 Rule Performance 24](#_Toc457555523)

[8.4 Garbage Collector Details 25](#_Toc457555524)

[8.5 Memory Pools Details 25](#_Toc457555525)

[9 Rules and Alerts 25](#_Toc457555526)

[9.1 Monitored Entities 25](#_Toc457555527)

[9.2 Set Actions and Clear Actions 27](#_Toc457555528)

[9.2.1 Set the Health of the Monitored Entity 27](#_Toc457555529)

[9.2.2 Send an Email 27](#_Toc457555530)

[9.2.3 Write to a log file 27](#_Toc457555531)

[9.2.4 Alert Only 27](#_Toc457555532)

[9.3 Placeholders in Alert Text 27](#_Toc457555533)

[9.4 Specifying Set and Clear Conditions 30](#_Toc457555534)

[9.5 Health Rules 30](#_Toc457555535)

[9.6 Alerts 30](#_Toc457555536)

[10 Export /Import of Application 30](#_Toc457555537)

[10.1 Export from TEA 30](#_Toc457555538)

[10.2 Import to TEA 30](#_Toc457555539)

[10.3 Export from TIBCO Administrator 30](#_Toc457555540)

[10.4 Import TIBCO Administrator Application 31](#_Toc457555541)

[10.4.1 GUI Approach -: 31](#_Toc457555542)

[10.4.2 CLI approach -: 32](#_Toc457555543)

[10.4.3 Sample CSV File -: 32](#_Toc457555544)

[11 Deployment Profiles 33](#_Toc457555545)

[11.1 Add Profile: 33](#_Toc457555546)

[11.2 Edit Profile: 33](#_Toc457555547)

[11.3 Delete Profile: 33](#_Toc457555548)

[12 Upload External Jars 33](#_Toc457555549)

13 i18n Support ………………………………………………………………………………………………………………………………...34

14 Alert Notifications ……………………………………………………………………………………………………………………….…34

# [Introduction](http://confluence.tibco.com/pages/viewpage.action?title=StreamBase+Channel+in+BE&spaceKey=be#StreamBaseChannelinBE-Introduction)

## [Purpose](http://confluence.tibco.com/pages/viewpage.action?title=StreamBase+Channel+in+BE&spaceKey=be#StreamBaseChannelinBE-Purpose) and Motivation

TEA provides a common monitoring and management console/platform for different Tibco products. Commonly used Tibco products such as EMS and BW6 have already adopted the TEA framework and are managed/monitored via the TEA console. There is a customer need for BE to adopt TEA as well.

Another motivation for adopting TEA is to make BE application management a better user experience as compared to BEMM. With BEMM, there are different places to go to for configuration and management. While deployment creation was a design time studio artifact, the rules management was via the dashboard agent configuration in the CDD. With TEA, the TEA UI becomes a single dashboard to manage and monitor a BE application deployment.

## [Audience](http://confluence.tibco.com/pages/viewpage.action?title=StreamBase+Channel+in+BE&spaceKey=be#StreamBaseChannelinBE-Audience)

This document is targeted towards QA, tech-pubs and engineering teams for creating functional test cases, documentation of product features and for implementing these functional specs.

## [Document Scope](http://confluence.tibco.com/pages/viewpage.action?title=StreamBase+Channel+in+BE&spaceKey=be#StreamBaseChannelinBE-DocumentScope)

The scope of this document is to list the functional specifications of the overall requirement stated as “BE/TEA integration”. It will not describe the technical details or the high/low level design of the TEA Agent or the server. In some cases, it might mention the technologies used to be read in context.

## [Assumptions](http://confluence.tibco.com/pages/viewpage.action?title=StreamBase+Channel+in+BE&spaceKey=be#StreamBaseChannelinBE-Assumptions)

It is assumed that readers are familiar with Tibco BusinessEvents product terminology and the BusinessEvents Cluster Deployment Descriptor and related terminology. It is also assumed that readers are familiar with other common Tibco terminologies such as “wrapper”, “TIBCO\_HOME” etc.

## [Overview](http://confluence.tibco.com/pages/viewpage.action?title=StreamBase+Channel+in+BE&spaceKey=be#StreamBaseChannelinBE-Overview)

The TEA deployment consists of two main components, namely the TEA-Server, which renders the UI and works with user sessions and product specific TEA-Agents that are responsible for providing product specific functionality such as managing deployments and monitoring its product. The TEA server also provides a mechanism to bind users to TEA Agent specific roles and permissions. The TEA Server can host/connect with multiple product agents thereby providing common dashboard across products to end users. The diagram below shows the overall TEA Agent Architecture.

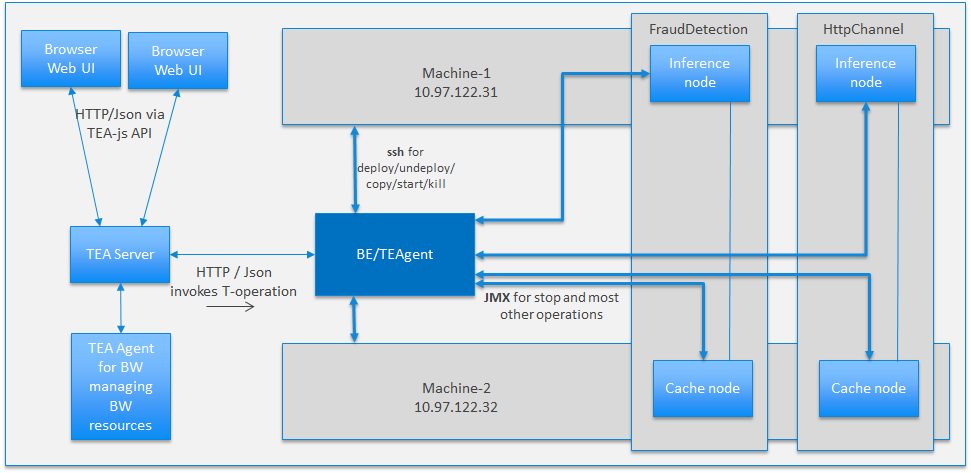


Fig.1

# User Interactions [Functional](http://confluence.tibco.com/pages/viewpage.action?title=StreamBase+Channel+in+BE&spaceKey=be#StreamBaseChannelinBE-FunctionalDesignDescription) Specifications

The user interactions functional use cases may be broadly classified as follows

**BE application management use cases**

These use cases deal with creating and managing a deployment blueprint for the BE application and providing useful views of the deployments. These include the ability to detect/define machines involved in a BE application deployment and the ability to associate/assign instances of BE processing units to these machines. They also include use cases to change configurations of these processing unit instances, and invoke MBeans defined by the application.

**BE application monitoring use cases**

These use cases deal with visualization of the performance indicators of individual processes or instances, the ability to visualize the performance of the application cluster as a whole and the ability to write simple rules to take action when certain performance indicators cross their configured thresholds.

## BE application management use cases

The following section covers application management use cases

### Add Machine

Add a Machine for use in a BE deployment. The following user inputs shall be captured

* *Machine name*: Name of the machine. Preferably the hostname
* *IP address*: ip address of the machine that will be used for communicating/connecting to this machine
* *The OS*: It is auto detected.If os is not detected and user tried to perform operations like deploy,undeploy etc then default OS is Windeows
* *Username*: This is the O/S user name to be used for all deployment related and start commands. **ssh** will be used for all such commands.
* *Password*: password for the above user. This should be stored in an encrypted form
* *SSH port*: The SSH port used by the SSH server on this machine.
* *Default Deployment Path*: The default path where deployment related artifacts will be installed
* Adding multiple BE home is now supported. Here user can add/discover BE Home details
* User manual can add BE home and TRA while the host
* User discovers the BE homes from host using IP address, ssh username,password,port.
* BE home and TRA path entry must be unique means if the BE Home and TRA pair exists then user cannot add it manually or skipped while discovering it from host .Each BE home details have one unique id.
* On Save of host TRA from each BE home is copied to repo.Name of TRA file is **<host\_id>\_<be\_home\_id>.tra**
* The remove BE home button is shown to those BE homes which are not yet used in any instance
* *BE\_HOME and BE\_TRA*: Update allowed only when there is not a single instance deployed to this host.
* Discover BE homes retrives BE homes which are greater than 5.3

### Machine Repository

The agent shall maintain a common machine repository that would be usable across multiple application deployments. Machines added as part of importing “.ST” files ~~or those detected by the T-server~~ should be available for deployments by different application deployments.

### Create New Application Deployment

A new application deployment is created to newly manage and monitor a BE application. The following user inputs shall be captured

* *Deployment name*: A name for the new deployment
* *Application CDD*: Browse the file system for the CDD to use
* *Application EAR*: Browse the file system for the EAR to use

### Delete Deployment

An ability to delete a previously defined deployment shall be provided. All instances associated with this machine need to be un-deployed before an application deployment can be deleted. Deleting an application deployment should not delete its associated machine entries.

### Create Instance

Once a new deployment is created, application instances need to be defined in the deployment. This involves binding a processing unit defined in the CDD to a machine on which it will run. The following user inputs shall be captured

* *Instance name*: (maps to the –name parameter of be-engine)
* *PU name*: (maps to the –u parameter of be-engine)
* *Machine name*: (one of the machines previously added to this deployment or to the agent). If machine is not authenticated then error message is shown.
* *BE Home*: List of BE home from selected host is shown.User need to select one BE home which he/she wants to use to start the engine.
* *JMX port*: The JMX port that will be used to communicate with this instance. It should not clash with any other port on this machine. If JMX port is not mentioned then suggestion for jmx port is given as follows

From selected host it suggests unused port (*which was previously used by any instance*) . If all ports from host are used then it returns max JMX port+1.

* *Deployment path*: The path on the machine where deployment artifacts will be copied. Deployment artifacts include the EAR and the CDD file, instance specific .TRA file and shell/batch files to start the instance. If an empty deployment path is provided, the default deployment path from the associated machine will be used.

### Copy Instance

A convenience mechanism to create similar instances from a given instance shall be provided. Once an instance is copied, it should be possible to edit it, to make specific changes to it. The following user inputs shall be captured

* The instance to copy
* The name of the new instance

During copy, it should be possible to change the following properties of the copied instance

* *Instance name*: (maps to the –name parameter of be-engine)
* *PU name*: (maps to the –u parameter of be-engine)
* *Machine name*: (one of the machines previously added to this deployment or to the agent).If machine is not authenticated then error message is shown.
* *BE Home*: List of BE home from selected host is shown.User need to select one BE home which he/she wants to use to start the engine.
* *JMX port*: The JMX port that will be used to communicate with this instance.
* *Deployment path*: The path on the machine where deployment artifacts will be copied. Deployment artifacts include the EAR and the CDD file, instance specific .TRA file and shell/batch files to start the instance. NOTE that this operation itself does not copy the deployment artifacts, just the application definition and properties. It would need explicit deployment.

The copy operation should copy all System Properties, BE Properties, JVM properties, Global Variables, Log settings from the source instance. It should also be possible to edit various properties once the copy is complete.

### Update Instance

It should be possible to update the following properties of an instance. Operation is to be allowed only for un-deployed instances.

* *Machine name*
* *BE Home*
* *Processing Unit*
* *JMX port*
* *Deployment path*

### Delete Instance

This is used to delete deployment instances. Operation is to be allowed only for un-deployed instances.

### Deploy Instance

Once an application instance is created, it needs to be deployed to its associated machine. Deployment includes copying the CDD, EAR to the deployment path of its associated machine and copying instance specific TRA file and shell scripts (or Windows batch file) to the deployment location

### Un-deploy Instance

An instance that is previously deployed can be un-deployed. To be allowed only for stopped instances. Once un-deployed, all associated local and deployed changes will be un-done.

### Start Instance

An ability to start the instance shall be provided.

(An ssh command will be used to start the instance-specific shell script at the deployment location.)

### Stop Instance

An ability to gracefully stop the instance shall be provided.

(A JMX MBean will be used to stop the instance.)

### Kill Instance

An ability to force kill (kill -9) the instance shall be provided.

### Hot-Deploy Instance

An ability to hot-deploy an EAR to a specific instance will be provided if the CDD setting in the associated PU is enabled for hot-deploy

### Hot-Deploy Application

An ability to hot-deploy a new EAR for the entire application will be provided. This option will only be enabled if there is at least one PU in the CDD with “Hot Deploy” enabled.

### Group Operation Support

The UI shall provide a way to select all or specific instances and to perform group operations of the following use cases: Deploy, Un-deploy, Start, Stop, Kill for the selected instances. If any of the instances is already in the target state, the system should silently ignore the operation for that instance. i.e., the application will not throw an exception.

For group operation of Hot-Deploy, hot-deployment will only take effect for those instances whose associated PUs are enabled for hot-deploy (via the CDD) For those instances where hot-deploy is not enabled via CDD, the new EAR will take effect upon subsequent restart.

For group operations of thread-dump, a zip file of thread dumps will be downloaded to the browser machine.

### Import Site Topology “ST” file

A convenience mechanism to import Site Topology files or “.ST” files shall be provided. Machines and PU instances defined in the ST file shall be imported as corresponding agent artifacts. The notion of a “Deployment Unit” of BEMM is dropped in BE/TEA. Processing units are directly bound to machines as Application Instances. During import, if there is already a machine defined for any of the machines in the ST file, the import process will automatically create a new machine definition for each such machine and update the dependent application instances to bind to the new machine. This is to avoid conflicting definitions of machines.

If all of the details of the machine in the ST file match with an existing repo machine entry then the existing machine definition is reused and a new entry is not created.

If the details don’t match a new machine entry is created in the repo and the dependent application instances will be bound to that new machine entry. If there is a name clash, it will get a uniquely generated name.

## Configuration Management Use cases

These use cases cover the general category of requirements of modifying instance configurations

### Edit Application Deployment

This provides a way to update existing deployments with a new CDD or a new EAR file. The state of the application instances will be changed to “Needs Deployment/Re-deployment”.

### Edit Machine Details

This provides an ability to change the machine definition. The following changes should be allowed

* *Machine Name*: Update allowed at all times. When this is changed, all related application instances should reflect the change in the host name. Should not allow a rename to another host with the same name.
* *IP Address*: Update allowed at all times, this is because there are cases where the physical machine has not changed but just its IP address.
* *Host OS*: Only allow update when there is not a single instance deployed to this host
* *BE\_HOME and BE\_TRA*: Update allowed only when there is not a single instance deployed to this host.
* *SSH username and password*: Update allowed at all time. All subsequent operations will use the updates values.
* *Default Deployment Path*: Update allowed only when there is not a single instance deployed to this host.

### JVM Properties

The ability to edit/update the following JVM properties should be provided

* Initial heap size (-Xms)
* Maximum heap size (-Xmx)

The instance needs to be deployed at least once for the user to be able to change these settings. This is to allow loading default values from the target machine’s BE-TRA file.

### System Properties

The ability to add/update and delete system properties shall be provided. The instance needs re-deployment for the changes to take effect.

### Group Operation Support

The ability to update System Properties, JVM properties and Global Variables as a group operation (i.e., for multiple, selected instances) shall be provided. Additionally the ability to Add/Delete System Properties only for similar valued properties across all selected instances shall be provided. This is to avoid conflicting changes when different instances have different values.

### Instance Operations

BE exposes several MBeans for management and monitoring. The UI shall provide a mechanism to invoke these operations for a given instance. Due to the large number of MBeans that are exposed and in order to keep the UI simple, there shall be a way to configure which of these are exposed via the TEA UI. By default the ones exposed by BEMM shall be exposed. Users can modify this list by changing the configuration. A change in configuration will need an agent restart.

## BE application monitoring use cases

The following section covers application monitoring use cases.

### Instance Status

The BE agent shall provide a visual indication its running state and its deployment state. The following parameters will be displayed

* Name of the instance
* Machine where running
* PID of the process
* JMX port being used
* Running State: Starting, Running, Stopping, Stopped
* Deployment State: Deployed, Needs Deployment, Un-deployed.
* CPU %
* Memory %

### CPU/Memory and other charts

The UI will provide real time charts for the following parameters:

* CPU usage
* memory usage
* Garbage Collectors statistics
* Thread usage statistics
* Memory Pools statistics

### Download Logs

An ability to view and optionally download an instance BE Logs/AS logs would be provided. From Group level BE/AS logs zip file would be downloaded for selected instance. From instance page on click of BE/AS logs last 25 lined are shown scroll line count is incremented by 25. When logs are displayed user can download the log file also.

### Thread-dumps

An ability to view and optionally download an instance thread dump would be provided.

### Log Levels

The UI shall provide a way to change log levels of a given instance. The user will be allowed to specify multiple logger patterns and set a level for each of the patterns. The runtime will evaluate these patterns and log levels will be applied accordingly. Log level changes can be applied directly to running instances or can be deployed so that the changes are permanent.

### Group Operation Support

The ability to change log levels and take thread dumps as a group operation will be provided.

For log levels, only those patterns which are common to all selected instances as well as having the same values in all selected instances will be changeable via group operations.

For thread dumps a thread zip file of thread dumps of selected instances will be downloaded to the browser machine.

## UI Views

The TEA console will show different views of the deployment. The following views are defined

### All Deployments View

This view shall display a summary view of all the deployments defined in the agent. For each deployment, it will display a summary of the instances defined in the deployment along with a summary of how many are running, stopped and in other states. This view will also have a convenience option to display the CDD being used for this deployment.

### All Instances View

This view shall display all the instance definitions of the deployment across PU’s and machines.

### Machines View

This view shall display all the machines in the deployment and a summary of instances grouped by machines.

### Processing Units View

This view shall display the processing units configured in the CDD and a summary of instances grouped by processing units.

### Agent View

This view shall display the different Agent Types configured in the CDD, and a summary of instances grouped by the instances by the Agent Classes defined in the CDD

### Navigability and Bread Crumbs

Each view shall have links to jump to or navigate to related entities. Bread-crumbs will be shown to track navigation paths.

## User Management

The TEA Agent uses TEA server framework for user management. The TEA Server has to first be configured with users who have permissions to login. Refer to the TEA Administration Guide for adding users. An administrator can then assign users to different groups and roles.

### Roles and Permissions

By default, the BE TEA Agent will provide the following user roles and permissions. The APP\_MANAGER role is the least restrictive (all permissions) and the VIEW\_ALL is the most restrictive role (no permissions) Deployer and Operator are two roles in between with varying permissions. The full table is as shown below. If the default roles are not sufficient, the TEA administrator can create new roles and assign permissions as per requirements.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Role** | **APP\_MANAGER** | **DEPLOYER** | **OPERATOR** | **VIEW\_ALL** |
| **Permissions** | **Enabled**? |  |  |  |
| Create/Import Deployment | Yes | No | No | No |
| Edit Deployment | Yes | No | No | No |
| Delete Deployment | Yes | No | No | No |
| Create Host | Yes | Yes | No | No |
| Edit Host | Yes | Yes | No | No |
| Delete Host | Yes | Yes | No | No |
| Create Instance | Yes | Yes | No | No |
| Edit Instance | Yes | Yes | No | No |
| Delete Instance | Yes | Yes | No | No |
| Start Instance | Yes | Yes | Yes | No |
| Stop Instance | Yes | Yes | Yes | No |
| Kill Instance | Yes | Yes | Yes | No |
| Hot Deploy | Yes | Yes | No | No |
| Copy Instance | Yes | Yes | No | No |
| Update Global Variables | Yes | Yes | No | No |
| Add/Update System Properties | Yes | Yes | No | No |
| Update JVM Property | Yes | Yes | No | No |
| Update Log Level | Yes | Yes | No | No |
| Suspend Agent | Yes | Yes | Yes | No |
| Resume Agent | Yes | Yes | Yes | No |

The TEA Administrator can also assign users to Roles. When a user logs in, only those operations will be permitted by the dashboard to which the user has permissions.

## Command-line Interface

The TEA Agent shall provide support for performing most of the application and configuration management operations from the Python based command-line interface. The TEA Server should be running and the TEA agent should be up & registered with the TEA server to successfully execute the CLI operations. The python script will take the following command arguments -

* TEA server URL
* TEA server username
* TEA user password
* Operation name
* Operation arguments

### ConfigurationMgmt.py

**EditMachine** - command to edit the machine.

editmachine -m MACHINENAME [-n NEWMACHINENAME] [-i IPADDRESS] [-o {Windows,Linux}] [-b BEHOME] [-t BETRA] [-u USER] [-p PWD] [-s SSHPORT] [-f DEPLOYMENTPATH]

**Editdeployment** – command to edit an application deployment.

editdeployment -d APPLICATIONNAME [-c CDDFILE] [-e EARFILE]

**Edit Instance** – command to edit an application instance.

editinstance -d APPLICATIONNAME -i INSTANCENAME [-u PU] [-m MACHINENAME] [-p JMXPORT] [-f DEPLOYMENTPATH] [-ju JMXUSER] [-jp JMXPASS]

**Save Global Variable** – update global variable for one or multiple instances. One variable at a time allowed. Some variables cannot be updated.

saveglobalvariable -d APPLICATIONNAME [-i [INSTANCES [INSTANCES ...]]] -n VARNAME -v VARVALUE

**Save System Property** – update/add a system property for one or multiple instances.

savesystemproperty -d APPLICATIONNAME [-i [INSTANCES [INSTANCES ...]]] -n PROPNAME -v PROPVALUE

**Save Jvm Property** – update s jvm property for one or multiple instances.

savejvmproperty -d APPLICATIONNAME [-i [INSTANCES [INSTANCES ...]]] –n PROPNAME -v PROPVALUE

### ApplicationsMgmt.py

**Deploy Instance** – deploy an application instance/s. Machine or pu or agent class should be given.

deploy -d APPLICATIONNAME [-m MACHINE | -u PU | -a AGENTCLASS] [-i [INSTANCES [INSTANCES ...]]]

**Undeploy Instance** – un deploy an application instance/s. Machine or pu or agent class should be given.

undeploy -d APPLICATIONNAME [-m MACHINE | -u PU | -a AGENTCLASS]

[-i [INSTANCES [INSTANCES ...]]]

**Start Instance** - start an application instance/s. Machine or pu or agent class should be given.

start -d APPLICATIONNAME [-m MACHINE | -u PU | -a AGENTCLASS] [-i [INSTANCES [INSTANCES ...]]]

**Stop Instance** - stop an application instance/s. Machine or pu or agent class should be given.

stop -d APPLICATIONNAME [-m MACHINE | -u PU | -a AGENTCLASS] [-i [INSTANCES [INSTANCES ...]]]

**HotDeploy Application** – hot deploy an application provided by ear file.

hotdeploy -d APPLICATIONNAME -e EARFILE

**Add Machine** – add a new machine.

addmachine -m MACHINENAME -i IPADDRESS -o {Windows,Linux} -b BEHOME –t BETRA -u USER -p PWD -s SSHPORT -f DEPLOYMENTPATH

**Create Deployment** – create a new application provided by cdd and ear file.

createdeployment -d APPLICATIONNAME -c CDDFILE -e EARFILE

**Import Deployment** – import an application provided by cdd, ear and st file.

importdeployment -d APPLICATIONNAME -c CDDFILE -e EARFILE -s STFILE

**Create Instance** – create a new instance of an application. Jmx user and password governed by policy in cdd.

createinstance -d APPLICATIONNAME -i INSTANCENAME -u PU -m MACHINENAME

-p JMXPORT [-f DEPLOYMENTPATH] [-ju JMXUSER] [-jp JMXPASS]

**Copy Instance** – copy an existing instance of an application. Jmx user and password governed by policy in cdd.

copyinstance -d APPLICATIONNAME -i INSTANCENAME -n NEWINSTANCENAME -u

PU -m MACHINENAME -p JMXPORT -f DEPLOYMENTPATH

[-ju JMXUSER] [-jp JMXPASS]

**Delete Instance** – Deletes an instance.

deleteinstance -d APPLICATIONNAME -i INSTANCENAME

**Arguments Explained:**

-t SERVERURL – tea server url. Though optional, has default value (http://localhost:8777) if not provided.

-u USERNAME – tea server user name.

-p USERPWD – tea server password.

Operation level:

-m MACHINENAME – the current machine name.

-n NEWMACHINENAME – the new machine name to be changed to.

-i IPADDRESS – the ip address of the machine.

-o {Windows,Linux} – the operating system of machine.

- b BEHOME – the behome location.

-t BETRA – the tra file location.

-u USER – the machine user name.

-p PWD –the machine user password

-s SSHPORT – the ssh port.

-f DEPLOYMENTPATH – the deployment path location.

-d APPLICATIONNAME – the application name

-c CDDFILE – the cdd file location of the application.

-e EARFILE - the ear file location of the application.

-i INSTANCENAME – instance name.

-u PU – processing unit name.

-p JMXPORT – jmx port number.

-ju JMXUSER – jmx user name.

-jp JMXPASS – jmx password

-n VARNAME – global variable name.

-v VARVALUE – global variable value.

-n PROPNAME – property name(system property, jvm property)

-v PROPVALUE – property value(system property value, jvm property value)

**Notes:**

[] – identifies optional parameter

[-t SERVERURL] – If not given default value is [http://localhost:8777](http://localhost:8777/)

[-i [INSTANCES [INSTANCES ...]]] – depicts multiple values can be given. Example –i instance1 ins2 ins3

## TEA Agent Self-monitoring View

The TEA Agent provides a monitoring view where user can monitor and see the stats related to the TEA Agent itself. The Self-monitoring view displays the following charts for both Hourly and 5 minute bucket:

* Average Memory Usage
* Average CPU Usage
* Thread Count

Note: Make sure the property :**be.tea.agent.jmx.port** is set to a unique value in the **be-teagent.props** file. It signifies the JMX port for the TEA Agent**.** It defaults to **5566**.

# TEA Agent functional Specification

The TEA Agent is the back end server process that provides management/monitoring functionality for a BE application/cluster. It communicates with the TEA Server for UI interactions and communicates with BE instances primarily using JMX. This section covers the functionality for the BE TEA Agent.

## TEA Agent as an Installer Option

The BE 5.3 installer will ship with an additional Installation Feature, the “Tea Agent”. A user may choose to not install this component during installation. By default, this feature shall be installed as part of the “Standard Installer”

If installed, a new folder called “teagent” shall be created in “BE\_HOME”. All BE/TEA Agent related artifacts will be installed in this folder.

## “Wrapper” based Command Line

A Tibco “Wrapper” based command line utility to start the agent will be provided. It will be named be-teagent and a corresponding be-teagent.tra will be shipped with proper defaults. This file can specify additional configuration files to be read as part of its application arguments.

## TEA Server Installation

It would be required to separately acquire and install the TEA Server. Refer to the TEA Server documentation for details.

## TEA Agent Configuration

The TEA Agent configuration will be a properties file with detailed documentation of each property and default values as applicable.

## TEA Server and TEA Agent

It should be possible to start the TEA Agent and the TEA Server in any order. If the TEA Agent is started before starting the TEA Server, the agent should retry registration with the TEA Server at a configurable interval.

## TEA Agent State persistence

The TEA Agent shall provide a configurable way for saving its state. By default, it should save its state to the local file system.

## TEA Agent Transport

The agent needs to communicate with remote machines for deployments. In the initial version, only SSH shall be supported. On Windows, users would have to install an SSH Server such as OpenSSH

*Note*: Certificates based login can be setup such that the client (TEA Agent) need not provide a password for each time it needs to execute a remote command.

## TEA Agent Fault Tolerance

Initial Version will not support fault-tolerance of the TEA Agent

# Additions

## Project specific Master TRA file

TEA Agent by default uses the TRA file specified in the host configuration as the master TRA file to configure instance specific TRA files. Sometimes, you would like to setup a project specific TRA file and use that as a template for creating instance TRA files. TEA Agent will support this functionality. Since the installation location of various products and thereby file system paths and classpaths referenced in the TRA are specific to a machine, it will be possible to upload a machine/project specific master TRA file to each machine that will participate in the deployment. All deployments on this machine will use this TRA file to create instance specific TRA files.

Alternately, it will also be possible to specify a location of the master TRA file to use on each host. This is for cases where the TRA file is already available at the specified location.

If the master TRA file is added later on after the initial creation of the deployment or if it is changed at any later point in time, all instances on that machine will be marked as “Needs Redeployment” to indicate that the master TRA file has changed.

With this feature, it would be possible to deploy out-of-box BE projects such as “RMS” via the TEA Agent interfaces.

## Upload and Load - Hot-deploy Classes and Rule Template Instances

It will be possible to hot-deploy decision table and rule template classes via the TEA Agent in addition to doing so via the WebStudio/RMS project. A facility to upload a ZIP file or a JAR file containing the hot-deployment classes will be provided. The file will be uploaded to the location as specified by the “be.engine.cluster.externalClasses.path” property in the CDD. The user can optionally invoke the “Load” function to initiate hot-deployment.

This functionality will only be enabled if the above property is found in the CDD.

The following new Permissions are added to support these additional functionalities:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Role** | **APP\_MANAGER** | **DEPLOYER** | **OPERATOR** | **VIEW\_ALL** |
| **Permissions** | **Enabled**? |  |  |  |
| UPLOAD\_TRA\_PERMISSION | Y | Y | N | N |
| UPLOAD\_CLASSES\_PEMISSION | Y | Y | N | N |
| UPLOAD\_CLASSES\_PEMISSION | Y | Y | N | N |
| UPLOAD\_RULE\_TEMPLATE\_PERMISSION | Y | Y | N | N |
| DEPLOY\_RULE\_TEMPLATE\_PERMISSION | Y | Y | N | N |

# Authentication and SSL Configuration

## Securing BE-TEA Agent / BE PU Instance JMX channel

### **JMX Authentication**

To enable user authentication for JMX connections from the BE-TEA Agent to BE PU Instances, you need to add the following as System Properties for each PU instance via the BE-Agent UI.

be.engine.jmx.connector.authenticate=true

be.auth.type=file|ldap

be.auth.file.location=<file that stores the users/passwords/roles>

Additional configuration is required. Refer to the “User Authentication” Section of the BE Administration Guide.

In the Application Instance configuration, specify the JMX username and password for each PU Instance.

### One way SSL between BE PU Instances (JMX Server) and BE TEA Agent (JMX client)

#### BE-TEA Agent side configuration

Add the following properties to be-teagent.tra

java.property.javax.net.ssl.trustStore=<location of all of BE instance certificate’s trust store>

java.property.javax.net.ssl.trustStorePassword=<truststore password>

Note: All BE instance public certificates must be stored in a single trust store

You may do so using the keytool utility as follows:

*keytool -import -alias pu1 -file <pu1 certificate> -keystore </path/to/mytruststore>*

*keytool -import -alias pu2 -file <pu2 certificate> -keystore </path/to/mytruststore>*

where pu1 and pu2 represent two BE PU instances.

#### BE PU Instance side configuration

Add the following as System Properties for each PU instance via the BE-TEA Agent UI

be.engine.jmx.connector.ssl=true/false(enable ssl or not)

javax.net.ssl.keyStore=<location of BE PU instance’s keystore file>

javax.net.ssl.keyStorePassword=<keystore password>

### Two way SSL between BE PU Instances (JMX Server) and BE TEA Agent (JMX client)

This is currently not supported.

## SSL Configuration for BE TEA Agent / TEA Server connection

Both one way and two way SSL is supported. Refer to the TEA product documentation for configuration details of securing the BE TEA/TEA server channel.

BE-Agent side properties should be added to be-teagent.tra and prefixed with “java.property.”

## SSL Configuration for Browser / TEA Server connection

Both one way and two way SSL is supported. Refer to the TEA product documentation for configuration details of securing the BE TEA/TEA server channel.

# BE Properties

We have added support to manage BE properties (different from “System Properties”). In BE, properties can be specified in the following places:

In property groups in CDD agent sections

In property groups in CDD PU sections

In property groups at the “Cluster” (the top level) in the CDD

For same name properties specified in multiple places in the CDD, the PU level value overrides the Agent level value which overrides the Cluster level value. This is called the “effective CDD value”

BE TEA Agent will display the “CDD Value” as computed above in the Instance level details

## PU Instance configuration

There will be a way for the user to override the effective CDD value via the TEA agent. When deployed, this value will be placed in the instance TRA file. The value in the instance TRA file will override the effective CDD value.

For CDD properties overridden this way, there will be a way for the user to delete the override.

There will also be a way for users to add and delete new properties from the TEA Agent (those that do not come from the CDD)

## Group Operation

When multiple instances are selected for overriding property values, those properties with the same “effective value” will be allowed to be modified via a group operation. The “effective value” of a property is:

If saved locally, the saved value.

Else if deployed via override, the deployed value

Else the “effective CDD” value if exists.

There will also be a way provided to override different valued properties, or to delete properties as a group operation.

# Global Variables

Handling of Global Variables is now similar to that of BE Properties. The default effective value of a Global Variable will be the one from the CDD if present, else as defined in the project

## PU Instance configuration

There will be a way for users to override or delete the overridden value of Global Variables. If overridden and deployed

## Group Operation

When multiple PU instances are selected, it would be possible to override the values of the Global Variables whose “effective value” is the same for all selected instances. The effective value is:

If saved locally, the saved value.

Else if deployed via override, the deployed value

Else the “effective CDD” value if exists,

Else the default value from the project.

There will also be a way provided to override different valued properties to a common value for all selected instances.

# Charts

The following charts will be provided

## Five Minute Statistics

The following charts will be shown with an aggregation interval of five minutes

### Average Used Memory

Five minute averages of Used Memory Percentage plotted for the last few hours.

### Average CPU

Five minute averages of CPU consumption plotted for the last few hours.

### Average RTC Transaction Latency

Five minute averages of RTC Transaction Latencies plotted for the last few hours.

NOTE: *This is only available for “Cache” based deployments.*

### Average RTC Transaction Throughput

Five minute averages of RTC Transaction throughput plotted for the last few hours.

NOTE: *This is only available for “Cache” based deployments.*

### Total Locks Held

Five minute totals of Local, Total and Cluster locks will be plotted for the last few hours.

NOTE: *This is only available for “Cache” based deployments.*

### Even Throughput

This chart will allow users to track event throughput for selected Destinations. Users can specify in a configuration file, the URI of Destinations whose performance is to be tracked/charted. Total throughput for these destinations for five minute intervals would be plotted for the last few hours.

### Event Throughput Configuration

To configure the destinations to be monitored event throughput , update file *beEntityMap.xml* , located at the location : *BE\_HOME/teagent/config/ .*

*BE\_HOME : The path to the BE installation (e.g. /opt/tibco/be/5.3)*

*This file needs to be updated with the destinations under a specific application. Create a new entry for the application in the following manner :*

<app name="*APPLICATION\_NAME*" >

<entity-group type="destination" >

<entity name="*DESTINATION\_NAME*" alias="*DESITNATION\_ALIAS*" />

</entity-group>

*</app>*

*APPLICATION\_NAME : Name of the application which will holds the destinations.*

*DESTINATION\_NAME: URI of the destination to be monitored*

*DESTINATION\_ALIAS : Update this attribute if an alias is to be shown in the chart for a particular destination.*

*NOTE: If no entry is present for an application in this file , the event throughput will display all the destinations. The property “max-series” in the file will decide the number of chart series to be shown. Update this property to limit the number of series.*

## Hourly Statistics

All the above charts at five minute aggregations would also be available at hourly aggregations. The hourly charts will be plotted for until the last one day.

## Rule Performance

This chart will show best performing rules since the start of the instance. The number of rules to be shown in the chart is configurable.

NOTE: *Since Rules statistics collection in itself causes performance degradation, data is not collected for this chart by default. To enable data collection to drive this chart, set the following BE properties in the BE PU instances:*

*com.tibco.be.metric.publish.enable=true*

*be.stats.enabled=true*

You can control the number of worst performing rules to be displayed by adding a <maxDataPoints> setting to EntityMetricViewConfig.xml file located in the teagent/config folderas shown below:

<section sectionId="3" displayName="Rule Statistics">

      <chart>

         <id>13</id>

         <chartType>column</chartType>

         <entity>agent</entity>

         <name>executionTimeChart</name>

         <description>chart which avg rule execution time per rule</description>

          <…………………>

<maxDataPoints>10</maxDataPoints>

<…………………>

## Garbage Collector Details

This chart will show the Garbage Collector details. (Whether to show it, will be configurable)

## Memory Pools Details

This chart will display the various Memory Pool usages such as PermGen/Eden Space etc. (Whether to show it, will be configurable)

# Rules and Alerts

In BE TEA Agent, *Monitored Entities* are those entities for which certain metrics are computed. The BE TEA Agent will allow users to write rules on these monitored entities, such that when certain metrics change, certain actions can be taken or alerts generated. An example rule could be "when the total number of running PU instances is less than 50% of the total PUs" then mark the cluster health as critical". While authoring rules, users would have to specify "Set Conditions" and "Set Actions". Set Conditions are those conditions which when satisfied would trigger "Set Actions" of the rule. Users would also have to specify "Clear Conditions" and "Clear Actions" - generally to counter the effects of the Set Condition/Set Actions. When "Clear Conditions" are met, its associated "Clear Actions" are triggered. Clear Conditions and Clear Actions only have meaning if corresponding Set Conditions have already been met.

## Monitored Entities

The following table shows the Monitored Entities and metrics that are available for use in rules with each of them.

|  |  |  |
| --- | --- | --- |
| **Monitored Entity** | **Metric** | **Description** |
| Processing Unit |  | An instance of any PU |
|  | Average CPU (%) | Average CPU usage percentage, averaged over 5 minutes. |
|  | Average Used Memory (%) | Average Memory usage percentage (evaluated over max allocated memory), averaged over 5 minutes. |
|  | Is Running | PU is in running or stopped state. |
|  | Thread Count | Count of the running threads in 5 minute groupings |
|  | Deadlocked Thread Count | Count of the deadlocked threads in 5 minute grouping |
|  | Processing Unit | The associated PU name |
|  | Processing Unit Name | The associated instance name. |
| RTC Transactions  *(Available for Cache deployments only)* |  | The RTC transactions carried out in the inference engine. |
|  | Pending Locks | The count of pending locks to release in 5 minute grouping |
|  | RTC Transaction Count | Number of RTC Transactions in 5 minute grouping |
|  | Processing Unit | The associated PU name |
|  | Processing Unit Name | The associated instance name. |
|  | Agent Name | The name of the agent. |
| Event Throughput |  | The throughput of the events in the inference engine. |
|  | Event Throughput | Throughput in Events per second in 5 minute intervals. |
|  | Destination URI | The URI of the destination where the event was received. |
|  | Processing Unit | The associated PU name |
|  | Processing Unit Name | The associated instance name. |
|  | Agent Name | The name of the agent. |
| BusinessEvents Rules  *(Will be in effect only if properties are set in the BE instances. Ref Sec 8.3)* |  | The Rules executed in the inference engine. |
|  | Rule Execution Time | The average rule execution time in milliseconds. |
|  | Rule Name | The rule URI |
|  | Processing Unit | The associated PU name |
|  | Processing Unit Name | The associated instance name. |
|  | Agent Name | The name of the agent. |
| Cluster |  | The cluster or application as a whole. |
|  | Processing Units in Running (%) | The percentage of PUs in Running state (of the total deployed PUs in the cluster.) |
|  | Processing Units in Normal (%) | The percentage of PUs in Normal state (of the total deployed PUs in the cluster.) |
|  | Processing Units in in Warning (%) | The percentage of PUs in Warning state (of the total deployed PUs in the cluster.) |
|  | Processing Units in in Critical (%) | The percentage of PUs in Critical state (of the total deployed PUs in the cluster.) |

## Set Actions and Clear Actions

The following Actions are supported as part of the Set and Clear Actions.

### Set the Health of the Monitored Entity

This sets the health of the metric in one of “Normal”, “Warning” or “Critical”. These actions can only be used with users with a different privilege. (See section 9.4)

### Send an Email

This action allows users to send emails. (SMTP server connection parameters need to be configured in the BE TEA Agent configuration)

### Write to a log file

This action allows users to write the set and clear conditions to a log file.

### Alert Only

Each of these actions is associated with an "Alert Text" and "Alert Details". Users will also be able to specify a Severity for the Alert: Low, Medium, High. There is also a “Normal” Severity. This can be used to set the Alert state of a Monitored Entity as “Normal”

## Placeholders in Alert Text

While setting the Alert Text, users can use certain tokens as placeholders for dynamic values. These placeholders will be substituted with actual values that triggered the rule at runtime.

The following placeholders or tokens are available for use in various actions.

|  |  |  |  |
| --- | --- | --- | --- |
| **Monitored Entity** | **Metric** | **Alert Token** | **Description** |
| Processing Unit |  |  | An instance of any PU |
|  | Average CPU (%) | ${average.cpu.usage} | Average CPU usage percentage, averaged over 5 minutes. |
|  | Average Used Memory (%) | ${average.memory.usage} | Average Memory usage percentage (evaluated over max allocated memory), averaged over 5 minutes. |
|  | Is Running | ${isrunning} | PU is in running or stopped state.(0/1) |
|  | Thread Count | ${total.threadcount} | Count of the running threads in 5 minute groupings |
|  | Deadlocked Thread Count | ${deadlocked.threadcount} | Count of the deadlocked threads in 5 minute grouping |
|  | Processing Unit | ${processing.unit.name} | The associated PU name |
|  | Processing Unit Name | ${instance.name} | The associated instance name. |
| RTC Transactions |  |  | Instances of PUs of type Inference Agent. |
|  | Pending Locks | ${pending.locks} | The count of pending locks to release in 5 minute grouping |
|  | RTC Transactions | ${transaction.rate} | Number of RTC Transactions in 5 minute grouping |
|  | Processing Unit | ${processing.unit.name} | The associated PU name |
|  | Processing Unit Name | ${instance.name} | The associated instance. |
|  | Agent Name | ${agent.name} | The name of the agent. |
| Event Throughput |  |  | The throughput of the events in the inference engine. |
|  | Event Throughput | ${event.throughput} | The value of the event throughput in millis. |
|  | Destination URI | ${destination} | The URI of the destination where the event was received. |
|  | Processing Unit | ${processing.unit.name} | The associated PU name |
|  | Processing Unit Name | ${instance.name} | The associated instance name. |
|  | Agent Name | ${agent.name} | The name of the agent. |
| BusinessEvents Rules |  |  | The Rules executed in the inference engine. |
|  | Rule Execution Time | ${berule.exec.time} | The average rule execution time in milliseconds. |
|  | Rule Name | ${berule.name} | The rule URI |
|  | Processing Unit | ${processing.unit.name} | The associated PU name |
|  | Processing Unit Name | ${instance.name} | The associated instance name. |
|  | Agent Name | ${agent.name} | The name of the agent. |
| Cluster |  |  | The cluster or application as a whole. |
|  | Processing Units Running (%) | ${percent.running} | The percentage of PUs in Running state (of the total deployed PUs in the cluster.) |
|  | Processing Units in Normal (%) | ${percent.normal} | The percentage of PUs in Normal state (of the total deployed PUs in the cluster.) |
|  | Processing Units in Warning (%) | ${percent.warning} | The percentage of PUs in Warning state (of the total deployed PUs in the cluster.) |
|  | Processing Units in Critical (%) | ${percent.critical} | The percentage of PUs in Critical state (of the total deployed PUs in the cluster.) |
| All |  |  |  |
|  | - | ${entity.health} | The health set to the rule entity at the Set/Clear action of the Rule(Only applicable for Set Health Action) |
|  | - | ${alert.timestamp} | The time at which the action was triggered. |
|  | - | ${alert.priority} | The priority of the alert created by the associated action. |
|  | - | ${alert.type} | The value signifying action was triggered as part of the trigger of the Set or Clear condition. |
|  | - | ${rule.owner.name} | The name of the rule owner |
|  | - | ${application.name} | The application/deployment name. |

## Specifying Set and Clear Conditions

The BE TEA Agent will provide a UI to specify Set and Clear conditions. Users will be able to build a nested expression with "AND" and "OR" operators to represent the set/clear condition.

## Health Rules

Health Rules are those rules which determine the health of a Monitored Entity. Since the health of a system cannot be different as seen by different users, only certain users with RULE\_ADMIN Role will be able to author "Health Rules". These are to be considered as "Global Rules" spanning across the user base. Health rules should be authored judiciously so as not to create conflicting rules that would confuse operators.

Additionally, users with the RULE\_AUTHOR\_ADMIN Role will also be able to modify/see all other rules created by non RULE\_AUTHOR\_ADMIN privileged users. If an rules admin user edits a non-admin rule and introduces a Health Action, the rule ownership will be transferred to the rule admin user.

## Alerts

Each rule action generates an Alert. The BE TEA Agent will provide a UI for Alert Management. It will allow users to query for alerts and to clear alerts. Users can view alerts generated by their own rules or generated by RULE\_ADMIN users’ rules. Users will only be able to clear alerts generated by their own rules. RULE\_ADMIN users will be able to view and clear all users’ alerts.

# Export /Import of Application

## Export from TEA

1. Users are allowed to export the application from TEA UI.From application level menu on click of ‘Export Deployment’ zip file is download to default download location of browser.
2. The zip file contains application cdd,application ear and xml file which has instance details like name,jmx port,host,behome ,GV,system properties .

## Import to TEA

1. The application exported from TEA is allowed to import in TEA from UI.From Import Deployment dropdown on click of BE TEA Agent Deployment ,path of exported application zip is asked, on click of save application is created .
2. While importing the application the application with same name must not be exit in TEA.

## Export from TIBCO Administrator

Many customers use Tibco Administrator (Tra based), to manage BE applications. For those customers who want to migrate to BE Agent, need to follow below steps for migration.

1. Run below command to export data from Tibco Administrator.

**./AppManage -batchExport -domain domain\_name -user username -pw password -exportDeployed -dir path\_to\_store\_exported\_data**

As an outcome of above export we get below files:

AppManage.batch – contains name of applications, and their respective ear and xml file names,Application Ear file,Application export config file(xml)

1. In order to migrate below activities need to done.
2. Along with the above files, add .cdd files of applications to the same folder manually. The name of the .cdd file should match name of application as per .batch file.
3. Add a csv file to give machine auths and some other details. This will be discussed in details further. A sample config.csv file is provided.
4. Pack all the above files into a single zip file.

## Import TIBCO Administrator Application

There are two ways the above exported data can be imported in BE Agent.

## GUI Approach -:

1. Keep zip file from above process in place.
2. Go to Tea and browse to BE Agent.
3. From Import Deployment dropdown click on TIBCO Administrator Deployment.
4. Provide zip file as said above and click migrate.
5. On the first page select the applications you want to migrate and proceed.
6. Check machines and application details and fill the missing data. Alternatively the missing data can be provided in config.csv file as mentioned above. CSV is though non mandatory in case of GUI approach (config.csv file explained in detail below, a sample file is also provided).
7. After everything verified, click create. Note that create button will not be enable unless all the mandatory data is provided.
8. Verify the created machines, applications, instances, Gv properties e.t.c

## CLI approach -:

Migration can also be done via command line interface.

1. For this approach, it is mandatory to have config.csv file for the missing data. Pack the config.csv file along with other files in a zip as explained above.
2. Set the python and other paths as explained in the section 2.6.
3. Run the below sample command to migrate application.

python adminToAgentMigration.py -t "Tea\_Server\_Url" -u admin -p admin migrate -z Zipfile\_Path

## Sample CSV File -:

1. A sample config.csv file is attached in python scripts folder.
2. The name of the csv file should be config.csv
3. The configuration data, like name of application and instance names should match the application and instance names as in exported data from administrator.
4. Format of csv file data

APPLICATIONNAME,INSTANCENAME,IP,JMXPORT

APPLICATIONNAME – Name of application

INSTANCENAME – Name of instance related to Application in first column.

IP – IP address

JMXPORT – Jmx Port of the instance

IP,OS,DEPLOYMENTPATH,SYSUSER,SYSPASS,SSHPORT

IP – IP address

OS – unix, windows

DEPLOYMENTPATH – deployment Path

SYSUSER – Machine user name

SYSPASS – Machine user password

SSHPORT – SSH Port(default 22)

# Deployment Profiles

As user might want to deploy the application in different environment using set of properties.To do that user need to create profile for specific environment. A profile is associated with application where user can add/override Global variables, System properties and BE properties.An application can have multiple profiles .

Application profiles are listed under Profiles Tab pn application instances page.From profiles page user can select the default profile of the application which is used by instances. When default profile is selected then application instances status becomes ‘Needs Deployment’. When instance/s deployed **–p parameter with <default\_profile>.properties** added in <application\_name> .sh file of application. At the time of instance deployment all profile files are copied to deployment location.

## Add Profile:

From profiles page user can add new profile where he can give the profile name and Global variables/System properties / BE properties.Save button is enabled only on change the any property(GV,System propertyP,BE Property).

## Edit Profile:

From profiles page user can update the profile by clicking edit icon. Popup is shown with existing properties when user can update Global variables/System properties / BE properties.Save button is enabled only on change the any property.If selected profile is default profile then instance status is changed to ‘Needs Deployment’

## Delete Profile:

From profiles page user can delete the profile by clicking the Delete icon If deleted pr profile is default profile then instance status is changed to ‘Needs Deployment’

# Upload External Jars

User might have external jars which he/she wants to use in his application. Usually those jars are kept in %BE\_HOME% /lib/ext/tpcl folder. From BE-TEA agent user can upload external jars on selected BE Home.

* From Master host page select the BE home and jar/zip of jars and click Finish.
* Files are uploaded to %BE\_HOME%/lib/ext/tpcl/beTeagentUpload folder.
* To upload multiple jars user need to create zip of jar and then upload a zip
* When user deploy the instances following entry is appended to TRA file which gets deployed

*tibco.class.path.extended %BE\_HOME%/lib/ext/tpcl/beTeagentUpload%PSP%%tibco.class.path.extended%*

# i18n Support

User may want to use multi byte characters as application/instance/machine name. To support multi byte characters user must have to add **java.property.file.encoding=UTF-8** property to TEA Server TRA file. After adding this property user can use the multi byte characters in application/instance/machine name.

Messages\_<locale>.properties file contains all key value pairs of messages used in BE-TEA agent. The default messages\_en\_US.properties file is located in ***BE\_HOME/teagent/config/messages/*** folder. If user wants to give another messages\_<locale>.properties file path, then user can set the new locale messages\_<locale>.properties file path against **be.tea.agent.message.file** property in be-teagent.props file.

# Alert Notifications

Every rule action generates an Alert. All the alerts are listed on Alert tab. User can see all the alerts after clicking on Alert tab. Alert notifications provide the notification of new alert generation on alert tab. It displays the number alerts generated count on the upper right side corner of the alert tab. When User clicks on alert tab, he will see newly generated alerts in bold font. After clicking refresh or tab change, font of new alert changes into normal.