TIBCO ActiveSpaces® Transactions

Administration

Software Release 2.5.8
Published November 10, 2017

Two-Second Advantage®

Important Information

Contents

List of Figures

List of Tables

List of Examples

About this book

This guide describes how to install, configure, and monitor TIBCO ActiveSpaces® Transactions applications. It provides both task oriented and a reference guide for TIBCO ActiveSpaces® Transactions administration.

It is intended for the following types of readers:

- Java developers who want to administer TIBCO ActiveSpaces® Transactions as part of application development.
- System operators deploying and administering TIBCO ActiveSpaces® Transactions applications.

This guide is organized into these general areas:

- An introduction to the available administration tools. This information is in Chapter 1.
- Practical sections on the management features in TIBCO ActiveSpaces® Transactions . These sections contain descriptions of administration tasks and the commands required to perform the tasks. The tasks are described using both the TIBCO ActiveSpaces® Transactions **Administrator** and the `administrator` command line.
- Reference material. This information is in Chapter 7.

Related documentation

This book is part of a set of TIBCO ActiveSpaces® Transactions documentation, which also includes:

TIBCO ActiveSpaces® Transactions Installation — This guide describes how to install the TIBCO ActiveSpaces® Transactions software.

TIBCO ActiveSpaces® Transactions Quick Start — This guide describes how to quickly get started using Java IDEs to develop TIBCO ActiveSpaces® Transactions applications.

TIBCO ActiveSpaces® Transactions Architect's Guide — This guide provides a technical overview of TIBCO ActiveSpaces® Transactions .

TIBCO ActiveSpaces® Transactions Java Developer's Guide — This guide describes how to develop TIBCO ActiveSpaces® Transactions applications.

TIBCO ActiveSpaces® Transactions Performance Tuning Guide — This guide describes the tools and techniques to tune TIBCO ActiveSpaces® Transactions applications.

TIBCO ActiveSpaces® Transactions System Sizing Guide — This guide describes how to size system resources for TIBCO ActiveSpaces® Transactions applications.

TIBCO ActiveSpaces® Transactions Javadoc — The reference documentation for all TIBCO ActiveSpaces® Transactions APIs.

Conventions

The following conventions are used in this book:

Bold — Used to refer to particular items on a user interface such as the **Event Monitor** button.

`Constant Width` **—** Used for anything that you would type literally such as keywords, data types, parameter names, etc.

`Constant Width Italic` **—** Used as a place holder for values that you should replace with an actual value.

Community

The TIBCO ActiveSpaces® Transactions online community is located at `https://devzone.tibco.com`. The online community provides direct access to other TIBCO ActiveSpaces® Transactions users and the TIBCO ActiveSpaces® Transactions development team. Please join us online for current discussions on TIBCO ActiveSpaces® Transactions development and the latest information on bug fixes and new releases.

1

Introduction

This chapter introduces the management tools that you use with TIBCO ActiveSpaces® Transactions . The tools discussed in this chapter are:

- Administrator: a web-based graphical administration tool
- JMX consoles

A command-line interface to all of the administrative capabilities is also provided. A complete reference to the command-line can be found in the section called “Command line interface” on page 121. For most tasks, TIBCO ActiveSpaces® Transactions Administrator or JMX consoles are easier to use. However, the command-line interface can be invoked by shell scripts for automation of specific tasks. It is also the only way to install and start a Domain Manager node.

More details on the information in these tools is described in other chapters of this guide.

Administrator

TIBCO ActiveSpaces® Transactions Administrator provides a graphical user interface (GUI) for managing TIBCO ActiveSpaces® Transactions domains and nodes. It automatically discovers all nodes on the network and presents them to you as a visual collection. You can log on to nodes or domains, and then "drill down" to monitor and control managed elements within them.

The TIBCO ActiveSpaces® Transactions Administrator login screen is shown in Figure 1.1.

Figure 1.1. Domain login dialog

Upon a successful login into the domain, you are automatically logged into all currently active managed nodes in the domain.

See Chapter 5 for details on defining principals and their credentials that can manage a domain.

Displaying domains and nodes

After you start and log in to TIBCO ActiveSpaces® Transactions Administrator, you see an overview screen that displays the `Domain Browser` in the left-hand pane and the `Manage Nodes` display in the right-hand pane, as shown in Figure 1.2. The left-hand pane contains these areas:

- **Domain** - the top level container for everything in the Domain Browser. This is the domain that was logged into.
- **Application Nodes** - a container for all application nodes being managed by the domain.

  Only managed nodes show up in the `Application Nodes` container.

- **Domain Groups** - a container for all defined domain groups.
- **Administration Node** - a container for the node hosting the Domain Manager.
- **High Availability Partitions** - a container for all partitions defined in the cluster.

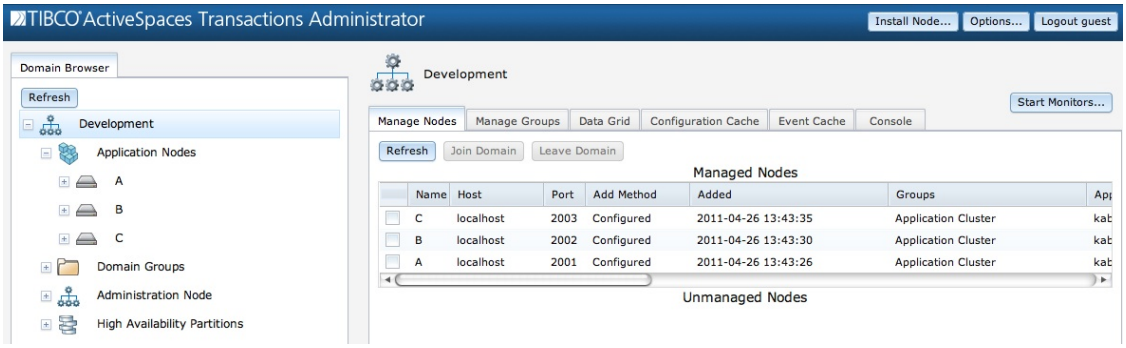

Figure 1.2. TIBCO ActiveSpaces® Transactions Administrator

When you select an application node, information about the node is displayed in the right-hand pane, as shown in Figure 1.3.

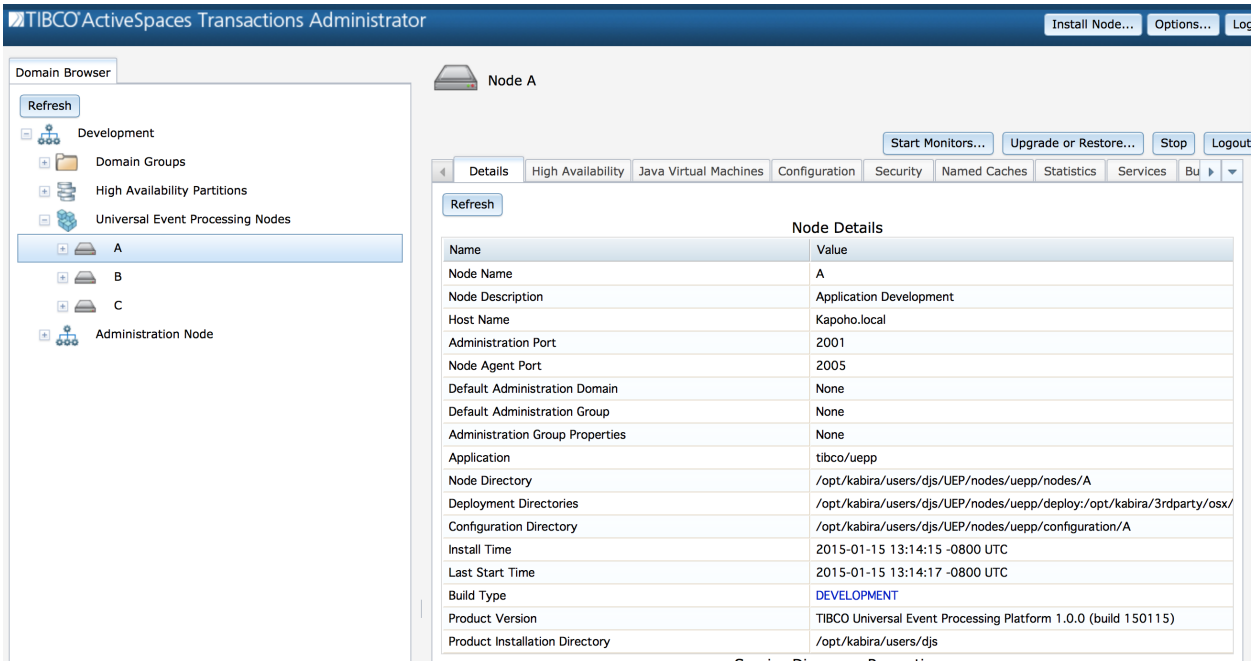

Figure 1.3. Application node display

Managing the elements on a node

Recall the managed element hierarchy shown in TIBCO ActiveSpaces® Transactions **Architect's Guide** – you can see that most elements are contained within a node. To access these elements in TIBCO ActiveSpaces® Transactions Administrator, you must be logged into the node (usually this happens automatically when you log into a domain). When you are logged into a node a set of tabs tabs appear as shown in Figure 1.3. These tabs provide detailed displays and control of the individual managed elements. The tabs for a node are:

- **Details** - generic node details
- **High Availability** - high availability status, including quorum status, discovered nodes, and known partitions.
- **Java Virtual Machines** - display and managed JVMs running on the node.
- **Configuration** - display and control node configuration data.
- **Security** - display security information in affect on the node.
- **Statistics** - display node runtime statistics.
- **Services** - control services, endpoints, and session.
- **Business State Machines** - control and monitor business state machines.
- **Console** - console access for the node.

Managing domains

When you log into TIBCO ActiveSpaces® Transactions Administrator, you are automatically logged into the Domain Manager node that is hosting the TIBCO ActiveSpaces® Transactions Administrator Web Server.

As shown in Figure 1.2, these tabs are available when the domain is selected to manage the domain:

- **Manage Nodes** - add and remove nodes from the domain.
- **Manage Groups** - add domain groups.
- **Configuration Cache** - manage the domain configuration cache.
- **Log Messages** - display log messages from managed nodes.
- **Console** - console access to the domain.

Log Messages

Nodes generate log messages for normal and exceptional conditions. These log messages are available in:

- Node log files.
- Domain Manager log message cache.
- Domain Manager event monitor.

No matter where log messages are viewed, they have the same content:

- **Time Stamp** - time log message was generated.
- **Priority** - message priority.
- **Identifier** - message identifier.

- **Originator** - message originator identifier.
- **Transaction** - transaction identifier for transaction in which message was logged.
- **Message** - a textual message.

In addition, log messages displayed from the Domain Manager log message cache or monitor also contain the node name that generated the log message. Here is an example log message displayed in the Domain Manager event monitor:

```
Node Name = A
Date Time = 2008-09-10 12:57:38
Event Topic = kabira.kts.security
Event Identifier = switchadmin::EventIdentifiers::OperatorActionSucceeded
Event Originator = switchadmin::PluginServiceImpl:1 (344579:8358104:7100:1 offset 67017096)

Message = Administrator command [display] on target [security] executed by principal
[guest] succeeded.
```

Log message cache The Domain Manager log message cache provides a historical cache of all messages raised by nodes being managed by a domain manager. The log message cache supports the following filters:

- **Node Name** - only show messages for a specific node.
- **Topic** - only show messages for a specific topic.
- **Identifier** - only show messages with a specific messages identifier.
- **Originator** - only show messages from a specific originator.
- **Contains** - only show messages that contain a specific phrase.
- **Start Time** - only show messages after a specific start time.
- **End Time** - only show messages before a specific end time. If not specified all messages from start time are displayed.

Figure 1.4 shows how to access the log message cache.

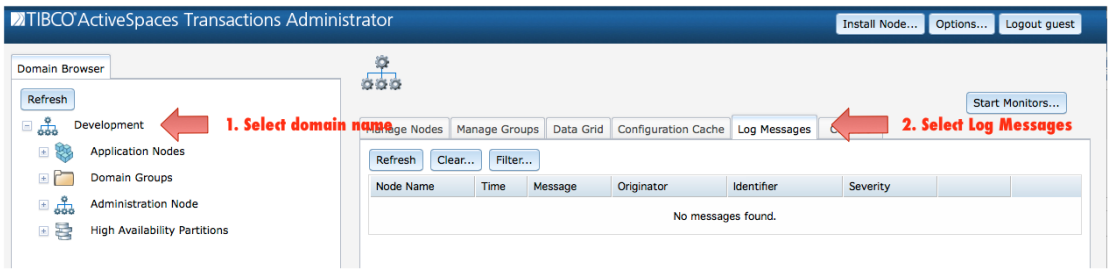

Figure 1.4. Accessing the log message cache

Log message monitor The log message monitor is available on the domain. It provides asynchronous notification of log messages reported by all nodes being managed by the domain. Figure 1.5 shows the running message monitor.

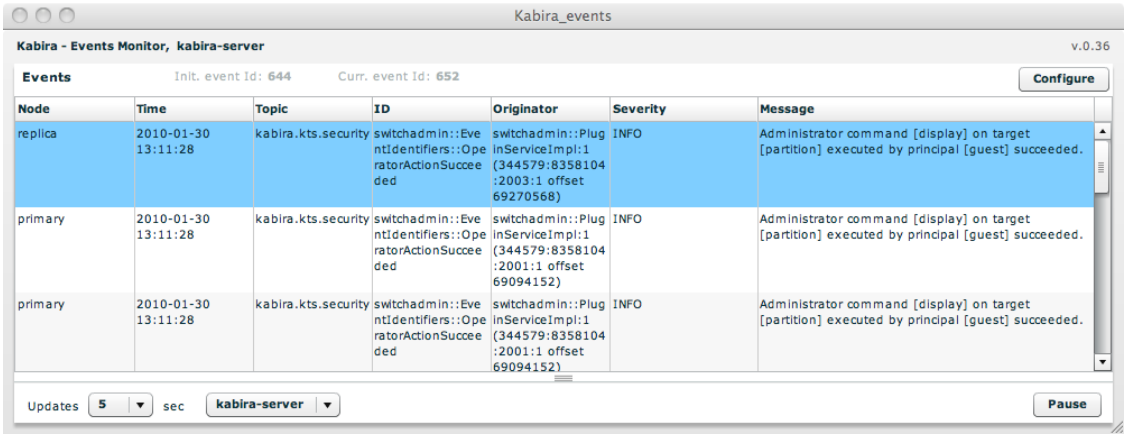

Figure 1.5. Log message monitor

Figure 1.6 shows how to access the log messages monitor.

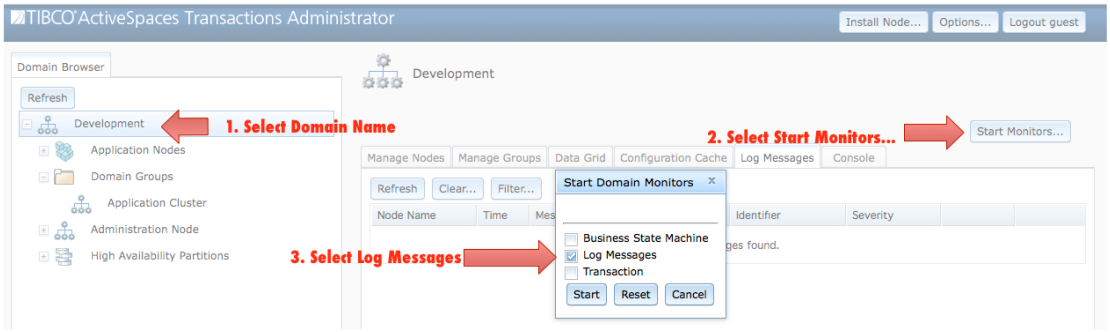

Figure 1.6. Accessing the log messages monitor

Performance monitors

TIBCO ActiveSpaces® Transactions Administrator provides the following monitors:

- CPU utilization monitor
- Shared memory monitor
- Transaction monitor

The CPU and shared memory monitors are available on any running node - this includes application and Domain Manager nodes. Figure 1.7 shows the CPU monitor and Figure 1.8 shows the shared memory monitor.

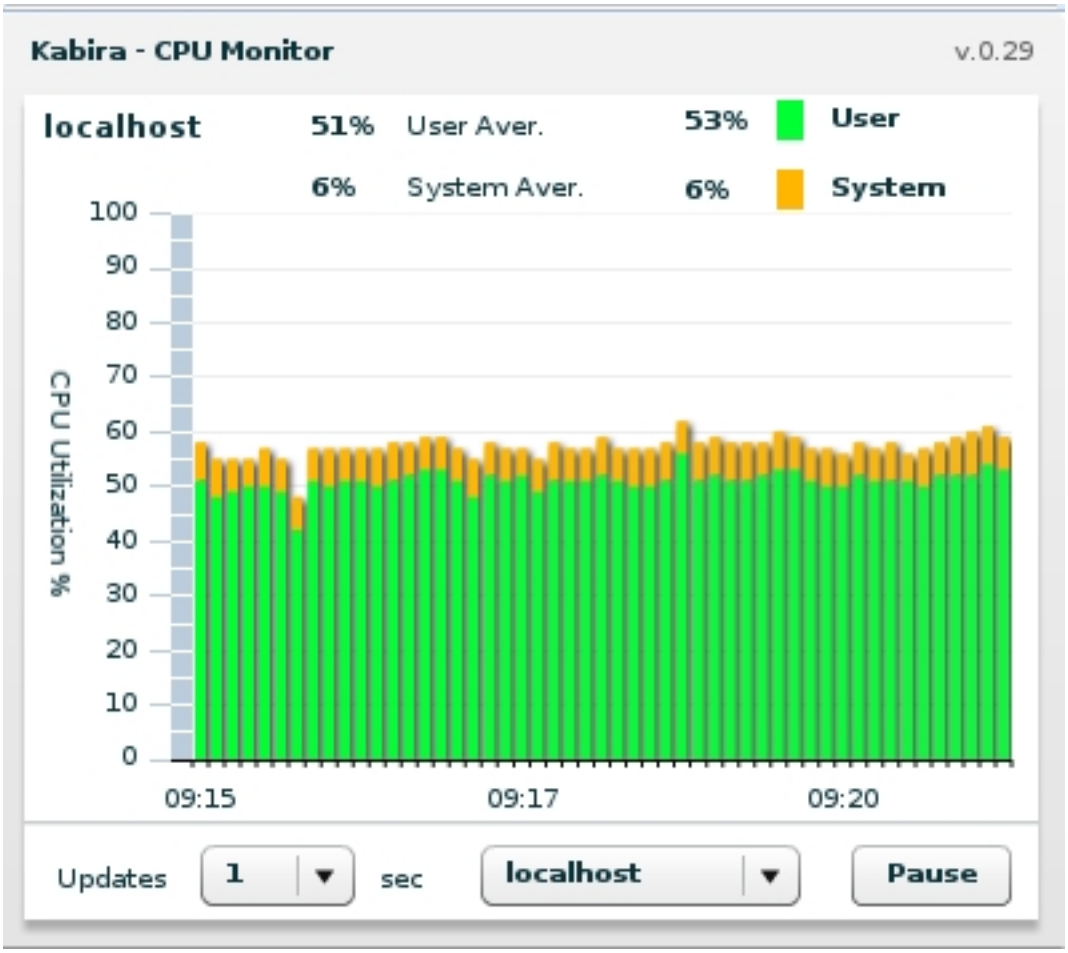

Figure 1.7. CPU monitor

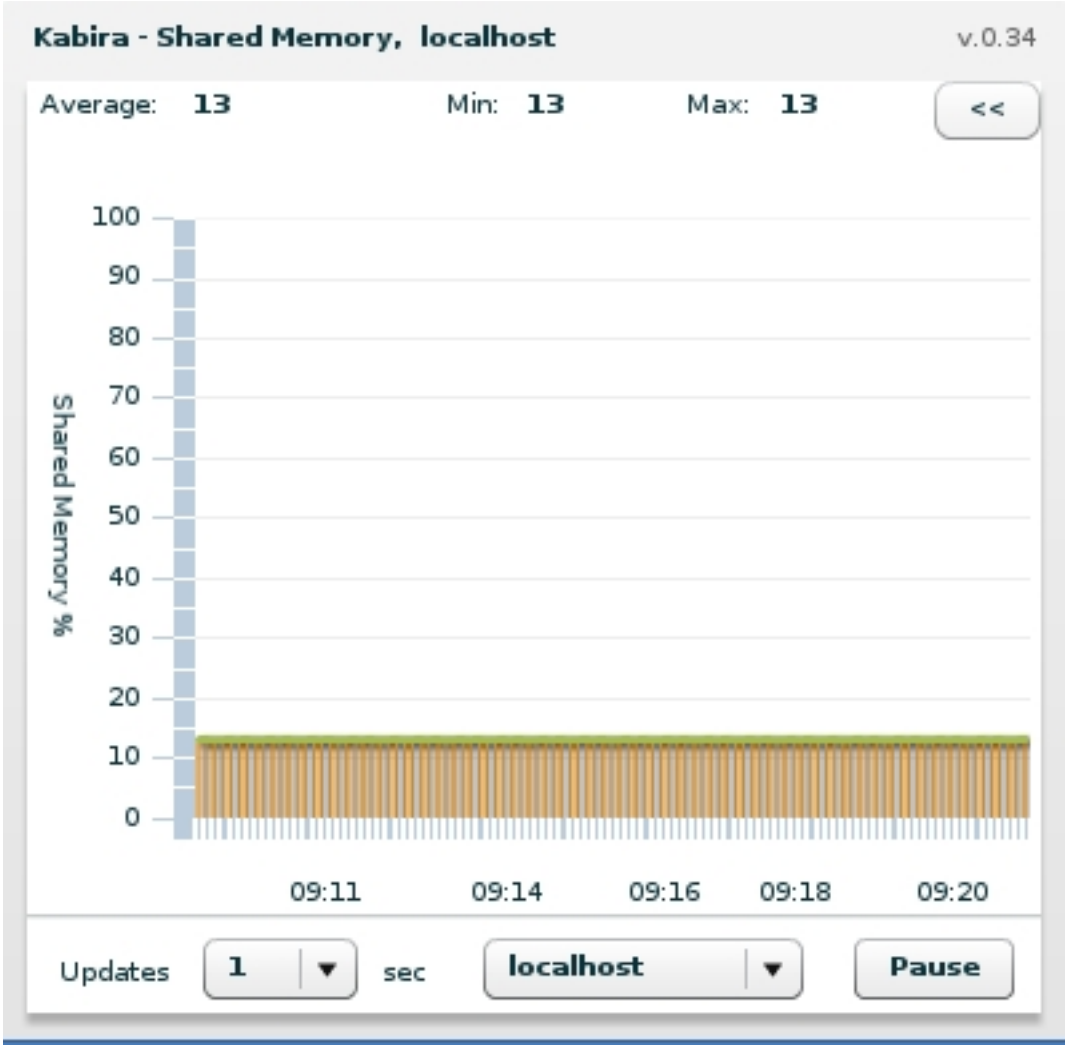

Figure 1.8. Shared memory monitor

The transaction monitor is available per node and on the domain. Starting the transaction monitor on the domain displays an aggregate transaction rate for all nodes being managed by the domain. Two different displays are available from the transaction monitor:

- total and average number of transactions (see Figure 1.9).
- minimum, average, and maximum transaction latency (see Figure 1.10).

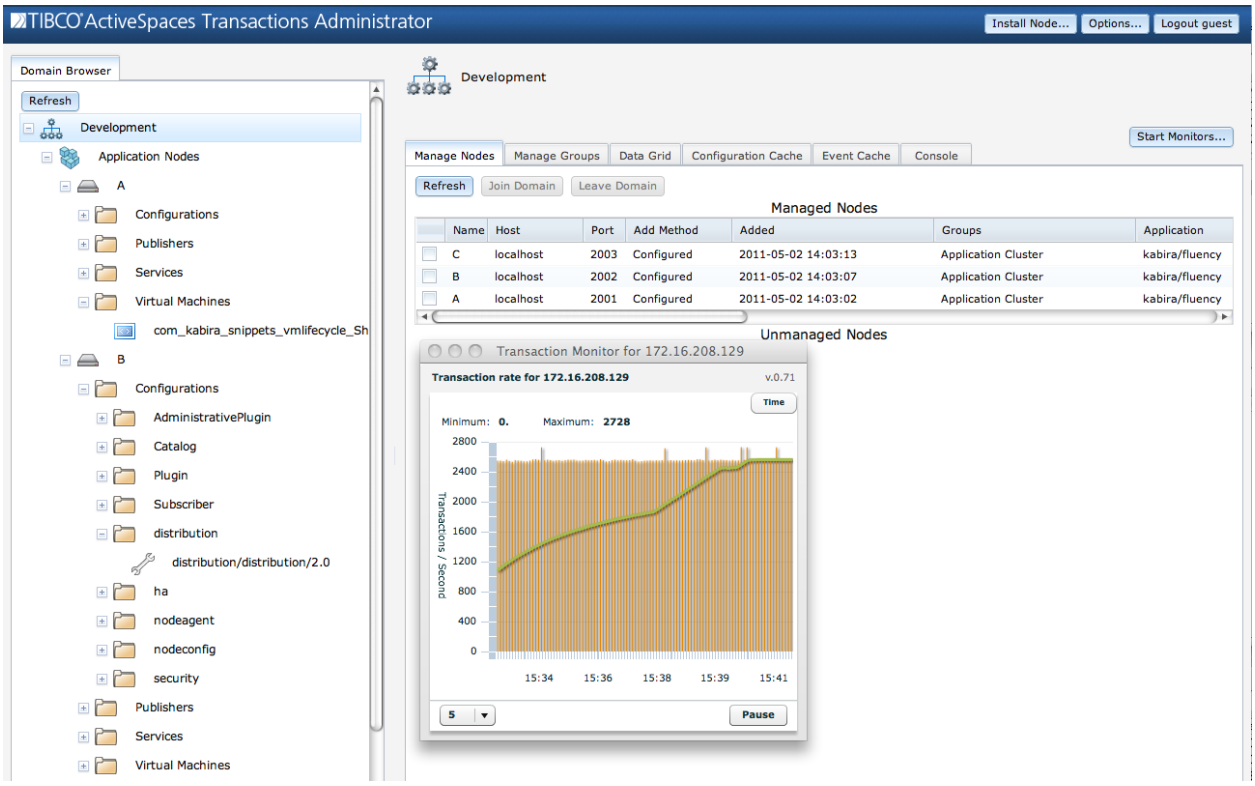

Figure 1.9. Transaction count

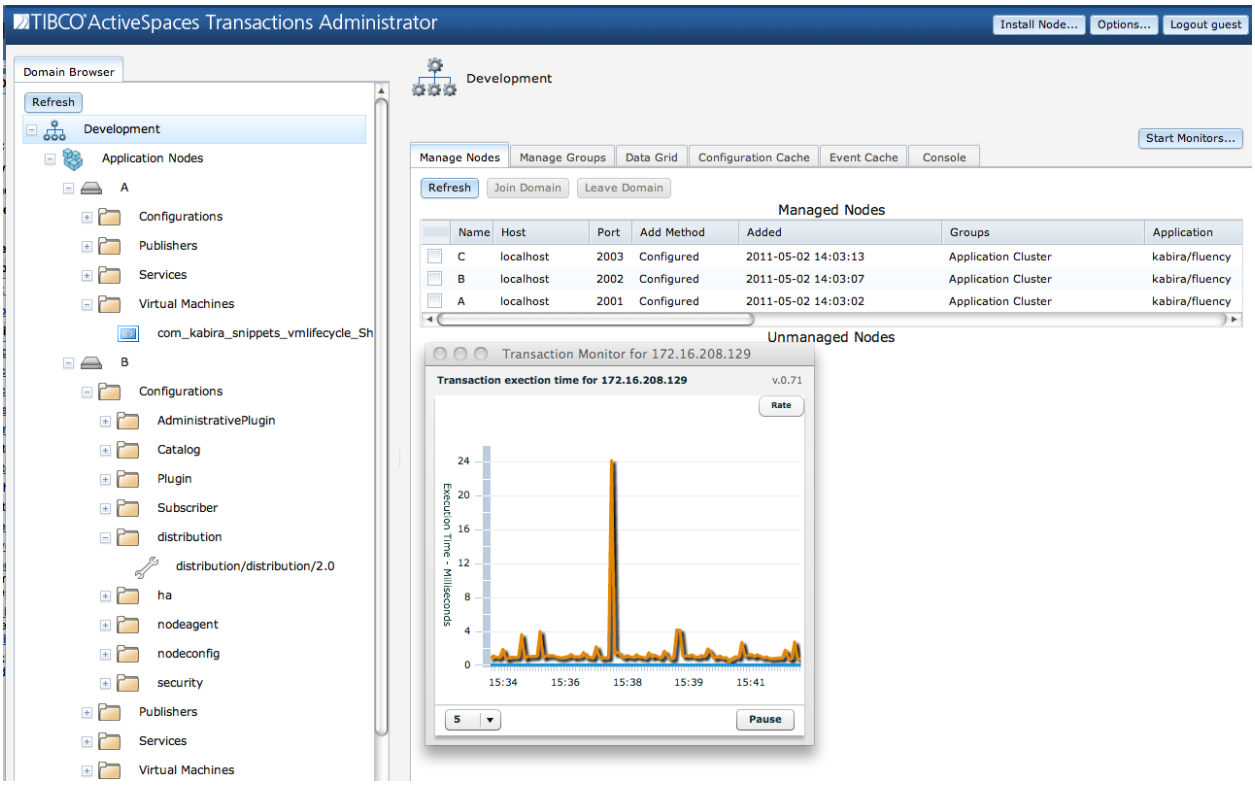

Figure 1.10. Transaction latency

Console commands

TIBCO ActiveSpaces® Transactions Administrator provides a way to perform most management tasks. However, sometimes solution specific administrative commands may be introduced that are not integrated with TIBCO ActiveSpaces® Transactions Administrator. The console tab provides a way to issue arbitrary `administrator` commands from within TIBCO ActiveSpaces® Transactions Administrator. Every management function is available from the command line, so the `Console` tab provides a way to access any features that TIBCO ActiveSpaces® Transactions Administrator does not already address. Click on the `Console` tab as shown in Figure 1.11.

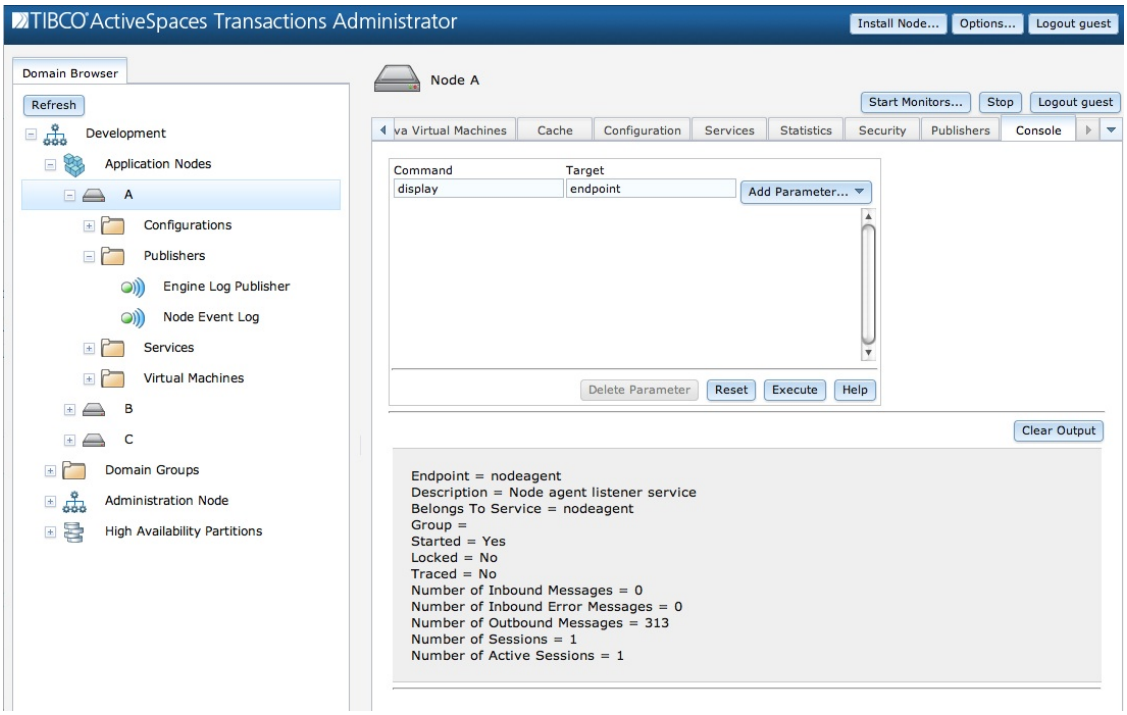

Figure 1.11. Console

The fields under the `Console` tab correspond to the elements of the `administrator` command line interface. the section called “Command line interface” on page 121 explains this interface and what these fields mean.

JMX

TIBCO ActiveSpaces® Transactions supports JMX access to all administrative commands. It also supports JMX notifications for all log events.

When a JVM is started in TIBCO ActiveSpaces® Transactions a JMX server is automatically started. The address of the JMX server can be found in the JVM details screen in the *JMX Service Address* and *JMX Service URL* fields. See Figure 1.12.

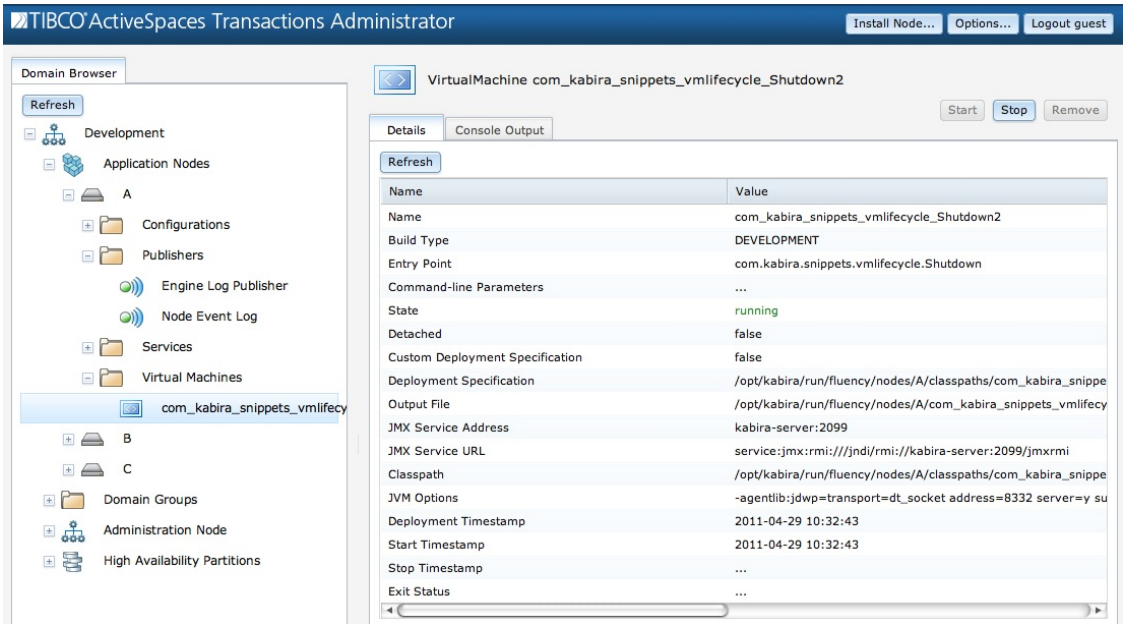

Figure 1.12. JMX address

Valid TIBCO ActiveSpaces® Transactions node credentials must be specified in the JMX console login panel to access the TIBCO ActiveSpaces® Transactions JVM. The user name and password specified must already be defined for the node.

Figure 1.13. JMX console log in

TIBCO ActiveSpaces® Transactions exposes MBeans for:

- Configuration
- Events
- Management

All of the TIBCO ActiveSpaces® Transactions Mbeans can be found in the `com.kabira.platform.management` name space.

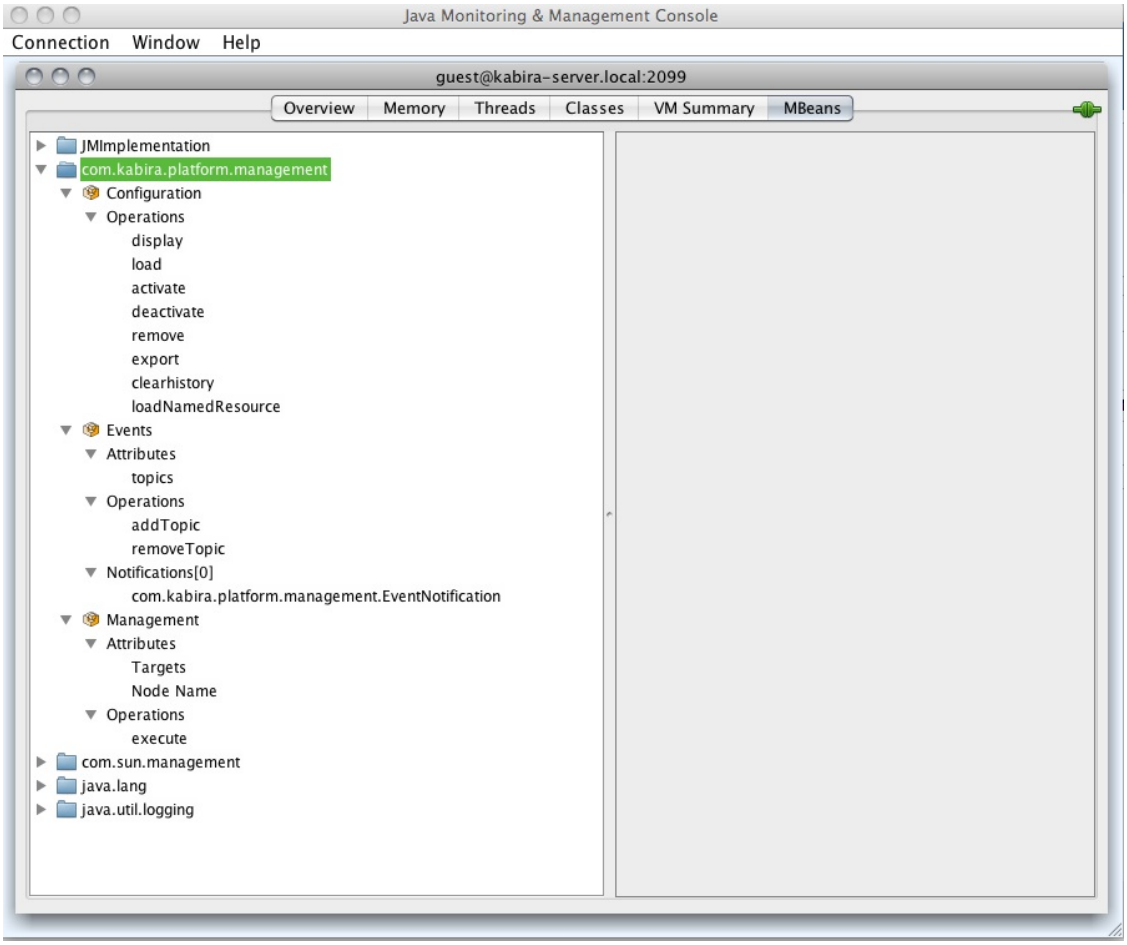

Figure 1.14. TIBCO ActiveSpaces® Transactions MBeans

Configuration MBean

The JMX configuration MBean supports execution of all TIBCO ActiveSpaces® Transactions configuration commands. Figure 1.15 shows an example of displaying node configuration loaded into a TIBCO ActiveSpaces® Transactions node.

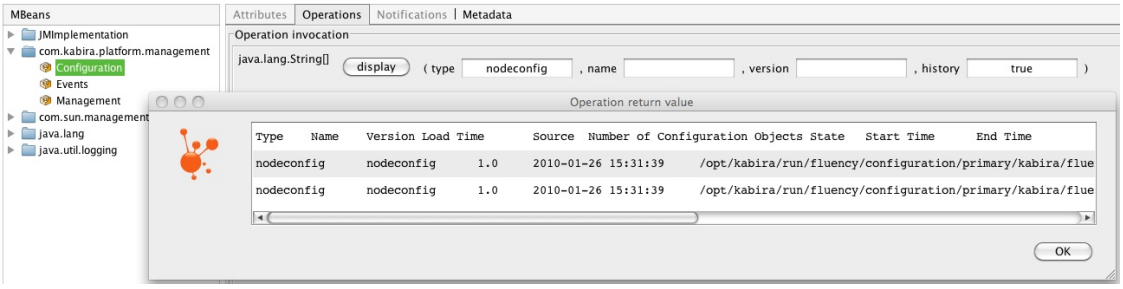

Figure 1.15. Configuration MBean

Event MBean

The JMX event MBean supports:

- dynamically adding and removing topics for which the MBean will receive events
- receiving event notifications

Figure 1.16 shows event notifications being displayed.

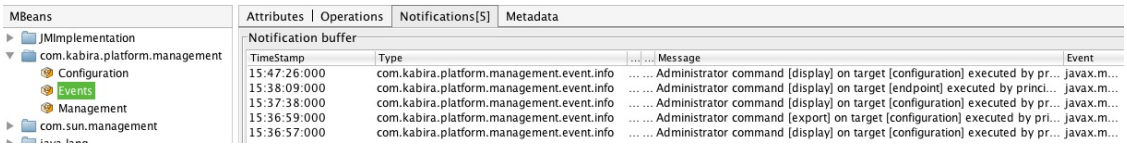

Figure 1.16. Events MBean

Management MBean

The JMX management MBean supports execution of any TIBCO ActiveSpaces® Transactions or application defined management command. Figure 1.17 shows an example of displaying all JVMs running on a TIBCO ActiveSpaces® Transactions node.

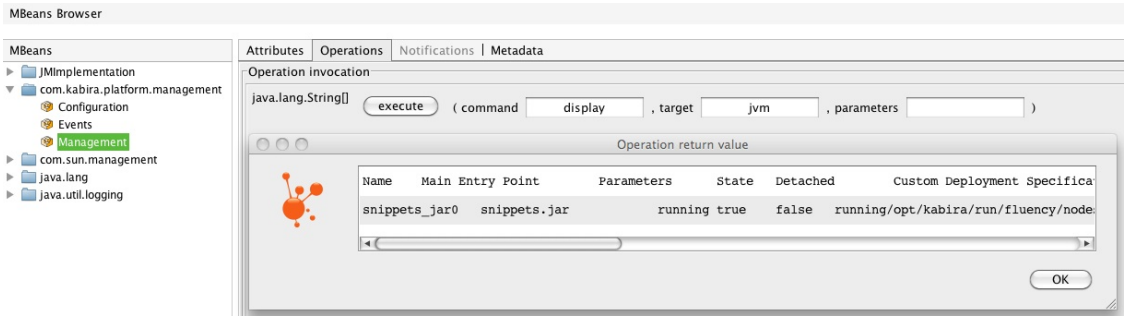

Figure 1.17. Management MBean

Problem reporting

Problem reporting is critical to allow TIBCO to continue to improve TIBCO ActiveSpaces® Transactions .

Providing the following information in all issues reported to TIBCO will help us resolve problems in a timely fashion.

- a concise description of what caused the problem, including these details:
  - Production or development environment.
  - Is the problem reproducible? If not how often does it occur?
  - Deployment tool crash or hang?

  - Node crash or hang?
  - Is the node running on a virtual machine?
- a simple program, or administrative commands, that recreates the problem.
- any screen shots that contain error information.
- a snapshot of the node.

If the application is being deployed from a client machine that is different than the server machine please include:

- Client operating system and version.
- Java version.

Snapshots of node log, configuration, and other information, are taken using the `create snapshot` command. A snapshot of a node can be taken whether a node is running or not.

To take a snapshot of a running node the `servicename` or `hostname/adminport` parameters are used. If a node is not running, the `installpath` parameter is used. Both cases are shown in Example 1.1 on page 16. Taking a snapshot of a running node has no impact on the node.

Example 1.1. Node snapshot command

```
#
#   Snapshot of a running node with a service name of A
#
administrator servicename=A create snapshot
    Created snapshot archive '/opt/RUN/./A/../snapshots/A/A.2015-04-10-09-42-33.zip'

#
#   Snapshot of a down node installed into a directory named A
#
administrator create snapshot installpath=A
    Created snapshot archive '/opt/RUN/A/../snapshots/A/A.2015-04-10-09-43-26.zip'
```

The created snapshot archive should be attached to the support issue.

2

Domain Management

This chapter describes TIBCO ActiveSpaces® Transactions domain manager. Installation, configuration, and administration of a management domain and the domain manager are discussed.

While there is no requirement for stand-alone TIBCO ActiveSpaces® Transactions nodes to be managed by a domain manager, it is recommend that all TIBCO ActiveSpaces® Transactions deployments run a domain manager. A domain manager provides:

- a single point of management control.
- web-based administration through TIBCO ActiveSpaces® Transactions Administrator.

Domain manager

This section describes how to install, manage, and configure a domain manager.

Installation

The `administrator` command line tool is used to install, start, stop, and remove a domain manager. TIBCO ActiveSpaces® Transactions must already be installed. See the TIBCO ActiveSpaces® Transactions **Installation** guide for details on installing TIBCO ActiveSpaces® Transactions .

Here are the commands to manage a Domain Manager.

```
//
//     Install a node running the domain manager application
//
administrator install node application=kabira/kdm

//
//     Start the domain manager - the adminport value was displayed during installation
//
administrator adminport=19149 start node

//
```

```
//     Display the URL for administrator
//
administrator adminport=19149 display manager

//
//     Stop the domain manager
//
administrator adminport=19149 stop node

//
//     Remove the domain manager node installation
//
administrator adminport=19149 remove node
```

By default a domain manager is installed with a domain named *tibco*. This can be changed through configuration. See the section called “Configuration” on page 23 for details.

The URL at which to access the TIBCO ActiveSpaces® Transactions Administrator is displayed using the `display manager administrator` command.

A default user named `guest` with a password of `guest` is also configured. This can also be changed using configuration.

It is strongly recommended that the default user and password be changed for production systems.

See Chapter 5 for details on security configuration and other other security considerations when using a domain manager.

Adding and removing groups and nodes

Multiple mechanisms are provided to add new groups and nodes to a domain. These mechanisms are:

- Operator command
- Domain configuration
- Node configuration (adding nodes to existing domains and groups only)

Nodes Nodes being managed by a domain manager are called *managed nodes*. These nodes are displayed in the `Managed Nodes` section on the `Manage Nodes` tab or using

```
administrator servicename=domainmanager display domain type=node
```

This information is displayed for each node managed by a domain:

- **Name** - Node name.
- **Host** - Host name on which node is running.
- **Port** - Node administration port.
- **Add Method** - How the node was added to the domain. `Configured` indicates that the node was explicitly configured into the domain. `Administrator` indicates that the node was added to the domain using an administrative command. `Discovered` indicates that the node was dynamically added to the domain using service discovery.

- **Added** - Date and time on which the node was added to the domain.
- **Groups** - Domain groups in which this node is included, if any.
- **Application** - The container type running on the node.
- **Description** - Node description, if any.
- **Node Agent Address** - Node agent address.
- **Connection Count** - The number of administrative connections currently open to this node.
- **Install Time** - Date and time on which this node was installed.
- **Last Start Time** - Date and time on which this node was last started.

To start managing a node in a domain it must be *attached* to the domain. Attaching a node to a domain using TIBCO ActiveSpaces® Transactions Administrator is shown in Figure 2.1. To start managing a node in a domain, the node is selected in the `Unmanaged Nodes` table and the `Join Domain` button is executed.

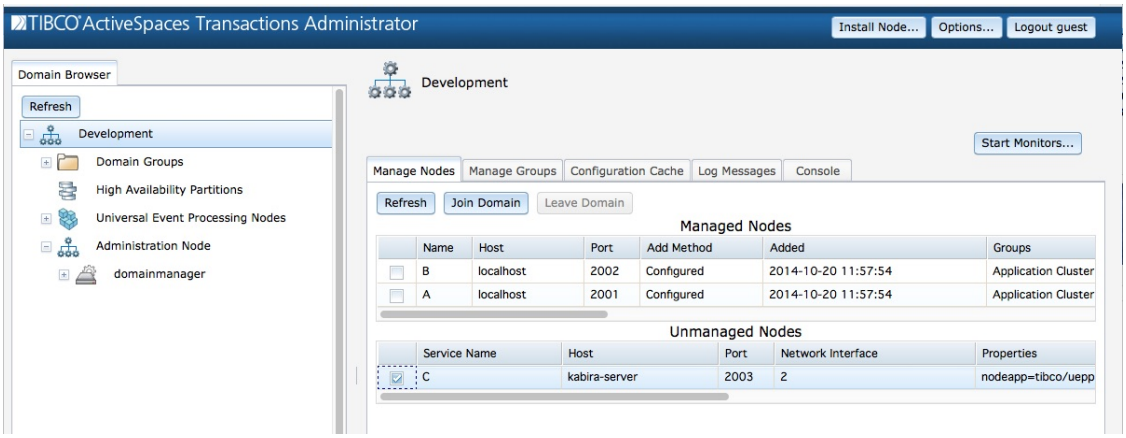

Figure 2.1. Attaching a node to a domain

The command line to attach a node to a domain is:

```
administrator servicename=domainmanager addnode domain name=A
```

To stop managing a node it must be *detached* from a domain. Detaching a node from a domain using TIBCO ActiveSpaces® Transactions Administrator is shown in Figure 2.2. To stop managing a node in a domain, the node is selected in the `Managed Nodes` table and the `Leave Domain` button is executed.

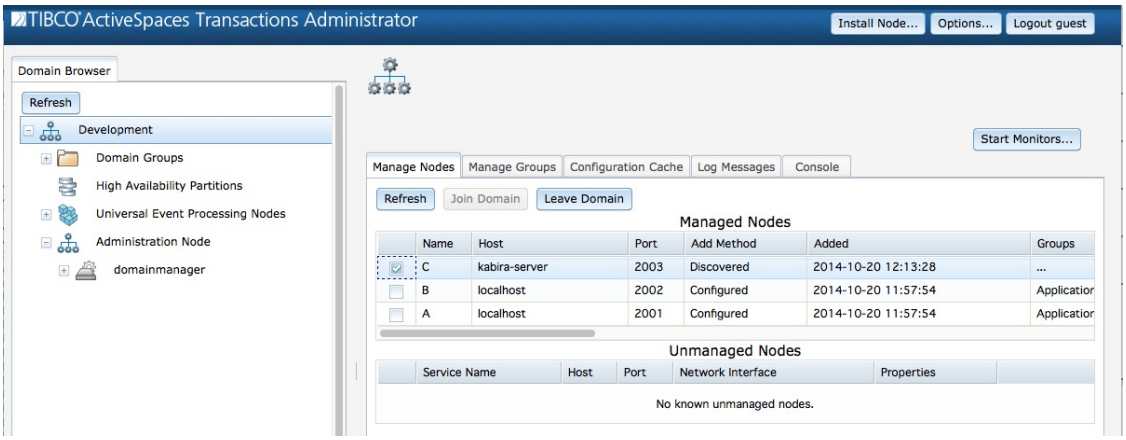

Figure 2.2. Detaching a node from a domain

The command line to detach a node from a domain is:

```
administrator servicename=domainmanager removenode domain name=A
```

The `nodeConfiguration` configuration value can also be used to configure new nodes in a domain. See the section called “Configuration” on page 23 for details on configuring nodes in a domain.

Finally, nodes can be configured to automatically add themselves to a domain. See the section called “Configuration” on page 48 for details.

Groups Domain groups are displayed in the `Domain Groups` section on the `Manage Groups` tab or using

```
administrator servicename=domainmanager display domain type=group
```

This information is displayed for each group:

- **Name** - Group name.
- **Properties** - Service properties used to determine group membership, if any.
- **Node** - Name of node in group.

Domain groups are created from the `Manage Groups` tab in TIBCO ActiveSpaces® Transactions Administrator. The create group dialog is shown in Figure 2.3.

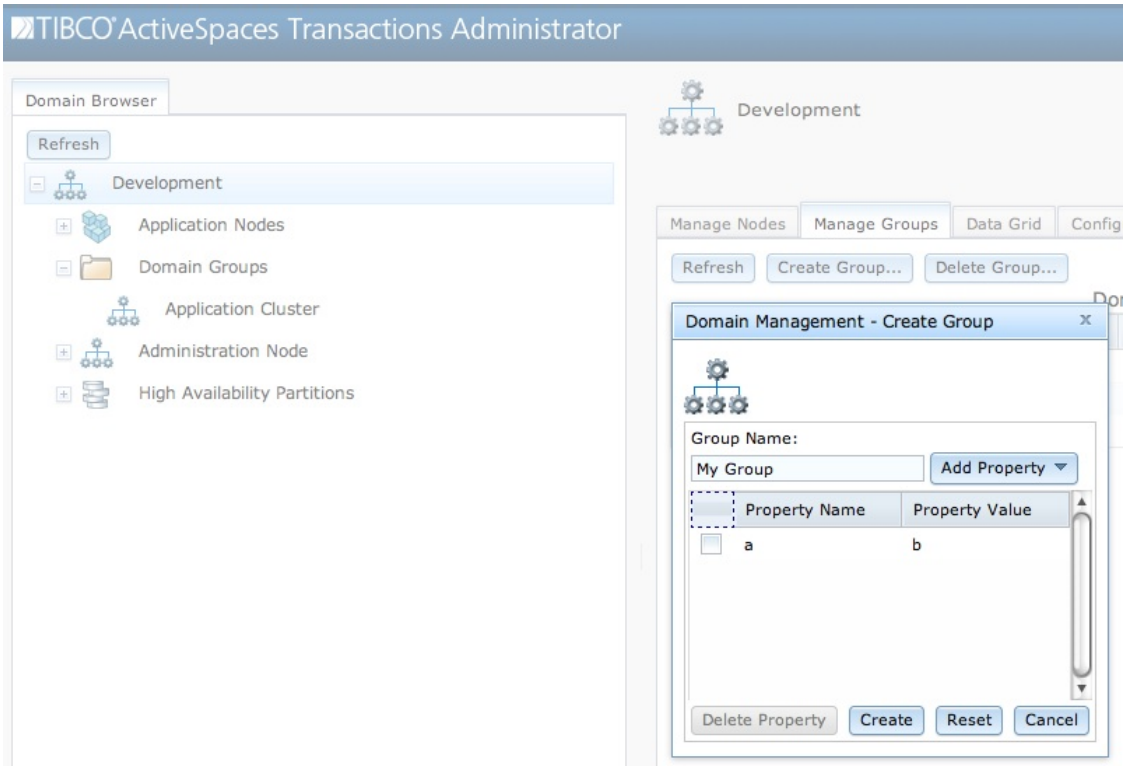

Figure 2.3. Creating a group

The property name and value fields are used to define service discovery properties that are used to dynamically match service discovery properties of TIBCO ActiveSpaces® Transactions nodes. If an TIBCO ActiveSpaces® Transactions node matches the service discovery properties defined for a group the node is automatically added to the group and domain as a managed node.

The command line to create a group in a domain is:

```
administrator servicename=domainmanager addgroup domain groupname=MyGroup
```

Groups are deleted from the group detail screen. The `Delete` button deletes the selected group. Figure 2.4 shows an example of deleting a group.

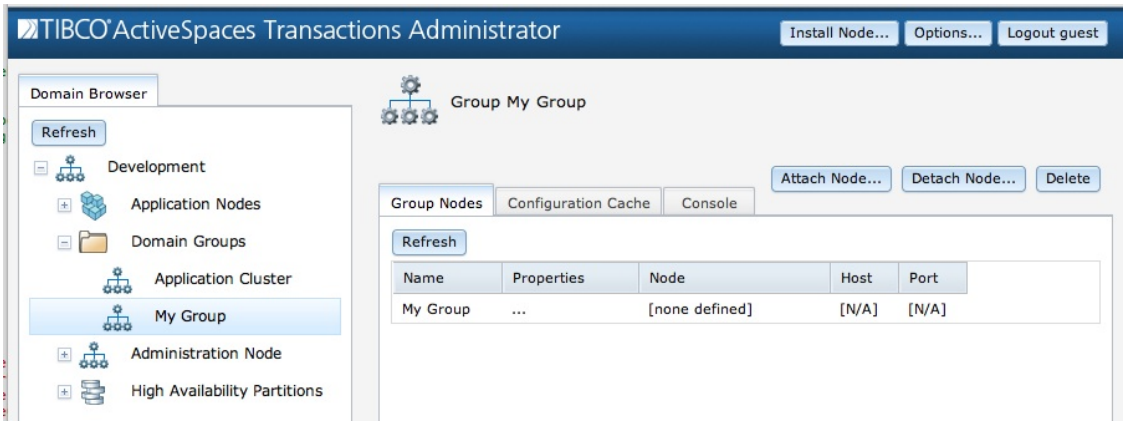

Figure 2.4. Deleting a group

Groups can also be deleted using this command line:

```
administrator servicename=domainmanager removegroup domain groupname=MyGroup
```

Nodes added to groups must already be managed by the domain. Once a node is managed by a domain it can be added to a specific group. Attaching a node to a group from TIBCO ActiveSpaces® Transactions Administrator is shown in Figure 2.5.

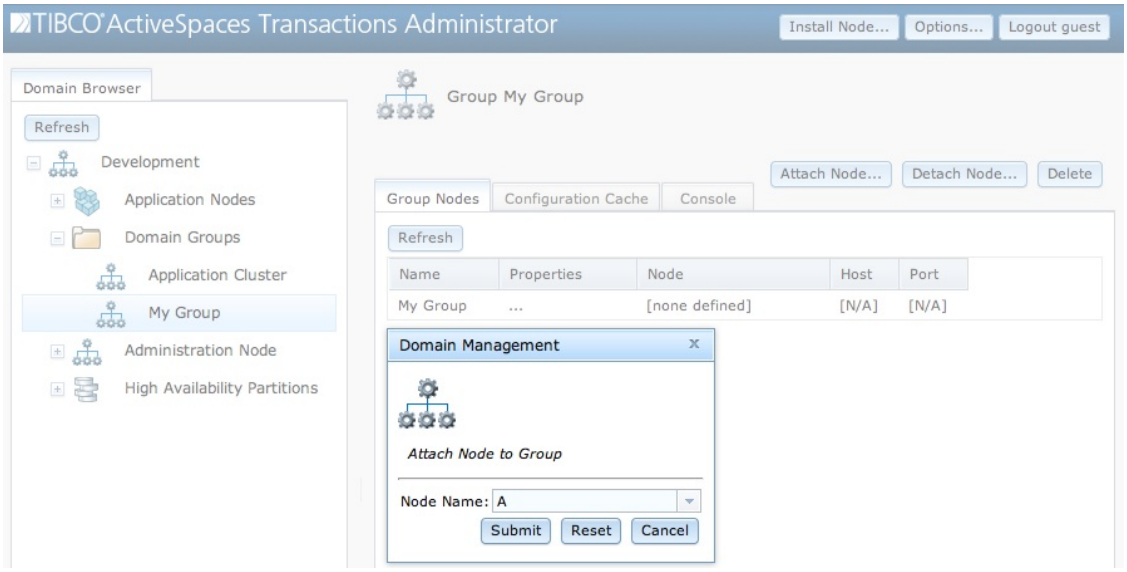

Figure 2.5. Attaching a node to a group

This command line can also be used to add a node to a group:

```
administrator servicename=domainmanager addgroupnode domain groupname="Application Cluster"
 name=A
```

Detaching a node from a group in TIBCO ActiveSpaces® Transactions Administrator is shown in Figure 2.6.

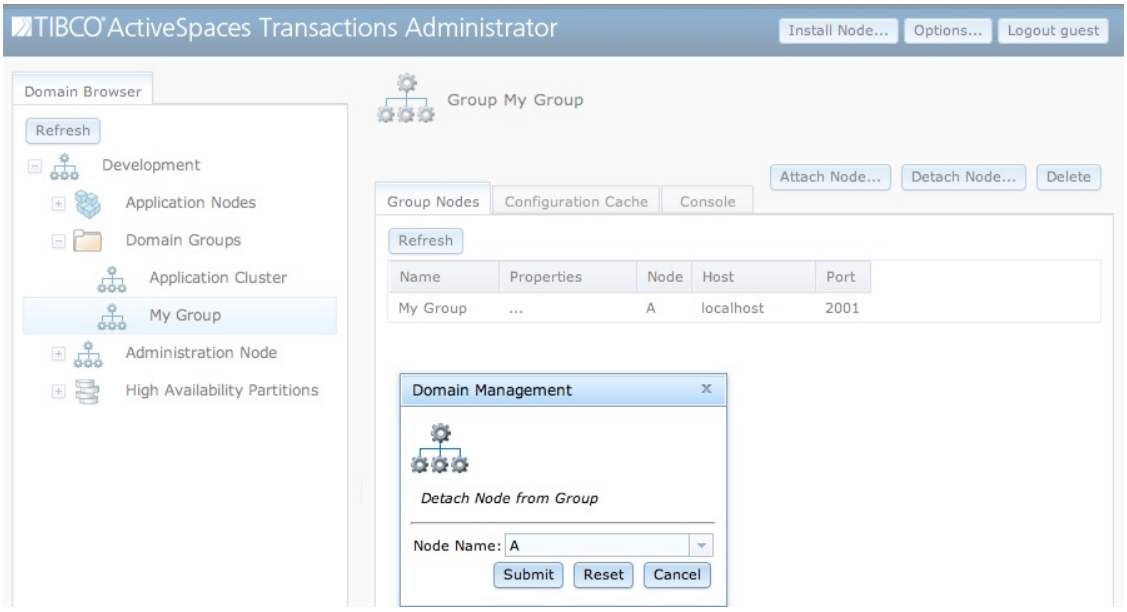

Figure 2.6. Detaching a node from a group

This is accomplished using the command line with:

```
administrator servicename=domainmanager removegroupnode domain groupname="Application
Cluster" name=A
```

The `groupConfiguration` configuration value can also be used to configure new groups in a domain. This configuration value optionally supports dynamically adding nodes to a group using service discovery properties. See the section called “Configuration” on page 23 for details on configuring groups in a domain.

Configuration

Domain configuration can be separated into these distinct areas:

- global parameters
- group definitions
- node definitions

Domain configuration has a configuration type of `kdm`.

Table 2.1 on page 23 defines the global domain configuration parameters.

Table 2.1. Global domain configuration

Name	Type	Description
`connectionInactivityTimeoutSeconds`	Integer	Number of seconds of inactivity before a cached connection to a managed node is closed. Default value is `3600` seconds.
`connectionReadTimeoutSeconds`	Integer	Number of seconds to wait for successful connection attempt to a managed node. If this value is exceeded, the connection

		to the managed node is dropped. A value of 0 blocks forever waiting for a response. Default value is 10 seconds.
`defaultNodeConfiguration`	String list	An optional list of configuration files which are loaded and activated on all nodes which join the domain. The configuration files must be specified as an absolute path.
`discoveryHost`	String	Service discovery host. Use this host name when looking for new services. The host name must be valid on the local machine. Default value is the default interface on the local machine.
`discoveryIntervalSeconds`	Integer	Service discovery interval in seconds. Controls the delay between attempts to discover new services. This value must be > 0. Default value is 30 seconds.
`domainName`	String	Domain name.
`enableAutoJoin`	Boolean	Control whether nodes are automatically added to the domain. A value of `true` causes the domain to automatically add nodes that request to be added to the domain as they are detected. See Table 3.8 on page 49 for details on configuring a node to automatically request to be added to a domain. A value of `false` requires all nodes to be explicitly configured or added by the administrator to the domain. See Table 2.3 on page 25 for details on explicitly configuring nodes in a domain. Default value is `true`.
`groupConfiguration`	Group list	An optional list of groups that are part of this domain. These groups are automatically added to the domain when the configuration is activated.
`nodeConfiguration`	Node list	An optional list of nodes which are part of this domain. Each entry in this list specifies an TIBCO ActiveSpaces® Transactions node which is automatically added to the domain when the configuration is activated.
`retryIntervalSeconds`	Integer	Number of seconds between retrying queued configuration commands to managed nodes.

Table 2.2 on page 24 defines the group configuration parameters. This configuration block is optional. No domain groups need to be defined in a domain.

Table 2.2. Group configuration

Name	Type	Description
`name`	String	Group name.
`properties`	String	An optional set of service properties of the form *`property1,property2,...,propertyN`*. If this configuration is defined, the domain manager initiates a search for all TIBCO ActiveSpaces® Transactions nodes with matching properties in their node service record. Any matching nodes are automatically added to this group.
`defaultNodeConfiguration`	String list	An optional list of configuration files which are loaded and activated on all nodes which join this group. The configuration files must be specified as an absolute path.

Table 2.3 on page 25 defines the node configuration parameters. This configuration is only required to statically define a node to be managed by this domain. This configuration is not required if all managed nodes are dynamically added to the domain.

Table 2.3. Node configuration

Name	Type	Description
`name`	String	Node name.
`host`	String	Host name.
`nodeAgentAddress`	String	The network address of the node agent listener for this node. The format of this configuration value is *<protocol>:<host name>:<port number>*. Protocol is one of `TCP` or `SSL`.
`port`	Integer	Administration port number.
`application`	String	Application name.
`groups`	String list	A list of domain groups for this node.
`description`	String	Node description.

An example domain configuration is below.

Example 2.1. Example domain configuration

```
//
//    Version 2.0 of domain manager configuration
//
configuration "kdm" version "2.0" type "kdm"
{
     configure kdm
     {
          DomainConfig
          {
               //
               //    Domain name of Development
               //
               domainName = "Development";

               //
               //    Retry queued configuration commands every 5 seconds
               //
               retryIntervalSeconds = 5;

               //
               //     Disable auto-join
               //
               enableAutoJoin = false;

               //
               //     Set connection inactivity interval to 3000 seconds
               //
               connectionInactivityTimeoutSeconds = 3000;

               //
               //     Set connection read timeout interval to 10 seconds
               //
               connectionReadTimeoutSeconds = 10;

               //
               //    Add A node to this domain.  The node is added to
               //    the Application Cluster
```

```
              //
              nodeConfiguration =
              {
                   {
                        name = "A";
                        host = "localhost";
                        port = 2001;
                        application = "kabira/ast";
                        description = "TIBCO ActiveSpaces® Transactions
application";
                        groups =
                        {
                             "Application Cluster"
                        };
                        nodeAgentAddress = "TCP:localhost:2005";
                   }
              };

              //
              //    No default configuration loaded for domain managed nodes
              //
              defaultNodeConfiguration = { };

              //
              //    Define a group in this domain
              //
              groupConfiguration =
              {
                   {
                        //
                        //    Group name of Application Cluster
                        //
                        name = "Application Cluster";

                        //
                        //    No service properties used to add nodes to this group
                        //
                        properties = "";

                        //
                        //    No default configuration loaded for nodes in this group
                        //
                        defaultNodeConfiguration = { };
                   }
              };
         };
    };
};
```

Centralized Logging

Domain managers support capturing log messages from all managed nodes. This section describes how to configure and manage capturing log messages from all managed nodes.

Configuration

Domain configuration has a configuration type of `eventcache`.

Table 2.4 on page 27 defines the log message cache configuration parameters.

Table 2.4. Log message cache configuration

Name	Type	Description
`topicDoNotCacheList`	String list	Optional list of message topics that should not be added to the cache. If a message is received with a topic in this list it is discarded without adding it to the cache.
`eventDoNotRepublishList`	String list	Optional list of message identifiers that should not be republished to TIBCO ActiveSpaces® Transactions Administrator and JMX event monitors.
`numberOfEvents`	Integer	The total number of messages that should be maintained in the message cache. When this number of messages is reached, the oldest message in the cache is discarded to add the new message.

An example log message cache configuration is below.

Example 2.2. Example log message cache configuration

```
//
//    Version 1.0 of an log message cache configuration
//
configuration "MessageCache" version "1.0" type "eventcache"
{
     configure eventcache
     {
          EventCacheConfiguration
          {
               //
               //     List of topics that should not be added to
               //     the cache
               //
               topicDoNotCacheList =
               {
                    "application.statistics"
               };

               //
               //    List of identifiers that should not be republished to
               //    the Administrator monitor
               //
               eventDoNotRepublishList =
               {
                    "channel::EndpointEvents::TraceEndpointHandleMessage",
                    "channel::EndpointEvents::TraceEndpointReceiveMessage",
                    "channel::EndpointEvents::TraceOutboundMessage",
                    "nodeagent::EventIdentifiers::KISTraceInfo",
                    "nodeagent::EventIdentifiers::KISTraceDebug"
                };

                //
                //     Number of messages to maintain in the cache
                //
                numberOfEvents = 1000;
          };
     };
};
```

Configuration cache

The configuration cache is automatically started and initialized when the Domain Manager is started. The only configuration required for the configuration cache is the `retryIntervalSeconds` configuration value in the domain configuration. See Table 2.1 on page 23 for details.

The configuration cache supports these commands to manage configuration data for one or more managed nodes:

- **Load** - load configuration into the configuration cache and one or more managed nodes.
- **Activate** - activate configuration for one or more managed nodes.
- **Deactivate** - deactivate configuration for one or more managed nodes.
- **Remove** - remove the configuration from the configuration cache and from one or more managed nodes.
- **Display** - display information about the configuration cache.
- **View Source** - view the cached configuration source.
- **Restore** - restore configuration data for one or more managed nodes. Configuration data is only restored to a node if the node was re-installed after the configuration data was loaded into the configuration cache.

All of the configuration cache commands can be executed on the entire domain, or a specific domain group.

Figure 2.7 is an example configuration cache information display.

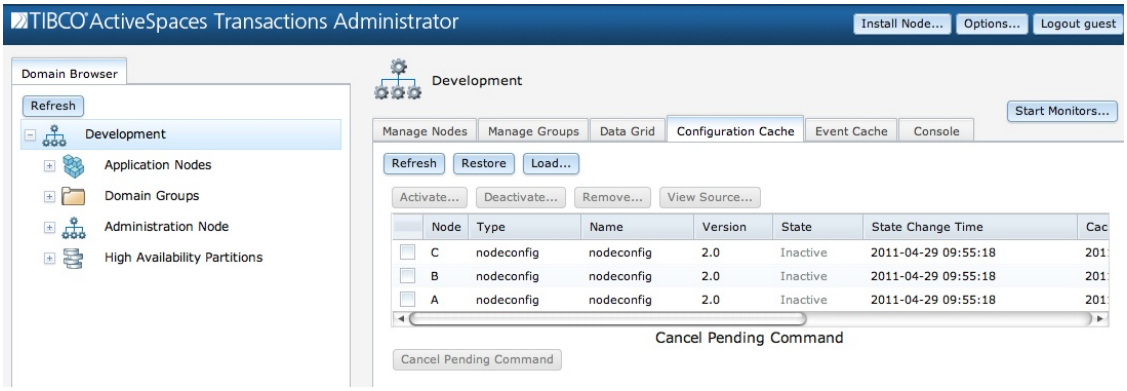

Figure 2.7. Configuration cache information display

The cached configuration display shows the following for configuration data in the cache:

- **Node** - managed node name.
- **Type** - configuration type.
- **Name** - configuration name.
- **Version** - configuration version.

- **State** - current state of configuration on managed node.
- **State Change Time** - Time-stamp for when the last state transition occurred.
- **Cache Load Time** - Time-stamp for when configuration was loaded into the configuration cache.
- **Source** - Source path.

Administrator

When a Domain Manager node is started, it automatically starts a Web Server to support the TIBCO ActiveSpaces® Transactions Administrator access to the domain.

The URL for the TIBCO ActiveSpaces® Transactions Administrator can be found using the `display manager` command after the domain manager has been installed and started. For example:

```
//
//     Display the URL for administrator
//
Manager Service Name = ActiveSpaces Transactions Adminstrator
Manager URL = http://127.0.0.1:8080
Manager State = Active
Service Discovery State = Enabled
Service Discovery Name = domainmanager
```

Configuration

The TIBCO ActiveSpaces® Transactions Administrator configuration has a configuration type of `km`.

Activating the TIBCO ActiveSpaces® Transactions Administrator configuration starts the Web Server. Deactivating the configuration stops the Web Server.

Table 2.5 on page 29 defines the TIBCO ActiveSpaces® Transactions Administrator configuration parameters.

Table 2.5. TIBCO ActiveSpaces® Transactions Administrator Web Server configuration

Name	Type	Description
`serviceName`	String	The service name of the TIBCO ActiveSpaces® Transactions Administrator Web Server that is published to service discovery. This name is also used in error messages raised by the server.
`description`	String	A description of the TIBCO ActiveSpaces® Transactions Administrator Web Server.
`listenerAddresses`	String list	A list of listener addresses on which the Web Server listens for requests. The format of these addresses is *<protocol>:<host name>:<port number>*. Protocol is one of `TCP` or `SSL`.

An example TIBCO ActiveSpaces® Transactions Administrator Web Server configuration is below.

Example 2.3. Example TIBCO ActiveSpaces® Transactions Administrator server configuration

```
//
//     Version 3.0 of a TIBCO ActiveSpaces® Transactions
 Administrator Web Server configuration
//
configuration "asta" version "3.0" type "km"
{
     configure kabman
     {
          Configuration
          {
               serviceName = "TIBCO ActiveSpaces® Transactions
 Adminstrator";
               description = "TIBCO ActiveSpaces® Transactions
 Administrator web server";
               listenerAddresses =
               {
                    "TCP:192.168.71.129:80"
               };
          };
     };
};
```

The IP address of the machine on which the Domain Manager is running must be resolvable by host name, not an IP address. If it is not, the TIBCO ActiveSpaces® Transactions Administrator web display will still work, but response time will be very slow. Example 2.4 on page 30 shows an example host file entry that will perform IP address resolution by name.

Example 2.4. /etc/hosts File

```
127.0.0.1 localhost
<machine IP address> <Domain Manager node name>
```

3

Node Management

This chapter describes how to configure TIBCO ActiveSpaces® Transactions nodes and deploy applications onto the nodes.

This chapter assumes that a Domain Manager has been installed and configured (see the section called “Installation” on page 17).

Node administration

This section discusses node administration. Node administration can be done using TIBCO ActiveSpaces® Transactions Administrator, JMX consoles, or `administrator`.

Installation

The following information is needed to perform a node installation on a host.

- Host name
- TIBCO ActiveSpaces® Transactions installation path on host (e.g. `/opt/tibco-home/<version>`)

Installation can be done remotely using TIBCO ActiveSpaces® Transactions Administrator as long as the TIBCO ActiveSpaces® Transactions installation is accessible from the host on which the node is being installed. The node installation screen is shown in Figure 3.1.

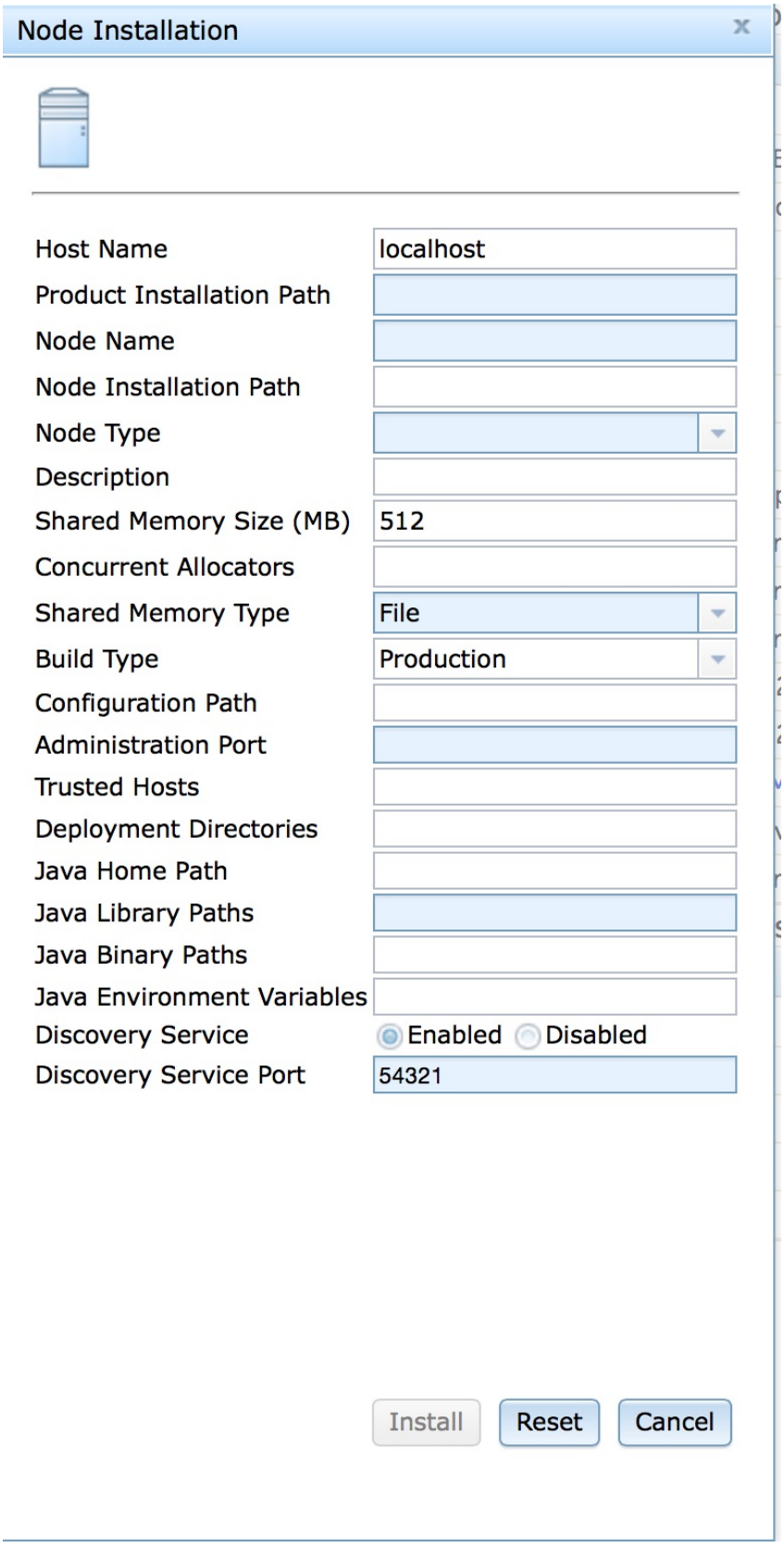
Node Installation
Host Name
localhost
Product Installation Path
Node Name
Node Installation Path
Node Type
Description
Shared Memory Size (MB)
512
Concurrent Allocators
Shared Memory Type
File
Build Type
Production
Configuration Path
Administration Port
Trusted Hosts
Deployment Directories
Java Home Path
Java Library Paths
Java Binary Paths
Java Environment Variables
Discovery Service
Enabled
Disabled
Discovery Service Port
54321
Install
Reset
Cancel

Figure 3.1. Node installation screen

This fields in this dialog are described in Table 3.1 on page 33.

Table 3.1. Node installation options

Option	Description
Host Name	Host name on which to perform node installation. This is a required field.
Product Installation Path	Path to TIBCO ActiveSpaces® Transactions installation on host. This must be an absolute path name. This is a required field.
Node Name	Node name. Node names cannot contain any spaces. This is a required field.
Node Installation Path	Installation path on host where node should be installed. This must be an absolute path name. This is a required field.
Node Type	One of `Application` or `Domain Manager`. `Application` installs a node for hosting applications. `Domain Manager` installs a Domain Manager.
Description	An optional node description.
Shared Memory Size (MB)	Shared memory size in megabytes. Default value is 512 MB.
Concurrent Allocators	Number of concurrent shared memory allocators. Default value is based on the number of cores on the system and the configured shared memory size, up to a maximum value of 64.
Shared Memory Type	One of `File` or `System V`. `File` causes the node to use file backed shared memory. `System V` causes the node to use System V shared memory.
Build Type	Controls the mode in which the TIBCO ActiveSpaces® Transactions runtime executes. One of `Production` or `Development`. `Production` should be used for production deployments. `Development` mode generates additional tracing and diagnostics and should only be used for development nodes.
Configuration Path	Path for default configuration files. This can be an absolute or relative path. If a relative path is specified, it is relative to the node installation directory. The default value is *<installation path>*`/../configuration/`*<node name>*.
Administration Port	Administration port. Default value is random.
Trusted Hosts	An optional comma separated list of host names to define as trusted hosts for this node.
Deployment Directories	An optional : separated list of directories where application JAR files can be installed. The directory names can be absolute or relative. All non-absolute paths are relative to the node Installation Path directory. If the directories do not exist, they are created during node installation. If a directory cannot be created, a warning message is generated, but node installation continues. The default value is *<installation path>*`/../deploy`. See the section called “Deployment directories” on page 46 for further details about deployment directories.

Java Home Path	Optional location of the JRE or JDK to be used by the node. This must be an absolute path to the Java Home to use. The default value is the shipped default JRE.
Java Library Paths	An optional list of JRE/JDK directories to add to the node's library search path. Relative paths are relative to the directory specified in the `Java Home Path` parameter. Use the `:` character to separate directories in the list. If this parameter is specified then the `Java Home Path` parameter must also be specified. The default value is configured to work the shipped default JRE.
Java Binary Paths	An optional list of JRE/JDK directories to add to the node's binary search path. Relative paths are relative to the directory specified in the `Java Home Path` parameter. Use the `:` character to separate directories in the list. If this parameter is specified then the `Java Home Path` parameter must also be specified. The default value is configured to work the shipped default JRE.
Java Environment Variables	An optional comma-separated list of *<name>=<value>* pairs specifying the environment variables to add to the node's environment for the JRE/JDK specified by the `Java Home Path` parameter. If this parameter is specified then the `Java Home Path` parameter must also be specified. Default value is an empty string.
Discovery Service	Enable or disable the discovery service. Default value is enabled.
Discovery Service Port	Discovery service port. Default value is 54321.

Figure 3.2 shows the installation of a node named `MyApplication`.

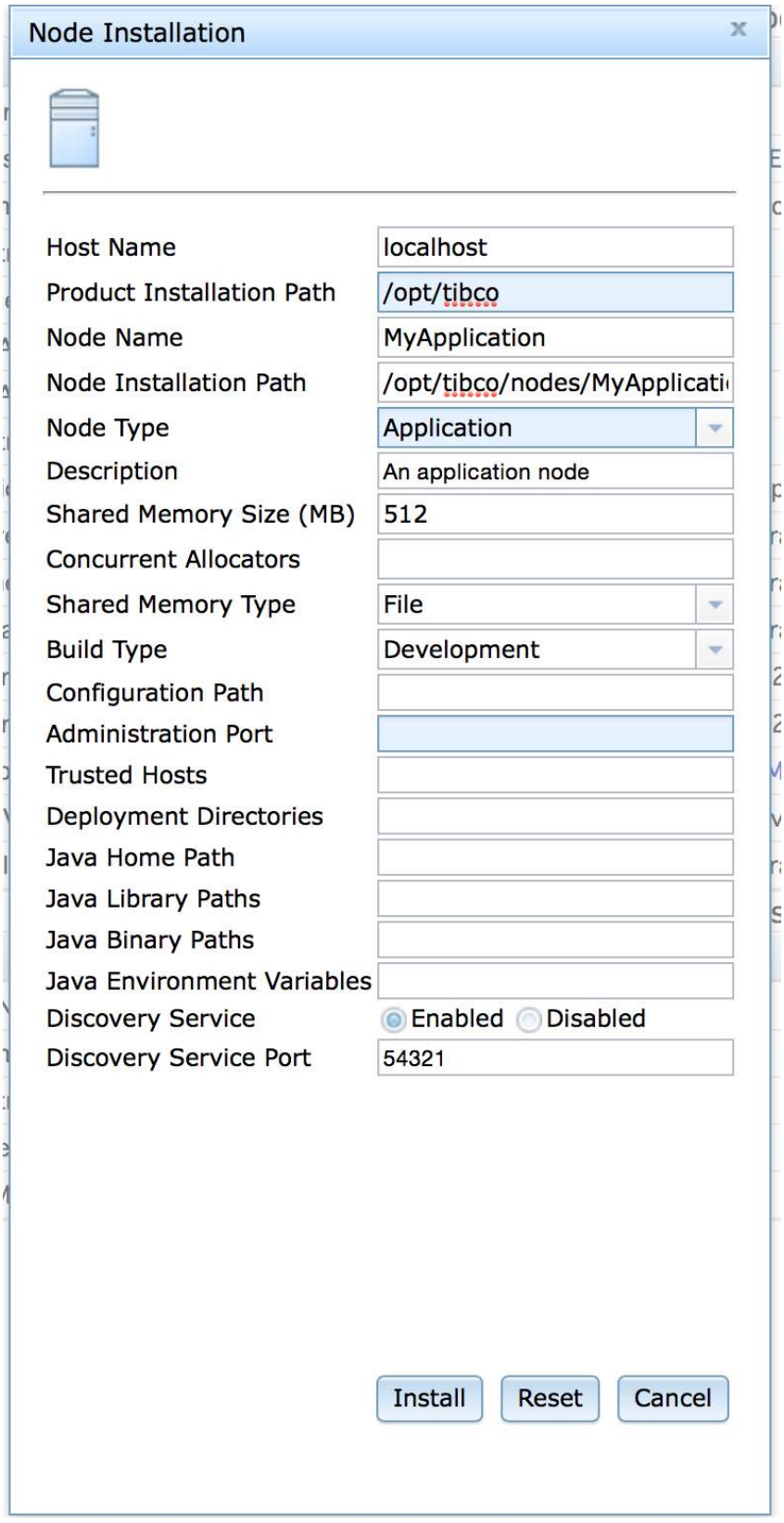

Figure 3.2. Installation of node MyApplication

The node can also be installed using this command on the machine `orion` with the `SW_HOME` environment variable set to `/opt/tibco`.

```
administrator install node nodename=MyApplication installpath=/opt/tibco/deploy/nodes
description="an application node"
```

After the installation completes, the node shows up as unmanaged in the domain as show in Figure 3.3.

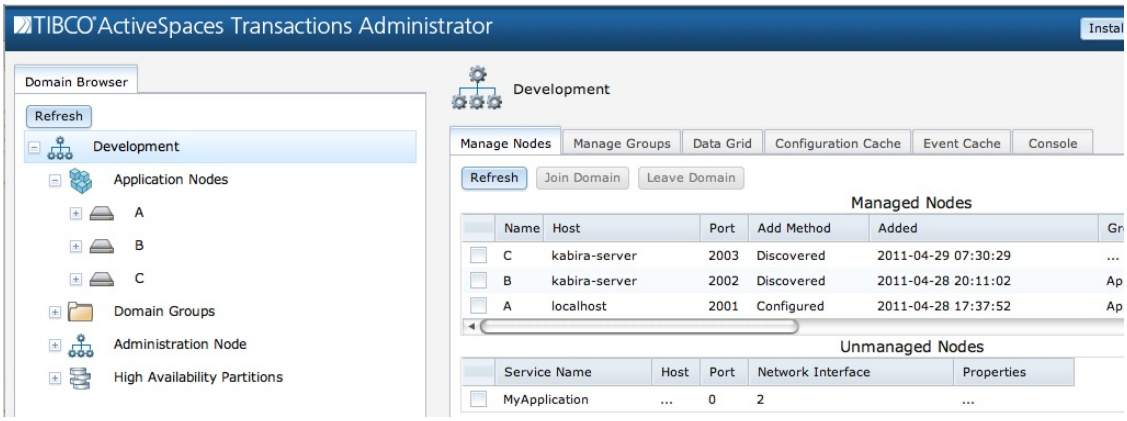

Figure 3.3. After installing MyApplication node

The node must be added to the domain to perform any further management functions. This is done by selecting the node in the `Unmanaged Nodes` table and clicking on the `Join Domain` button.

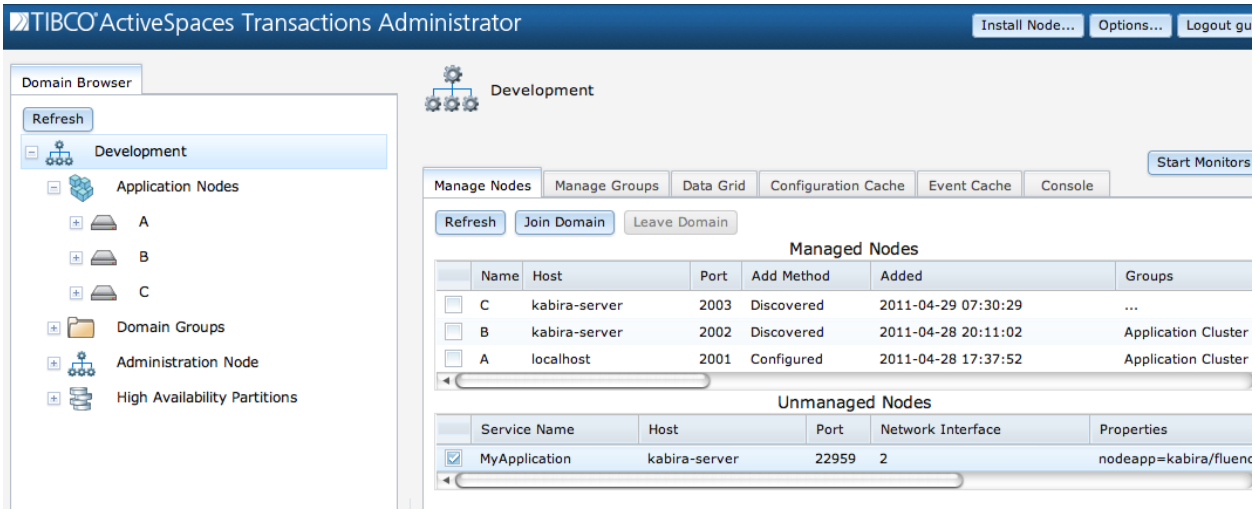

Figure 3.4. Adding installed node to domain

To complete the installation the node must be started. This is done by selecting the node in the Domain Browser window and clicking on the `Start...` button as shown in Figure 3.5, or using this command:

```
administrator hostname=kabira-server adminport=22959 start node
```

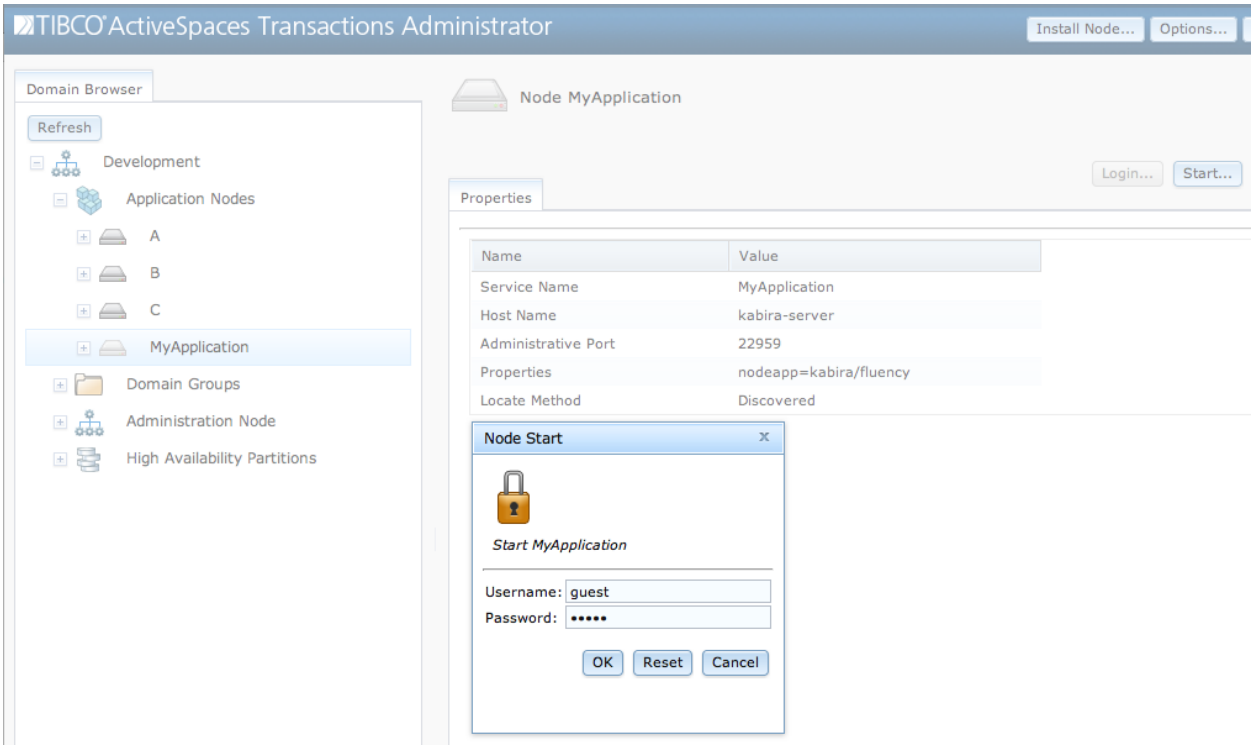

Figure 3.5. Starting node

Overriding and augmenting default node configuration As part of node installation, default configuration information is installed. This configuration information can be overridden before the node is started. Application specific configuration can also be installed after a node has been installed.

The default path where configuration for a node is located is *<node directory>*`/../configuration/`*<node name>*. This path can be changed at node installation time with the `Configuration Path` option. The configuration directory has these top level directories:

- **node** - contains the default node configuration files. The contents of this directory should not be changed.
- **application** - application specific configuration files and overrides of the default node configuration.

The `application` directory has the following behavior:

- created during node installation if it does not exist.
- all contained configuration files are loaded and activated in `ASCII` sort order during node start-up, after the node configuration was loaded and activated.
- individual configuration files are copied into an `activated` or `rejected` directory in the `application` directory depending on whether the configuration was successfully loaded and activated or not.
- a failure to load and/or activate a configuration file will cause node start-up to fail.

Configuration files copied to the `activated` directory are named *<filename>.<current date>*.

Configured files copied to the rejected directory are named *<filename>.<current date>* and a log file containing the failure reason is created. The log file is named *<filename>.<current date>*`.log`.

Removing a node does not impact the `application` configuration directory in any way.

Inherited environment A node inherits all environment variables set in the environment in which the node was installed. This provides a mechanism to make environment variables available to deployed applications. The inherited environment is printed in the system coordinator's log file. Any changes to environmental variables after the node is installed do not affect the node's inherited environment.

There is special handling of the executable search path, e.g. `PATH`, and the shared library search path, e.g. `DYLD_LIBRARY_PATH`. Both the executable and library search paths inherited from the environment are appended to the paths defined by the node. This ensures that a setting in the inherited environment cannot cause incompatibilities with the node's required environment.

Default Node Security When a node is installed, the following security policy is automatically configured:

- Operating system user
- Installation user
- Trusted host configuration

A principal is defined when a node is installed using the user name determined from the operating system context in which the node installation was done. This is the name of the user logged into the operating system when performing a node installation. This principal has no password defined and they are granted `switchadmin` role privileges, which give that principal full administrative access to the node.

A principal definition is automatically defined for the user name and password specified when installing a node. This principal definition uses the user name as the principal name and the password as the text credential that was specified to the node installation command. This principal definition is also granted the `switchadmin` role.

The operating system user name and the user name used to install a node may be the same or different. Both users are granted the same privileges and access to the node. The only difference is that the operating system user can perform administrative commands on the local node without specifying a user name or password because of the default trusted host configuration described next.

The default trusted host configuration defines `localhost` and the actual host name where the node is being installed, as trusted hosts. This allows the operating system user to administer the node without specifying a username or password when executing administrative commands from the same machine on which the node was installed. This is accomplished by using the operating system identify when executing administrative commands if no user name and password is specified. Since the operating system user was defined as a principal with administrative privileges on the node during installation, they are allowed to execute any administrative command.

A default user named `guest` with a password of `guest` is configured for both TIBCO ActiveSpaces® Transactions and Domain Manager nodes. This can be changed using configuration.

It is strongly recommended that the default user and password be changed for production systems.

See Chapter 5 for details on the security architecture and configuration.

Starting

When a node is started the following occurs:

1. Start runtime services.
2. Start node management services.
3. Load default configuration.
4. Load application configuration and default configuration overrides (if any).
5. Initiate distribution initialization (if configured).
6. Start any JVMs that are deployed on the node.

The following steps are required to start a node from TIBCO ActiveSpaces® Transactions **Administrator**

1. Log into the node
2. Start the node

This command can also be used to start a node:

```
administrator servicename=A start node
```

Stopping

When a node is stopped the following occurs:

1. Stop any JVMs that were deployed to the node.
2. Terminate any distributed communication with other nodes.
3. Terminate node management services.
4. Terminate runtime services.

The following steps are required to stop a node from TIBCO ActiveSpaces® Transactions **Administrator**

1. Log into the node
2. Stop the node

This command can also be used to stop a node:

```
administrator servicename=A stop node
```

A stopped node can be restarted to resume application execution.

Removing

An node must be stopped before it can be removed. When an node is removed the following occurs:

1. Terminate system coordinator
2. Remove the node directory and all contained files

The following steps are required to remove a node from TIBCO ActiveSpaces® Transactions **Administrator**

1. Log into the node
2. Remove the node

This command can also be used to remove a node:

```
administrator servicename=A remove node
```

To reclaim System V shared memory a node must be removed using the procedure described in this section. System V shared memory is not released by removing a node directory manually.

An node must be reinstalled after it has been removed.

Upgrading

Applications running on a node can be upgraded without impacting other nodes in the cluster. When a node is upgraded with a new application version, the following steps are taken:

1. Audit that all classes being upgraded are in a partition with at least one replica node. This ensures that the objects can be migrated to another node during the upgrade process to avoid any data loss and to provide continuous service availability. This audit can be disabled by setting the Remove Non-Partitioned Objects option to True.

   

   Setting the Remove Non-Partitioned Objects option to True can cause data loss, because non-replicated objects will be removed during the upgrade process. Use this option with extreme caution.

2. Leave the cluster. This causes all active partitions to migrate to their replica nodes.
3. Stop the node.
4. Update deployment directories for the node if new ones were provided in the upgrade or restore node dialog.
5. Remove all objects that are being upgraded. Replicated objects are still available on the nodes where the partitions migrated.
6. Restart the node.
7. Add the node as a replica to all partitions that were migrated off the node.

8. Join the cluster.

When the upgrade process completes, the new application version has been installed on the node, and the node is active in the cluster as a replica. The final step required to bring the node back online is to migrate partitions that should be active on this node back to the node. See the section called "Migrating partitions" on page 98 for details.

Upgrading a node is initiated by clicking on the `Upgrade or Restore...` button on the node screen in TIBCO ActiveSpaces® Transactions Administrator as shown in Figure 3.6.

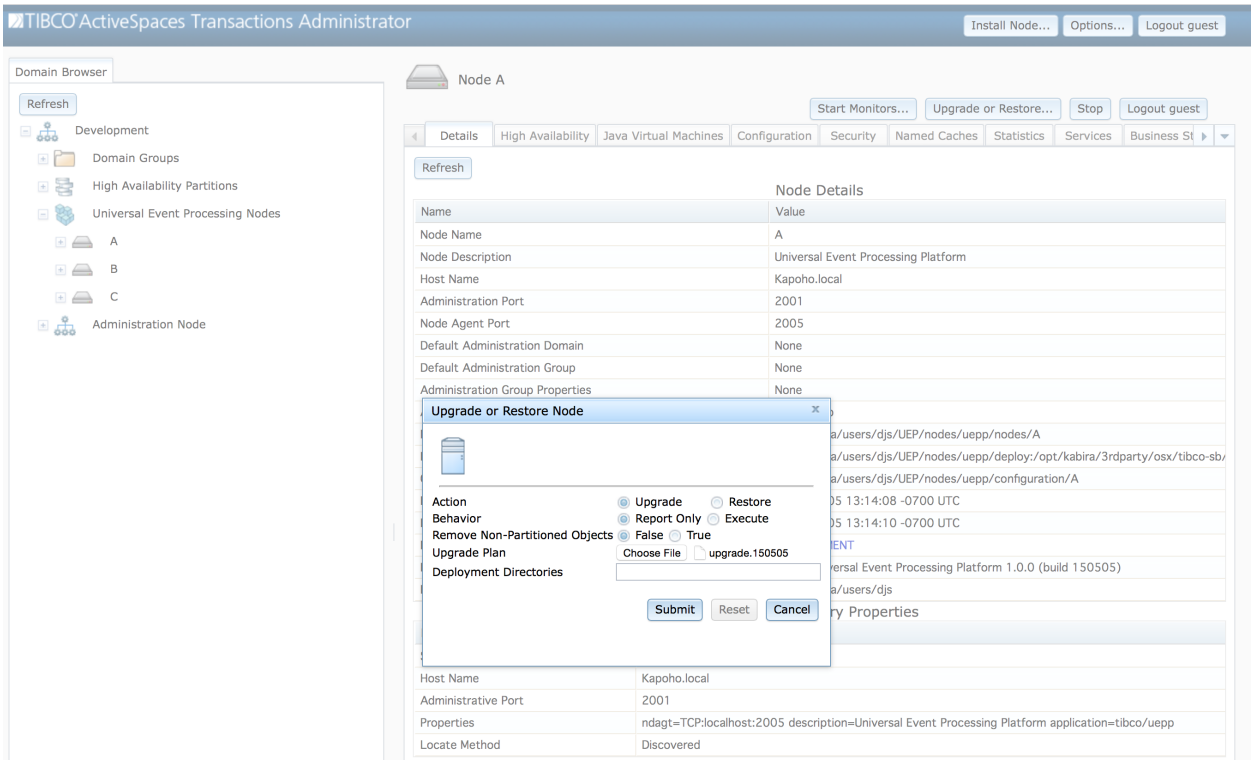

Figure 3.6. Upgrade or restore node dialog

The Table 3.2 on page 41 table defines the values in the upgrade or restore dialog.

Table 3.2. Upgrade or restore options

Name	Options	Description
`Action`	Upgrade or Restore	Control whether an upgrade or a restore should be done.
`Behavior`	Report Only or Execute	Report Only generates an upgrade or restore report, but does not execute the action. Execute performs the requested action.
`Remove Non-Partitioned Objects`	True or False	Setting this option to True will allow the upgrade or restore to proceed even if there are non-partitioned objects in shared memory. The default value of False will cause an upgrade or restore to fail if there are non-partitioned objects in shared memory.

`Upgrade Plan`	Upgrade plan file name.	Specify the file containing the upgrade plan for the application.
`Deployment Directories`	An optional `:` separated list of directories on the server where application JAR files are installed.	The default deployment directories are specified when the node is installed. Specifying this value replaces the current value for the node.

Before performing the upgrade, the updated application JAR files must be installed into the node deployment directories. It is recommended that different deployment directories be used for each version of an application being upgraded. This makes it easy to restore the node to a previous application version by resetting the deployment directories to their previous value. Deployment directories are changed using the `Deployment Directories` field in the upgrade or restore dialog.

To perform the upgrade, select the `Upgrade` action in the dialog and set the behavior to `Execute` and then click on the `Submit` button.

It is recommended that the `Report Only` option be executed before performing an upgrade (or restore) to verify the impact of the action.

An upgrade can also be performed using this command:

```
administrator servicename=A upgrade node upgradefile=upgrade.150505
```

Current application versions on all nodes in a cluster can be displayed on the Application Nodes screen as shown in Figure 3.7, or using these commands:

```
//
// Current active applications running in domain
//
administrator servicename=domainmanager display domain type=application

//
//  Display version inconsistencies
//
administrator servicename=domainmanager domainname=Development display cluster
type=classmismatches
```

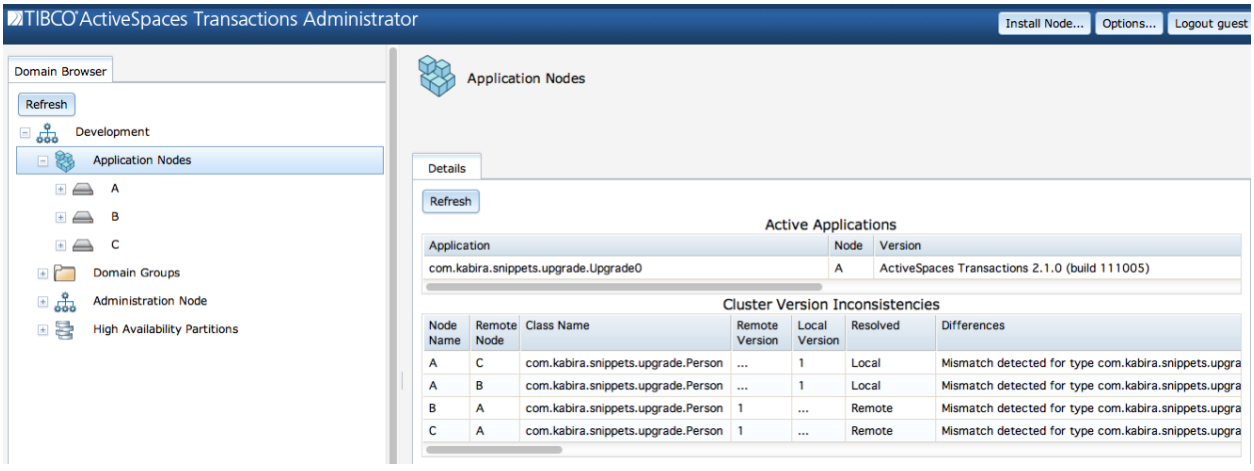

Figure 3.7. Cluster version display

This information is displayed for each application running in the domain:

- **Application** - application name.
- **Node** - node on which application is running.
- **Version** - product version.

This information is displayed for any application version inconsistencies in the cluster:

- **Node Name** - Node reporting inconsistency.
- **Remote Node** - Node to which inconsistency exists.
- **Class Name** - Name of class that is inconsistent.
- **Remote Version** - Remote class version.
- **Local Version** - Local class version.
- **Resolved** - Which node is resolving the inconsistency.
- **Differences** - Description of differences.

Restoring

Restoring a node to a previous application version is done using the upgrade or restore node dialog in Figure 3.6 described in the section called “Upgrading” on page 40, or this command:

```
administrator servicename=A upgrade node restorefile=upgrade.150505
```

Node restoration is useful to back-out a new application version that is causing errors. When a node is restored to a previous application version, the following steps are taken:

1. Audit that all classes being restored are in a partition with at least one replica node. This ensures that the objects can be migrated to another node during the restore process to avoid any data loss and to provide continuous service availability. This audit can be disabled by setting the Remove Non-Partitioned Objects option to True.

   

   Setting the Remove Non-Partitioned Objects option to True can cause data loss, because non-replicated objects will be removed during the restore process. Use this option with extreme caution.

2. Leave the cluster. This causes all active partitions to migrate to their replica nodes.
3. Stop the node.
4. Update the deployment directories for the node if new ones were provided in the upgrade or restore node dialog.
5. Remove all objects that are being restored. Replicated objects are still available on the nodes where the partitions migrated.
6. Restart the node.
7. Add the node as a replica to all partitions that were migrated off the node.
8. Join the cluster.

When the restore process completes, the previous application version has been re-installed on the node, and the node is active in the cluster as a replica. The final step required to bring the node back online is to migrate the partitions that should be active on this node back to the node. See the section called “Migrating partitions” on page 98 for details.

Before performing the restore, the previous version of the application JAR files must be re-installed into the deployment directories for the node. If different deployment directories were specified when the application was upgraded, the previous ones should be used, and specified in the `Deployment Directories` field of the upgrade or restore node dialog. The `Mismatch File` specified in the upgrade or restore node dialog must be the same one that was used to upgrade to the new application version.

Named caches

Named caches are used to control the amount of shared memory allocated to managed objects on a node. Named cache management is available from the `Named Caches` tab on an application node display. Named caches are defined per node.

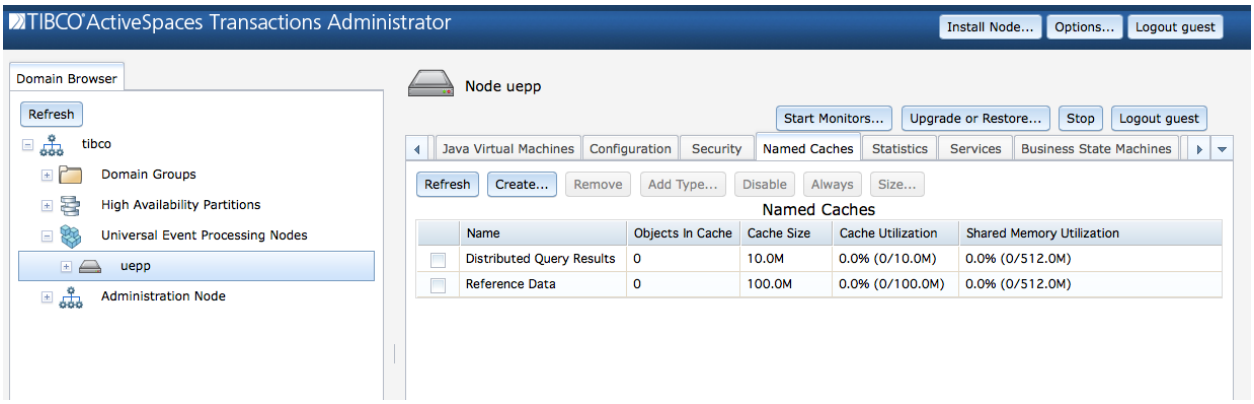

Figure 3.8. Named cache management

The `Named Caches` section contains this information:

- **Name** - the cache name.
- **Objects In Cache** - total number of objects currently in the named cache.
- **Cache Size** - the amount of the shared memory allocated to the named cache.
- **Cache Utilization** - percentage of cache currently being utilized.
- **Shared Memory Utilization** - percentage of total shared memory currently being used for the named cache.
- **Types In Cache** - the types currently contained in the named cache.
- **Flusher Sleep Interval** - the sleep interval, in seconds, for the background flusher for the named cache.
- **Maximum Objects Per Flush** - the maximum number of objects that will be flushed each time the flusher wakes up.

Named cache statistics are an approximation.

This information is also available using this command:

```
administrator servicename=A display cache
```

To define a new named cache click on the Create... button and specify a cache name, or use this command:

```
administrator servicename=A create cache name=MyCache
```

To remove a named cache, select a row in the `Named Cache` table, and click on the Remove button, or use this command:

```
administrator servicename=A remove cache name="Reference Data"
```

To move a type from one named cached to another, select a row in the `Named Cache` table for the named cache to which the type should be moved and click on the Add Type... button, or use this command:

```
administrator servicename=A add cache name=MyCache type=example.managed.X
```

The caching policy for a named cache can be set to one of these values:

- **Unlimited** - objects are always cached in shared memory.
- **No Caching** - objects are never cached in shared memory. Objects are always fetched from a remote node.
- **Sized** - the shared memory cache size is specified. Objects are flushed when the specified size is exceeded.

To change the caching policy for a named cache, select the row in the `Named Caches` table for the named cache to change and click the appropriate button:

- **Disable** - sets the caching for the selected type to `No Caching`.
- **Always** - sets the caching for the selected type to `Unlimited`.
- **Size…** - brings up a dialog to set a specific cache size for the selected type.

The caching policy can also be set using this command:

```
administrator servicename=A set cache name=MyCache size=50%
```

The cache size is specified in one of these ways:

- Percentage of total shared memory available on node.
- Absolute size in bytes, kilobytes, megabytes, or gigabytes

Changing a caching policy may impact system performance if currently cached objects are no longer cached.

Deployment directories

Deployment directories provide a mechanism for installing JAR files on a node that are implicitly added to a JVM's class path.

Deployment directories are not added to a JVM's class path, only the JAR files contained in deployment directories are added to the class path. This implies that class or property files in a deployment directory will not found using a JVMs class path.

Nested directories in a deployment directory are not searched for JAR files, they are ignored. The only exception is if they are a JVM-specific deployment directory (see below).

Node-wide deployment directories

Node-wide deployment directories are optionally specified during node installation and upgrade. See the Deployment Directories Node Installation option: Table 3.1 on page 33.

Node-wide deployment directories apply to all JVMs deployed in the node.

Node-wide deployment directories are created automatically at node installation time, if they do not already exist.

JVM-specific deployment directories

JVM-specific deployment directories apply to the JVM whose name matches the directory name. See the **TIBCO ActiveSpaces® Transactions Java Developer's Guide** for information on setting the JVM name with the deploy tool.

JVM-specific deployment directories are optional, and are not automatically created.

JVM-specific deployment directories are top level sub-directories located within existing node-wide deployment directories. They must be manually created.

If using JVM-specific deployment directories, it is recommended that the JVM name be specified when deploying the JVM. Otherwise, the deploy tool will generate a unique JVM name, that will not be known in advance.

For example, assume that the node deployment directories were set to `/a/jars` and `/b/jars` at node installation time, and two JVM-specific deployment directories named `JVM1` and `JVM2` were created under them with the JAR files installed as shown in Figure 3.9.

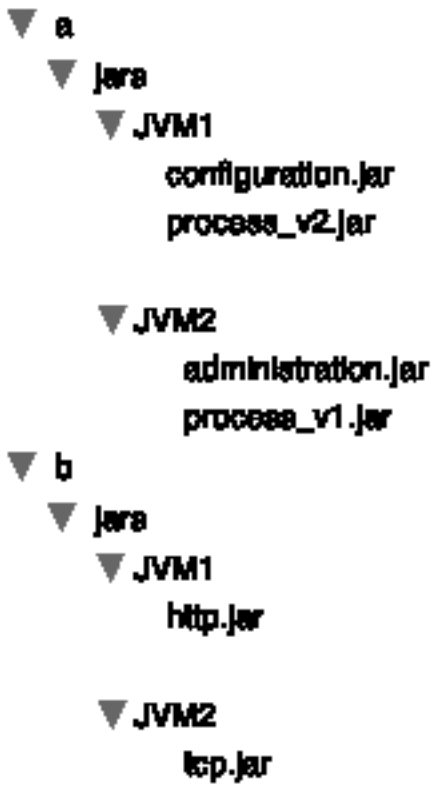

Figure 3.9. Deployment Directories

When `JVM1` and `JVM2` are started they will be have these JAR files added to their class paths:

- JVM1 - `/a/jars/JVM1/configuration.jar:/a/jars/JVM1/process_v2.jar:/b/jars/JVM1/http.jar`
- JVM2 - `/a/jars/JVM2/administration.jar:/a/jars/JVM2/process_v1.jar:/b/jars/JVM2/tcp.jar`

Deploy directory search ordering

When a JVM is started, all configured node deployment directories are searched in the order specified at node installation, or upgrade, for sub-directories with the same name as the JVM being started. If a matching sub-directory is found, all JAR files found within are sorted, by name, in ascending ASCII order, and added to the JVM's class path.

Next any JAR files found in deployment directories are automatically added to the JVM's class path. The top level of the deployment directories are scanned in the order specified at node installation or upgrade time, and within each deployment directory the JAR files are sorted, by name, in ascending ASCII order, and added to the JVM's class path.

Configuration

This section describes how to configure nodes and their associated node agents. It also discusses how configuration life cycle is managed.

Node configuration

Node configuration has a configuration type of `nodeconfig`.

Node configuration is separated into these distinct areas:

- Node wide values
- Per JVM values

The node configuration is separated into these distinct areas:

- Node description configuration
- Memory throttling configuration
- Shared memory IPC configuration
- Deadlock resolution configuration
- Default domain configuration

Table 3.3 on page 48 defines the node description configuration parameters.

Table 3.3. Node description configuration

Name	Type	Description
`defaultDescription`	String	Node description
`properties`	Array of name/value pairs.	An array of service properties. These properties are added to the node service record.

The optional memory throttling configuration controls:

- Utilization percentage of shared memory at which to start throttling. This is set to the value of the `thresholdPercentage` configuration parameter.
- Utilization percentage of shared memory at which to start stalling. This is set to the value of the `thresholdPercentage` plus 30 percent.
- Utilization percentage of shared memory at which to generate a low memory warning. This is set to the value of the `thresholdPercentage` plus 30 percent, but not exceeding 90 percent. A low memory warning is always generated at 90 percent utilization.

Throttling and stall states are used internally to prevent memory exhaustion under extreme loads. They can also be used by an application for the same reasons using the `com.kabira.platform.swbuiltin.EngineServices.throttle()` API. For details on the `throttle` API see the product javadoc.

Table 3.4 on page 49 defines the memory throttling configuration parameters.

Table 3.4. Memory throttling configuration

Name	Type	Description
thresholdPercentage	Integer	A number treated as a percentage at which to start shared memory throttling. The value must be between 0 and 100. There is no default value. This value is required if the memory throttling configuration block is specified.
frequency	Integer	The number of shared memory utilization checks per second. There is no default value. This value is required if the memory throttling configuration block is specified.

Table 3.5 on page 49 defines the shared memory cache flusher configuration parameters.

Table 3.5. Flusher configuration

Name	Type	Description
flushIntervalSeconds	Integer	The number of seconds to sleep between runs of the flusher. A value of 0 disables the flusher. The default value is 1 seconds.
maximumTypes	Integer	Control the number of types that will be flushed per flusher run. A value of 0 indicates that there is no limit. The default value is 0.
maximumObjectsPerType	Integer	Control the number of objects per type that will be flushed per flusher run. A value of 0 indicates no limit. The default value is 0.

Table 3.6 on page 49 defines the shared memory IPC configuration parameters.

Table 3.6. Shared memory IPC configuration

Name	Type	Description
noDestinationTimeoutSeconds	Integer	The number of seconds to block waiting for a target JVM for a method invocation. The current transaction is aborted when the timeout value is exceeded. The default value is 10 seconds.

Table 3.7 on page 49 defines the deadlock resolution configuration parameters.

Table 3.7. Deadlock resolution configuration

Name	Type	Description
maximumBackoffMilliseconds	Integer	The maximum amount of time, in milliseconds, to backoff during deadlock resolution. Deadlock backoff times will never exceed this value. The default value is 10000 milliseconds (10 seconds).

Table 3.8 on page 49 defines the domain configuration parameters.

Table 3.8. Default domain configuration

Name	Type	Description

name	String	Name of domain to join.
group	String	Name of domain group to join. The group must be in the domain configured in name.

JVM configuration is defined per JVM by name. Multiple JVM configuration blocks may be defined. The JVM name is optionally defined when the JVM is deployed using the jvmname deployment tool option. If the jvmname option is not specified when the JVM is deployed, a unique name is generated automatically.

JVM configuration supports specifying:

- Dispatch thread configuration - the simultaneous number of remote method invocations into the JVM.
- Timer thread configuration - the number of concurrent timers, and maximum timer resolution.
- Runtime tracing configuration

All JVM configuration values, other than the JVM name, are optional. If a JVM configuration is deactivated, all of the configuration parameters are set to their default value.

Table 3.9 on page 50 defines the JVM configuration parameters.

Table 3.9. JVM configuration

Name	Type	Description
jvmName	String	JVM name. This parameter is required.
minimumDispatch-Threads	Integer	Minimum number of dispatch threads. This value is optional. Default value is 10.
maximumDispatch-Threads	Integer	Maximum number of dispatch threads. This value is optional. Default value is 2000.
timerParallelism	Integer	Number of timers that can be executing concurrently. This value is optional. Default value is 1.
timerResolutionMilli-seconds	Integer	The maximum timer resolution. This is the interval at which timers are examined. Higher resolution timers have more impact on system performance. This value is optional. Default value is 1000.
traceFatalFilter	Bit string	Trace flag for fatal messages. This value is optional. Default value is 0x3FFFFFF.
traceWarningFilter	Bit string	Trace flag for warning messages. This value is optional. Default value is 0x3FFFFFF.
traceInfoFilter	Bit string	Trace flag for informational messages. This value is optional. Default value is 0x3FFFFFF.
traceDebugFilter	Bit string	Trace flag for debug messages. This value is optional. Default value is 0x0 (all messages disabled).

The bit field values for the trace configuration parameters are defined in Table 3.10 on page 51.

Table 3.10. Trace flag values

Value	Description
0x00000001	System services. Low level operating system abstraction layer in runtime.
0x00000002	Event bus. Shared memory inter-process dispatch mechanism.
0x00000004	Object services. Managed object runtime component.
0x00000008	System management.
0x00000100	Distribution.
0x00000400	Exceptions.
0x00004000	Java. Class loader and JVM interaction with runtime.
0x00008000	System coordinator.
0x00010000	Engine. Process container for a JVM.
0x00020000	Shared memory allocator.
0x00100000	Distributed discovery.
0x00200000	Distribution protocol. Trace the distribution Protocol Data Units.
0x01000000	Security.

Example 3.1 on page 51 is an example node and JVM configuration.

Example 3.1. Node configuration example

```
configuration "Node A" version "3.0" type "nodeconfig"
{
     configure switchadmin
     {
          configure Node
          {
               //
               //     Set the node description and add
               //     a service property of foo=bar to the
               //     node service record
               //
               Description
               {
                    defaultDescription = "Development";
                    properties =
                    {
                         {
                              name = "foo";
                              value = "bar";
                         }
                    };
               };

               //
               //     Shared memory flusher control
               //
               Flusher
               {
                    flushIntervalSeconds = 2;
                    maximumTypes = 3;
                    maximumObjectsPerType = 4;
               };

               //
               //     Set a memory utilization threshold of 50%
```

```
               //     and check 5 times a second
               //
               MemoryThrottle
               {
                    thresholdPercentage = 50;
                    frequency = 5;
               };

               //
               //     Set the no destination timeout value to 10 seconds
               //     for the Shared memory IPC mechanism
               //
               EventBus
               {
                    noDestinationTimeoutSeconds = 10;
               };


               //
               //    Set the maximum deadlock backoff resolution to 5 seconds
               //
               Deadlock
               {
                    maximumBackoffMilliseconds = 5000;
               };

               //
               //     Join the Application Cluster group in
               //     the Development domain
               //
               Domain
               {
                    name = "Development";
                    group = "Application Cluster";
               };
          };

     };

     configure switchadmin
     {
          JVM
          {
               jvmName = "My JVM";

               //
               //    Configure dispatch threads
               //
               minimumDispatchThreads = 8;
               maximumDispatchThreads = 16;

               //
               //    Allow four concurrent timers with a 500 millisecond resolution
               //
               timerParallelism = 4;
               timerResolutionMilliseconds = 500;

               //
               //    Update runtime trace flags
               //
               traceDebugFilter = 0x0;
               traceInfoFilter = 0x3FFFFFF;
               traceWarningFilter = 0x3FFFFFF;
               traceFatalFilter = 0x3FFFFFF;
          };
      };
};
```

Node agent configuration

Node agent configuration has a configuration type of `nodeagent`.

Table 3.11 on page 53 defines the node agent configuration parameters.

Table 3.11. Node agent configuration

Name	Type	Description
`enableEventForwarding`	Boolean	A value of `true` enables the node agent network listener. A value of `false` disables the node agent event listener preventing events from being published to Domain Managers.
`enableTraceToEvent`	Boolean	A value of true enables publishing of log file entries as events. A value of false disables the publishing of log file entries as events.
`numberOfMessages`	Integer	The size of the log file trace buffer in number of messages. If the number of messages in the trace buffer exceed this value, new messages are discarded until there is room in the buffer.
`listenAddress`	String	The node agent network listener address. The format of this string is *<protocol>:<host name>:<port number>.* A value of "0" for the port number field will cause the node agent to pick a random port number. Protocol must be one of `TCP` or `SSL`.
`topicNameList`	String list	List of event topics. All events on these topics will be sent to remote Domain Managers.

The node agent `listenAddress` configuration must match the Domain Manager `nodeAgentAddress` (see Table 2.3 on page 25) configuration value if service discovery is not being used for managed node discovery. If these two values do not match, events will not be forwarded to the Domain Manager by the node. The `listenAddress` configuration value defines a network listener. The `nodeAgentAddress` configuration value defines the remote address that the Domain Manager, acting as a network client, should connect with to communicate with the node.

Example 3.2 on page 53 is an example node agent configuration.

Example 3.2. Example node agent configuration

```
configuration "nodeagent" version "2.0" type "nodeagent"
{
     configure nodeagent
     {
          NodeAgentConfiguration
          {
               enableEventForwarding = true;
               enableTraceToEvent = true;
               numberOfMessages = 1000;
               listenAddress = "TCP::0";
               topicNameList =
               {
                    "kabira.kis",
                    "kabira.kts"
               };
          };
```

```
    };
};
```

Managing configuration

A complete discussion of configuration life cycle can be found in the **TIBCO ActiveSpaces® Transactions Java Developer's Guide**. In summary configuration follows this life-cycle:

- Configuration is loaded into a node
- Configuration is activated
- Configuration is deactivated
- Configuration is removed from a node

Configuration is loaded into a node using either the TIBCO ActiveSpaces® Transactions Administrator dialog show in Figure 3.10 or this command:

```
administrator servicename=A load configuration source=myConfiguration.kcs
```

The `Ignore memory threshold` check box is used to allow configuration data to be loaded in shared memory even if the configured shared memory throttle utilization is exceeded (see Table 3.4 on page 49). Without this override, attempting to load configuration data into congested shared memory will fail.

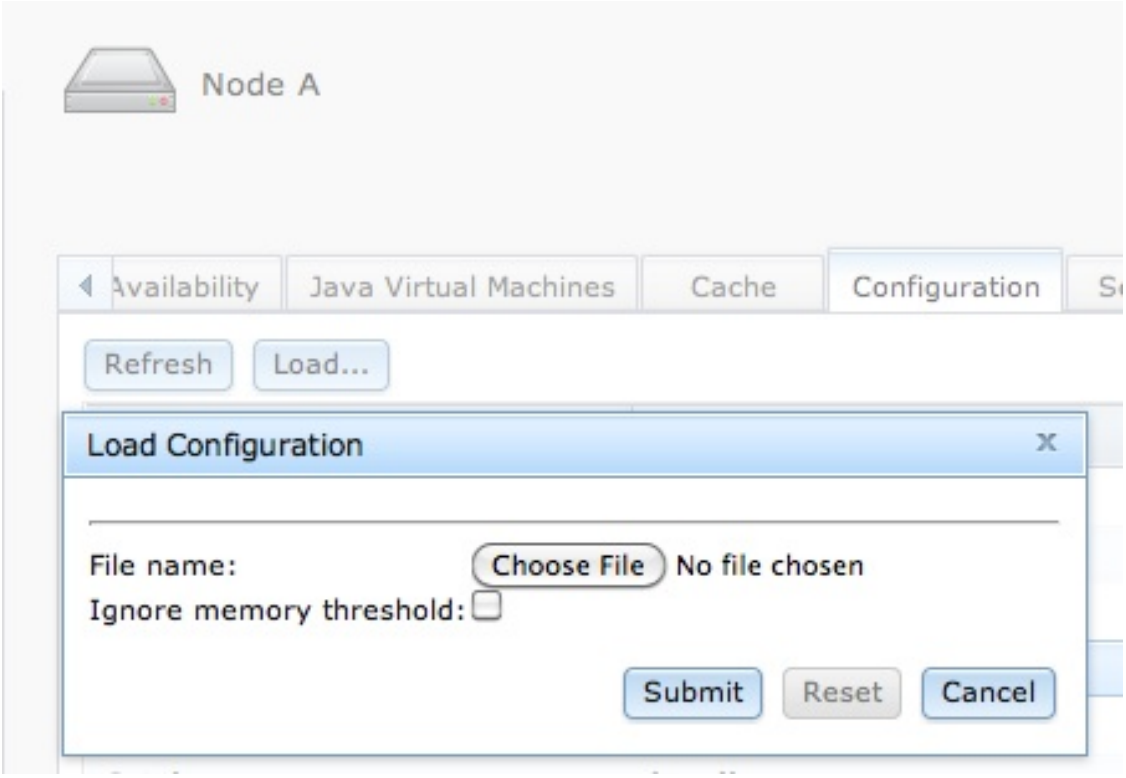

Figure 3.10. Loading configuration

All loaded configuration data can be displayed from the `Configuration` tab for a node as shown in Figure 3.11, or using this command:

```
administrator servicename=A display configuration
```

This information is displayed for each loaded configuration file:

- **Type** - Configuration type.
- **Name** - Configuration name.
- **Version** - Configuration version.
- **State** - Current state.
- **State Change Time** - Time of last state change.
- **Load Time** - Time configuration file was loaded.
- **Initial Activation Time** - Initial time configuration was activated.
- **Source** - Configuration source file path.
- **Number of Configuration Objects** - Number of objects created from the configuration file.
- **Principal** - User that last changed configuration state.

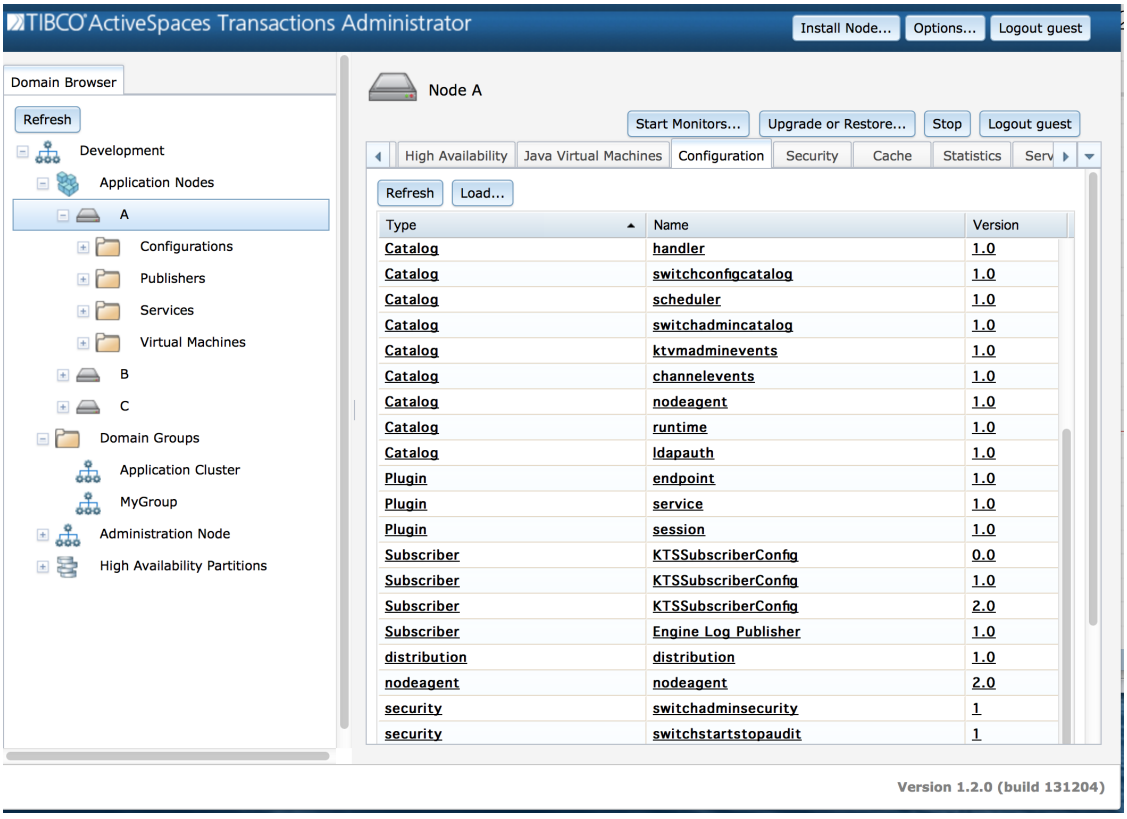

Figure 3.11. Loaded configuration

Configuration activation, deactivation, and removal is done from the TIBCO ActiveSpaces® Transactions Administrator configuration screen show in Figure 3.12.

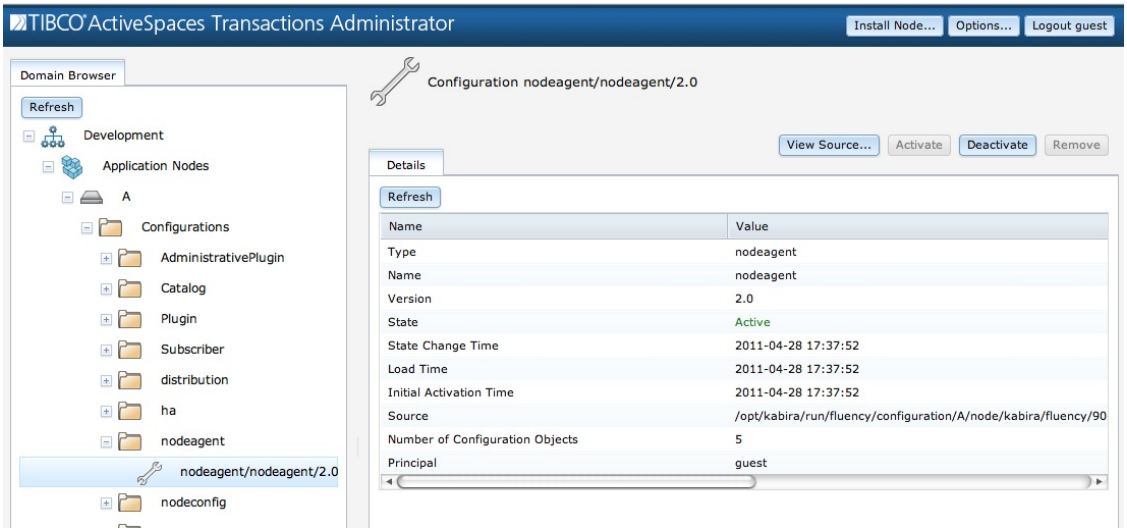

Figure 3.12. Managing configuration

Configuration state changes and also be done using these commands:

```
//
//  Activate a deactive configuration
//
administrator servicename=A activate configuration type=distribution name=distribution
version=1.0

//
//  Deactivate an active configuration
//
administrator servicename=A deactivate configuration type=distribution name=distribution
 version=1.0

//
//  Remove configuration from shared memory
//
administrator servicename=A remove configuration type=distribution name=distribution
version=1.0
```

The View Source... button shown in Figure 3.12 can be used to export loaded configuration in a format that can be modified and reloaded. Configuration can also be exported using this command:

```
administrator servicename=A export configuration type=distribution name=distribution
version=1.0
```

4

Java Virtual Machines

This chapter describes how to manage and configure Java Virtual Machines running in an TIBCO ActiveSpaces® Transactions node.

Creating a Java virtual machine

TIBCO ActiveSpaces® Transactions nodes support running multiple Java Virtual Machines. The number of JVMs started is application specific.

JVMs are created using the deployment tool described in the section called “Deployment tool” on page 57.

Deployment tool

Java Virtual Machines are created using the TIBCO ActiveSpaces® Transactions deployment tool which is packaged in the `deploy.jar` file located in the TIBCO ActiveSpaces® Transactions SDK. The deployment tool supports these general forms:

```
java [local JVM options] -jar deploy.jar [options] <target> [application parameters]

java [local JVM options] -jar deploy.jar [options] help
```

A complete reference for the deployment tool options can be found in the **TIBCO ActiveSpaces® Transactions Java Developer's Guide** or using the `help` command. The deployment tool supports two modes when starting a JVM to execute application code. These modes are controlled with the `detach` option to the deployment tool. The modes are:

- Detached - `detach = true`
- Attached - `detach = false`

The behavior of the deployment tool in detached mode depends on the value of the *<target>* parameter. If the *<target>* parameter is a fully scoped name of a `main` method to execute in the

JVM, all Java classes must be available on the server in a deployment directory. If the *<target>* parameter is an archive file, e.g. a JAR, the archive is copied to the node before starting the JVM.

The deployment tool returns to the caller after starting a JVM, executing the `main` method, and then waiting for application startup status for `detachtimeout` seconds, where `detachtimeout` is a deployment tool option. If the application exits with an error before the `detachtimeout` period expires, an error is returned to the deployment tool. If the `detachtimeout` period expires and the application is still running, success is returned to the deployment tool. Detached mode is recommended for deployment of production systems.

When using detached mode with a main method *<target>*, all classes must be available on the server. If they are not available program execution will fail with a `java.lang.NoClassDefFoundError`.

Attached mode supports both a JAR file, or a fully scoped name of a `main` method, in the `<target>` parameter. The JAR or class specified on the command line is copied to the server before the JVM is started. The JVM is started and the `main` method is executed. The deployment tool continues to block waiting for the JVM to exit. While the JVM is active, class files are shipped to the server from the client if the class is not already available on the server. Attached mode is very useful during development to support transparent integration with Java IDEs.

To summarize, the following steps are taken by the deployment tool:

1. Archive or class file (attached mode only) specified in the *<target>* parameter is copied to the node.
2. The JVM is started.
3. Specified `main` method is executed.
4. In detached mode the deployment tool returns to the caller, in attached mode it blocks waiting for the JVM to exit

When the deployment tool is running in attached mode, keep-alive messages are continuously sent between the deployment tool and all application nodes. If an application node does not receive a keep-alive message from the deployment tool within a configurable amount of time (`keepaliveseconds` option to the deployment tool) the node terminates the JVM running the application. This ensures that applications running in attached mode are not abandoned if there are client or network errors. Keep-alive messages are not used when the application is deployed in detached mode.

Example 4.1 on page 58 shows a simple application that starts a daemon thread and then returns from `main`.

Example 4.1. Simple application

```
//     $Revision: 1.1.2.1 $

package com.kabira.snippets.vmlifecycle;

/**
 *  Using daemon threads
 * <p>
 * <h2> Target Nodes</h2>
 * <ul>
 * <li> <b>domainnode</b> = A
 * </ul>
```

```
 */
public class Daemon
{
    /**
     * Application thread
     */
    public static class MyThread extends Thread
    {
         @Override
         public void run()
         {
              try
              {
                   System.out.println("thread sleeping...");

                   Thread.sleep(5000);
              }
              catch (InterruptedException ex)
              {
                   //     Handle exception
              }
         }
    }

    /**
     * Main entry point
     * @param args  Not used
     */
    public static void main(String[] args)
    {
          //
          //     Create a new thread
          //
          MyThread     t = new MyThread();

          //
          //     Mark the thread as a daemon thread
          //
          t.setDaemon(true);

          //
          //     Start the thread
          //
          t.run();

          //
          //     Returning from main - causes the JVM to exit
          //
          System.out.println("returning from main");
    }
}
```

This application is deployed to an TIBCO ActiveSpaces® Transactions node with the following command. Notice that the class containing the `main` method is specified on the command line.

```
java -jar deploy.jar username=guest password=guest \
     hostname=kabira-server.local adminport=2001 \
     com.kabira.snippets.vmlifecycle.Daemon

INFO: deploy.jar version: [TIBCO TIBCO ActiveSpaces® Transactions
 2.2.0 (build 111214)] starting at [Fri Dec 16 07:30:53 PST 2011]
INFO: JVM remote debugger agent running on [kabira-server.local:41895] ...
INFO: node version: [TIBCO TIBCO ActiveSpaces® Transactions
 2.2.0 (build 111214)]
INFO: Starting application [com.kabira.snippets.vmlifecycle.Daemon] ...
```

```
INFO: AST component [com.kabira.snippets.vmlifecycle.Daemon] started on JVM
[com_kabira_snippets_vmlifecycle_Daemon10].
WARNING: loopback ip address choosen, this agent may not connect to remote agents
ip_address=127.0.0.1 port=50004
Listening for transport dt_socket at address: 41895
INFO: JMX Management Service started at:
   kabira-server:2099
   172.16.208.130:2099
   service:jmx:rmi:///jndi/rmi://kabira-server:2099/jmxrmi
thread sleeping...
returning from main
INFO: Application [com.kabira.snippets.vmlifecycle.Daemon] exited with status [0].
```

The application is now running in a JVM on the target TIBCO ActiveSpaces® Transactions node.

Setting Java virtual machine options Virtual machine options are set using command line options to the deployment tool. For example to change the minimum and maximum memory sizes for the JVM being started the following command would be used:

```
java -jar deploy.jar username=guest password=guest hostname=192.168.71.129 \
     adminport=2001 detach=true -Xms256m -Xmx512m com.kabira.snippets.jvmlifecycle.Daemon
```

Class resolution

During JVM execution all class files must be available in the node's class path, or resolved using the deployment tool's CLASSPATH when running in attached mode.

When the deployment tool is running in attached mode, the CLASSPATH of the JVM running the deployment tool (`deploy.jar`) is used to resolve class references if they cannot be resolved in the node's class path.

Deployment directories are used to add JAR files to a node's class path. By default nodes define the deployment directory as:

```
#
#   Path relative to the node installation directory
#
../../deploy
```

All JAR files copied into the deploy directory are automatically added to the node's class path. See the section called “Installation” on page 31 for details on changing the default deployment directory.

JVM Administration

After a JVM has been created in a node it can be started, stopped, and removed. If a JVM is removed it must be redeployed.

Figure 4.1 shows the screen that is used to monitor and control a JVM. The information on this screen can also be displayed using this command:

```
administrator servicename=A display jvm name=sandbox_Timer0
```

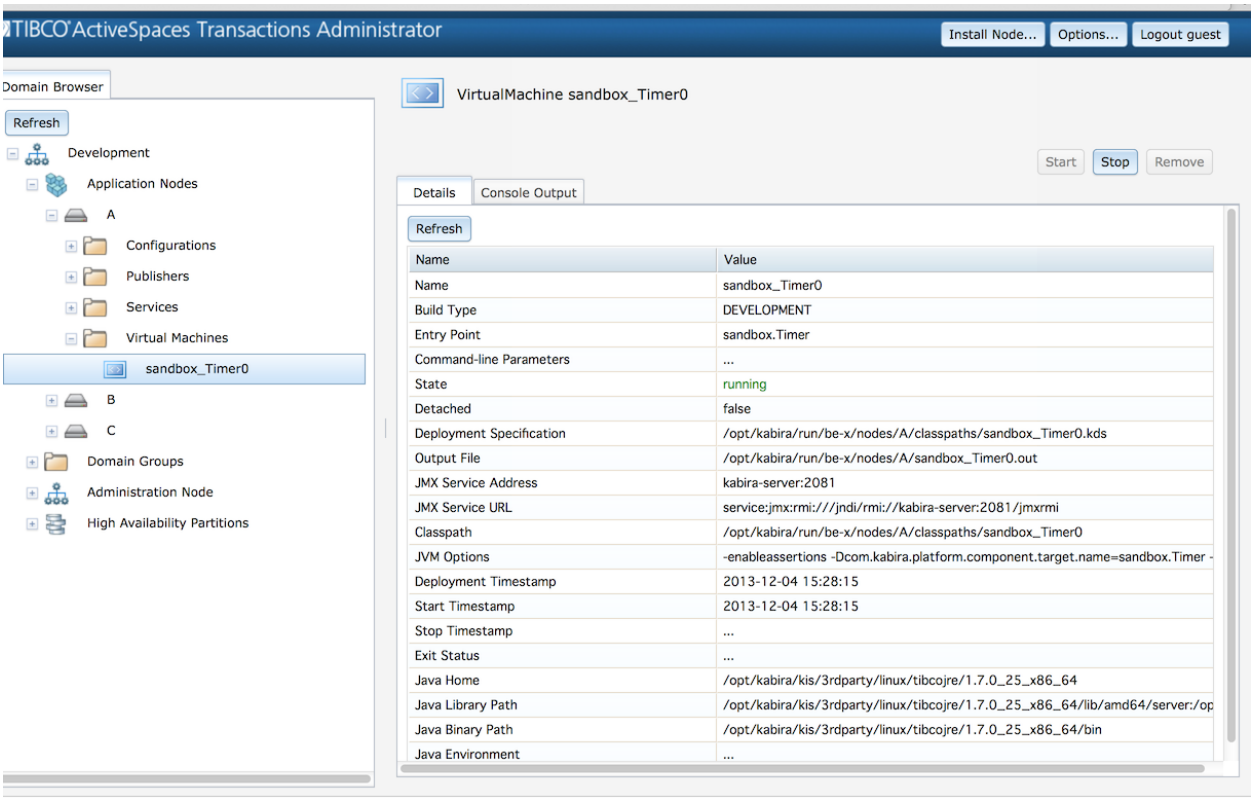

Figure 4.1. JVM administration

This information is displayed for a deployed JVM:

- **Name** - JVM name.
- **Build Type** - `DEVELOPMENT` or `PRODUCTION` build.
- **Entry Point** - Class containing `main`.
- **Command-line Parameters** - Application parameters specified at deployment.
- **State** - Current state.
- **Detached** - Boolean indicating detached state.
- **Deployment Specification** - Generated internal deployment specification.
- **Output File** - Path to file containing standard output for the JVM.
- **JMX Service Address** - Network address to connect with JMX.
- **JMX Service URL** - Network URL to connect with JMX.
- **Classpath** - Class path used when starting JVM.
- **JVM Option**s - Options used when starting JVM.
- **Deployment Timestamp** - Time JVM was deployed.

- **Start Timestamp** - Time JVM was last started.
- **Stop Timestamp** - Time JVM was last stopped.
- **Exit Status** - Exit status last time JVM was stopped.
- **Java Home** - Java JDK/JRE path.
- **Java Library Path** - JDK/JRE native library path.
- **Java Binary Path** - JDK/JRE binary executable path.
- **Java Environment** - Environment variables passed to JVM when it is started.

Once a JVM is deployed it supports being started, stopped, and removed. These are accomplished using the Start, Stop, and Remove buttons shown on Figure 4.1. These actions can also be performed from the command line:

```
//
//  Start a deployed JVM
//
administrator servicename=A start jvm name=sandbox_Timer0

//
//  Stop a running JVM
//
administrator servicename=A stop jvm name=sandbox_Timer0

//
//  Remove the JVM from the node.  It must be redeployed
//
administrator servicename=A remove jvm name=sandbox_Timer0
```

5

Security

This chapter describes how to manage TIBCO ActiveSpaces® Transactions security. The TIBCO ActiveSpaces® Transactions security model uses these concepts:

- System users are formally known as *principals*.
- A principal is defined with a *credential* (password) and they are granted one or more *roles*.

Principals are defined using administration commands, or optionally configuration.

A user who installs a node is automatically granted administrative access to the node. This user has full administrative control of the installed node when logged in from the same host on which the node was installed. It is also possible to specify a different user at node installation. That user is also granted full administrative access to the installed node. See the section called “Default Node Security” on page 38 for details.

Apart from the user that installed the node, other users may administer the node only if they are granted the administrative role. See the section called “Roles” on page 70 for complete details on the default security roles.

All access to a node is controlled by a username and password that is defined for a principal. This is true for all supported administrative clients, including JMX tools.

When using TIBCO ActiveSpaces® Transactions Administrator, you need to also have access permission to the domain management node itself as well as all the managed nodes. See the section called “Domain security model” on page 64 for more details.

You can use administration clients without authentication to view some general *public* properties for each node on the network. These are the properties published by the discovery service.

Access to all other node details - and to the managed elements contained within the node - is controlled via the security service. To access these elements, you need to be authenticated. In TIBCO ActiveSpaces® Transactions Administrator you do this by logging in to the domain, after which your credential is cached for future accesses to all nodes managed by the domain. When using the

command line client, you must provide the authentication with each command. JMX credential management is specific to the JMX tool.

The security model is defined more formally in Figure 5.1.

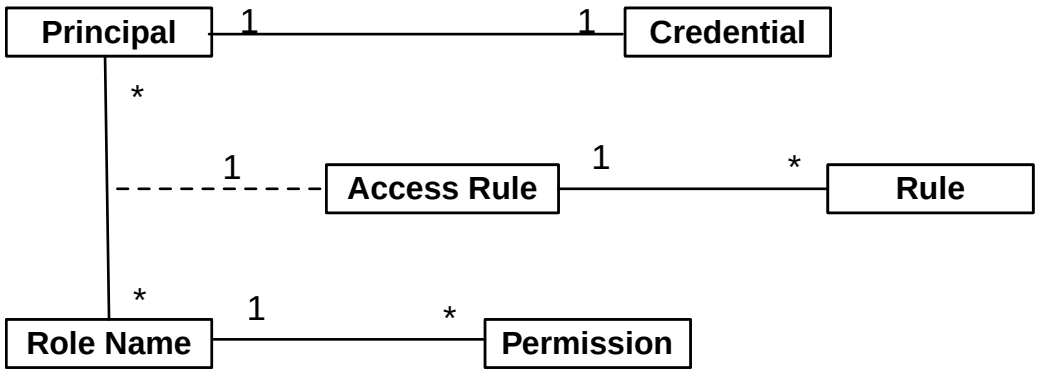

Figure 5.1. Security Model

The concepts in the security model are:

- **Principal** - an entity that can be positively identified and verified via a technique known as authentication
- **Credential** - password used to control access to information or other resources.
- **Role Name** - a logical grouping of access control rules.
- **Access Rule** - associate permission with a specific role name.
- **Rule** - an access control directive for a specific resource (type) in the system.

A principal definition contains one or more role names that define the access privileges granted to the principal.

Domain security model

The introduction of a Domain Manager does not alter the TIBCO ActiveSpaces® Transactions security model, but it does introduce some additional configuration requirements on the administrative domain. Administrative commands issued to managed nodes through a domain manager must use one of these authentication mechanisms:

- the principal and credential provided to the administrative command must be defined on both the domain manager node and the target managed node(s) for the command.
- the target managed node(s) must have the domain manager host defined as a trusted host (see the section called “Trusted Hosts” on page 67).

The choice of authentication mechanism is dependent on local security polices and whether the application is deployed into a trusted network. For example, the use of trusted host authentication would not be appropriate on an untrusted network.

Figure 5.2 shows the use of common principal and credential information in an administrative domain.

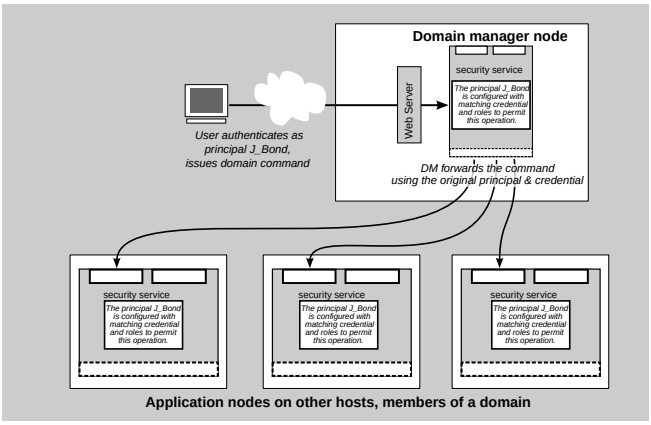

Figure 5.2. Common principal and credential authentication policy

Principal authentication

TIBCO ActiveSpaces® Transactions authentication is based on principals, or users of the system. A principal is uniquely identified by a user name, and has a set of attributes. These attributes include:

- **User Name** - Unique identifier for a principal.
- **Password** - Credential for this principal.
- **Roles** - One or more roles granting access to administrative commands.
- **Password Expiration** - The duration in days that the password is valid.
- **Remote Access** - Allow access only from trusted hosts, or from any host.
- **Password Required** - Define whether a password is always required, or only from untrusted hosts.

These attributes are defined using administrative commands, or loading and activating a configuration file.

It is strongly recommended that administrative commands are used to define principals to avoid storing a clear text password, which is required when using configuration files. The remainder of this section only uses the administrative commands to manage user definitions.

the section called “Add principal” on page 74 shows how to add a user to a node.

Authentication verifies that a given principal is valid and that the presented credentials match the credentials defined for the principal. Once authentication succeeds, all work done in the context of the current administration command will be done in an authenticated session for that principal. Any access to secured resources during that session will require that the principal was granted a role which has access privileges for the resource. Principal definitions may be updated, for example to change the credential, or the set of granted roles. See the section called “Update principal” on page 80 for details on how to update an existing user definition.

Authentication Sources

Authentication is performed using one or more *authentication sources*. An authentication source provides a mechanism to authenticate a principal. There is always at least one authentication source configured for a node. It is named `Local`. The `Local` authentication source uses the principal definitions on the local node to perform authentication.

It is also possible to configure multiple external LDAP authentication sources. An external LDAP authentication source contains both principal and role definitions for authorized users.

The `Local` authentication source is always used to authenticate the user that installed the node. This ensures that the node can be administered even if all external authentication sources are unavailable.

When multiple authentication sources are configured, a priority is defined which controls the order the authentication sources are used to attempt to authorize a principal. During an authentication attempt, an authentication source returns one of the following:

- Authentication succeeded - the authentication succeeded.
- Authentication failed - the authentication failed because the provided credential was invalid, or the principal was unknown.
- Authentication source not available - a failure prevented access to the authentication source.

A successful authentication returned by an authentication source allows a principal access to a node.

An authentication failure returned by an authentication source prevents a principal from accessing a node.

An authentication source not available error, causes the authentication request to be sent to the next highest priority configured authentication source, if one. If there are no more configured authentication sources, the authentication attempt fails and the principal is denied access to the node.

An example authentication source configuration is shown in Example 5.1 on page 66.

Example 5.1. Authentication source configuration example

```
//
//   Authentication source configuration
//
configuration "authentication sources" version "1.0" type "security"
{
    configure security
    {
        configure Authentication
        {
            AuthenticationSources
            {
                //
                //    Prioritized list of authentication sources.
                //
                sourceList =
                {
                    {
                        name = "ldap-west";
                    },
                    {
                        name = "ldap-east";
```

```
                    },
                    {
                        // The internal authentication source
                        name = "Local";
                    }
                };
            };
        };
    };
};
```

This configuration will cause the authentication sources to be attempted in this order:

1. `ldap-west`
2. `ldap-east`
3. `Local`

Trusted Hosts

The default `Local` authentication source in a node supports the configuration of trusted hosts, which allows for expedited authentication of principals when the authentication request originates from a network connection from a configured trusted host. Authentication from a trusted host passes without consideration for credentials. That is, if the authentication request originates from a trusted host, TIBCO ActiveSpaces® Transactions trusts the host-based authentication mechanism (e.g. UNIX login) to have verified the identity of the principal.

Trusted hosts are not supported for external authentication sources.

A user must have the **Password Required** attribute set to **Untrusted Host Only** to take advantage of the trusted host facility. See the section called “Add principal” on page 74.

An example of using trusted hosts is shown below:

```
//
//  Command executed on host1
//
administrator hostname=host2 adminport=1234 display node
```

When the `display node` command is executed on `host1`, the name of the user that executed the command is determined from the operating system. The user name is sent to the node on which the command is being executed (`host2` in the example) and if the command came from a trusted host (`host1` in the example), authentication succeeds.

Trusted hosts are defined using configuration, as illustrated in the following example:

```
//
//  Define trusted  hosts
//
configuration "trusted-hosts" version "1.0" type "security"
{
    configure security
    {
        configure Hosts
        {
            Host
            {
                name = "host1";
```

```
            };
            Host
            {
                name = "host2";
            };
            Host
            {
                name = "host3.some.domain.org";
            };
            Host
            {
                name = "host4.tibco.com";
            };
        };
    };
};
```

The above example configuration defines these trusted hosts:

- `host1` - a simple host name.
- `host2` - another simple host name.
- `host3.some.domain.org` - a fully qualified host name.
- `host4.tibco.com` - another fully qualified host name.

Trusted hosts can also be specified at node installation. See the section called “Installation” on page 31.

Secure Shell (SSH)

The `administrator` command-line tool provides administrative access to TIBCO ActiveSpaces® Transactions nodes running on both local and remote hosts. Some commands sent to nodes running on remote hosts are executed via Secure Shell (SSH) connections. SSH is a tool which provides authentication and confidentiality for network connections to remote hosts.

SSH requires configuration of authentication keys for the user who executes the remote request. Authentication keys for SSH are generated using the `ssh-keygen` command-line tool. See the `ssh-keygen` manual page (run `man ssh-keygen` on a Unix host) for further information on how to generate authentication keys, and where the keys must be stored for SSH to make use of them.

Access control

TIBCO ActiveSpaces® Transactions provides an access control facility, which enables secure access to administration commands.

Access control is role-based, and is configured using configuration files. Each configuration file contains one or more access control *rules*. Each rule defines a set of privileges granted to specific *roles*.

For example, the following configuration file defines access control policy for an administrative command `myCommand` in target `p.mytarget`:

```
configuration "mytarget" version "1.0" type "security"
{
    configure security
    {
```

```
        configure AccessControl
        {
            Rule
            {
                name="p.mytarget.myCommand";
                accessRules =
                {
                    {
                        roleName = "switchadmin";
                        permission = Execute;
                    };
                };
            };
        };
    };
};
```

Once the above configuration is activated, access control will enforce that only principals who have been assigned the `switchadmin` *role* are allowed to execute the `p.mytarget.myCommand` command.

In addition to access control rules for specific commands, access control also supports configuration of access control policy for an administrative target as a whole, using the following configuration options:

- `lockAllElements` = *<true|false>* - Prohibit access to all commands in the target.
- `AccessAllOperationsAndAttributes` permission - grants full access to all commands in the target to the given role.

For example, the following configuration specifies that all commands in the target `p.mytarget` can only be executed by principals who have been assigned the `switchadmin` role:

```
configuration "mytarget" version "2.0" type "security"
{
    configure security
    {
        configure AccessControl
        {
            Rule
            {
                name="p.mytarget";
                lockAllElements=true;
                accessRules =
                {
                    {
                        roleName = "switchadmin";
                        permission = AccessAllOperationsAndAttributes;
                    };
                };
            };
        };
    };
};
```

The above example shows how to configure secure access to an entire administrative target for a specific role, without having to explicitly define access control rules for every command in the target.

Failing to specify `lockAllElements` on an administrative target allows execution of all unprotected commands on an administrative target by any principal.

Roles

The TIBCO ActiveSpaces® Transactions access control supports a set of predefined roles that are used to define access control for principals. Application specific roles are also supported.

The predefined roles are:

- `switchadmin` - allows modification of the operational state of a node. This includes control of a node's lifecycle, e.g. Installation, starting, stopping, and removal.
- `switchmonitor` - allows monitoring of the operational state of a node.

TIBCO ActiveSpaces® Transactions nodes have a default access control policy installed. The default access control policy restricts administration functions to principals using the predefined roles.

Each of these roles is described in more detail below.

switchadmin The `switchadmin` role assigns administrative privileges to principals. The `switchadmin` role is automatically granted to the user who installed the node. That user always has full administrative control of the node when logged in on the same host. All operations which modify the operational behavior of a TIBCO ActiveSpaces® Transactions node may only be executed by principals which have `switchadmin` role privileges. Operational control of a node occurs via `administrator`, TIBCO ActiveSpaces® Transactions Administrator, and JMX. This role should be granted to principals that can perform operations that modify the node state.

switchmonitor The `switchmonitor` role assigns system monitoring privileges to principals. This role is granted execute permission to all display-type administrative operations. Consequently, this role should be granted to principals who can monitor node status. Such principals will be able to display the state of a TIBCO ActiveSpaces® Transactions node, but will be unable to execute administrative operations which change the operational state of the node.

Application defined roles As discussed in the section called “Access control” on page 68, it is also possible to define application specific roles. This section provides a complete example of defining application specific roles, including the definition of principals to use the roles.

Two application roles are defined in Example 5.2 on page 70. They are:

- `exchange-administrator` - A role to administrator an exchange. This role is granted `AccessAllOperationsAndAttributes` permission to the market administration target (`com.tibco.demo.exchange.admin.MarketTarget`). This allows this role to execute all market administrative commands.
- `exchange-trader` - A role to display market status. This role is only granted *execute* permission to a display market administrative command (`com.tibco.demo.exchange.admin.MarketTarget.display`).

Example 5.2. Application defined roles

```
configuration "exchange-security" version "1.0" type "security"
{
    configure security
    {
        configure AccessControl
        {
            Rule
            {
```

```
                name = "com.tibco.demo.exchange.admin.MarketTarget";
                lockAllElements = true;
                accessRules =
                {
                    {
                        roleName = "exchange-administrator";
                        permission = AccessAllOperationsAndAttributes;
                    }
                };
            };
            Rule
            {
                name = "com.tibco.demo.exchange.admin.MarketTarget.display";
                accessRules =
                {
                    {
                        roleName = "exchange-trader";
                        permission = Execute;
                    }
                };
            };
        };
    };
};
```

Example 5.3 on page 71 defines two new principals to use the application roles defined in Example 5.2 on page 70. The defined principals are:

- `trader` - this principal can display market and node status because they were granted both the `exchange-trader` and `switchmonitor` roles.
- `administrator` - this principal can manage exchange markets and nodes because they were granted the `exchange-administrator`, `switchadmin` and `switchmonitor` roles.

Example 5.3. Principal definitions

```
configuration "exchange-users" version "1.0" type "security"
{
    configure security
    {
        configure Principals
        {
            //
            //    Traders
            //
            Principal
            {
                name = "trader";
                textCredential = "trader";
                credentialRequired = true;
                roles =
                {
                    "exchange-trader",
                    "switchmonitor"
                };
            };

            //
            //    Administrator
            //
            Principal
            {
                name = "administrator";
                textCredential = "administrator";
                credentialRequired = true;
```

```
                roles =
                {
                    "exchange-administrator",
                    "switchadmin",
                    "switchmonitor"
                };
            };
        };
    };
};
```

Administration

Security monitoring and administration is done from the node `Security` tab. The following commands are accessed from the `Security` tab.

- `Add` - add a new principal in the node's `Local` authentication source. See the section called “Add principal” on page 74.
- `Audit` - audit the administrative commands security configuration. See the section called “Audit security” on page 76.
- `Export` - export the node's `Local` authentication source's user configuration. See the section called “Export user configuration” on page 76.
- `Remove` - remove a principal definition from the node's `Local` authentication source. See the section called “Remove principal” on page 79.
- `Reset` - reset a password in the node's `Local` authentication source. See the section called “Reset password” on page 78.
- `Update` - update a principal definition, other than password, in the node's `Local` authentication source. See the section called “Update principal” on page 80.

Figure 5.3 shows the security information displayed from the `Security` tab.

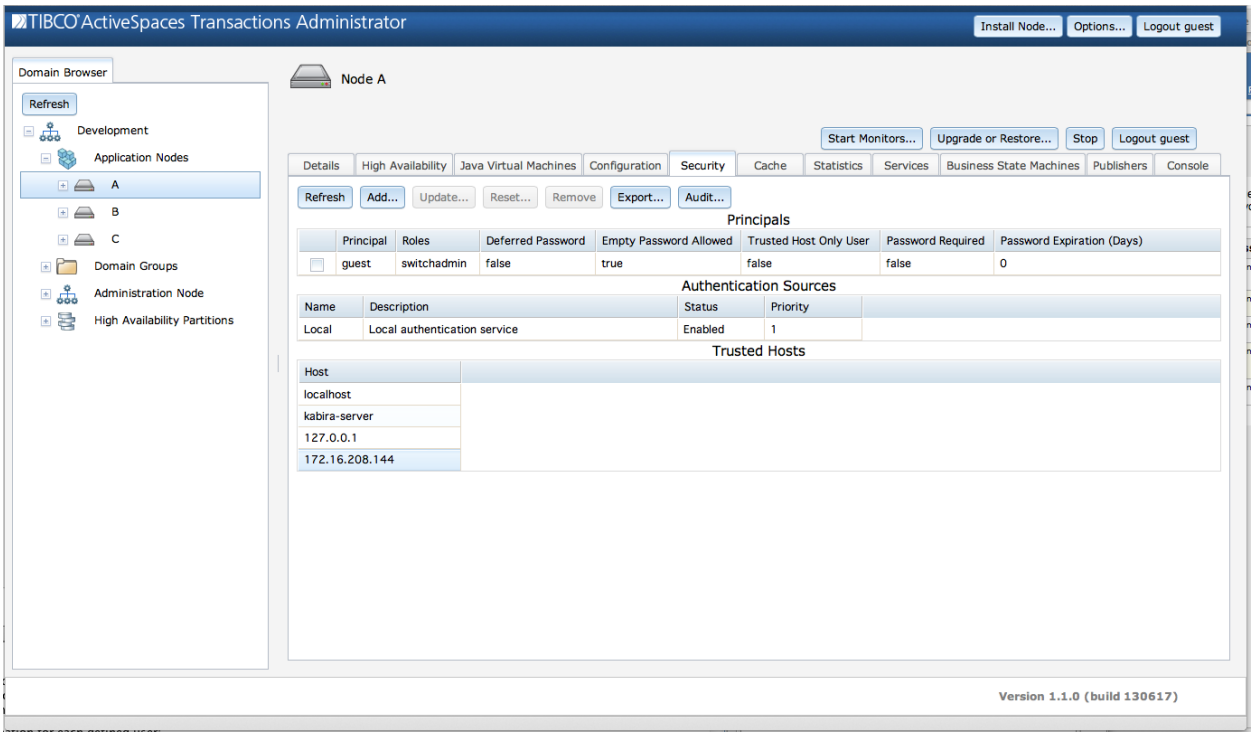

Figure 5.3. Security information

This screen consists of these sections:

- **Principals** - all principals defined in the `Local` authentication source for this node.
- **Authentication Sources** - all authentication sources being used by this node.
- **Trusted Hosts** - the trusted host defined by this node.

The `Principals` section shows this information for each principal defined in the node's `Local` authentication source:

- **Principal** - Principal name.
- **Roles** - Roles granted to this principal.
- **Deferred Password** - A value of `true` indicates that this principal's password was reset, and the new password will be set the next time they access the node. A value of `false` indicates that this principal is either not using deferred password definition, or they have accessed the node and set a new password.
- **Empty Password Allowed** - Empty password support is deprecated. It will be removed in a future release.
- **Trusted Host Only User** - A value of `true` indicates that this principal can only access this node from a trusted host. A value of `false` indicates that this principal can access this node from any host.

- **Password Required** - A value of `true` indicates that this principal must always provide a password - they cannot use trusted hosts. A value of `false` indicates that this user can use trusted hosts without providing a password. A password is always required from a non-trusted host.
- P**assword Expiration (Days)** - The password expiration time in days. A value of zero indicates that the password does not expire.

This information can also be displayed using:

```
administrator servicename=A display security type=principals
```

The `Authentication Sources` section shows this information for each configured authentication source:

- **Name** - Authentication source name.
- **Description** - Authentication source description.
- **Status** - `Enabled` if the authentication source is being used for authentication. `Disabled` if the authentication source is not being used for authentication. If there is no active authentication source configuration then the `Local` authentication source will be the only `Enabled` authentication source.
- **Priority** - Numeric priority of the authentication source. The lower the number the higher the priority. The highest priority is one. This field has no value if the authentication source status is `Disabled`.

This information can also be displayed using:

```
administrator servicename=A display security type=authenticationsources
```

The `Trusted Hosts` section shows this information for each configured trusted host:

- **Host** - host name or IP address for a configured trusted host.

This information can also be displayed using:

```
administrator servicename=A display security type=hosts
```

Add principal

Principals are added to a the node's `Local` authentication source using the `Add Principal` dialog shown in Figure 5.4 accessed from the `Add...` button.

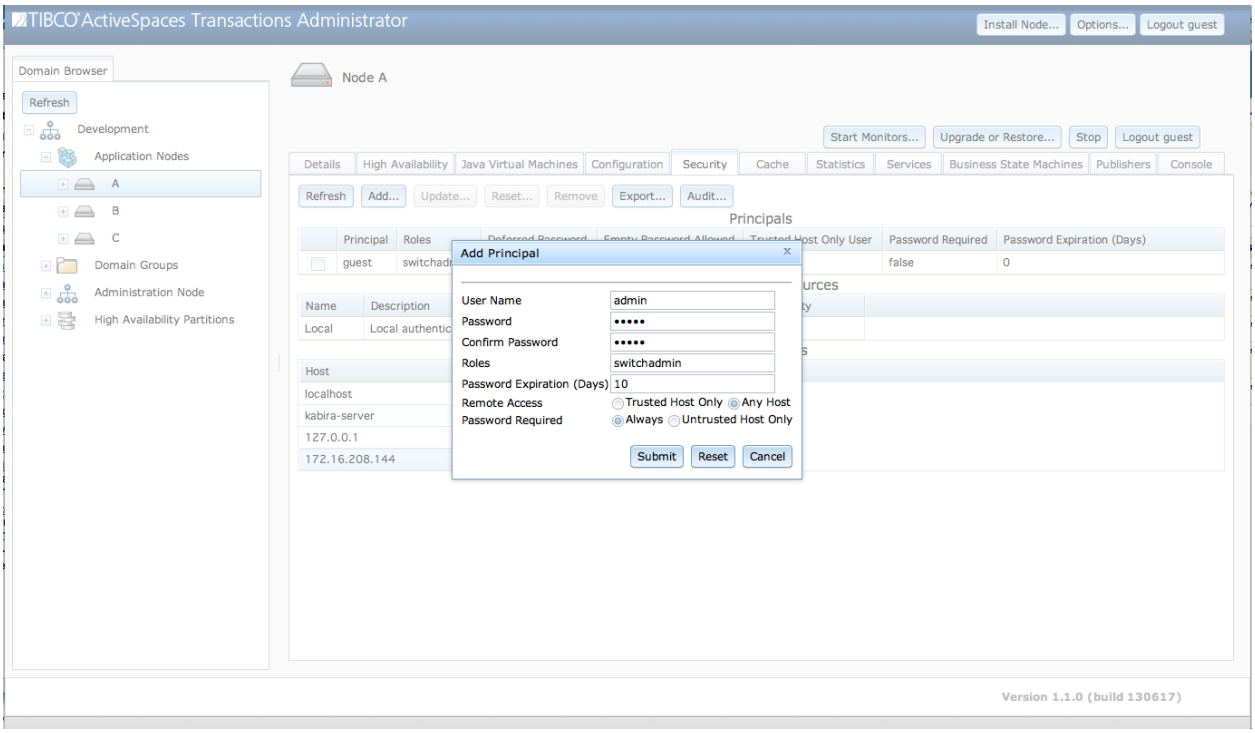

Figure 5.4. Add principal

The fields in the `Add Principal` dialog are:

- **User Name** - A unique user name for this principal.
- **Password** - Initial password for this principal.
- **Confirm Password** - Password confirmation.
- **Roles** - A space separated list of roles to assign to this principal.
- **Password Expiration (Days)** - Password expiration in days. A value of zero indicates that the password does not expire.
- **Remote Access** - Control hosts from which this principal can access this node. `Trusted Host Only` indicates that this principal can only access this node from a trusted host. `Any Host` indicates that this principal can access this node from any host.
- **Password Required** - Control when this principal must provide a password to access this node. `Always` indicates that this principal must always provide a password. They cannot use the trusted host facility. `Untrusted Host Only` indicates that this principal is only required to provide a password from an untrusted host.

Clicking on the `Submit` button will add the new principal to the node after validating that the password values match.

Principals can also be added using:

```
administrator servicename=A add security \
   username=admin roles=switchadmin passwordexpirationdays=10 \
   trustedhostuser=false passwordrequired=true
```

Audit security

When a node starts a security audit is automatically run as part of node startup. It can also be run after a node is started to validate any changes in security configuration, or application features added at runtime.

Security audits are done using the `Security Audit` dialog shown in Figure 5.5 accessed from the `Audit...` button.

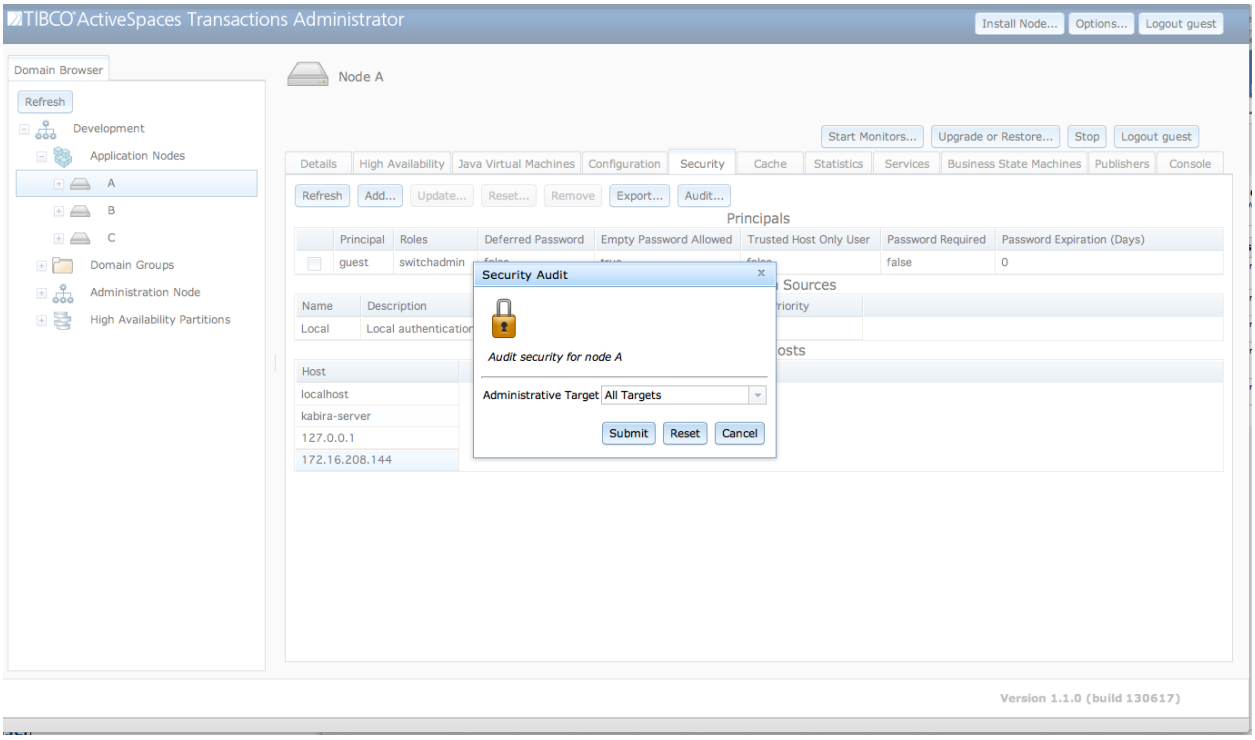

Figure 5.5. Audit security

The fields in the `Security Audit` dialog are:

- **Administrative Target** - A drop-down list of all administrative targets installed on the node. The default `All Targets` value will audit all installed administrative targets, or a specific target to audit can be selected from the drop-down list.

Clicking on the `Submit` button will perform the audit.

A security audit can also be performed using:

```
administrator servicename=A audit security
```

Export user configuration

Configuration for all principals defined on a node can be exported using the `Export Users` dialog show in Figure 5.6. This dialog is accessed from the `Export...` button. Exported user configuration can be reloaded and activated on a node using the standard node configuration mechanisms described in the section called “Managing configuration” on page 54.

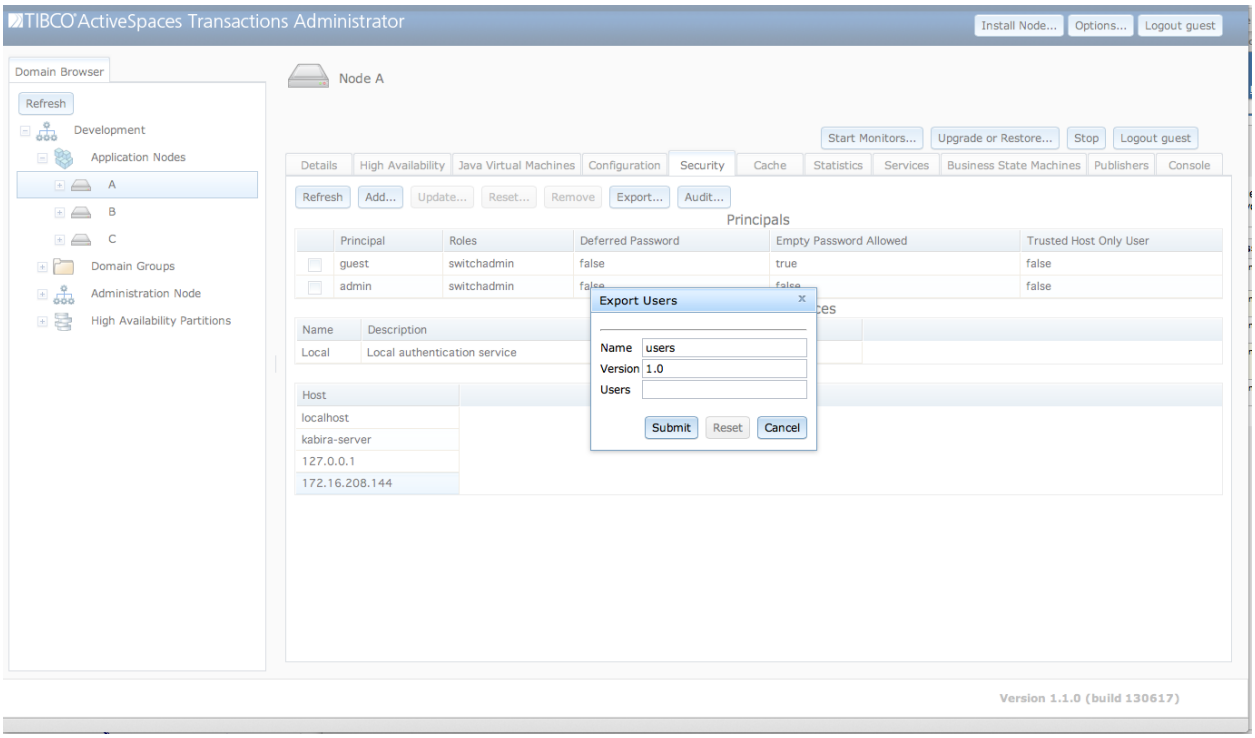

Figure 5.6. Export user configuration

The fields in the `Export Users` dialog are:

- **Name** - Configuration name used for export.
- **Version** - Configuration version used for export.
- **Users** - Optional space separated list of users to export. If specified, only the users in this list are exported.

When the `Submit` button is clicked, another window is displayed which contains the user configuration (see Figure 5.7). Notice that the credential information is encoded as an opaque value in the `opaqueCredential` field - no clear text passwords are displayed.

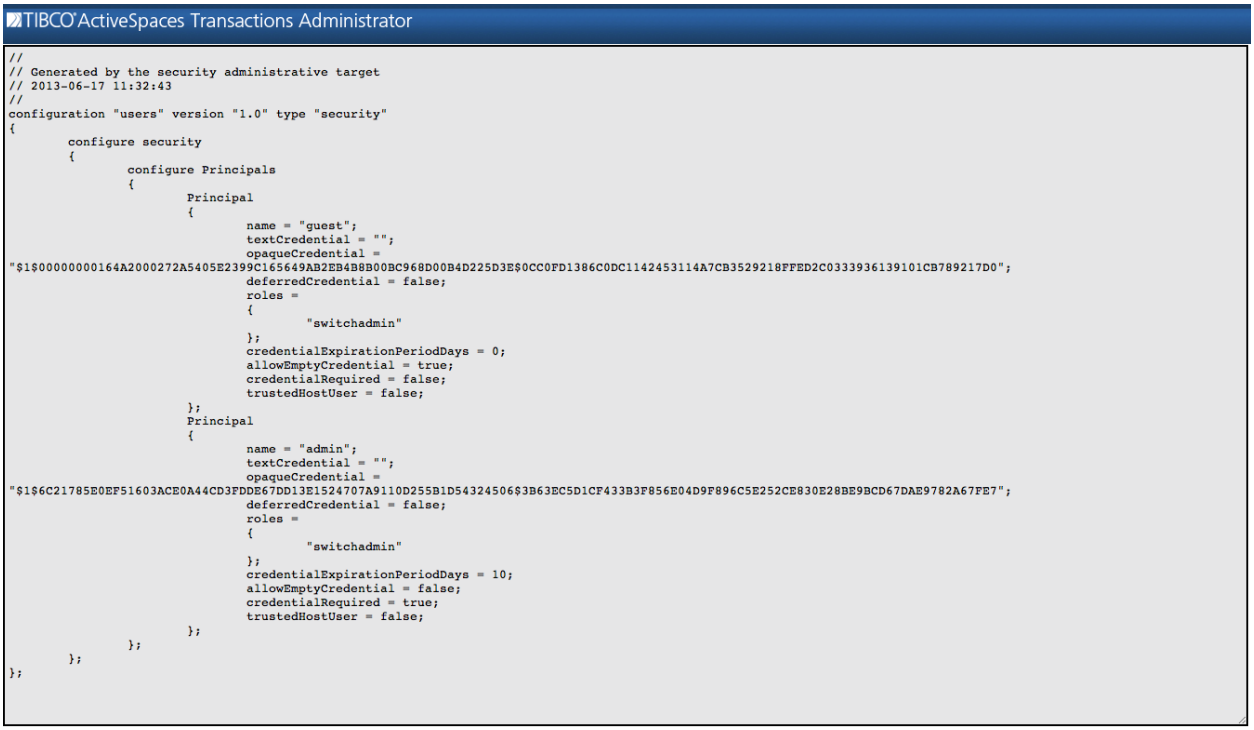

Figure 5.7. Exported user configuration

Security configuration can also be exported using this command:

```
administrator servicename=A export security name=users version=1.0
```

Reset password

Passwords are reset using the `Reset Password` dialog shown in Figure 5.8 accessed from the `Reset...` button.

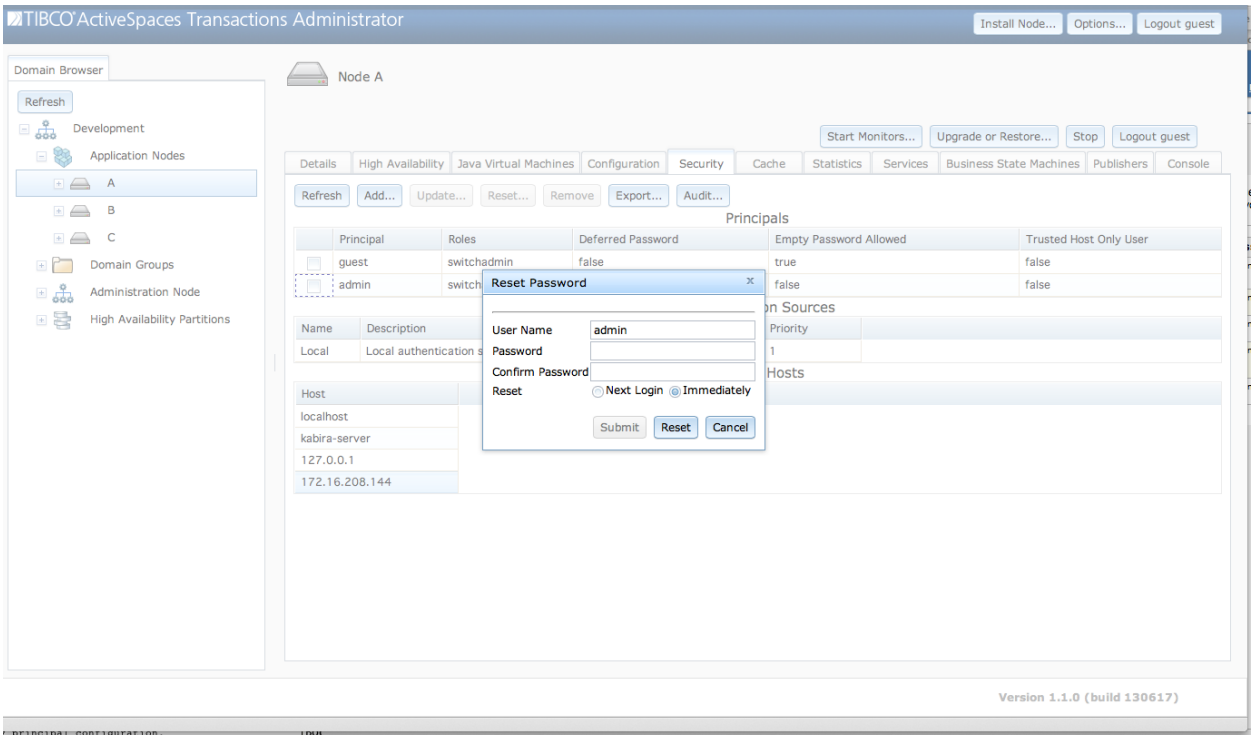

Figure 5.8. Reset password

The fields in the `Reset Password` dialog are:

- **User Name** - User name being reset. This field is read-only. It is set to the user selected in the `Principals` table.
- **Password** - New password, or empty if `Next Login` is set in `Reset`.
- **Confirm Password** - Confirm password.
- **Reset** - `Next Login` indicates that the password is set using the password provided by the user's next authentication. If `Next Login` is checked, no password can be specified in this dialog. `Immediately` indicates that the password is reset immediately. The new password must be specified in this dialog.

When the `Submit` button is clicked the password has been reset.

Passwords can also be reset using this command:

```
administrator reset security username=admin
```

Remove principal

Principals are removed from a node by selecting a principal in the `Principals` table as shown in Figure 5.9 and clicking on the `Remove` button.

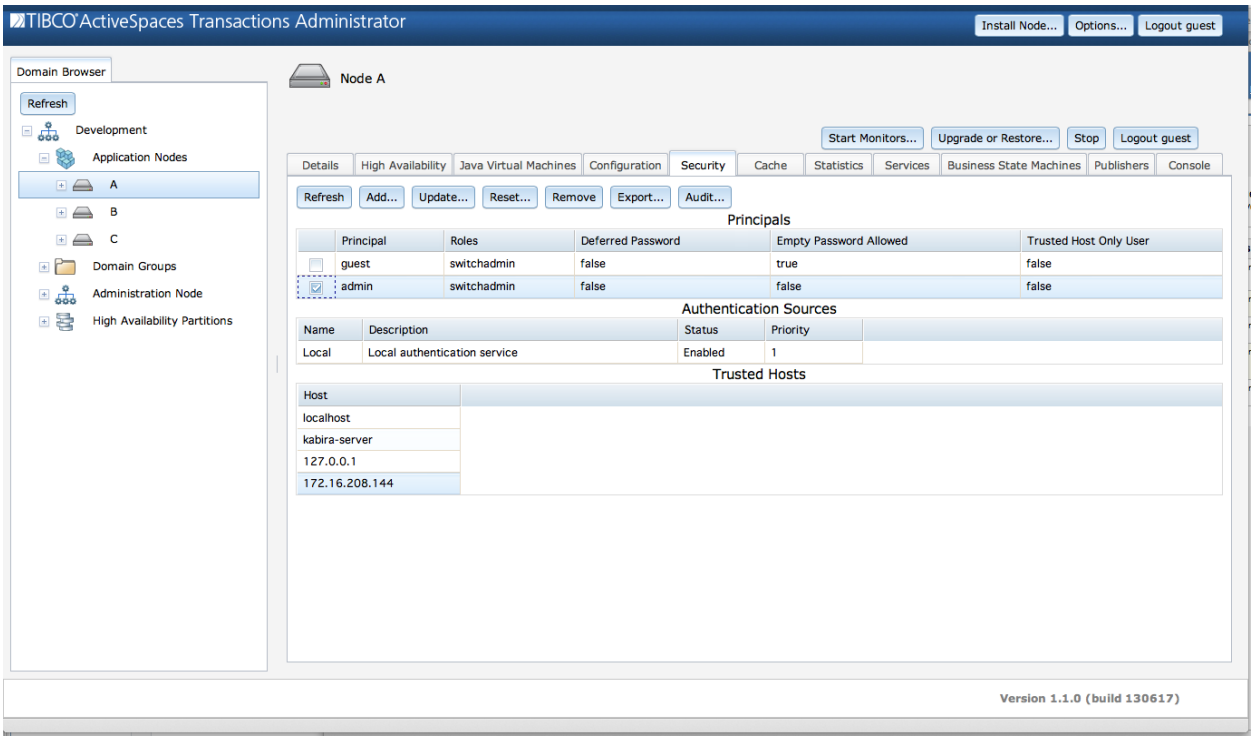

Figure 5.9. Remove principal

When the `Submit` button is clicked the principal has been removed.

Principals can also be removed using this command:

```
administrator remove security username=admin
```

Update principal

Principals are updated using the `Update Principal` dialog shown in Figure 5.10 accessed from the `Update...` button.

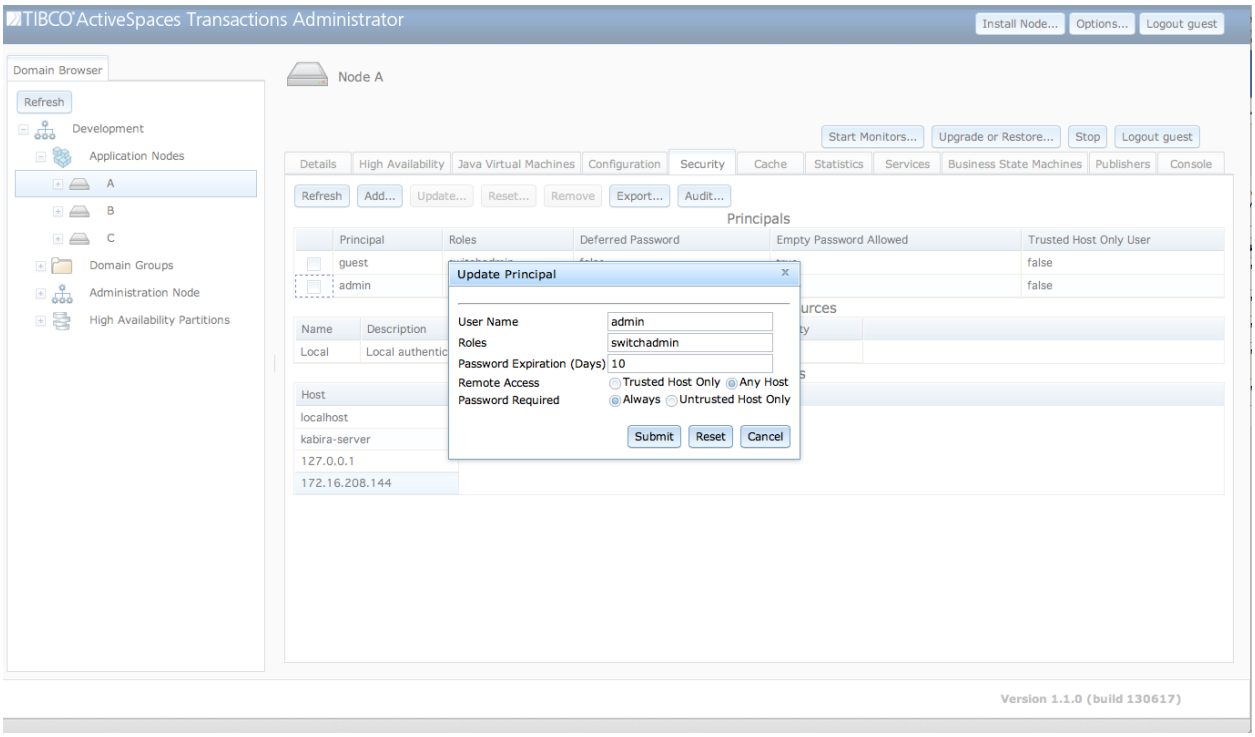

Figure 5.10. Update principal

The fields in the `Update Principal` dialog are:

- **User Name** - User name being updated. This field is read-only. It is set to the user selected in the `Principals` table.
- **Roles** - A space separated list of roles for this principal.
- **Password Expiration (Days)** - Password expiration in days. A value of zero indicates that the password does not expire.
- **Remote Access** - Control hosts from which this principal can access this node. `Trusted Host Only` indicates that this principal can only access this node from a trusted host. `Any Host` indicates that this principal can access this node from any host.
- **Password Required** - Control when this principal must provide a password to access this node. `Always` indicates that this principal must always provide a password. They cannot use the trusted host facility. `Untrusted Host Only` indicates that this principal is only required to provide a password from an untrusted host.

The fields contain the current values for the principal when the dialog is initially displayed. When the `Submit` button is clicked any changed values are updated for the principal.

Principals can also be updated using this command:

```
administrator servicename=A update security \
    username=admin roles=switchadmin \
    passwordexpirationdays=10 trustedhostuser=false passwordrequired=true
```

Configuration

Security policy configuration can be separated into these distinct areas:

- principal definition
- access control rules
- trusted hosts
- authentication sources

Security configuration has a configuration type of `security`.

Table 5.1 on page 82 defines the principal configuration parameters.

It is strongly recommended that all principal configuration be done using administrative commands instead of configuration to avoid exposing clear text passwords in configuration files.

It is illegal to activate security configuration that modifies the active principal executing the configuration command. Security configuration that modifies or removes principals should be activated by a principal which is not referenced in the configuration file.

Table 5.1. Principal configuration

Name	Type	Description
`name`	String	Principal name.
`textCredential`	String	Optional text credential. Default value is the empty string.
`opaqueCredential`	String	Optional opaque text credential. Default value is the empty string. Opaque text credentials are generated when user security configuration is exported - they should not be set manually. See the section called “Export user configuration” on page 76.
`deferredCredential`	Boolean	Optionally indicate whether credential definition be deferred until the initial authentication event. Default value is false.
`roles`	Role list	A comma separated list of roles.
`credentialExpiration-PeriodDays`	Integer	An optional expiration time for the credential in days. Default value is 0 (no expiration).
`allowEmptyCredential`	Boolean	Optionally indicate whether empty text credentials are ever allowed for this principal. Default value is true.
`credentialRequired`	Boolean	Optionally indicate whether a credential is always required. If true the principal must always present credentials during authentication, and cannot use the trusted host facility. Default value is false.
`trustedHostUser`	Boolean	Optionally indicate whether the principal may only be authenticated when connecting from a trusted host. Default value is false.

Access control configuration is done in two parts - the access rules and the rule itself.
Table 5.2 on page 83 defines the configuration values for an access rule.

Table 5.2. Access control rule configuration

Name	Type	Description
roleName	String	Name of role associated with rule.
permission	Enumeration - Execute or AccessAllOperationsAndAttributes.	Define the granted permissions for the access control rule. Execute can only be specified on administrative commands. AccessAllOperationsAndAttributes can only be specified on administrative targets.

Table 5.3 on page 83 defines the configuration values for a rule.

Table 5.3. Rule configuration

Name	Type	Description
name	String	Class name. This is a fully scoped class name of the administrative target or command being protected.
lockAllElements	Boolean	A value of true disables all unauthenticated access to administrative commands in the target. A value of false enables all access. Default value is false.
accessRules	Array of access rules.	The access control rules for the type defined in this rule.

Table 5.4 on page 83 defines the trusted host configuration parameters.

Table 5.4. Trusted hosts configuration

Name	Type	Description
name	String	Trusted host name. Either a fully-qualified domain name, or a simple name.

Table 5.5 on page 83 defines the authentication source configuration parameters.

Table 5.5. Authentication source configuration

Name	Type	Description
sourceList	Array of source names.	An array of source names in priority order. No default value.
name	String	Unique source name in sourceList array. The source name must match an available authentication source. The name of the node local authentication source is Local. See Example 5.1 on page 66 for an example. No default value.

6

Distribution and High Availability

This chapter describes configuring and controlling distribution and high availability.

Distribution status

The current distribution status is available on the `Distribution Status` section of the `High Availability` tab for a node (see Figure 6.1). The information in this display is:

- **Node Name** - local node name.
- **Distribution State** - the current state of distribution on the local node (see the section called "Distribution states" on page 86 for details).
- **Quorum State** - current quorum state of the local node (see **TIBCO ActiveSpaces® Transactions Architect's Guide** for details).
- **Quorum Description** - description of current quorum status.
- **Number of Active Nodes** - current number of active remote nodes. An active remote node has connections established.
- **Number of Discovered Nodes** - total number of discovered remote nodes. These nodes may have been discovered using dynamic or static discovery.
- **Number of Undiscovered Nodes** - number of remote nodes that have never been discovered. These nodes are configured to use static discovery and have never had an active connection.
- **Number of Connections to Remote Nodes** - total number of connections to all active remote nodes.
- **Number of Type Mismatches** - total number of type mismatches detected from remote nodes.
- **Location Code** - location code of local node.

- **Calculated CRC of System Types** - Calculated check-sum of system types. Nodes with the same calculated value can communicate successfully. This value may change between releases.

This information can also be displayed using:

```
administrator servicename=A display cluster type=local
```

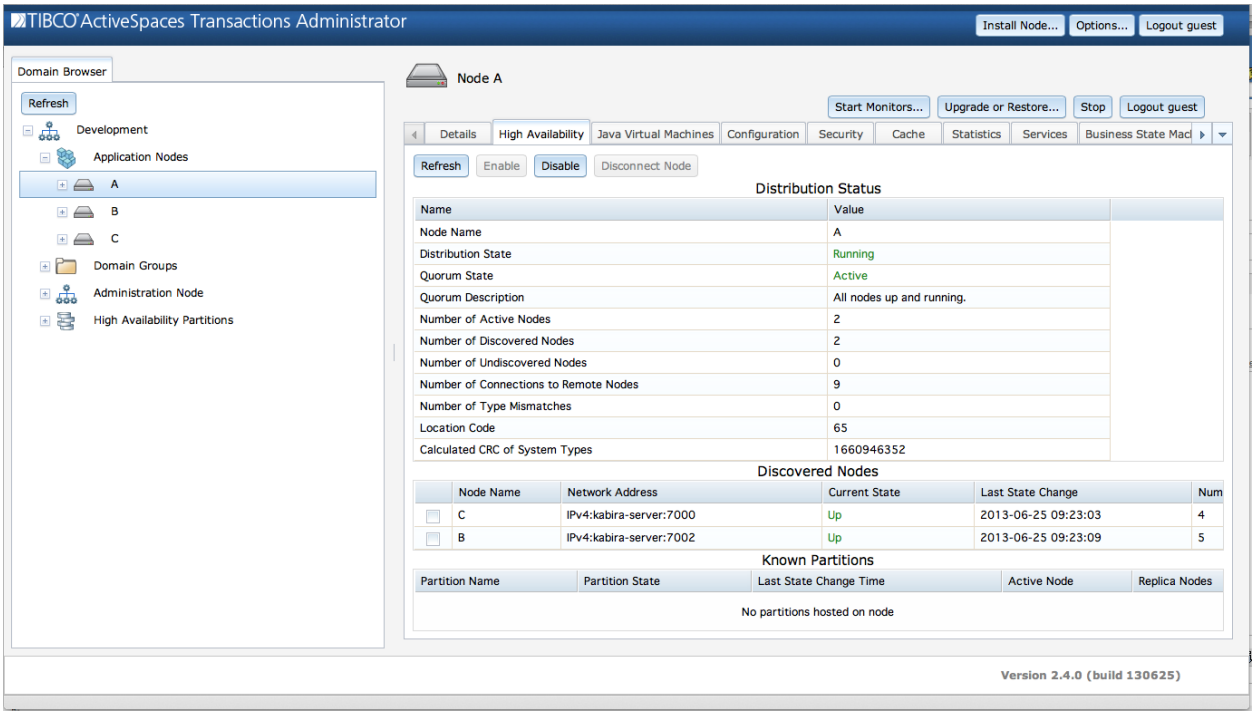

Figure 6.1. Distribution status

Starting and stopping distribution

Distribution services are hosted in the first application JVM that is started. They are stopped when this JVM is stopped. In addition, a distribution configuration must be loaded and activated to enable distribution. Deactivating the distribution configuration terminates distribution, even if there is a running application JVM in which they could be hosted. See the section called “Configuration” on page 109 for details on distribution configuration.

The current distribution status is available on the `Distribution Status` section of the `High Availability` tab for a node (see Figure 6.1).

Stopping distribution on an active node causes all high-availability services on the node to be disabled. This causes any partitions active on the node to failover.

Distribution states

Distribution can be in one of the following states:

Table 6.1. Distribution states

State	Description

`Stopped`	Distribution is stopped either because there are no active application JVMs to host it, or distribution configuration has been deactivated. There is no active connectivity to any remote nodes.
`Running`	Distribution is active on this node. There is active connectivity to all discovered nodes.

Figure 6.2 shows the state machine that controls the transitions between the distribution states.

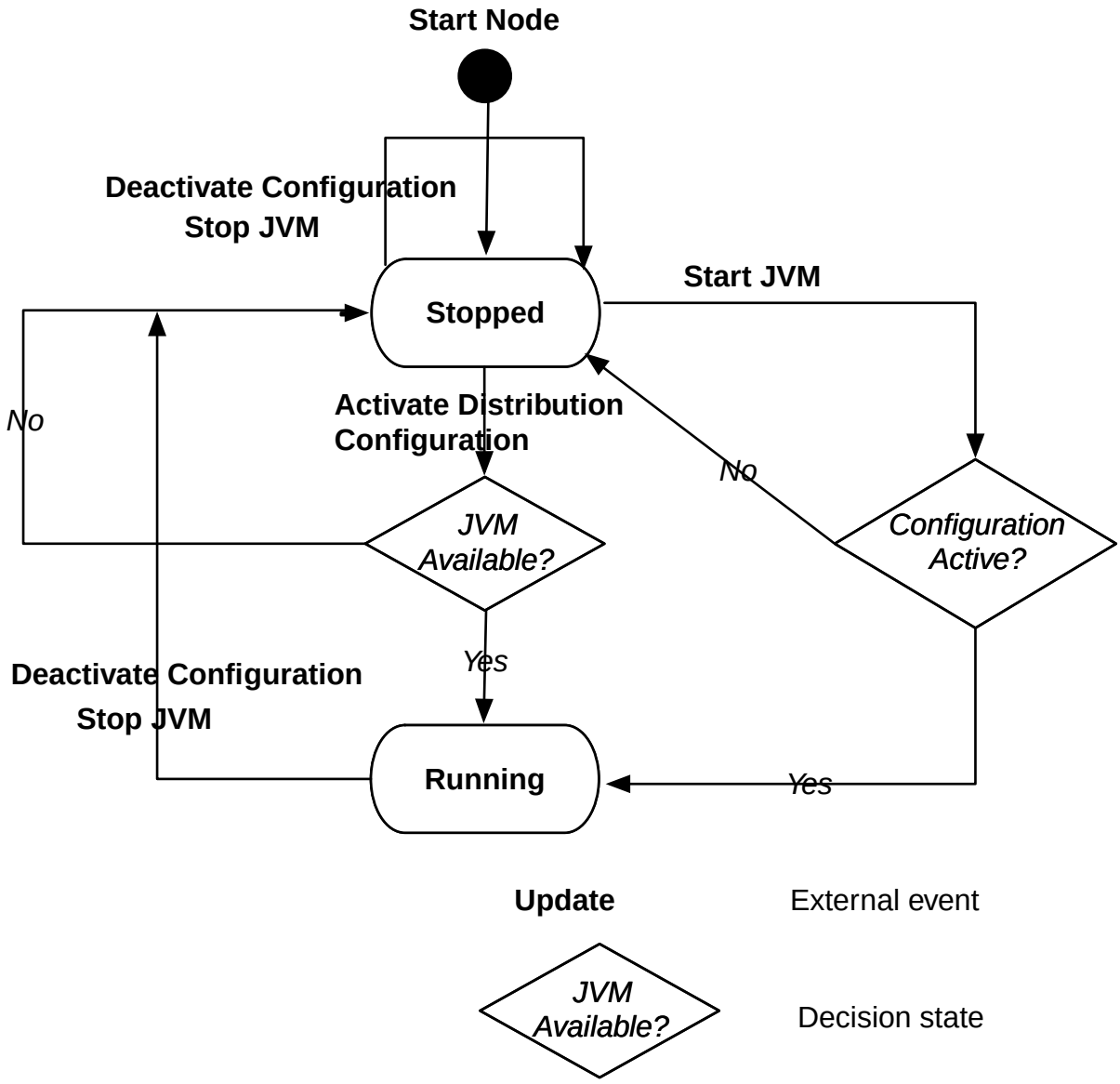

Figure 6.2. Distribution states

Node connectivity

Connectivity between nodes is established by either dynamic or static discovery. Each node in a cluster must agree on the dynamic and static discovery values for node discovery to work correctly.

- **dynamic discovery** - all nodes in the cluster must have both dyanmic discovery and service discovery enabled and use the same discovery network address information. Dyanmic discovery is enabled in the distribution configuration using the `DynamicDiscovery.enabled` configuration value (see Table 6.5 on page 113). The service discovery values are set at node installation. See the section called “Installation” on page 31 for details.

- **static discovery** - each node must define a `StaticDiscovery.RemoteNode` configuration block for all remote nodes to which it will be communicating. See Table 6.6 on page 113 for details.

Connectivity between nodes can be verified on the `Discovered Nodes` section of the `High Availability` tab for a node (see Figure 6.1). For each remote node that has been discovered this information is displayed:

- **Node Name** - remote node name.

- **Network Address** - network address used to connect to node.

- **Current State** - current state of remote node.

- **Last State Change** - the time of the last state change.

- **Number of Connections** - current number of active TCP connections to the node.

- **Number of Queued PDUs** - current number of protocol data units queued for this remote node when using asynchronous replication. This value is always 0 if synchronous replication is being used.

- **Discovered** - how the node was discovered.

- **Location Code** - internally generated location identifier for this node.

This information can also be displayed using:

```
administrator servicename=A display cluster type=remote
```

Node discovery

Operators can force distribution to attempt to discover remote nodes using the `Discover...` button on the `High Availability` tab as shown in Figure 6.3. Clicking on the `Discover` button in the dialog attempts to establish connectivity to the specified nodes.

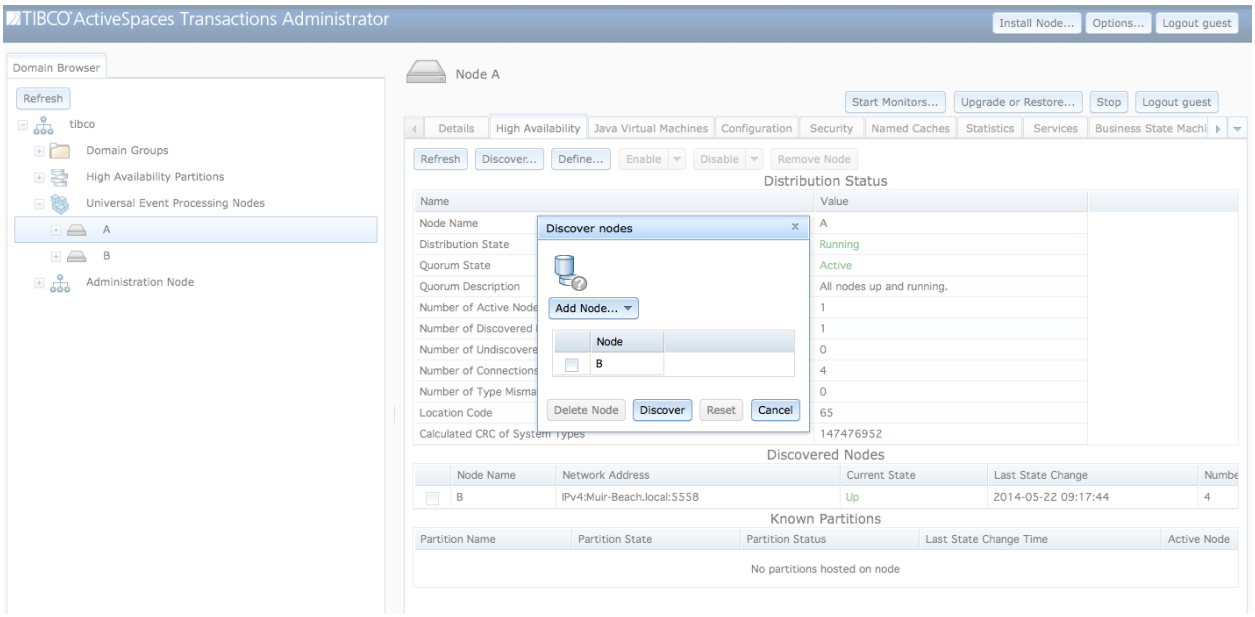

Figure 6.3. Discovering nodes

Nodes can also be discovered using this command:

```
administrator servicename=A discover cluster nodes=B,C
```

Discovered nodes in the `Down` state can be removed by selecting a node in the `Discovered Nodes` section and clicking on the **Remove Node** button. This is useful when connectivity has been lost to a remote node, or the node has been permanently removed from the cluster.

Nodes can also be removed using this command:

```
administrator servicename=A removenode distribution remotenode=B
```

Partition status

TIBCO ActiveSpaces® Transactions Administrator provides these different views of partition status:

- Cluster wide summary
- Details on a specific partition
- Node summary

Figure 6.4 shows the cluster partition summary screen that is accessed from the `High Availability Partitions` folder in the `Domain Browser`. This screen provides a summary of all partitions that are defined in the cluster for every node managed by the Domain Manager. A row is displayed for each node with that node's view of the partition. The fields in the `Partitions` table are:

- **Reporting Node** - the node name that is reporting the information on the partition.
- **Partition Name** - the partition name.
- **Partition State** - the state of the partition on the node in the `Reporting Node` field.

- **Partition Status** - the status of the partition on the node in the `Reporting Node` field.
- **Last State Change Time** - the last time the partition state changed on the node in the `Node Name` field.
- **Active Node** - the active node for the partition from the perspective of the node in the `Node Name` field.
- **Replica Nodes** - the replica nodes for the partition from the perspective of the node in the `Node Name` field.
- **Replicate To Existing** - defines whether replication occurs to active replica nodes during a migration (`true`) or not (`false`).
- **Object Batch Size** - number of objects locked during a migration or update of this partition.
- **Number of Threads** - number of threads used during partition migration.
- **Restore From Node** - node partition should be restored from in a multi-master scenario.
- **Mapped Types** - list of types mapped into this partition.
- **Broadcast Definition Updates** - Broadcast changes in partition state to all nodes in the cluster.
- **Sparse Audit Option** - Audit node list if a sparse partition.
- **Remote Enable Action** - Control remote enabling of the partition.

This information can also be displayed using this command:

```
administrator servicename=domainmanager domainname=Development display partition
```

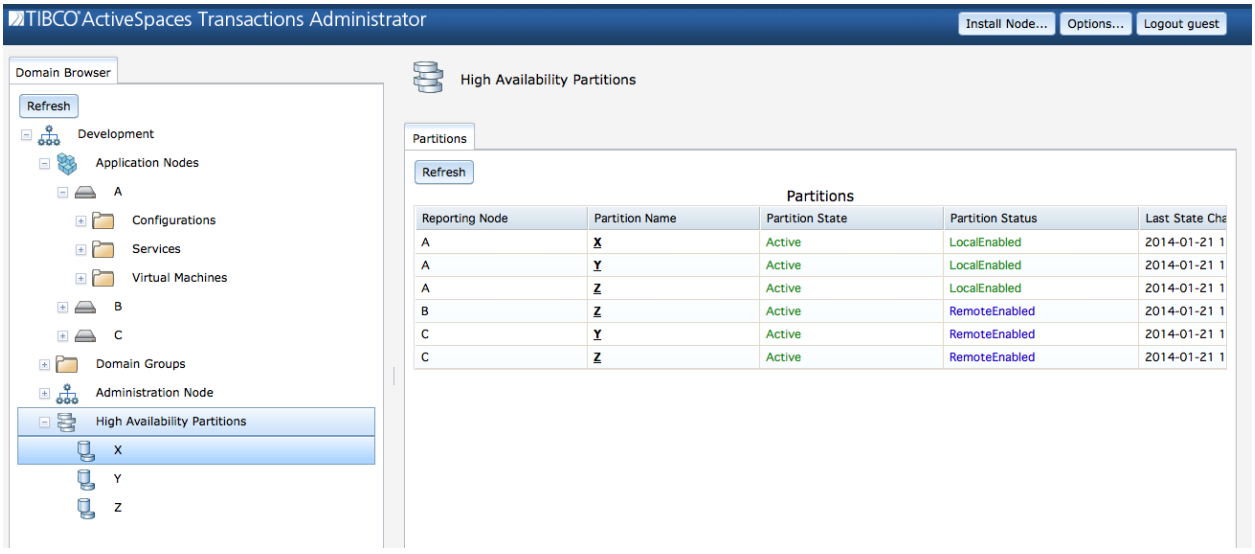

Figure 6.4. Cluster partition summary

Figure 6.5 shows the partition summary screen that is accessed by selecting a specific partition under the `High Availability Partitions` folder in the `Domain Browser`. This screen provides a summary of a specific partition defined in the cluster. The fields in the `Active Node` section of this screen are:

- **Node Name** - the node reporting this information.
- **Active Node** - the active node for the partition.
- **Partition State** - the state of the partition on the active node.
- **Partition Status** - the status of the partition on the active node.
- **Last State Change Time** - the last time the partition state changed on the active node.
- **Replicate To Existing** - defines whether replication occurs to active replica nodes during a migration (`true`) or not (`false`).
- **Object Batch Size** - number of objects locked during a migration or update of this partition.
- **Number of Threads** - number of threads used during partition migration.
- **Restore from Node** - node partition should be restored from in a multi-master scenario.
- **Mapped Types** - list of types mapped into this partition.
- **Broadcast Definition Updates** - whether partition definition updates should be broadcast to other nodes in the cluster.
- **Sparse Audit Option** - the sparse audit option value.
- **Replica Audit Option** - the replica audit option value.
- **Remote Enable Action** - the remote partition enable option value.

The `Replica Nodes` section of this screen has a row for each replica node defined for the partition. The fields in this section of the screen are:

- **Node Name** - this node is a replica node for the partition.
- **Partition State** - the state of the partition on the replica node.
- **Partition Status** - the status of the partition on the replica node.
- **Type** - indicates whether the partition uses `Asynchronous` or `Synchronous` replication to this node.
- **Last State Change Time** - the last time the partition state changed on the replica node.

The information in both of these sections can also be obtained using this command:

```
administrator servicename=domainmanager domainname=Development display partition name="Z"
```

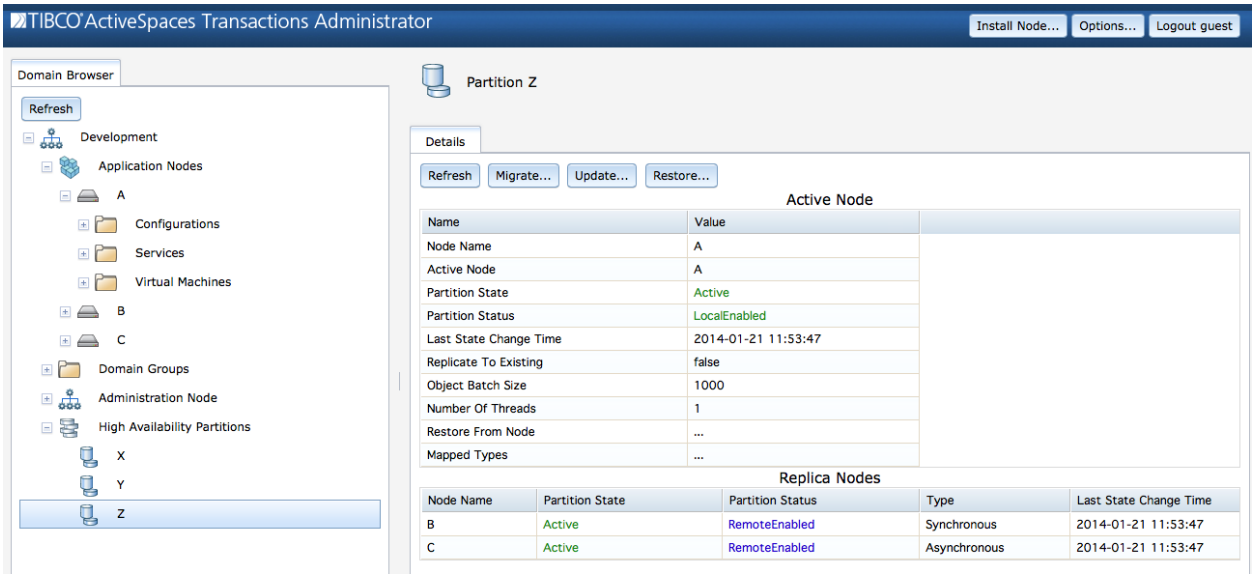

Figure 6.5. Partition summary

Figure 6.6 shows the node partition summary screen that is accessed from the `High Availability` tab in the node display. This screen provides a summary of all partitions that are known by the node in the `Known Partitions` section. A row is displayed for each partition known by a node. A partition is known by a node when the node is the active or replica node for a partition, or the partition was previously an active or replica node, but the partition was migrated off of the node.

The fields in the `Known Partitions` section are:

- **Partition Name** - the partition name.
- **Partition State** - the state of the partition on this node.
- **Partition Status** - the status of the partition on this node.
- **Last State Change Time** - the last time the partition state changed on this node.
- **Active Node** - the active node for the partition from the perspective of this node.
- **Replica Nodes** - the replica nodes for the partition from the perspective of this node.
- **Replicate To Existing** - defines whether replication occurs to active replica nodes during a migration (`true`) or not (`false`).
- **Object Batch Size** - number of objects locked during a migration or update of this partition.
- **Number of Threads** - number of threads used during partition migration.
- **Restore From Node** - node partition should be restored from in a multi-master scenario.
- **Mapped Types** - list of types mapped into this partition.
- **Broadcast Definition Updates** - whether partition definition updates should be broadcast to other nodes in the cluster.

- **Sparse Audit Option** - the sparse audit option value.
- **Replica Audit Option** - the replica audit option value.
- **Remote Enable Action** - the remote partition enable option value.

This information is also available using this command:

```
administrator servicename=A display partition name=Z
```

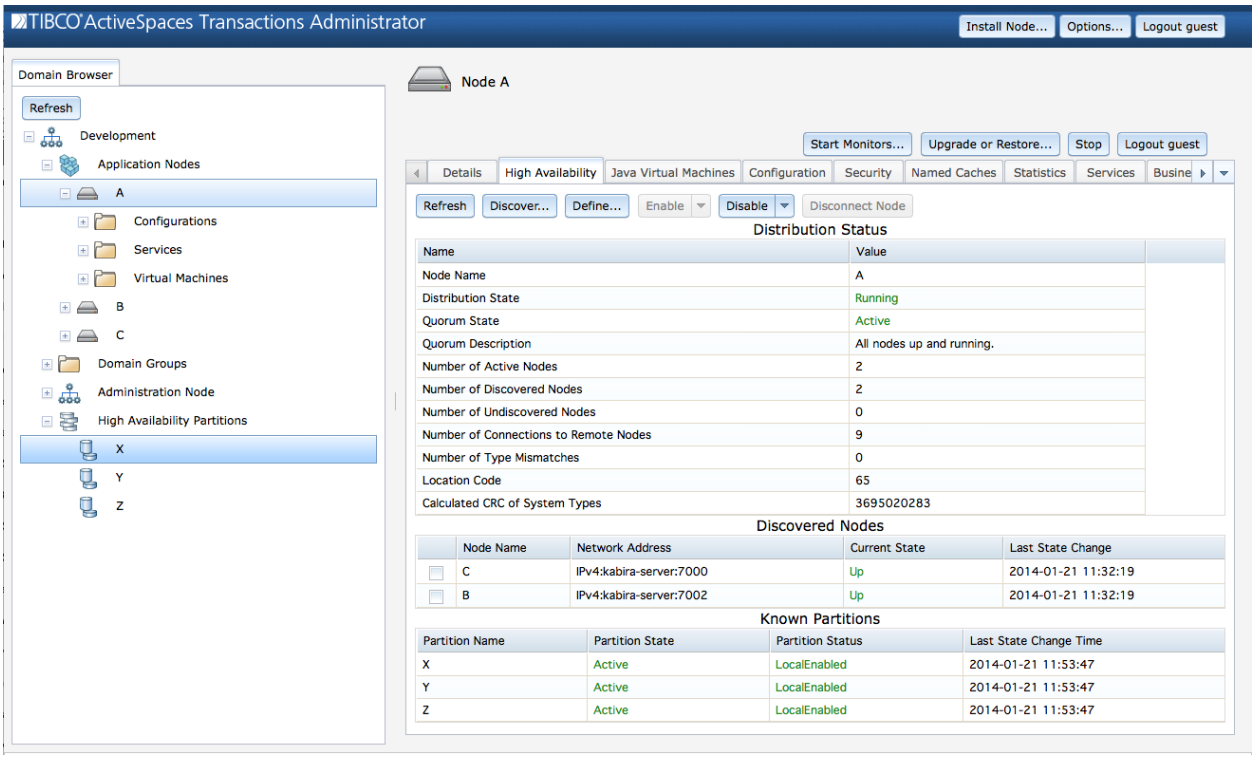

Figure 6.6. Partition node summary

Adding a node to a cluster

The following steps are required to add a node to a cluster that will host partitions.

1. Install the node.
2. Start the node.
3. Deploy the application.
4. Discover all nodes in the partition definitions that will be defined on this node.
5. Define all partitions that will be hosted on this node - active and replicas.
6. Enable the partitions on the node.
7. Enable application requests.

The rest of this section describes how to define and enable partitions on the new node.

Partitions can be automatically defined by the application or manually by the administrator. Figure 6.7 shows the definition of partition XX with these properties:

- **Active Node** - An active node of A.
- **Replica Nodes** - `Do Not Synchronize` replicas when partition is enabled.
- **Objects per Transaction** - Number of objects to migrate per transaction.
- **Number of Threads** - Number of threads to use during object migration.
- **Restore from Node** - Restore from this node following a multi-master scenario.
- **Broadcast Updates** - Broadcast partition updates to all discovered nodes.
- **Sparse Audit Node List** - Verify node list if a sparse partition. This is not a sparse partition, so this property is ignored.
- **Replica Audit** - Wait for the replica node to be active.
- **Remote Enable Action** - A remote enable partition command will enable this partition.

The partition has a single replica node B that uses `Synchronous` replication.

The partition definition dialog is accessed from the node screen on which the partition should be defined.

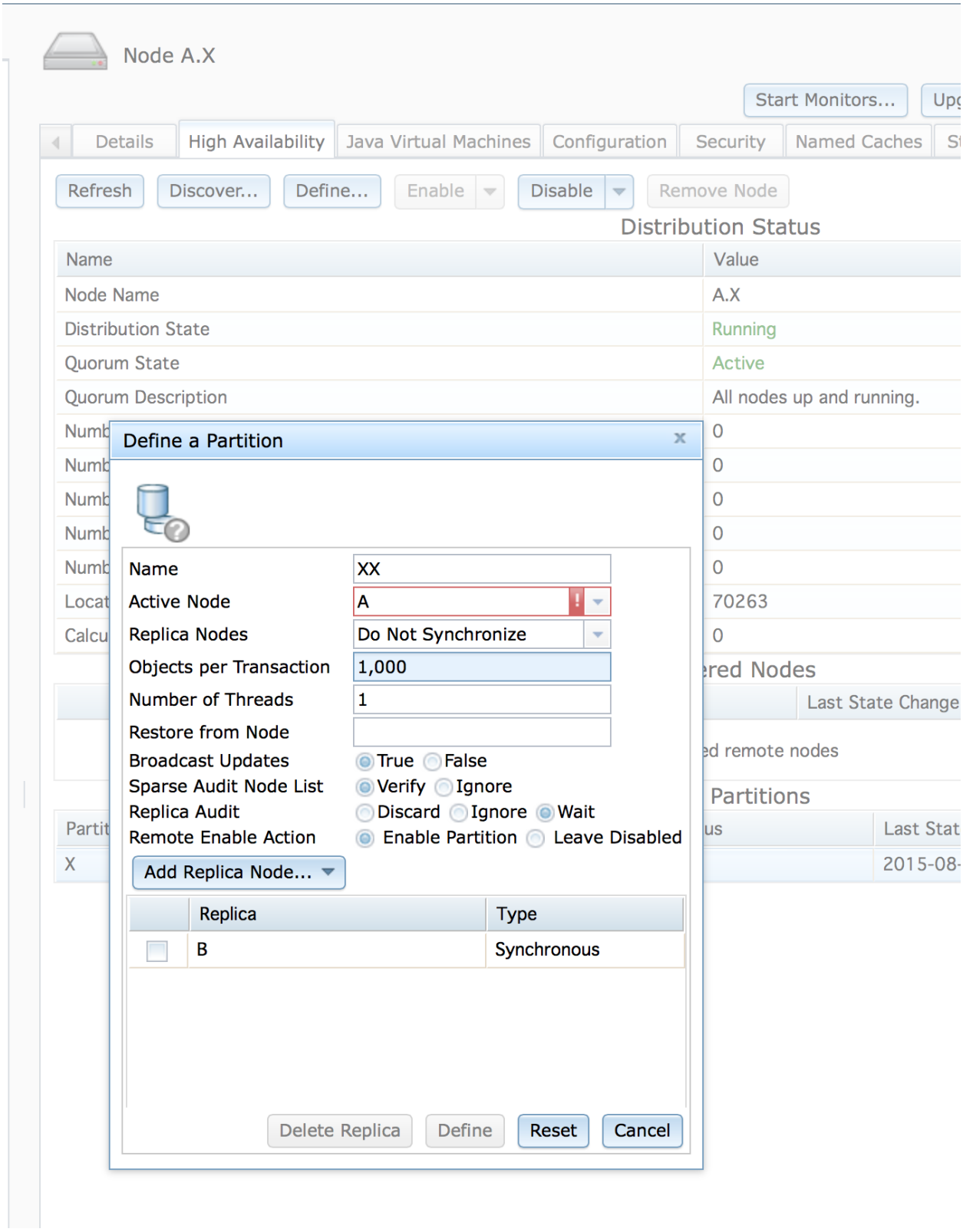

Figure 6.7. Defining a partition

When the partition definition is completed, the partition is in an `Initial` state on the active node as shown in Figure 6.8.

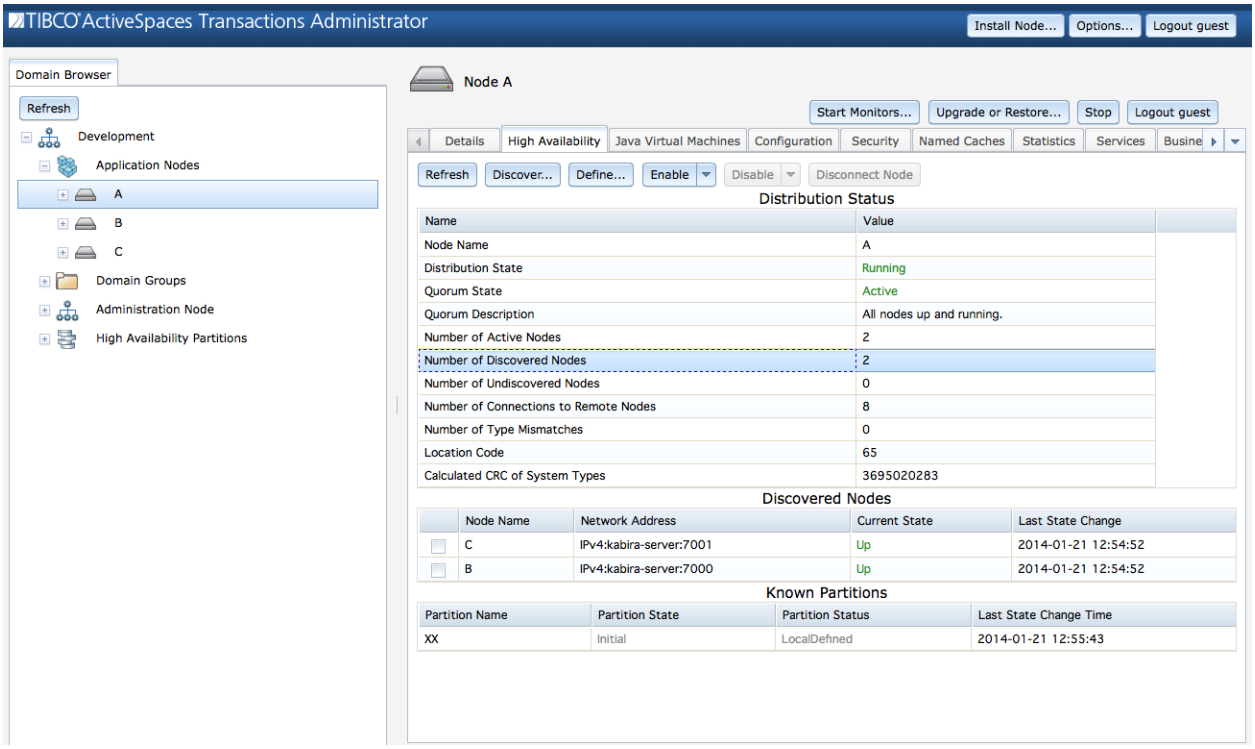

Figure 6.8. Initial partition state

The `Enable` button on the `High Availability` tab is used to change the partition state to `Active`. When the partition state transitions to `Active` all object migrations required to enable the partition have completed. At this point any partitions defined on the node are now active in the cluster with objects being maintained between the active node and any defined replica nodes. Figure 6.9 shows partition XX as `Active` on node A.

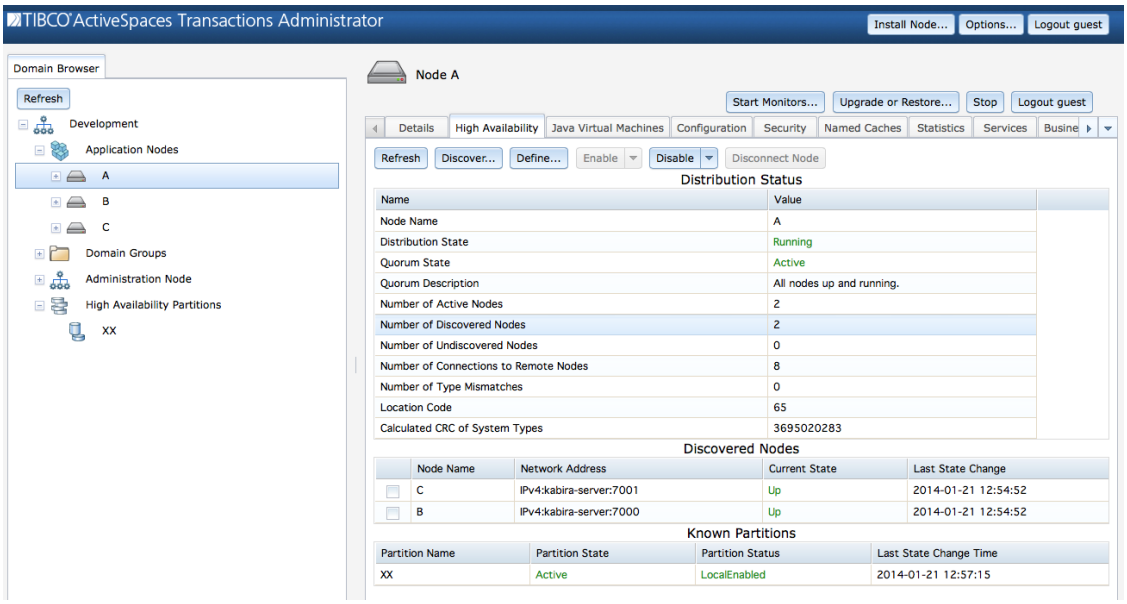

Figure 6.9. Partitions on active node

These commands can also be used to perform the same functionality:

```
administrator servicename=A define partition name=XX activenode=A replicas=B
administrator servicename=A join cluster
```

Figure 6.10 shows partition XX on the replica node B. The partition is also `Active` on replica node B.

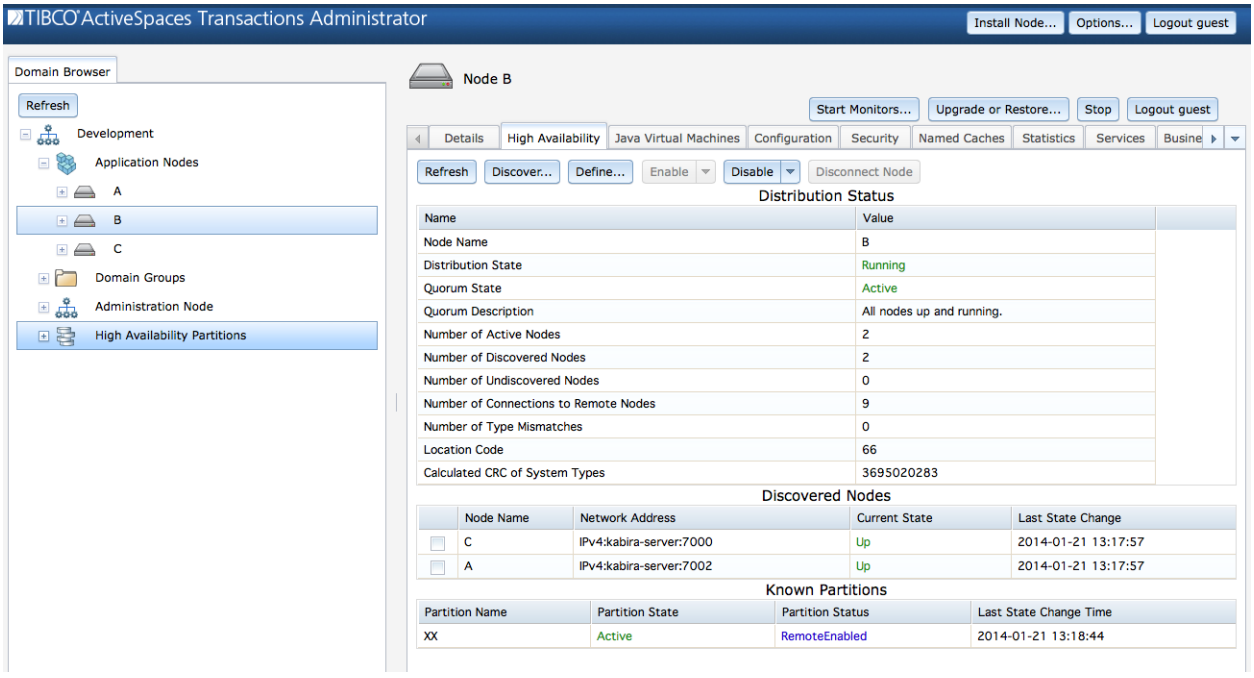

Figure 6.10. Partition on replica node

The partition information for partition XX on node B can also be displayed using this command:

```
administrator servicename=B display partition name="XX"
```

Adding a partition to a node

Partitions can be dynamically added to nodes either by the application or by the administrator. The steps to add a new partition to an active node are identical to the steps defined in the section called “Adding a node to a cluster” on page 93 to add a new node to the cluster, specifically:

1. Define the partition as show in Figure 6.7.
2. Enable the partition on the active node using the `Enable` button on the `High Availability` tab as shown in Figure 6.11.

These commands can also be used to define and enable a new partition:

```
administrator servicename=A define partition name="YY" activenode=A replicas=C
administrator servicename=A join cluster
```

Notice that the `Partition State` for the new partition is `Initial` (see Figure 6.11). The `Initial` partition state indicates that an enable needs to be done on this node to activate the new partition.

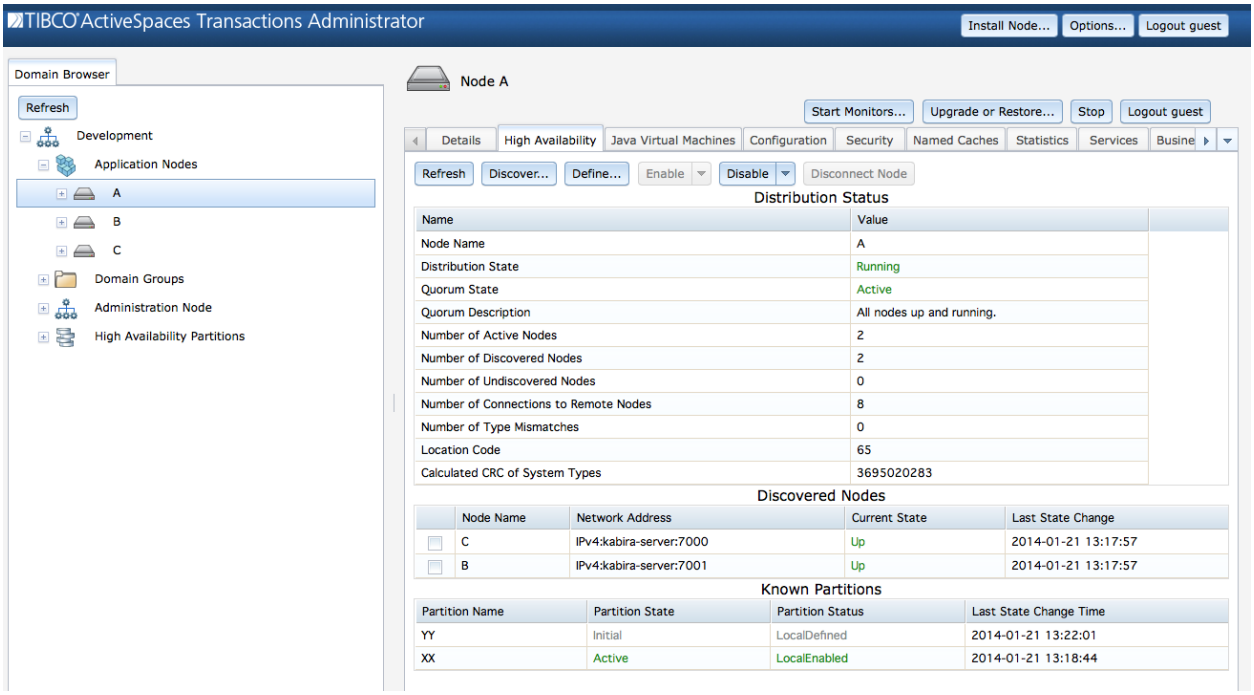

Figure 6.11. Adding a new partition

Migrating partitions

Partition definitions can be changed at any time on a running system without impacting application availability. These changes include:

- Changing the active node of a partition.
- Adding or removing replica nodes.
- Changing the replication type of replica nodes.

This class of changes is called *partition migration* because they may cause object data to be moved between nodes. The rest of this section shows an example of adding a new replica node to a partition to demonstrate partition migration.

The following steps are required to add a new replica node to an existing partition.

1. Migrate the partition to add the replica node.

Partitions can be automatically modified by the application or manually by the administrator. Figure 6.12 shows the definition of `Partition 1` being migrated to add a new asynchronous replica node `C`. The partition migration dialog is accessed from the partition details screen. In addition to changing the active and replica nodes for a partition, these partition properties can be set in the migration partition dialog:

- **Replica Nodes** - Whether to force synchronization or not of replica nodes during a migration.
- **Objects per Transaction** - Number of objects to migrate per transaction.
- **Number of Threads** - Number of threads to use for object migration.
- **Sparse Audit Node List** - Verify node list if migrating a sparse partition.
- **Replica Audit** - Replica node audit to be used during the migration.

These partitions properties are only used for this migration. They do not override the partition properties set when the partition was defined.

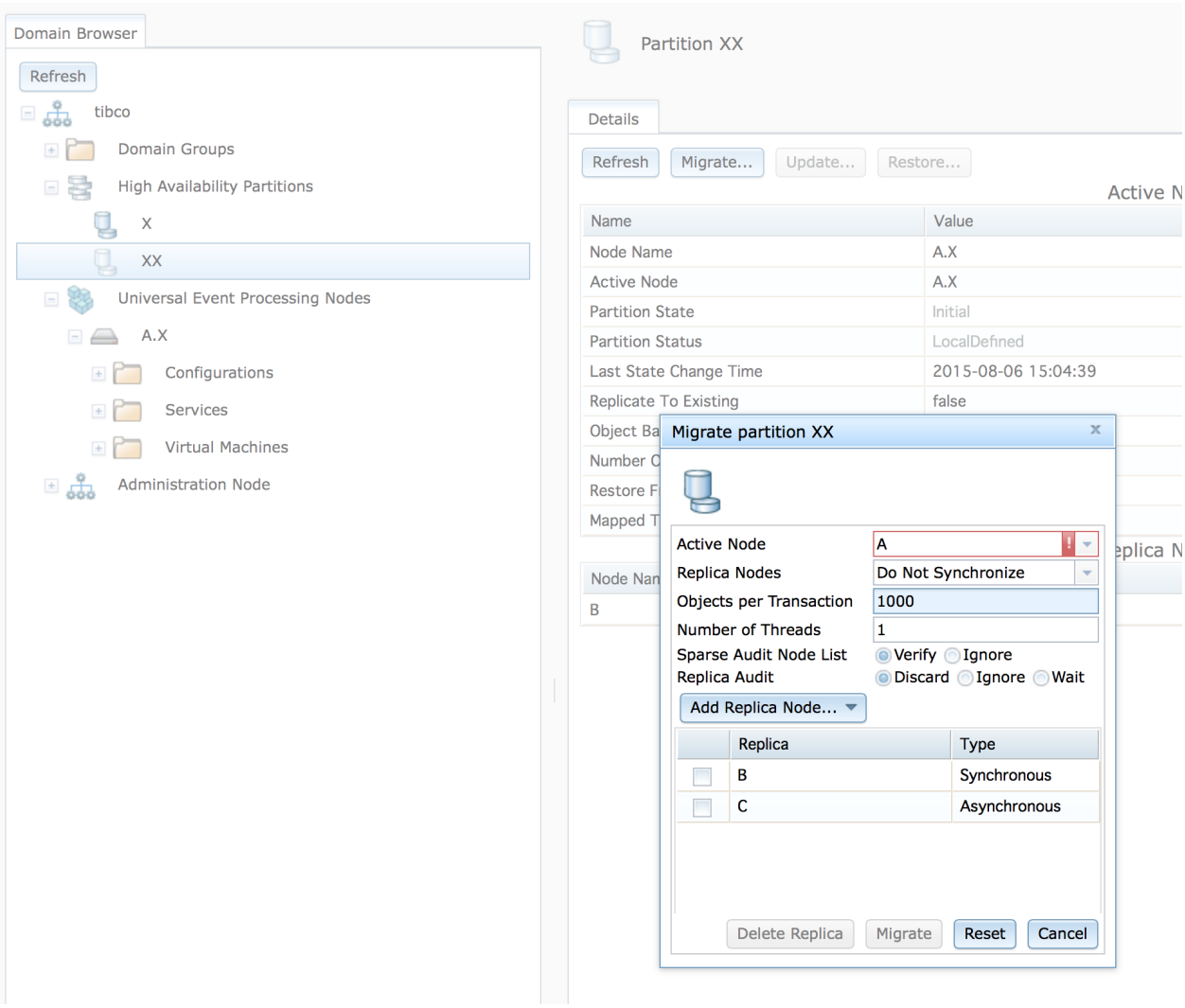

Figure 6.12. Migrating a partition

When the dialog is submitted, the partition definition is updated in the cluster and object data is copied from the current active node (A) to the new replica C.

Partition migration can also be triggered using this command. This command must be executed on the current active node for the partition.

```
administrator servicename=A migrate partition name="XX" activenode=A replicas=B,C
replicatypes=synchronous,asynchronous
```

Removing a node from service

Nodes can be gracefully removed from a cluster by disabling all partitions on the node. When a node is disabled all active partitions on the node failover to the next highest priority replica and all replication to the node is terminated. If there is no replica node defined for a partition that is active on the node being disabled, the partition is abandoned after confirmation from the administrator.

The following steps are required to remove a node from a cluster:

1. Disable all active partitions.

Figure 6.13 shows partition XX as `Unavailable` after being disabled on node C. Partitions can be disabled in two ways:

- **Normal** - this is the default. Disable will fail if any partitions would be abandoned.
- **Force** - Disable will succeed, even if partitions would be abandoned.

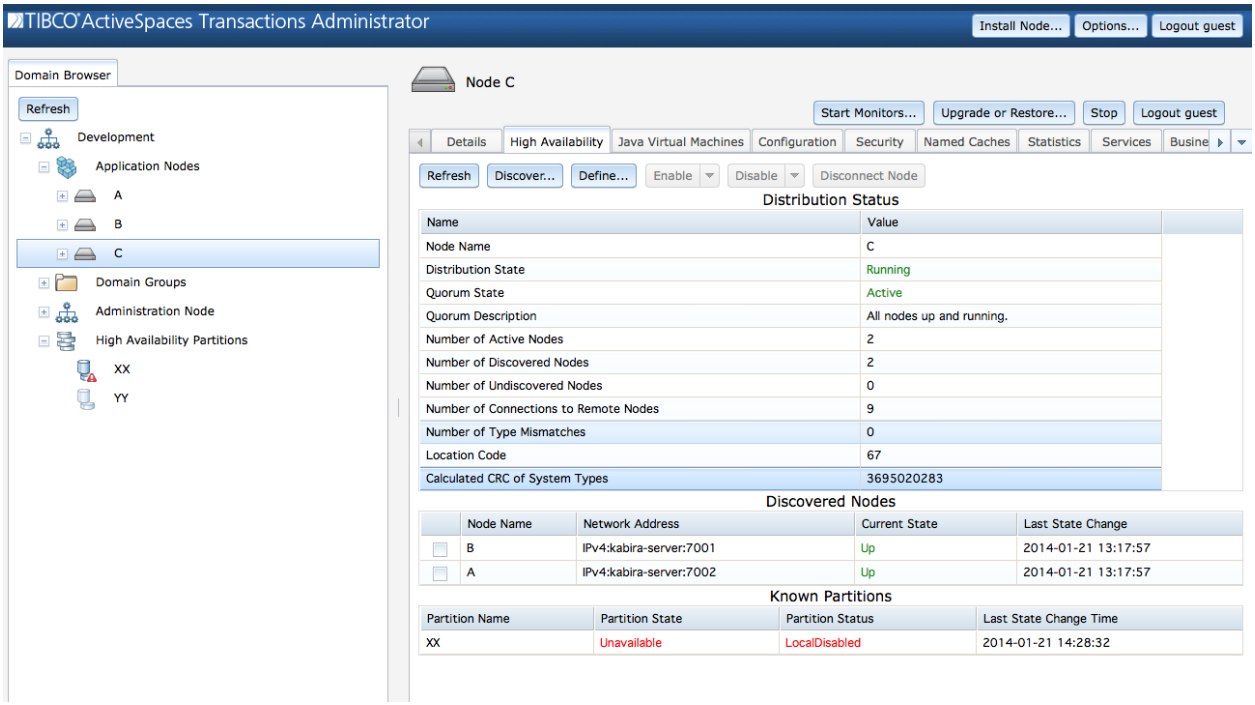

Figure 6.13. Disabling partitions on a node

A node can also leave a cluster using this command.

```
administrator servicename=C leave cluster
```

Replacing one node with another

Replacing one node with another requires the following steps:

1. Migrate the partition definition from the node being taken out of service to the new node.
2. Disable the node being taken out of service.

Figure 6.14 shows the migrate partition dialog with node A replaced with node C. When this dialog is submitted, all of the objects in partition XX are migrated from node A to node C, and the partition's new active node is C.

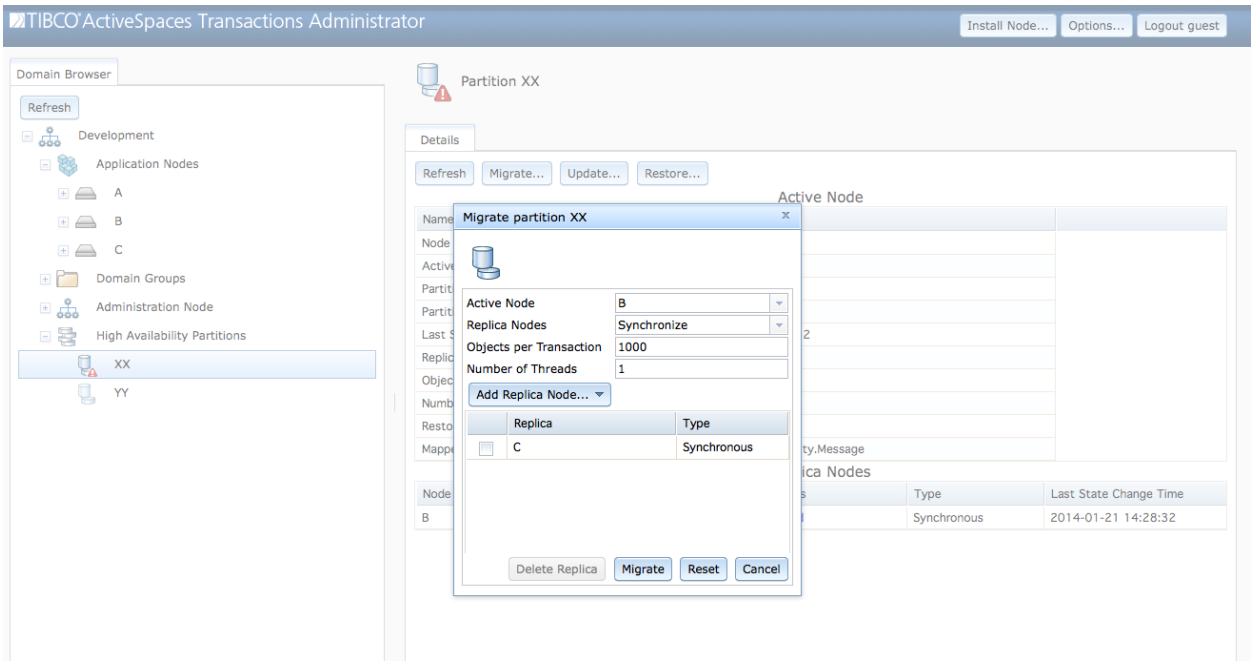

Figure 6.14. Replacing a node

This can also be accomplished with this command. This command must be executed on the current active node for the partition - node A in this example.

```
administrator servicename=A migrate partition name="XX" activenode=C replicas=B
```

After the partition migration has completed, the partition has successfully migrated from node A to node C as seen in Figure 6.15.

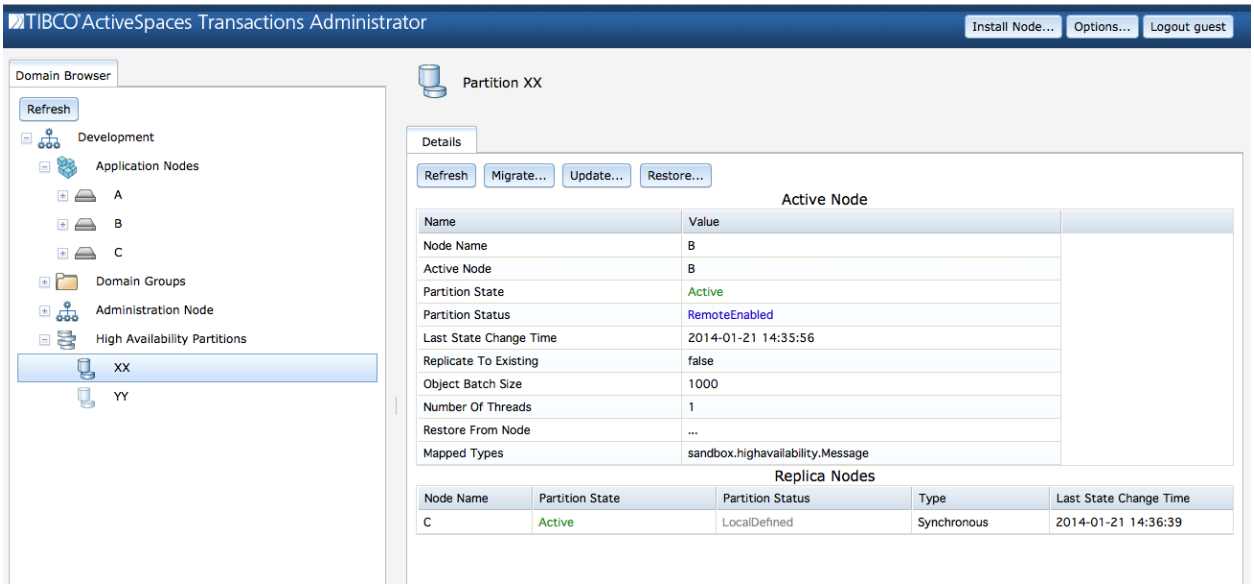

Figure 6.15. Migrated partition

After the partition migration is successful, the node can be taking out of service as described in the section called “Removing a node from service” on page 100.

Restoring a node to service

To bring a node back in service following a failure, it must be restored. The steps required to restore a node to service are identical to adding a new node to the cluster. See the section called “Adding a node to a cluster” on page 93 for details.

Updating partitions

Object to partition mapping can be dynamically updated on a running system. This provides a mechanism for reallocating work among nodes in a cluster.

Applications must specifically support dynamic partition updates. Not all applications will support this feature.

Object to partition mapping is done for a specific partition. The actual effect of the update is application specific. The `Update` button is used to dynamically update the object to partition mapping as shown in Figure 6.16.

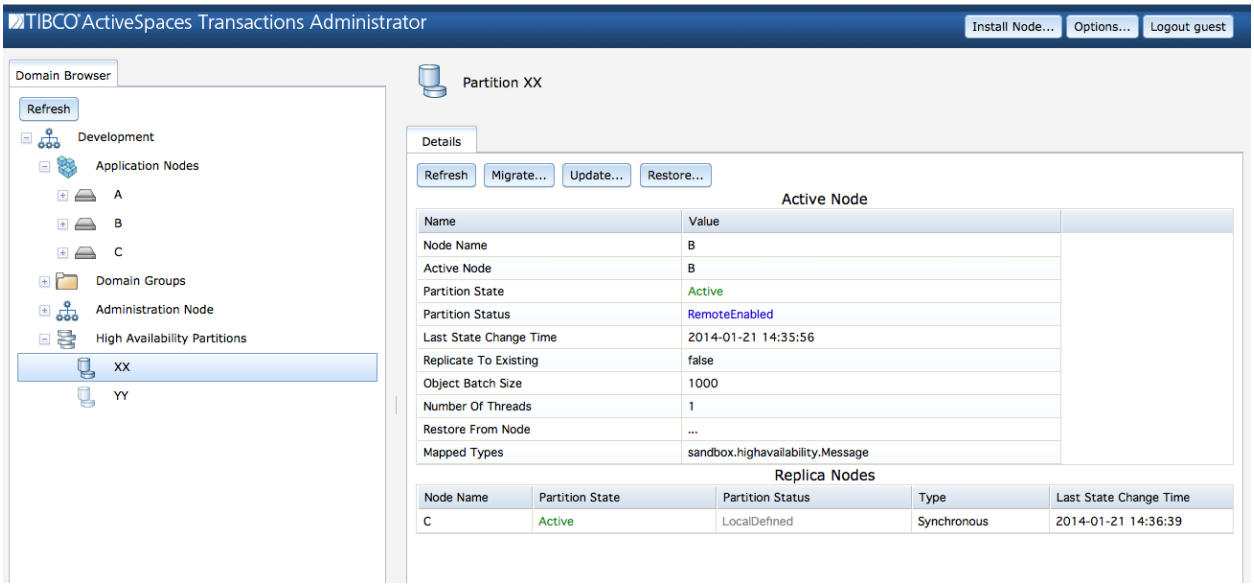

Figure 6.16. Updating a partition

Updating a partition can be also be done using this command. This command must be executed on the current active node for the partition.

```
administrator servicename=B update partition names="XX"
```

Node quorum management

Node quorum monitoring is controlled by configuration. Node quorum monitoring is enabled using the `nodeQuorum` configuration option. By default, node quorum monitoring is disabled.

When node quorum monitoring is enabled, the number of active application nodes required for a quorum is determined using one of these methods:

- minimum number of active remote nodes in the cluster (`minimumNumberQuorumNodes` configuration value).
- percentage of votes from currently active nodes in the cluster (`nodeQuorumPercentage` and `nodeQuorumVoteCount` configuration values).

Domain manager nodes are not included in node quorum calculations. Only application nodes count towards a node quorum.

When node quorum monitoring is enabled, high availability services are `Disabled` if a node quorum is not met. This ensures that a partition can never be active on multiple nodes. When a node quorum is restored, the node state is set to `Partial` or `Active` depending on the number of active remote nodes and the node quorum mechanism being used. Once a quorum has been reestablished partitions must be migrated back to the node. See the section called "Migrating partitions" on page 98. The current quorum state is displayed on the `High Availability` tab for a node (see Figure 6.1 for an example).

When using the minimum number of active remote nodes to determine a node quorum, the node quorum is not met when the number of active remote nodes drops below the configured `minimumNumberQuorumNodes` configuration value.

Figure 6.17 shows a four node example cluster using minimum number of remote nodes to determine node quorum. Each node shows their configured minimum remote node values.

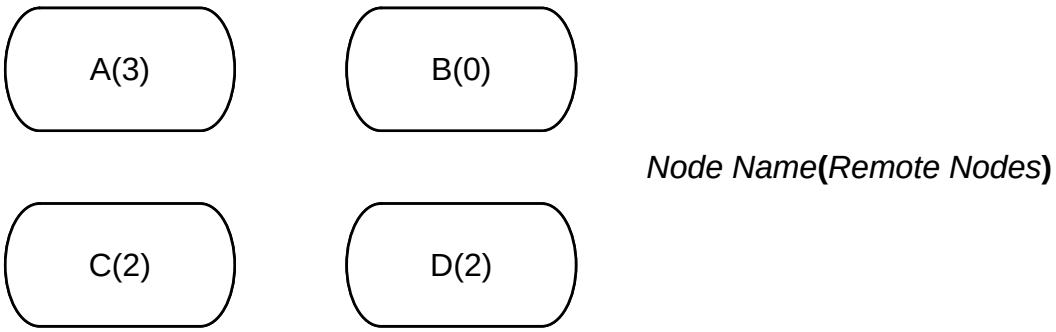

Figure 6.17. Minimum node quorum cluster

Figure 6.18 shows different scenarios based on network failures. All machines remain active. For each case the disabled and active nodes are shown, along with the total number of visible remote nodes, for the sub-cluster caused by the network failure.

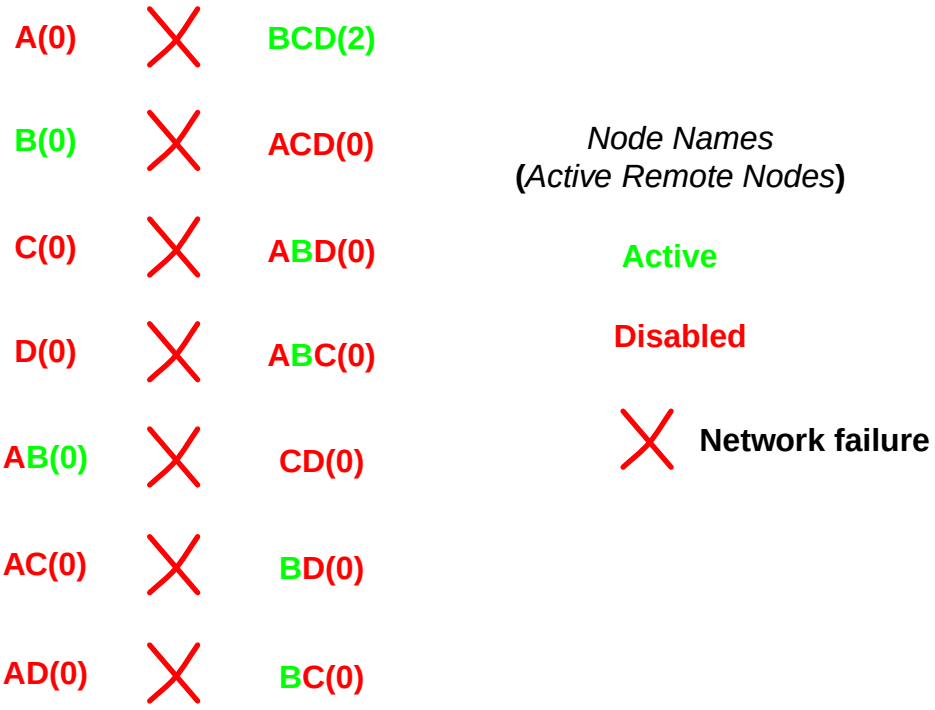

Figure 6.18. Minimum node quorum status - network failures

When using voting percentages, the node quorum is not met when the percentage of votes in a cluster drops below the configured `nodeQuorumPercentage` value. By default each node is assigned one vote. However, this can be changed with the `nodeQuorumVoteCount` configuration value. This allows certain nodes to be given more weight in the node quorum calculation by assigning them a larger number of votes.

Figure 6.19 shows a four node example cluster using voting percentages to determine node quorum. Each node shows their configured voting values. The node quorum percentage is set at 51%.

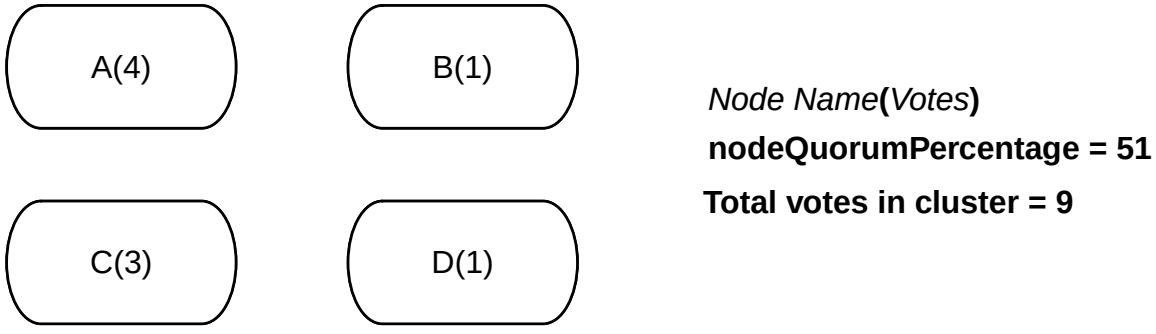

Figure 6.19. Voting quorum cluster

Figure 6.20 shows different scenarios based on network failures. All machines remain active. For each case the disabled and active nodes are shown, along with the total number of votes, and percentage, for the sub-cluster caused by the network failure.

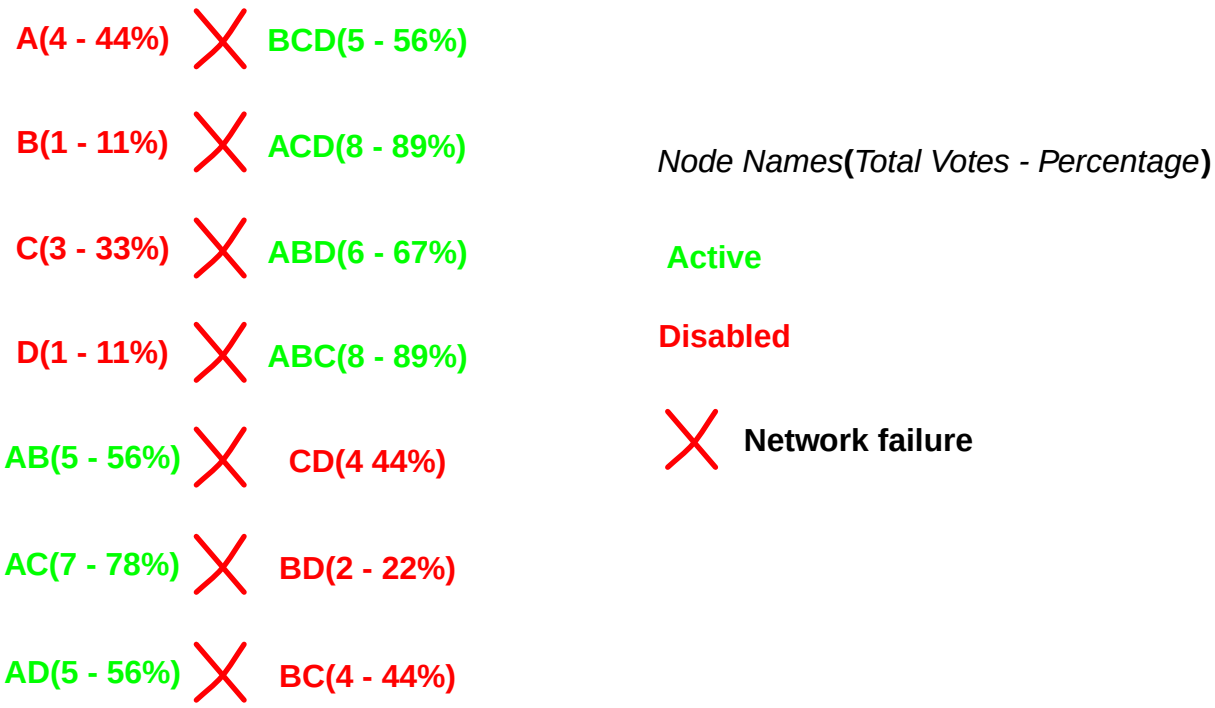

Figure 6.20. Voting node quorum status - network failures

Figure 6.21 shows different scenarios based on network and machine failures. For each case the disabled and active nodes are shown, along with the total number of votes, and percentage, for the sub-cluster caused by the network failure. This shows the advantages of using voting to determine node quorum status - the cluster can remain active following a single failure, while still allowing the cluster to be disabled if multiple nodes or networks fail.

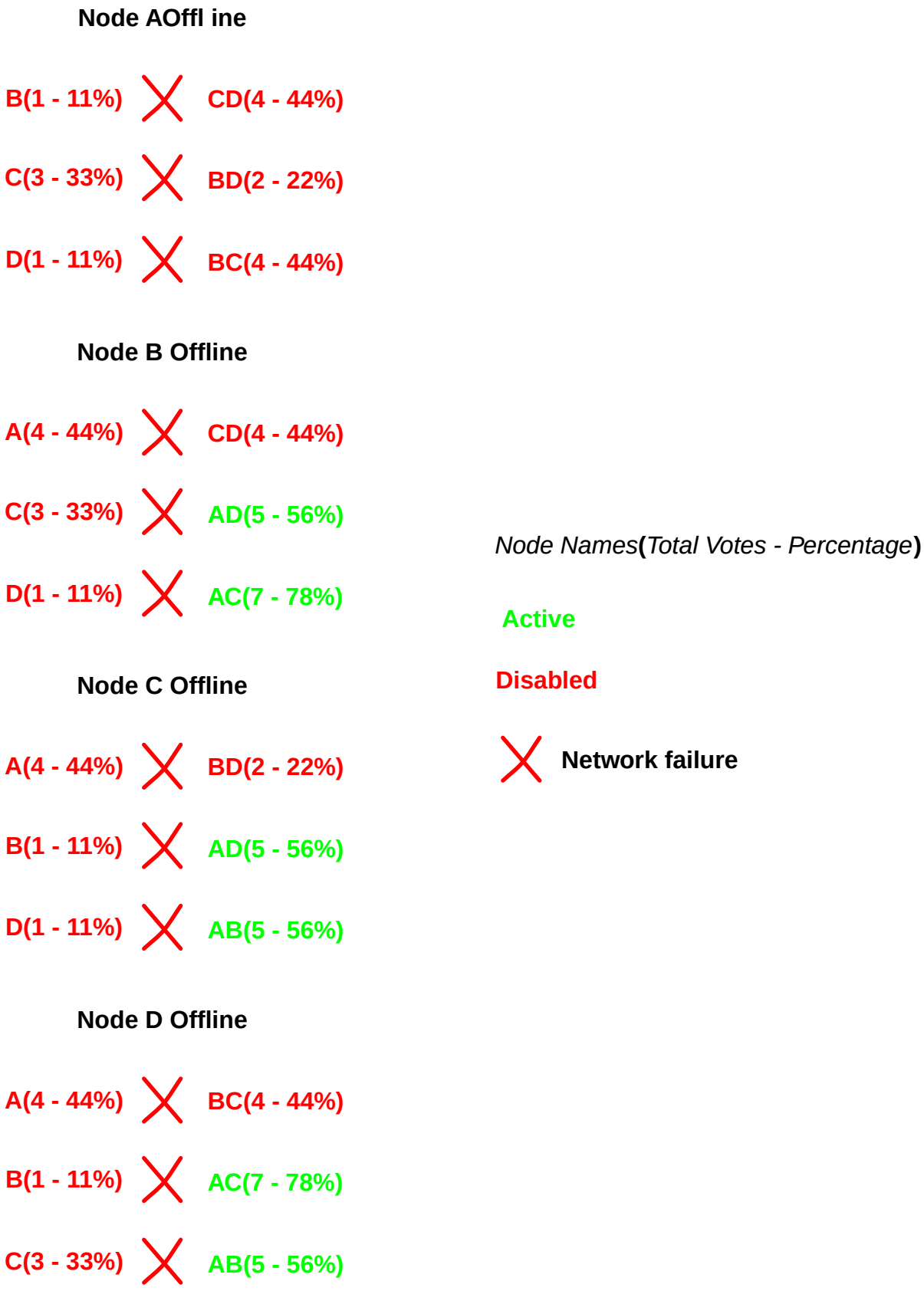
Node AOffl ine
B(1 - 11%)
CD(4 - 44%)
C(3 - 33%)
BD(2 - 22%)
D(1 - 11%)
BC(4 - 44%)
Node B Offline
A(4 - 44%)
CD(4 - 44%)
C(3 - 33%)
AD(5 - 56%)
D(1 - 11%)
AC(7 - 78%)
Node C Offline
A(4 - 44%)
BD(2 - 22%)
B(1 - 11%)
AD(5 - 56%)
D(1 - 11%)
AB(5 - 56%)
Node D Offline
A(4 - 44%)
BC(4 - 44%)
B(1 - 11%)
AC(7 - 78%)
C(3 - 33%)
AB(5 - 56%)
Node Names(Total Votes - Percentage)
Active
Disabled
Network failure

Figure 6.21. Voting node quorum status - network and machine failures

When a new high availability configuration is activated that changes the node quorum values the changes are not immediate. All changes are immediately propagated to all nodes in the cluster, but they do not take affect until a node leaves and rejoins the cluster, or a remote node fails. This ensures that a misconfiguration does not cause nodes to be taken offline unexpectedly.

The configuration values for node quorum are summarized in Table 6.2 on page 111.

Recovering partitions active on multiple nodes

There are cases where an application can tolerate operating with partitions active on multiple nodes for periods of time. If this is acceptable behavior for an application, the `nodeQuorum` configuration option should be set to `disabled`. When node quorum is disabled, the administrator must manually restore the cluster when the connectivity problem has been resolved.

The cluster partition summary display (see Figure 6.4) can be used to determine if partitions are active on multiple nodes. Before attempting to restore the partitions active on multiple nodes, connectivity between all nodes must have been reestablished. See the section called “Node connectivity” on page 87 for details on determining the status of node connectivity in a cluster.

The following steps are required to restore a cluster with partitions active on multiple nodes:

1. Define how to restore all partitions active on more than one node (see Figure 6.22).
2. Enable partitions on all nodes specified in the `Restore To Node` field in the restore partition dialog (see Figure 6.23).

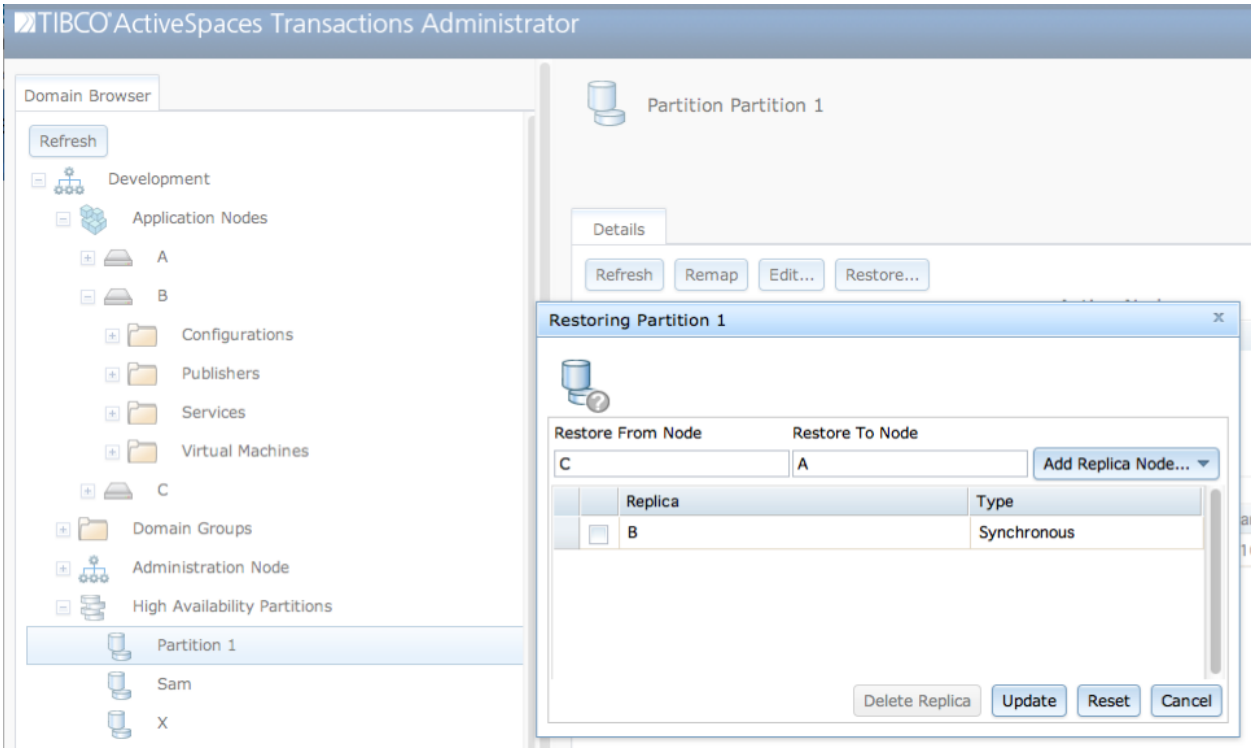

Figure 6.22. Restoring a partition

Figure Figure 6.22 shows partition X being restored from node C to node A. The partition must currently be active on both the from and to node specified in the restore node dialog. When partitions are enabled on node A, the partition objects will be merged with the objects on node C.

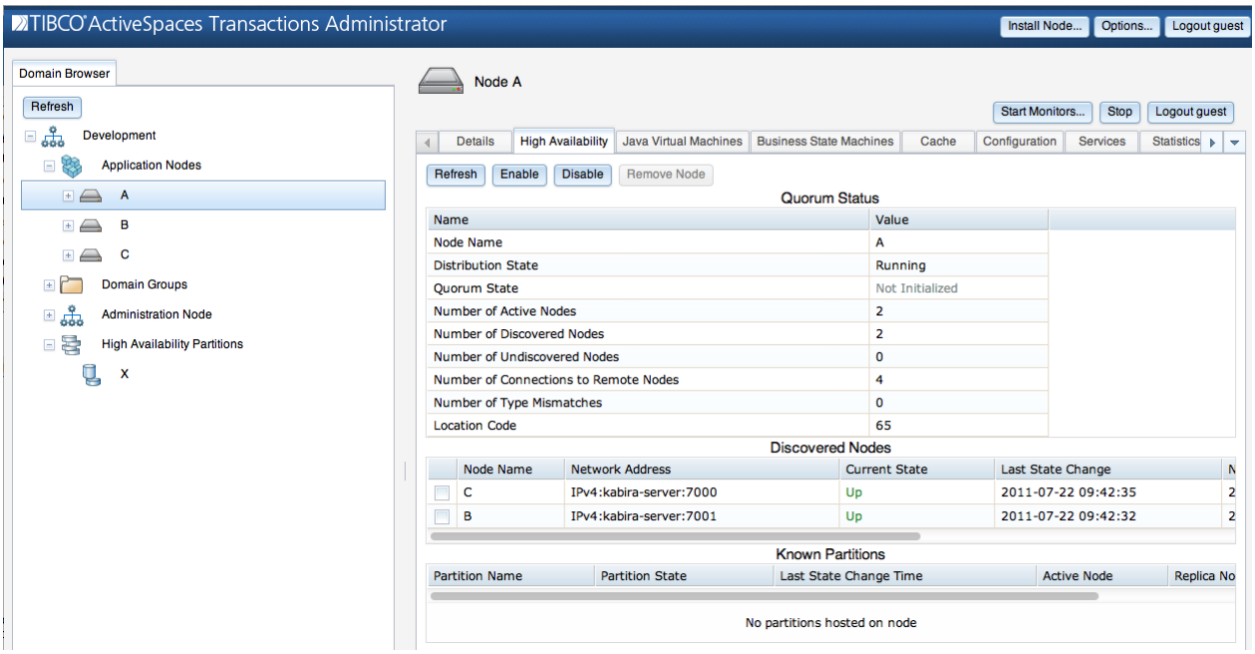

Figure 6.23. Enabling partitions

Clicking on the `Enable` button on the `High Availability` tab for a node (see Figure 6.23), causes all partitions being restored to this node to be merged with the partition specified as the from node in the restore partition dialog, and then the partition is made active on this node.

When these steps are complete, the cluster has been restored to service and all partitions now have a single active node.

Restoring a cluster after a multi-master scenario can also be performed using these commands.

```
administrator servicename=A define partition name="Partition 1" activenode=A replicas=B,C
 restorefromnode=C
administrator servicename=A join cluster type=merge
```

Configuration

Distribution configuration is described in this section. Distribution configuration is used for:

- high availability cluster behavior
- distributed transaction behavior
- network information
- dynamic and static discovery parameters

Distribution configuration has a configuration type of `distribution`.

Distribution services are enabled when there is an active distribution configuration on a node and an application JVM executing in which to host distribution. Distribution services are disabled when there is no active distribution configuration on a node, or there are no active application JVMs running.

Distribution configuration defines a nested configuration block named `Distribution`. The distribution configuration block contains these nested configuration blocks:

- `HighAvailability` - high availability configuration
- `Transaction` - distributed transaction configuration
- `Transport` - network configuration
- `DynamicDiscovery` - dynamic service discovery configuration
- `StaticDiscovery` - static service discovery configuration. Contains a separate configuration block for each remote node being defined.

Example 6.1 on page 110 shows how the distribution configuration and configuration blocks are used.

Example 6.1. Distribution configuration

```
//
//    Define a distributed configuration named sampleDistribution
//    This is version 1.0 of this configuration
//
configuration "sampleDistribution" version "1.0" type "distribution"
{
     //
     //    Required configuration block name space for distribution configuration
     //
     configure switchadmin
     {
          //
          //    Distribution configuration block
          //
          configure Distribution
          {
               //
               //    High availability configuration
               //
               HighAvailability
               {
                    ...
               };

               //
               //    Distributed transation configuration
               //
               Transaction
               {
                    ...
               };

               //
               //    Distribution transport configuration
               //
               Transport
               {
                    ...
```

```
            };

            //
            //    Dynamic service discovery configuration
            //
            DynamicDiscovery
            {
                 ...
            };

            //
            //    Static service discovery configuration
            //
            configure StaticDiscovery
            {
                 //
                 //     Remote node configuration
                 //
                 RemoteNode
                 {
                      ...
                 };
                 ...
            };
        };
    };
};
```

The configuration values supported by the distribution configuration blocks are summarized in the tables below.

Table 6.2 on page 111 defines the high availability configuration block values.

Table 6.2. High availability configuration

Name	Type	Description
nodeQuorum	Enumeration - Enable or Disable.	Control node quorum behavior. Default value is Disable.
minimumNumberQuorumNodes	Integer	Number of nodes required for a quorum. This value is ignored if nodeQuorum is set to Disable. minimumNumberQuorumNodes must be set to 0 if nodeQuorumPercentage is set to a non-zero value. Default value is 0. See the section called "Node quorum management" on page 103 for more details on the use of this configuration value.
nodeQuorumPercentage	Integer	Percentage of votes required for a quorum, including the votes for this node. This value is ignored if nodeQuorum is set to Disable. nodeQuorumPercentage must be set to 0 if minimumNumberQuorumNodes is set to a non-zero value. Default value is 0. See the section called "Node quorum management" on page 103 for more details on the use of this configuration value.
nodeQuorumVoteCount	Integer	Number of votes for this node when calculating the node quorum percentage. This value is ignored if nodeQuorum is set to Disable or nodeQuorumPercentage is 0. Default value is 1. See the section called "Node quorum manage-

		ment" on page 103 for more details on the use of this configuration value.
keepAliveSendIntervalSeconds	Integer	Keep-alive send interval. Must be a positive number. Default value is 1 second.
nonResponseTimeoutSeconds	Integer	Keep-alive non-response timeout interval following a send failure on a network interface to a remote node. When the non-response timeout expires on all configured interfaces, the remote node is marked down. Must be a positive number. Default value is 2 seconds.
deferredWritesEnabled	Boolean	Control deferred writes distribution protocol. A value of `true` causes writes to be deferred until a transaction commits. A value of `false` causes writes to be done immediately following a field modification. Default value is `true`.

Table 6.3 on page 112 defines the distributed transaction configuration block values.

Table 6.3. Distributed transaction configuration

Name	Type	Description
timeoutSeconds	Integer	The number of seconds to wait for a distributed lock. If this time value expires, a deadlock is thrown and the transaction is retried. Must be a positive number. Default value is 60 seconds.
numberCompletedTransactions	Integer	The maximum number of committed or aborted transactions to retain for each remote node. This is used for recovery processing to determine the outcome of global transactions if a remote node crashes and is restarted with a loss of shared memory. Default value is 1000.

Table 6.4 on page 112 defines the distribution transport configuration block values. Distribution can be configured to use either TCP, SSL, or Infiniband as the underlying transport protocol. Listener addresses are specified using the `listenerAddressList` configuration value. The possible transport protocol specifiers are:

- `IPv4[/[TCP|SSL]]` - Specify an IPv4 network address using either TCP or SSL as the underlying transport protocol. The `TCP` specifier is optional - both `IPv4` and `IPv4/TCP` specify TCP. `IPv4/SSL` specifies SSL.
- `IPv6[/[TCP|SSL]]` - Specify an IPv6 network address using either TCP or SSL as the underlying transport protocol. The `TCP` specifier is optional - both `IPv6` and `IPv6/TCP` specify TCP. `IPv6/SSL` specifies SSL.
- `IPoSDP` - specify an Infiniband transport address.

Table 6.4. Distribution transport configuration

Name	Type	Description
listenerAddressList	String array	Listener address list. A list of TCP or Infiniband (Linux only) network addresses on which to start a listener. Remote nodes communicate to this node using these listeners. The format of each specified network listener address is: `IPv4[/[TCP|SSL]]|IPv6[/[TCP|SSL]]|IPoSDP:<host`

		address>:<port number> The *<host address>* default value is all interfaces. The default *<port number>* is 5557. For example to listen on all IPv4 interfaces on port number 5557 would require the following entry: {IPv4::}; To listen on all IPv6 interfaces on port 5558 using SSL would require the following entry: {IPv6/SSL::5558}; To listen on a specific Infiniband interface would required the following entry: {IPoSDP:myhost-name:5558}; To listen on all of these interfaces requires the following entry: {IPv4::,IPv6/SSL::5558,IPoSDP:myhost-name:5558};
numberSearch-Ports	Integer	The number of ports to search before reporting a listener start failure. The search is started at the configured network listener port number and then incremented by one on each failure up to this value. A value of 0 disables port search. Default value is 20.
nodeActive-Timeout-Seconds	Integer	The amount of time in seconds to wait for a remote node to move into the Active state before a resource unavailable exception is raised. The wait is done for a partition's active node when defining a partition, or for each remote node when a discover cluster administration command is executed. This value must be > 0. Default value is 60 seconds.
tcpNoDelayEn-abled	Boolean	Control enabling of TCP_NODELAY socket option on connection establishment. Default value is true.
maximumPDUS-izeBytes	Integer	Control maximum protocol data unit (PDU) size used for distributed communications. This value must be > 4000. Default value is 1000000 bytes.

Table 6.5 on page 113 defines the dynamic discovery configuration block values. Dynamic discovery of nodes depends on service discovery being enabled. If service discovery is disabled, dynamic discovery cannot be enabled. See the section called “Installation” on page 31 for details.

Table 6.5. Dynamic discovery configuration

Name	Type	Description
enabled	Boolean	Enable or disable dynamic discovery. Dyanmic discovery can only be enabled if the discovery service is enabled. Default value is true.

Table 6.6 on page 113 defines the static discovery RemoteNode configuration block values.

The listener address of the remote node is specified using the networkAddressList configuration value. The transport protocol specifier in the networkAddressList uses the same format as the listenerAddressList described in Table 6.4 on page 112.

Table 6.6. Static discovery configuration

Name	Type	Description
name	String	Remote node name. This field is required. There cannot be multiple configurations with the same node name.
networkAd-dressList	String array	Network address list. This field is required. A list of TCP or Infiniband (Linux only) network addresses over which the node can be reached. Connections to this remote node are done using round-robin in the

		order in which the addresses are specified in this list. The format of each specified network address is `IPv4[/[TCP|SSL]]|IPv6[/[TCP|SSL]]|IPoSDP:<host address>:<port number>`. For example to connect over `IPv4` using TCP to an `acme.com` host at port 5557 would require the following entry: `{IPv4:acme.com:5557}`

Updating configuration

Distribution configuration can be updated on a running node to change any of the configuration values. The majority of the distribution configuration values can be updated without impacting distribution connectivity between nodes - the distribution services are not disabled. An active distribution configuration can be replaced with a new version with different configuration values. When replacing an active distribution configuration with an updated version, the configuration *name* must be the same. For example:

```
//
//  Original distribution configuration version
//
configuration "connectivity" version "1.0" type "distribution"
{
};

//
//  Updated distribution configuration version - has same name - connectivity
//
configuration "connectivity" version "2.0" type "distribution"
{
};
```

Changing these configuration values require the distribution services to be disabled first, by deactivating the current configuration:

- The transport `listenerAddressList` (Table 6.4 on page 112).
- All dynamic discovery configuration values - `enabled`, `broadcastPort`, and `broadcastHost` (Table 6.5 on page 113).
- Removing an active statically defined remote node (Table 6.6 on page 113).

If an active distribution configuration is replaced with a new version with any of the above values changed, the activation of the new version will fail audit and the previous version will remain active.

Configuration Examples

Example 6.2 on page 114 shows an example of a high availability configuration.

Example 6.2. High availability configuration

```
configuration "distribution" version "1.0" type "distribution"
{
    configure switchadmin
    {
        configure Distribution
        {
            HighAvailability
            {
                nodeQuorum = Enable;
```

```
                minimumNumberQuorumNodes = 1;
                keepAliveSendIntervalSeconds = 1;
                nonResponseTimeoutSeconds = 2;
            };
        };
    };
};
```

Example 6.3 on page 115 shows an example of a distributed transaction configuration.

Example 6.3. Distributed transaction configuration

```
configuration "distribution" version "1.0" type "distribution"
{
     configure switchadmin
     {
         configure Distribution
         {
            Transaction
            {
                timeoutSeconds = 60;
                numberCompletedTransactions = 1000;
            };
         };
     };
};
```

Example 6.4 on page 115 shows an example of a distribution transport configuration.

Example 6.4. Distribution transport configuration

```
configuration "distribution" version "1.0" type "distribution"
{
    configure switchadmin
    {
        configure Distribution
        {
            Transport
            {
               listenerAddressList =
               {
                   "IPv4::7000", // listen on all IPv4 interfaces
                   "IPv6/SSL::7100:, // listen on all IPv6 interfaces and use SSL
                   "IPoSDP:ib_nodeA:2000" // listen on Infiniband ib_nodeA interface
               };
               numberSearchPorts = 20;
               nodeActiveTimeoutSeconds = 60;
               tcpNoDelayEnabled = true;
               maximumPDUSizeBytes = 1000000;
            };
        };
    };
};
```

Table 6.5 on page 113 shows an example of a dynamic discovery configuration.

Example 6.5. Dynamic discovery configuration

```
configuration "distribution" version "1.0" type "distribution"
{
    configure switchadmin
    {
        configure Distribution
        {
```

```
            DynamicDiscovery
            {
                enabled = true;
            };
        };
    };
};
```

Table 6.6 on page 113 shows an example of a static discovery configuration.

Example 6.6. Static discovery configuration

```
configuration "distribution" version "1.0" type "distribution"
{
    configure switchadmin
    {
        configure Distribution
        {
            configure StaticDiscovery
            {
                RemoteNode
                {
                    name = "A";
                    networkAddressList =
                    {
                        "IPv4/SSL:a-networkaddress:22222", // IPv4 address using SSL
                        "IPv6:a-networkaddress:33333" // IPv6 address

                    };
                };
                RemoteNode
                {
                    name = "B";
                    networkAddressList =
                    {
                        "IPoSDP:a-networkaddress:2000" // Infiniband address
                    };
                };
            };
        };
    };
};
```

This configuration defines two remote nodes named A and B. These remote nodes must have their distribution listener addresses configured to match the network addresses specified in the `RemoteNode` configuration block. Specifically node A must have specified the `listenerAddressList` in Example 6.7 on page 116 and node B must have specified the `listenerAddressList` in Example 6.8 on page 117.

Do not mix Infiniband and Ethernet addresses in the same `RemoteNode` configuration block. Network addresses in the `RemoteNode` `networkAddressList` configuration are used round-robin in the order they are specified. Mixing Infiniband and Ethernet addresses will use a mix of these two connection types to the remote node, defeating the performance advantages of using Infiniband.

Example 6.7. Node A transport configuration

```
configuration "node-a" version "1.0" type "distribution"
{
    configure switchadmin
    {
        configure Distribution
        {
```

```
            Transport
            {
               listenerAddressList =
               {
                   "IPv4/SSL:a-networkaddress:22222",
                   "IPv6:a-networkaddress:33333"

               };
               ...
            };
        };
    };
};
```

Example 6.8. Node B transport configuration

```
configuration "node-b" version "1.0" type "distribution"
{
    configure switchadmin
    {
        configure Distribution
        {
            Transport
            {
               listenerAddressList =
               {
                   "IPoSDP:a-networkaddress:2000"
               };
               ...
            };
        };
    };
};
```

7

Reference

This chapter provides reference material for

- Required network ports
- Administrator command line interface
- Configuration file syntax
- Default distribution configuration

Required network ports

This section summarizes the equired network ports and how they are configured. In general, these network ports must be allowed through host firewalls for correct operation.

Table 7.1. Network ports

Service	Protocol	Default Port Number	Configuration
Administration	SSL using IPv4	Randomly assigned valid free port > 1024.	See Table 3.1 on page 33
Administration Web Server - Domain Manager	SSL or TCP using IPv4	Randomly assigned valid free port > 1024.	See Table 2.5 on page 29
Distribution Transport	SSL or TCP using IPv4, IPv6, or IPoSDP (Infiniband)	Attempts to find a free port starting at 5557 up to 5577.	See Table 6.4 on page 112
JMX Administration	TCP using IPv4	Randomly assigned valid free port > 1024.	Set when deploying applications using the deployment tool.

Node Agent - Domain Manager	SSL or TCP using IPv4	Randomly assigned valid free port > 1024.	See Table 3.11 on page 53
Service Discovery	UDP broadcast over IPv4	54321	See the section called “Installation” on page 31

Figure 7.1 shows how the required network ports are used to communicate between nodes and administration clients.

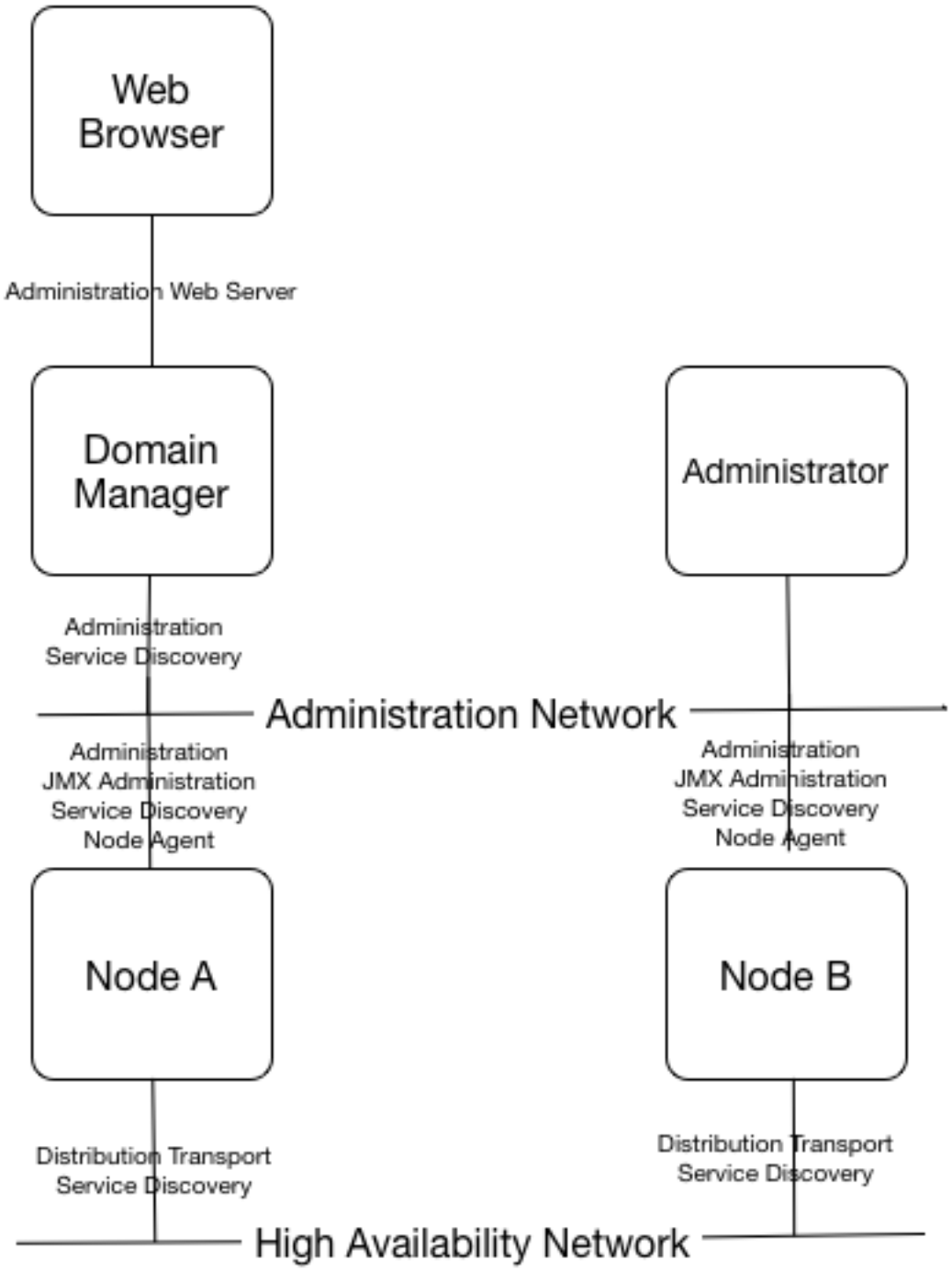

Figure 7.1. Network ports

Command line interface

This section describes the command-line interface (CLI) used to manage nodes. This CLI is invoked using the `administrator` command. The `administrator` command can perform any task that

can be done using the TIBCO ActiveSpaces® Transactions Administrator. In fact, the TIBCO ActiveSpaces® Transactions Administrator issues exactly the same commands as `administrator` (though it also caches and displays some information in a more user-friendly form).

The `administrator` command line interface uses a combination of a *command* and a *target*, possibly including *parameters,* to specify an *action*. A *command* is generally a verb, and the *target* is generally a noun.

Parameters are always specified using *<name>*=<value> pairs. There are two types of parameters - *global* and *action specific*. Global parameters are valid for all actions, with a few exceptions. Different mandatory and optional action specific parameters are implied by the action. Global parameters are always specified to the left of the command. Action specific parameters are always specified to the right of the target.

This is the general `administrator` syntax:

```
administrator global-parameters command target action-specific-parameters
```

The `administrator` executable is located in the TIBCO ActiveSpaces® Transactions installation at:

```
$SW_HOME/distrib/kabira/bin/administrator
```

It is convenient to include this path in the PATH variable to save typing it every time.

Getting help

`administrator` has extensive built-in help. To access the top level help use `administrator help`.

```
administrator help

  administrator [<global-parameter>=<value>,...]
      <command> <target> [<parameter>=<value>,...]
      Command line syntax

  administrator {[hostname=<host>] adminport=<port>
      | servicename=<name>} help
      List of available targets on a node.

  administrator {[hostname=<host>] adminport=<port>
      | servicename=<name>} help <target>
      Help for a specific target on a node.

  administrator help node
      Help for the node target.

  administrator help services
      Help for the services target.

  administrator help snapshot
      Help for the snapshot target.

  administrator help globals
      Help for global parameters
```

These different help options can be seen from the usage message:

- List of all available targets on a node. A running node, `adminport` and `hostname`, or `servicename`, is required to access this help. See Example 7.1 on page 123.

- Help for a specific target on a node. A running node, `adminport` and `hostname`, or `service-name`, is required to access this help. See Example 7.2 on page 123.
- Node target help. No running node is required to access this help. See Example 7.3 on page 124.
- Services target help. No running node is required to access this help. See Example 7.4 on page 124.
- Snapshot target help. No running node is required to access this help. See Example 7.5 on page 124.
- Global parameter help. No running node is required to access this help. See Example 7.6 on page 124.

Example 7.1. Target list help

```
administrator adminport=21447 help
  ldap
  jvm
  endpoint
  space
  distribution
  globaltransaction
  configuration
  session
  partition
  service
  security
  cluster
  cache
  node
  snapshot
  statistics
```

Example 7.2. Specific target help

```
administrator adminport=21447 help jvm
  display jvm
    [ name=<string value, default = ""> ]
        The JVM name.

    Display the state of a JVM.

  stop jvm
    name=<string value>
        The JVM name.

    Stop a JVM.

  start jvm
    name=<string value>
        The JVM name.

    Start a JVM.

  remove jvm
    name=<string value>
        The JVM name.

    Remove a JVM from the node.

Description:

    Manage JVMs on a node.
```

Example 7.3. Node target help

```
administrator help node
  display node

  getadminport node
        [ installpath=<value> ]
        [ nodename=<value> ]

    install node
      [ adminhost=<value> ]
      [ adminport=<value> ]
      [ application=<value> ]
      [ buildtype=<value> ]
      [ configpath=<value> ]
      [ deploydirectories=<value> ]
      [ description=<value> ]
      [ discoveryenabled=<true|false> ]
      [ discoveryport=<value> ]
      [ installpath=<value> ]
      [ javabinarypath=<value> ]
      [ javaenvironment=<value> ]
      [ javahome=<value> ]
      [ javalibrarypath=<value> ]
      [ memoryallocators=<value> ]
      [ memorysize=<value> ]
      [ memorytype=<value> ]
      [ nodename=<value> ]
      [ trustedhosts=<value> ]
      [ usehostaliases=<value> ]

   ...
```

Example 7.4. Services target help

```
administrator help services
  display services
    [ servicetype=<value> ]
    [ servicename=<value> ]
    [ timeout=<seconds> ]
    [ detailed=<boolean> ]
    [ properties=<property list> ]

  ...
```

Example 7.5. Snapshot target help

```
administrator help snapshot
  create snapshot
      [ description=<value> ]
      [ destination=<value> ]
      [ includeclasspaths=<false|true> ]
      [ installpath=<value> ]

      Create a snapshot archive.

  ...
```

Example 7.6. Global parameters help

```
administrator help globals
  adminport:      Administration port (no default).
  delimiter:      Enable delimited tabular output using parameter value
                  as delimiter.  Default is formatted text output.
```

```
  discoveryport   Port to use for service discovery.
                  Defaults to 54321.
  discoveryhost   The hostname to use to select which network
                  interface gets used for sending the discovery
                  request. The default uses the system's host name.
  domaingroup:    Target all nodes in domain group (no default).
  domainname:     Target all nodes in domain (no default).
  domainnode:     Target specific node (no default).
  hostname:       Host name (default localhost).
  password:       User password (no default).
  servicename:    Service name (no default).
  username:       User name (default login name).
```

Accessing a node

Accessing a node to perform an administrative action requires:

- Valid credentials, i.e. a user name and password
- Network addressing information

Credentials and network address information are specified using global parameters. See the section called “Global parameters” on page 127 for details.

If a username is specified but a password is not, then `administrator` prompts for a password. If username is not specified, the user name in the shell environment in which `administrator` is executed is used as the user name. See Chapter 5 to learn how to configure access control.

Node address information is required for most commands (those that apply to a node). You can specify an explicit hostname and port number, or alternatively you can use a service name to access a node. See the section called “Service discovery” on page 125 for details on using a service name.

Administration actions can also be addressed to multiple nodes simultaneously. This is supported by targeting an action at a domain manager node and specifying that the command should be targeted at all nodes in the domain, a subset of the nodes, or a specific node.

Service discovery You do not need to supply a specific host name or port to address a node; you can use the node's published service name. If you need to find the nodes running on the network, you can perform discovery manually using the `display services` action.

Here is example output from the `display services` action:

```
administrator display services
Service Name = B
Service Type = node
Network Address = Kapoho.local:2002

Service Name = A
Service Type = node
Network Address = Kapoho.local:2001

Service Name = C
Service Type = node
Network Address = Kapoho.local:2003

Service Name = domainmanager
Service Type = node
Network Address = Kapoho.local:2000
```

This output indicates that administration actions could be targeted at a node named A using a service name of A instead of using a host name of `Kapoho.local` and a port number of 2001.

Domain manager Administrative actions can be sent to a domain manager node for distribution to managed application nodes by specifying the host name and port number, or the service name, of the domain manager node. Global variables are used to indicate the final target of the administration command. See the section called "Global parameters" on page 127 for details on the syntax. The final target if an action can be:

- A single managed application node.
- Multiple managed application nodes in the same domain group.
- All application nodes being managed in the domain.

Domain manager nodes also support administrative actions similar to application nodes to allow them to be managed using the same targets and commands as application nodes.

The following example sends a `display cluster` action to all nodes in the domain by specifying the domain name and the network address of the domain manager node. The annotated output shows that the output comes from each node in the domain. There are three nodes in the domain - A, B, and C.

```
administrator servicename=domainmanager domainname="Development" display cluster type=local
Node Name = C // Output from node C
Node Name = C
Distribution State = Running
Quorum State = Active
Quorum Description = All nodes up and running.
Number of Active Nodes = 2
Number of Discovered Nodes = 2
Number of Undiscovered Nodes = 0
Number of Connections to Remote Nodes = 9
Number of Type Mismatches = 0
Location Code = 67
Calculated CRC of System Types = 3695020283

Node Name = B // Output from node B
Node Name = B
Distribution State = Running
Quorum State = Active
Quorum Description = All nodes up and running.
Number of Active Nodes = 2
Number of Discovered Nodes = 2
Number of Undiscovered Nodes = 0
Number of Connections to Remote Nodes = 9
Number of Type Mismatches = 0
Location Code = 66
Calculated CRC of System Types = 3695020283

Node Name = A // Output from node C
Node Name = A
Distribution State = Running
Quorum State = Active
Quorum Description = All nodes up and running.
Number of Active Nodes = 2
Number of Discovered Nodes = 2
Number of Undiscovered Nodes = 0
Number of Connections to Remote Nodes = 8
Number of Type Mismatches = 0
Location Code = 65
Calculated CRC of System Types = 3695020283
```

Global parameters

Table 7.2 on page 127 summarizes the supported global parameters.

Table 7.2. Global parameters

Name	Example	Description
`adminport`	`adminport=2001`	Administration port on target node. Required if `servicename` not specified. No default.
`delimiter`	`delimiter=|`	Format output as a table with the first row containing column names, and all other rows containing the columnar data. The delimiter character is used to separate all columns in all rows. Default is to output formatted plain text.
`discoveryport`	`discoveryport=54321`	Port to use for service discovery. Default value is 54321.
`discoveryhost`	`discoverhost=myhost`	The hostname to use to select which network interface gets used for sending the discovery request. The default value is the local machine's host name.
`domaingroup`	`domaingroup=EastCoast`	Target action at all application nodes in the specified domain group managed by a domain manager. Only valid if network address specifies a domain manager node. No default value.
`domainname`	`domainname=Development`	Target action at all application nodes in the specified domain managed by a domain manager Only valid if network address specifies a domain manager node. No default value.
`domainnode`	`domainnode=A`	Target action at a specific application node managed by a domain manager. Only valid if network address specifies a domain manager node. No default value.
`hostname`	`hostname=venus`	Host name where target node is running. This can be a simple name, a fully qualified name, or an IP address. Default value is `localhost`.
`password`	`password=guest`	User password to access target node. It is strongly recommended that passwords never be specified on the command line. See the section called "Accessing a node" on page 125 for details on specifying a password. No default.
`servicename`	`servicename=A`	The service name of the target node. See the section called "Service discovery" on page 125 for details on using service discovery. If `servicename` is specified, `hostname` and `adminport` are not required. No default.
`username`	`username=guest`	User name to access target node. Default value is login name of user executing `administrator`.

Supported targets

`administrator` supports both built-in targets and application specific targets. Application targets may be added during development of an application. See the TIBCO ActiveSpaces® Transactions **Java Developer's Guide** for details. Application specific targets are not discussed further here.

The supported built-in targets are show in Table 7.3 on page 128.

Table 7.3. Built-in administration targets

Target	Description
`cache`	Named cache management
`cluster`	Cluster management.
`configuration`	Configuration management
`distribution`	Manage distribution services.
`domain`	Manage administration domain. Only available in a domain manager node.
`endpoint`	Manage channel endpoint (Java channels only)
`jvm`	Java virtual machine management.
`ldap`	Control external LDAP authentication sources.
`manager`	Display status of TIBCO ActiveSpaces® Transactions Administrator server. Only available in domain manager node.
`node`	Node management.
`partition`	Partition management.
`security`	Security management.
`service`	Manage channel services (Java channels only)
`services`	Discover nodes using service discovery.
`session`	Manage channel sessions (Java channels only)
`snapshot`	Create a diagnostic snapshot of a node.
`statistics`	Capture and display runtime statistics.

Details on each of these targets can be found in the following sections. These sections provide a brief description of the supported commands and targets, along with any parameters. Examples are provided to help clarify a command's usage.

Global parameters are not shown in the examples unless they are required to clarify command usage.

Channels

Channels are managed using the `endpoint`, `service`, and `session` targets.

These targets can only be used to manage custom Java channels.

endpoint target Manage channel endpoints.

clearstatistics command

Clear endpoint statistics.

```
clearstatistics endpoint
clearstatistics endpoint name=myEndpoint
```

Table 7.4. clearstatistics command parameters

Name	Description	Required
`name`	Optional endpoint name. If omitted, statistics are cleared for all endpoints.	No. No default.

disabletrace command

Disable tracing for an endpoint. The valid trace options are:

- `endpoint` - trace endpoint level messages.
- `pdu` - trace protocol data units.
- `session` - trace session level messages.

A value of `all` can be used to indicate all trace options.

```
disabletrace endpoint name=myEndpoint
disabletrace endpoint name=myEndpoint options="session,pdu"
```

Table 7.5. disabletrace command parameters

Name	Description	Required
`name`	Endpoint name.	Yes.
`options`	Optional tracing options. A comma separated list of trace options to disable. Valid values are `endpoint`, `session`, `pdu`, or `all`. Specified values will be subtracted from current trace values.	No. Default value is `all`.

display command

Display endpoint information.

The `name`, `state`, `servicename`, and `group` parameters are used to filter the displayed endpoints. Only one of these parameters can be specified. If none of these parameters are specified, all endpoints are displayed.

```
display endpoint
display endpoint name=myEndpoint
display endpoint state=Started
display endpoint group=myGroup
display endpoint servicename=myService
```

Table 7.6. display command parameters

Name	Description	Required
`name`	Optional endpoint name. Only display endpoint with this name.	No. No default value.

Name	Description	Required
state	Optional endpoint state. Valid values are Stopped or Started. Only display endpoints with specified state.	No. No default value.
servicename	Optional service name. Only display endpoints owned by this service.	No. No default value.
group	Optional group name. Only display endpoints owned by this group.	No. No default value.

enabletrace command

Enable tracing for an endpoint. The valid trace options are:

- endpoint - trace endpoint level messages.
- pdu - trace protocol data units.
- session - trace session level messages.

A value of all can be used to indicate all trace options.

```
enabletrace endpoint name=myEndpoint
enabletrace endpoint name=myEndpoint options="session,pdu"
```

Table 7.7. enabletrace command parameters

Name	Description	Required
name	Endpoint name.	Yes.
options	Optional tracing options. A comma separated list of trace options to enable. Valid values are endpoint, session, pdu, or all. Specified values will be added to current trace values.	No. Default value is all.
count	Optional number of message to trace. Tracing is disabled after count messages. Use a value of zero to enable tracing until it is explicitly disabled using the disabletrace command.	No. Default value is 0.

lock command

Lock an endpoint.

```
lock endpoint name=myEndpoint
```

Table 7.8. lock command parameters

Name	Description	Required
name	Endpoint name.	Yes.

start command

Start an endpoint.

```
start endpoint name=myEndpoint
```

Table 7.9. start command parameters

Name	Description	Required
name	Endpoint name.	Yes.

startall command

Start all endpoints.

The group parameter can be used to only start endpoints in a specific group, otherwise all endpoints are started.

```
startall endpoint
startall endpoint group=myGroup
```

Table 7.10. startall command parameters

Name	Description	Required
group	Optional group name. Only start endpoints in this group.	No. No default.

stop command

Stop an endpoint.

```
stop endpoint name=myEndpoint
```

Table 7.11. stop command parameters

Name	Description	Required
name	Endpoint name.	Yes.

stopall command

Stop all endpoints.

The group parameter can be used to only stop endpoints in a specific group, otherwise all endpoints are stopped.

```
stopall endpoint
stopall endpoint group=myGroup
```

Table 7.12. stopall command parameters

Name	Description	Required
group	Optional group name. Only stop endpoints in this group.	No. No default.

unlock command

Unlock an endpoint.

```
unlock endpoint name=myEndpoint
```

Table 7.13. unlock command parameters

Name	Description	Required
name	Endpoint name.	Yes.

service target Manage channel services.

display command

Display service information.

The name, state, and group parameters are used to filter the displayed services. Only one of these parameters can be specified. If none of these parameters are specified, all services are displayed.

```
display service
display service name=myService
display service state=Started
display service group=myGroup
```

Table 7.14. display command parameters

Name	Description	Required
name	Optional service name. Only display service with this name.	No. No default value.
state	Optional service state. Valid values are Stopped or Started. Only display services with specified state.	No. No default value.
group	Optional group name. Only display services owned by this group.	No. No default value.

start command

Start a service.

```
start service name=myService
```

Table 7.15. start command parameters

Name	Description	Required
name	Service name.	Yes.

startall command

Start all services.

The group parameter can be used to only start services in a specific group, otherwise all services are started.

```
startall service
startall service group=myGroup
```

Table 7.16. startall command parameters

Name	Description	Required
group	Optional group name. Only start services in this group.	No. No default.

stop command

Stop a service.

```
stop service name=myService
```

Table 7.17. stop command parameters

Name	Description	Required
`name`	Service name.	Yes.

stopall command

Stop all services.

The `group` parameter can be used to only stop services in a specific group, otherwise all services are stopped.

```
stopall service
stopall service group=myGroup
```

Table 7.18. stopall command parameters

Name	Description	Required
`group`	Optional group name. Only stop services in this group.	No. No default.

session target Manage channel sessions.

clearstatistics command

Clear session statistics.

```
clearstatistics session
clearstatistics session name=mySession
```

Table 7.19. clearstatistics command parameters

Name	Description	Required
`name`	Optional session name. If omitted, statistics are cleared for all session.	No. No default.

display command

Display session information.

The `name`, `endpointname`, and `servicename` parameters are used to filter the displayed sessions. Only one of these parameters can be specified. If none of these parameters are specified, all sessions are displayed.

```
display session
display session name=mySession
display session endpointname=myEndpoint
display session servicename=myService
```

Table 7.20. display command parameters

Name	Description	Required
`name`	Optional session name. Only display session with this name.	No. No default value.
`endpointname`	Optional endpoint name. Only display sessions owned by this endpoint.	No. No default value.
`servicename`	Optional service name. Only display sessions owned by this service.	No. No default value.

start command

Start a session.

```
start session name=mySession
```

Table 7.21. start command parameters

Name	Description	Required
`name`	Session name.	Yes.

stop command

Stop a session.

```
stop session name=mySession
```

Table 7.22. stop command parameters

Name	Description	Required
`name`	Session name.	Yes.

Diagnostics

Diagnostic information is captured by the `snapshot` target. The `snapshot` target creates an archive containing log files and other information required to help troubleshoot problems. A snapshot archive can be captured for an active or a failed node.

Snapshots are taken against active nodes using the `adminport` and `hostname` or `servicename` global parameters.

Snapshots are taken against failed nodes using the `installpath` parameter.

snapshot target Create and manage snapshot archive files.

create command

Create a snapshot archive. The created archive is named *<node name>*.*<timestamp>*.zip.

```
create snapshot
create snapshot description="issue 4567"
create snapshot destination=/tmp
create snapshot includeclasspaths=true
create snapshot installpath=/opt/tibco/nodes/A
```

Table 7.23. create command parameters

Name	Description	Required
`description`	Optional description to include in archive. If specified a file named `description.txt` is included in the snapshot archive with the specified description.	No. No default value.
`destination`	Optional destination directory for archive.	No. Default directory is *<node directory>*`/../snapshots/`*<node name>*.
`include-classpaths`	Optional boolean value on whether or not to include the contents of a node's classpath directories.	No. Default value is `false`.
`installpath`	Optional node installation path. This parameter can be used to specify the node directory to create a snapshot archive if the node is not operational.	Yes, if node not operational, otherwise optional. No default value.

Domain Manager

Domain management is accomplished using the `domain` and `manager` targets. These targets are only supported on Domain Manager nodes.

domain target Manage an administrative domain.

addgroup command

Add a group to the administrative domain.

```
addgroup domain groupname=myGroup
addgroup domain groupname=myGroup properties="prop1=a,prop2=b"
```

Table 7.24. addgroup command parameters

Name	Description	Required
`groupname`	Group name.	Yes.
`properties`	Comma separated list of *<name>*=*<value>* service property pairs. If a node has these properties published in their ndoe service record, they will be automatically added to the group.	No. No default.

addgroupnode command

Add a managed node to a group. The node must already be managed by the domain and the group must be defined.

```
addgroupnode domain groupname=myGroup name=myNode
```

Table 7.25. addgroupnode command parameters

Name	Description	Required
`groupname`	Group name.	Yes.

Name	Description	Required
name	Node name.	Yes.

addnode command

Add a new node to the administrative domain. The node must be started.

```
addnode domain name=myNodeServiceName
addnode domain name=myNodeServiceName timeout=10
```

Table 7.26. addnode command parameters

Name	Description	Required
name	Service name of node to add.	Yes.
timeout	Optional service lookup timeout in seconds. If the lookup of the node service is not completed in this time, the command fails.	No. Default value is 5.

display command

Display domain information. The supported display types are summarized in Table 7.27 on page 136.

Table 7.27. Display types

Type	Description	Name Filter	Detailed Output
application	Deployed applications.	Application name.	Supported.
domain	Managed domain information.	Not used.	Unsupported.
group	Domain group information.	Group name.	Supported.
node	Managed node information.	Node name.	Unsupported.
service	Nodes visible to service discovery on the network that are not managed by the domain.	Node service name.	Supported.
session	User sessions with managed nodes.	User name.	Unsupported.

```
display domain
display domain type=application name=myApplication
display domain type=service detailed=true
```

Table 7.28. display command parameters

Name	Description	Required
name	Name filter. Name filtering is only supported on a subset of the display types. See Table 7.27 on page 136 for details.	No. No default value.
type	Display type indicator. Valid values are application, domain, group, node, service, or session.	No. Default value is domain.
detailed	Control detailed display output. Detailed output is only supported on a subset of the display types. See Table 7.27 on page 136 for details.	No. Default value is false.

endsession command

Terminate all user sessions with application nodes. If a user has no sessions with application nodes, this command quietly does nothing.

```
endsession domain
endsession domain username=aUser
```

Table 7.29. endsession command parameters

Name	Description	Required
username	Terminate sessions for this user.	No. Default value is login name of user executing administrative action.

removegroup command

Remove a group from the administration domain. Any nodes contained in this group are still managed by the domain.

```
removegroup domain groupname=myGroup
```

Table 7.30. removegroup command parameters

Name	Description	Required
groupname	Group name.	Yes.

removegroupnode command

Remove a managed node from a group. The node is still managed by the domain.

```
removegroupnode domain groupname=myGroup name=myNode
```

Table 7.31. removegroupnode command parameters

Name	Description	Required
groupname	Group name.	Yes.
name	Node name.	Yes.

removenode command

Remove a node from management by this domain.

```
removenode domain name=myNodeServiceName
```

Table 7.32. removenode command parameters

Name	Description	Required
name	Service name of node to add.	Yes.

manager target Display information on the TIBCO ActiveSpaces® Transactions **Administrator** server.

display command

Display the status of the TIBCO ActiveSpaces® Transactions **Administrator** server.

```
display manager
```

High Availability and Distribution

High availability and distribution management is accomplished using the `cluster`, `distribution`, and `partition` targets.

cluster target Manage the high-availability cluster

discover command

Verify connectivity to remote nodes.

This command ensures that the local node has network connectivity to all of the nodes in the `nodes` parameter. Success is reported if connectivity exists, otherwise a failure is returned.

```
discover cluster nodes="A,B"
```

Table 7.33. discover command parameters

Name	Description	Required
`nodes`	Comma separated list of nodes.	Yes.

display command

Display cluster information.

The `type` controls the information displayed. Valid values are:

- **classmismatches** - Display class mismatch information between the local and discovered remote nodes.
- **configuration** - Display local node configuration information.
- **local** - Display local node status information.
- **remote** - Display information about discovered remote nodes.

```
display cluster
display cluster type=configuration
display cluster type=local
display cluster type=remote
display cluster type=classmismatches
```

Table 7.34. display command parameters

Name	Description	Required
`type`	Optional information type to display. Valid values are one of `classmismatches`, `configuration`, `local`, or `remote`.	No. Default value is `remote`.

join command

Join the cluster.

Object migration is performed and active nodes are updated based on the partition definitions on the local node. This command blocks until all required object migration is complete. This is true for both synchronous and asynchronous partition definitions. A periodic percent complete message is returned while the join is executing. When this command returns any object migrations required to join the cluster have completed.

The `type` parameter controls the type of join that is performed. The valid join types are:

- **normal** - Join following a normal node restart with no partitioned objects in shared memory.
- **purge** - Join used to clear partitioned objects in shared memory. All partitioned objects on the local node are removed from shared memory before any object migrations occur.
- **merge** - Join used to recover from a multiple master scenario. Partitioned objects on the local node are merged with objects from the current active nodes specified using the `restorefrom-node` parameter. See the section called “define command” on page 141 for details.

This command will fail if connectivity cannot be established to all required nodes.

Concurrent execution of this command will fail.

```
join cluster
join cluster type=normal
join cluster type=purge
join cluster type=merge
```

Table 7.35. join command parameters

Name	Description	Required
`type`	Optional join type. Valid values are `normal` `purge`, or `merge`.	No. Default value is `normal`.

leave command

Leave the cluster.

The local node is removed from all partition definitions. If the local node was the active node, the next node in the node list is made the active node. If there are no replica nodes defined for partitions with the local node as the active node, the `type` parameter must be set to `force` to leave the cluster since these partitions will be abandoned.

This command will fail if connectivity cannot be established to all required nodes.

Concurrent execution of this command will fail.

```
leave cluster
leave cluster type=normal
leave cluster type=force
```

Table 7.36. leave command parameters

Name	Description	Required
`type`	Optional leave type. Valid values are `normal` and `force`. A value of `force` allows the local node to leave a cluster even if a partition will be abandoned.	No. Default value is `normal`.

distribution target Manage distribution services.

removenode command

Remove a discovered node.

```
removenode distribution remotenode=A
```

Table 7.37. removenode command parameters

Name	Description	Required
`remotenode`	Remote node name.	Yes.

globaltransaction target Manage global transactions for a node.

abort command

Abort a local transaction initiated by a global transaction. The abort command cannot be run on the node initiating the global transaction. It can only be run on remote nodes involved in the global transaction.

```
abort globaltransaction tranId="serializable:98752449084876475:221:33:360614189986813"
```

Table 7.38. abort command parameters

Name	Description	Required
`tranId`	Global transaction identifier.	Yes.

commit command

Commit a local transaction initiated by a global transaction. The commit command cannot be run on the node initiating the global transaction. It can only be run on remote nodes involved in the global transaction.

```
commit globaltransaction tranId="serializable:98752449084876475:221:33:360614189986813"
```

Table 7.39. commit command parameters

Name	Description	Required
`tranId`	Global transaction identifier.	Yes.

display command

Display global transactions running on this node.

```
display globaltransaction
display globaltransaction verbose=true
```

Table 7.40. display command parameters

Name	Description	Required
`verbose`	Include additional global transaction details. The `verbose` option only displays additional details if debug tracing is enabled for distribution.	No. Default value if `false`.

partition target Manage partitions.

define command

Define, or redefine, a partition definition.

Partitions should be defined on all nodes that need to be aware of the partition. This includes the active node, any replica nodes, and nodes that are using sparse partitions.

A `join cluster` command (the section called "join command" on page 139) or `enable partition` command (the section called "enable command" on page 143) must be executed after defining a partition to update the cluster with the new partition definitions.

Concurrent execution of this command will fail.

```
define partition name=MyPartition activenode=A
define partition name=MyPartition activenode=A replicas="B,C"
define partition name=MyPartition activenode=A replicas="B,C"
replicatypes="synchronous,asynchronous"
define partition name=MyPartition activenode=A replicas="B,C" replicaudit=discardreplica
define partition name=MyPartition activenode=A forcereplication=true
define partition name=MyPartition activenode=A objectspertransaction=10000
define partition name=MyPartition activenode=A numberofthreads=10
define partition name=MyPartition activenode=A restorefromnode=B
define partition name=MyPartition activenode=A broadcastupdates=false
define partition name=MyPartition activenode=A remoteenableaction=leavedisabled

//
//  Defines MyPartition on node D which is not the active or replica
//  node for the partition, i.e. a sparse partition.
//
servicename=D define partition name=MyPartition activenode=A replicas="B,C"
servicename=D define partition name=MyPartition activenode=A replicas="B,C"
sparseaudit=ignorenodelist
```

Table 7.41. define command parameters

Name	Description	Required
`name`	Partition name.	Yes.
`activenode`	Active node.	Yes.
`replicas`	Comma-separated list of replica nodes.	No. No default.
`replica-types`	Comma-separated list of replica types. Valid values are `synchronous` or `asynchronous`. If specified, the number of entries in the `replicatypes` list must be the same as the number of entries in the `replicas` parameter.	No. Default value is `synchronous`.

Name	Description	Required
broadcastupdates	Control broadcasting of partition updates to all nodes in the cluster when the partition is enabled. This option is used to support multiple-master testing. It should not be set to true in production deployments.	No. Default value is true.
forcereplication	Boolean controlling whether or not to force the copy of partitioned objects to all replica nodes when the partition's objects are migrated during a join cluster or a partition failover.	No. Default value is false.
objectspertransaction	The number of objects locked per transaction when the partition's objects are migrated during a join cluster or a partition failover. A value of 0 means that all objects are migrated in a single transaction.	No. Default value is 1000.
numberofthreads	The number of threads used when performing a partition migration during a join cluster or a partition failover. If the number of partition instances is less than the objectspertransaction property value, only one thread will be used. If the value of the objectspertransaction property is zero, the numberofthreads property is ignored, and all work is done in a single transaction.	No. Default value is 1.
remoteenableaction	Control whether this partition can be made active by an enable partition command on a remote node. Valid values are enablepartition and leavedisabled. Setting this property to enablepartition will allow this partition to be enabled by remote enable partitition commands. Setting this property to leavedisabled will prevent this partition from being enabled by remote enable partition commands.	No. Default value is enablepartition.
replicaaudit	An enumeration of the possible audits done when enabling a partition with replica nodes. Valid values are ignorereplica, discardreplica, or waitactive. ignorereplica ignores any non-active replica nodes when a partition is enabled. discardreplica removes any non-active replica nodes from the node list when a partition is enabled. waitactive waits for any non-active replica nodes when a partition is enabled. This parameter is quietly ignored if set for a partition without replica nodes.	No. Default value is waitactive.
restorefromnode	Define the node that the partition should be restored from when recovering from a multi-master scenario. The actual restore occurs when the join cluster type=merge command executes.	No. No default value. A cluster wide broadcast is used to determine the active node if not specified.
sparseaudit	An enumeration of the possible audits done when enabling a sparse partition. Valid values are verifynodelist and ignorenodelist. verifynodelist will audit that the active and replica nodes defined for the sparse partition match the active node's definition when the partition is enabled. ignorenodelist does not perform this audit. This property is quietly ignored if set for a non-sparse partition.	No. Default value is verifynodelist.

disable command

Disable a partition.

The partition is disabled on the local node. If the local node was the active node, the next node in the node list is made the active node. If there are no replica nodes defined for the partition with the local node as the active node, the `type` parameter must be set to `force` to disable the partition since the partition will be abandoned.

This command will fail if connectivity cannot be established to all required nodes.

Concurrent execution of this command will fail.

```
disable partition name=MyPartition
disable partition name=MyPartition type=normal
disable partition name=MyPartition type=force
```

Table 7.42. disable command parameters

Name	Description	Required
`name`	Partition name.	Yes.
`type`	Optional disable type. Valid values are `normal` and `force`. A value of `force` will disable the partition even if the partition will be abandoned.	No. Default value is `normal`.

display command

Display partition information.

```
display partition
display partition name=MyPartition
```

Table 7.43. display command parameters

Name	Description	Required
`name`	Optional partition name. If not specified, all partitions are displayed.	No. No default value.

enable command

Enable a partition.

Object migration is performed and the active node is updated based on the partition definition on the local node. This command blocks until the enable is complete.

The `type` parameter controls the type of enable that is performed. The valid enable types are:

- **normal** - Enable following a normal node restart with no partitioned objects in shared memory.
- **purge** - Enable used to clear partitioned objects in the partition being enabled. All partitioned objects on the local node in the partition being enabled are removed from shared memory before any object migrations occur.
- **merge** - Enable used to recover from a multiple master scenario. Partitioned objects on the local node in the partition being enabled are merged with objects from the current active node specified

using the `restorefromnode` parameter. See the section called “define command” on page 141 for details.

This command will fail if connectivity cannot be established to all required nodes.

Concurrent execution of this command will fail.

```
enable partition name=MyPartition
enable partition name=MyPartition type=normal
enable partition name=MyPartition type=purge
enable partition name=MyPartition type=merge
```

Table 7.44. enable command parameters

Name	Description	Required
`name`	Partition name.	Yes.
`type`	Optional join type. Valid values are `normal` `purge`, or `merge`.	No. Default value is `normal`.

migrate command

Migrate a partition.

This command can only be executed on the current active node for a partition and the partition state must be active.

When the migration completes, the partition definition has been updated with the values specified in the `activenode`, `replicas`, and `replicatypes` parameters. Any of these values may be different than the original partition definition. Each object in the partition is migrated as needed to the active and replica nodes as specified by the `activenode` and `replicas` parameters.

This command blocks until partition migration has completed.

Concurrent execution of this command will fail.

```
migrate partition name=MyPartition activenode=B
migrate partition name=MyPartition activenode=B replicas="A,C"
migrate partition name=MyPartition replicas="A,C" replicatypes="asynchronous,asynchronous"
migrate partition name=MyPartition activenode=B replicas="A,C" forcereplication=true
migrate partition name=MyPartition activenode=B replicas="A,C" objectspertransaction=100
migrate partition name=MyPartition activenode=B replicas="A,C" numberofthreads=5
migrate partition name=MyPartition activenode=B replicas="A,C" replicaaudit=discardreplica
migrate partition name=MyPartition activenode=B replicas="A,C" sparseaudit=ignorenodelist
```

Table 7.45. migrate command parameters

Name	Description	Required
`name`	Partition name.	Yes.
`activenode`	Active node for the partition.	No. Default value is the local node.
`replicas`	Comma-separated list of replica nodes.	No. No default.

Name	Description	Required
`replicatypes`	Comma-separated list of replica types. Valid values are `synchronous` or `asynchronous`. If specified, the number of entries in the `replicatypes` list must be the same as the number of entries in the `replicas` parameter.	No. Default value is `synchronous`.
`forcereplication`	Boolean controlling whether or not to force the copy of the partition's objects to all existing replica nodes during the partition migration. Note that this value applies only to this partition migration. It does not override the `forcereplication` value set when defining the partition.	No. Default value is `false`.
`objectspertransaction`	The number of objects locked per transaction during the partition migration. A value of 0 means that all objects are migrated in a single transaction. Note that this value applies only to this partition migration. It does not override the `objectspertransaction` value set when defining the partition.	No. Default value is 1000.
`numberofthreads`	The number of threads to use when performing partition migration. If the number of partition instances is less than the `objectspertransaction` property value, only one thread will be used. If the value of the `objectspertransaction` property is zero, the `numberofthreads` property is ignored, and all work is done in a single transaction. Note that this value applies only to this partition migration. It does not override the `numberofthreads` value set when defining the partition.	No. Default value is 1.
`replicaaudit`	An enumeration of the possible audits done when enabling a partition with replica nodes. Valid values are `ignorereplica, discardreplica`, or `waitactive`. `ignorereplica` ignores any non-active replica nodes when a partition is enabled. `discardreplica` removes any non-active replica nodes from the node list when a partition is enabled. `waitactive` waits for any non-active replica nodes when a partition is enabled. This parameter is quietly ignored if set for a partition without replica nodes.	No. Default value is `waitactive`.
`sparseaudit`	An enumeration of the possible audits done when enabling a sparse partition. Valid values are `verifynodelist` and `ignorenodelist`. `verifynodelist` will audit that the active and replica nodes defined for the sparse partition match the active node's definition when the partition is enabled. `ignorenodelist` does not perform this audit. This property is quietly ignored if set for a non-sparse partition.	No. Default value is `verifynodelist`.

update command

Update partition mappings.

This command updates the partition mapping of all instances in the partitions specified in the `names` parameter. The actual updates that occur is dependent on the installed partition mappers.

This command blocks until the update is complete.

Concurrent execution of this command will fail.

```
update partition names="MyPartition,MyOtherPartition
update partition names="MyPartition,MyOtherPartition" objectspertransaction=10000
update partition names="MyPartition,MyOtherPartition" numberofthreads=25
```

Table 7.46. update command parameters

Name	Description	Required
`names`	Comma separated list of partitions to update.	Yes.
`objectsper-transaction`	The number of objects locked per transaction during the partition update. A value of 0 means that all objects are updated in a single transaction. Note that this value applies only to this partition update. It does not override the `objectspertransaction` value set when defining the partition.	No. Default value is 1000.
`numberof-threads`	The number of threads to use when performing the partition update. If the number of partition instances is less than the `objectspertransaction` property value, only one thread will be used. If the value of the `objectspertransaction` property is zero, the `numberofthreads` property is ignored, and all work is done in a single transaction. Note that this value applies only to this partition update. It does not override the `numberofthreads` value set when defining the partition.	No. Default value is 1.

Java Virtual Machines

Java virtual machine management within a node is accomplished using the `jvm` target.

jvm target Manage the life-cycle of Java Virtual Machines.

display command

Display the status of deployed JVMs. If `name` parameter not specified information is displayed for all deployed JVMs.

```
display jvm
display jvm name=myJvmName
```

Table 7.47. display command parameters

Name	Description	Required
`name`	JVM name.	No. No default value.

remove command

Remove a stopped JVM.

```
remove jvm name=myJvmName
```

Table 7.48. remove command parameters

Name	Description	Required
name	JVM name.	Yes.

start command

Start a deployed JVM

```
start jvm name=myJvmName
```

Table 7.49. start command parameters

Name	Description	Required
name	JVM name.	Yes.

stop command

Stop a running JVM.

```
stop jvm name=myJvmName
```

Table 7.50. stop command parameters

Name	Description	Required
name	JVM name.	Yes.

Named Caches

Named cache management is accomplished using the cache target.

cache target Manage named caches.

add command

Add a new type to an existing named cache.

```
add cache name=MyCache type=managed.Type
```

Table 7.51. add command parameters

Name	Description	Required
name	Cache name.	Yes.
type	Fully scoped type name.	Yes.

create command

Create a named cache. Cache names are unique on a node. If the cache being created already exists, this command quietly does nothing. Newly created caches have an unlimited cache size.

```
create cache name=MyCache
```

Table 7.52. create command parameters

Name	Description	Required
name	Cache name.	Yes.

display command

Display information about named caches on a node. If name is not specified, information about all caches is displayed.

```
display cache
display cache name=MyCache
```

Table 7.53. display command parameters

Name	Description	Required
name	Cache name.	No. No default value.

remove command

Remove a named cache. All objects in the cache are restored to their default caching behavior.

```
remove cache name=MyCache
```

Table 7.54. remove command parameters

Name	Description	Required
name	Cache name.	Yes.

set command

Set or change cache size. When changing a cache size, any required object flushing will occur asynchronously.

Cache sizes can be set as a percentage of total shared memory in the node, or as an absolute size. When setting size using an absolute size these suffixes are supported:

- None - size in bytes.
- K - size in kilobytes.
- M - size in megabytes.
- G - size in gigabytes.

Percentage values are specified using a % suffix.

```
set cache name=MyCache size=0%    // Disables caching
set cache name=MyCache size=100%  // Unlimited cache size
set cache name=MyCache size=50%   // Size cache to 50% of node shared memory
set cache name=MyCache size=0     // Disables caching
set cache name=MyCache size=25000 // Set cache size to 25000 bytes
set cache name=MyCache size=25K   // Set cache size to 25 kilobytes
set cache name=MyCache size=2M    // Set cache size to 2 megabytes
set cache name=MyCache size=1G    // Set cache size to 1 gigabyte
```

Table 7.55. set command parameters

Name	Description	Required
`name`	Cache name.	Yes.
`size`	Cache size - percentage of shared memory or an absolute value.	Yes.

Nodes

Configuration management for a node is accomplished using the `configuration` target. Node management is accomplished using the `node` target. Nodes can be discovered using service discovery using the `services` target.

configuration target Manage configuration.

activate command

Activate configuration.

If there is already an active configuration with the same `type` and `name`, it is deactivated as part of activating the current configuration.

```
activate configuration type=MyConfigurationType name=myConfiguration version="1.0"
```

Table 7.56. activate command parameters

Name	Description	Required
`type`	Configuration type.	Yes.
`name`	Configuration name.	Yes.
`version`	Configuration version.	Yes.

clearhistory command

Clear configuration state change history.

Configuration change history that meets the specified filter criteria defined using the parameters to this command is cleared. The filter parameters have this precedence - `type`, `name`, `version`, `date`. To specify a lower precedent filter value, all higher precedent values must be specified. For example, if `name` is specified, `type` also must be.

The `date` parameter is used to remove configuration history older than the specified date.

```
clearhistory configuration
clearhistory configuration type=MyConfigurationType
clearhistory configuration type=MyConfigurationType name=myConfiguration
clearhistory configuration type=MyConfigurationType name=myConfiguration version="1.0"
clearhistory configuration type=MyConfigurationType name=myConfiguration version="1.0"
date="2005-07-31-23:59:59"
```

Table 7.57. clearhistory command parameters

Name	Description	Required
`type`	Configuration type.	No. No default value.
`name`	Configuration name.	No. No default value.

Name	Description	Required
`version`	Configuration version.	No. No default value.
date	Older than date. Format is *`%Y-%m-%d-%H:%M:%S`* where month (%m) is specified as a number (1-12), hour (%H) is between 0 and 23 inclusive, and a dash separates the date and time. For example, `2005-07-31-23:59:59`. Time value must be specified in the local timezone of the node.	No. No default value.

deactivate command

Deactivate configuration.

This command will leave the node with no active configuration. It is generally recommended to use the `activate` command to atomically deactivate and activate new configuration.

```
deactivate configuration type=MyConfigurationType name=myConfiguration version="1.0"
```

Table 7.58. deactivate command parameters

Name	Description	Required
`type`	Configuration type.	Yes.
`name`	Configuration name.	Yes.
`version`	Configuration version.	Yes.

display command

Display configuration.

The `type`, `name`, and `version` parameters are used as filters to control which configuration is displayed. The parameters have this precedence - `type`, `name`, `version`. To specify a lower precedent parameter, all higher precedent parameters must be specified. For example, if `name` is specified, `type` also must be.

```
display configuration
display configuration type=MyConfigurationType
display configuration type=MyConfigurationType name=myConfiguration
display configuration type=MyConfigurationType name=myConfiguration version="1.0"
display configuration history=true
```

Table 7.59. display command parameters

Name	Description	Required
`type`	Configuration type.	No. No default value.
`name`	Configuration name.	No. No default value.
`version`	Configuration version.	No. No default value.
history	Boolean to control display of configuration state change history. A value of `true` will display the history.	No. Default value is `false`.

export command

Export configuration data.

The exported configuration is in format that can be reloaded after making any required changes.

```
export configuration type=MyConfigurationType name=myConfiguration version="1.0"
```

Table 7.60. export command parameters

Name	Description	Required
`type`	Configuration type.	Yes.
`name`	Configuration name.	Yes.
`version`	Configuration version.	Yes.

load command

Load configuration data.

The configuration file has been loaded into shared memory after this command completes - but it is not active. It needs to be activated using the `activate` command to take affect.

```
load configuration source=myConfiguration.kcs
load configuration source=myConfiguration.kcs forcelocal=true
load configuration source=myConfiguration.kcs ignorethrottle=true
```

Table 7.61. load command parameters

Name	Description	Required
`source`	Configuration file.	Yes.
`forcelocal`	If `true`, this parameter loads the configuration file from the node's file system, rather than loading the configuration file from `administrator` client using the network connection to the node. If `true`, any path information in the `source` parameter must be valid for the node.	No. Default is `false`.
`ignorethrottle`	By default configuration files cannot be loaded if a node's shared memory utilization exceeds the configured throttle threshold (See Table 3.4 on page 49). Setting this parameter to `true` will load the configuration file even if shared memory utilization exceeds the configured throttle value. Set this value to `true` with caution since it can cause memory exhaustion on a running system.	No. Default is `false`.

loadactivate command

Load and activate configuration data.

The configuration files have been loaded into shared memory and activated after this command completes.

```
loadactivate configuration sourcelist="myConfiguration.kcs:myOtherConfiguration.kcs"
loadactivate configuration sourcelist=myConfiguration.kcs forcelocal=true
loadactivate configuration sourcelist=myConfiguration.kcs ignorethrottle=true
```

Table 7.62. load command parameters

Name	Description	Required
`sourcelist`	A colon separated list of configuration files.	Yes.
`forcelocal`	If `true`, this parameter loads the configuration file from the node's file system, rather than loading the configuration file from `administrator` client using the network connection to the node. If `true`, any path information in the `source` parameter must be valid for the node.	No. Default is `false`.
`ig-norethrottle`	By default configuration files cannot be loaded if a node's shared memory utilization exceeds the configured throttle threshold (See Table 3.4 on page 49). Setting this parameter to `true` will load the configuration file even if shared memory utilization exceeds the configured throttle value. Set this value to `true` with caution since it can cause memory exhaustion on a running system.	No. Default is `false`.

remove command

Remove configuration data from shared memory.

The configuration being removed cannot be active. Configuration that is removed from shared memory must be reloaded using either the `load` or `loadactivate` commands.

```
remove configuration type=MyConfigurationType name=myConfiguration version="1.0"
```

Table 7.63. remove command parameters

Name	Description	Required
`type`	Configuration type.	Yes.
`name`	Configuration name.	Yes.
`version`	Configuration version.	Yes.

node target Manage node life-cycle.

display command

Display node information.

```
display node
```

getadminport command

Get the administrative port for a node.

The only valid global parameter for this command is `hostname`. The `hostname` global parameter can be used to execute this command on a remote machine. SSH access is required to the machine specified in the `hostname` parameter. See the section called “Secure Shell (SSH)” on page 68 for details.

```
getadminport node
getadminport node installpath=/opt/tibco/RUN/A
hostname=acme.com getadminport node installpath=/opt/tibco/RUN/A
```

Table 7.64. getadminport command parameters

Name	Description	Required
installpath	Installation path.	No. Default is *<current working directory>/<local host name>*

install command

Install a node.

The only valid global parameter for this command is hostname. The hostname global parameter can be used to execute this command on a remote machine. SSH access is required to the machine specified in the hostname parameter. See the section called "Secure Shell (SSH)" on page 68 for details.

The javahome parameter can be used to configure the JRE or JDK used by the node. Usually, this parameter is not required since you will typically want the node to use the JRE shipped with the product.

The javalibrarypath, javabinarypath, and javaenvironment parameters can be used to update the node's environment, as required, for the JRE/JDK configured with the javahome parameter. The default values for these parameters are set for the JRE shipped with the product. If your JRE/JDK is a different version of that JRE then the default values for these options are likely correct for your JRE.

```
install node
install node application=kabira/kdm
install node adminport=1234 nodename="MyNode"
install node buildtype=DEVELOPMENT memorytype=sysvshm memorysize=2048 memoryallocators=4
install node description="my application node"
install node installpath=/opt/tibco/A configpath=/opt/tibco/configuration
deploydirectories=/opt/tibco/deploy
install node javahome=$JAVA_HOME
install node trustedhosts="host1,host2,host3" usehostaliases=false
hostname=acme.com install node installpath=/opt/tibco/RUN/A nodename=A
```

Table 7.65. install command parameters

Name	Description	Required
adminhost	Set the host address used to receive administrative commands.	No. Default is all network interfaces.
adminport	Set the port used to receive administrative commands. Note that the adminport parameter follows install node, and is not the adminport global parameter. Use of the adminport global parameter is illegal for the install command, since the node is not running when the command is executed.	No. Default is a random port.
application	Type of node to install. Supported node types are application or a domain manager. To install an application node, this parameter should be omitted. To install a domain manager node this parameter should have a value of kabira/kdm.	No. Default is an application node.
buildtype	Build type of node to be installed. Valid values are DEVELOPMENT and PRODUCTION.	No . Default is PRODUCTION.

Name	Description	Required
configpath	Directory in which to install configuration files. May be absolute or relative to the current directory.	No. Default is *<installpath>*/../configuration/*<nodename>*
deploydirectories	One or more directories where application JAR files can be installed. The directory names can be absolute or relative, and are separated with a :. All non-absolute paths are relative to the node installpath directory. If the directories do not exist, they are created during node installation. If a directory cannot be created, a warning message is generated, but node installation continues. Removing a node has no impact on the deploydirectories directories and their contents. All of the JAR files contained in these directories are automatically added to the class path for all JVMs started in the node. The directories are added in the order they are specified in the deploydirectories parameter. All file names within each directory are added using an ASCII sort order on the file names.	No. Default is *<installpath>*/../deploy.
description	A description string for the node, which will be part of the node service record for the node.	No. Default is an empty string.
discoveryenabled	A boolean property to enable or disable the discovery service.	No. Default is true.
discoveryport	Discovery port number.	No. Default value is 54321.
installpath	Directory in which to install the node. May be absolute or relative to the current directory. The directory cannot exist.	No. Default is *<current working directory>*/*<node name>*
javabinarypath	A list of JRE/JDK directories to add to the node's binary search path. Relative paths are relative to the directory specified in the javahome parameter. Use the : character to separate directories in the list. If this parameter is specified then the javahome parameter must also be specified.	No. Default is built-in value.
javaenvironment	A comma-separated list of name/value pairs specifying the environment variables to add to the node's environment for the JRE/JDK specified by the javahome parameter.	No. Default is an empty list.
javahome	Location of the JRE or JDK to be used by the node.	No. Default is built-in value.
javalibrarypath	A list of JRE/JDK directories to add to the node's library search path. Relative paths are relative to the directory specified in the javahome parameter. Use the : character to separate directories in the list. If this parameter is specified then the javahome parameter must also be specified.	No. Default is built-in value.
memoryallocators	Number of concurrent shared memory allocators.	No. Default value is based on number of cores in the machine.

Name	Description	Required
memorysize	Size of shared memory in megabytes.	No. Default value is 512.
memorytype	Type of memory to use for system shared memory. The value file indicates a memory mapped file. The value sysvshm is System V Shared memory.	No. Default value is file.
nodename	Node name.	No. Default value is local host name.
trustedhosts	Specifies trusted hosts for node. The value to this parameter is a comma-separated list of host names.	No. Default value is empty list.
usehostaliases	Boolean value that specifies whether the node should look for host aliases (in /etc/hosts and directory services) when defining trusted hosts.	No. Default value is true.

remove command

Remove a stopped node.

When a node is removed, the node coordinator is stopped if it is still running, and the node directory is deleted. If the node configuration directory was located outside the node directory, it is left intact.

Use the hostname and adminport or servicename global parameters to identify the node to be removed.

If the remove command fails because it cannot connect to the node, then use the installpath parameter to identify the node directory. To remove a failed node on a remote machine use the hostname global parameter. SSH access is required to the machine specified in the hostname parameter. See the section called "Secure Shell (SSH)" on page 68 for details.

```
remove node
remove node installpath=/opt/tibco/RUN/A
hostname=acme.com remove node installpath=/opt/tibco/RUN/A
```

Table 7.66. remove command parameters

Name	Description	Required
installpath	Installation path.	No. No default value.

restore command

Restore a node that was previously upgraded using the upgrade command.

The JAR files being restored must be copied into one of the deployment directories for the node before executing the restore command.

The restore command performs the following steps if the node is running:

1. Leave the cluster. All partitions with this node as the active node will migrate to a replica.
2. Stop the node.
3. Update the deployment directories for the node if deploydirectories is set on the command line.

4. Remove all managed objects for classes being restored.

5. Restart the node.

6. Add this node as a replica to all partitions that were active on this node and join the cluster.

The `restore` command performs the following steps if the node is not running:

1. Update the deployment directories for the node if `deploydirectories` is set on the command line.

2. Remove all managed objects for classes being restored.

3. Start the node.

Upon successful completion a restore report is generated containing information on what was restored.

```
restore node restorefile=version2.txt
restore node restorefile=version2.txt execute=true
restore node restorefile=version2.txt execute=true deploydirectories=/opt/tibco/deployV2
```

Table 7.67. restore command parameters

Name	Description	Required
`restorefile`	File containing the list of classes to restore. This is the `upgradefile` that was passed to the upgrade command. All non-absolute paths are relative to the node `installpath` directory.	Yes.
`allownonpartitioned`	A boolean value indicating whether a restore should be allowed if there are instances of non-partitioned objects in shared memory. Setting this value to `true` allows restores to continue, but all non-partitioned objects are deleted as part of the restore process.	No. Default value is `false`.
`execute`	A boolean value indicating whether the command should be executed. A value of `false` only generates the restore report. A value of `true` generates the report and also executes the command.	No. Default value is `false`.
`deploydirectories`	Updated deployment directories for the node. See Table 7.65 on page 153 for more details on this parameter.	No. Default is value set when node was installed.

start command

Start a node or restart the node coordinator.

The `start` command can be used to start a node, or restart the node coordinator after a `terminate` command. The `installpath` parameter controls whether the `start` command is being used to start the node, or to restart the node coordinator.

```
start node
start node maxretries=50
start node installpath=/opt/tibco/A
```

Table 7.68. start command parameters

Name	Description	Required
`installpath`	Node installation directory. May be absolute or relative to the current directory. This parameter must be specified to restart the node coordinator.	No. Default is *<current working directory>/<node name>.*
`maxretries`	Amount of time in seconds to wait for node to start.	No. Default value is 300 seconds.

stop command

Stop a running node.

When a node is stopped, the node coordinator is left running, and will not be stopped unless the node is removed. Executing the `start` command will restart all application processes.

```
stop node
```

terminate command

Terminate the node coordinator.

The `start` command can be used to restart the node coordinator.

```
terminate node
```

upgrade command

Upgrade a node.

This command upgrades the classes specified in the `upgradefile` parameter. This command must be used to upgrade nodes in a cluster. It can be run against a newly installed node, or a running node.

The new JAR files for the upgrade must be copied into a deployment directory for the node before performing the `upgrade` command.

The `upgrade` command performs the following steps if the node is running:

1. Validate that all classes being upgraded have a partition mapper installed, unless the `allownonpartitioned` parameter is set to `true`.
2. Leave the cluster. All partitions with this node as the active node will migrate to a replica.
3. Stop the node.
4. Update the deployment directories for the node if `deploydirectories` is set on the command line.
5. Remove all managed objects for classes being upgraded.
6. Restart the node.
7. Add this node as a replica to all partitions that were active on this node and join the cluster.

The `upgrade` command performs the following steps if the node is not running:

1. Update the deployment directories for the node if `deploydirectories` is set on the command line.
2. Remove all managed objects for classes being restored.
3. Start the node.

Upon successful completion an upgrade report is generated containing information on what was upgrade.

```
upgrade node upgradefile=version2.txt
upgrade node upgradefile=version2.txt execute=true
upgrade node upgradefile=version2.txt execute=true deploydirectories=/opt/tibco/deployV2
upgrade node upgradefile=version2.txt jvmnames=myApplication allownonpartitioned=true
```

Table 7.69. upgrade command parameters

Name	Description	Required
`upgradefile`	File containing the list of classes to upgrade. This file is generated by the upgrade tool. All non-absolute paths are relative to the node `installpath` directory.	Yes.
`allownonpartitioned`	A boolean value indicating whether an upgrade should be allowed if there are instances of non-partitioned objects in shared memory. Setting this value to `true` allows upgrades to continue, but all non-partitioned objects are deleted as part of the upgrade process.	No. Default value is `false`.
`execute`	A boolean value indicating whether the command should be executed. A value of `false` only generates the upgrade report. A value of `true` generates the report and also executes the command.	No. Default value is `false`.
`deploydirectories`	Updated deployment directories for the node. See Table 7.65 on page 153 for more details on this parameter.	No. Default is value set when node was installed.
`jvmnames`	A comma-separated list of JVM names to be used when searching for JVM specific deploy directories, when updating into a freshly installed node.	No. Default value is an empty list.

services target Use service discovery to discover nodes on the local network.

display command

Display services.

```
display services timeout=30
display services timeout=30 servicename=A
display services timeout=30 detailed=true
display services timeout=30 properties="property2=value1,property2=value2"
```

Table 7.70. display command parameters

Name	Description	Required
`servicetype`	Service type to discover.	No. Default is show all service types.

Name	Description	Required
`servicename`	Service name to discover.	No. Default is show all discovered services.
`timeout`	Number of seconds to wait for results.	No. Default is two seconds.
`detailed`	Boolean value controlling detailed output.	No. Default is `false`.
`properties`	Comma separated list of *<name>=<value>* pairs used to filter returned services.	No. Default value is an empty list.

Security

Control of external LDAP servers used for authorization is accomplished using the `ldap` target. User management is done using the `security` target.

ldap target Manage LDAP authentication servers.

disable command

Disable an LDAP server.

```
disable ldap server=myLdapServer
```

Table 7.71. disable command parameters

Name	Description	Required
`server`	Configured LDAP server name.	Yes.

display command

Display the status of LDAP servers.

The `server` and `source` parameters are used to filter the output. If neither is specified, all configured LDAP servers will be displayed. If `server` is specified, only that server will be displayed. If `source` is specified, all LDAP servers configured for that authentication source will be displayed.

```
display ldap
display ldap server=myLdapServer
display ldap source=myLdapAuthenticationSource
```

Table 7.72. display command parameters

Name	Description	Required
`server`	Configured LDAP server name.	No. No default value.
`source`	Configured authentication source.	No. No default value.

enable command

Enable an LDAP server.

```
enable ldap server=myLdapServer
```

Table 7.73. enable command parameters

Name	Description	Required
server	Configured LDAP server name.	Yes.

security target Manage node security.

add command

Add a user.

The user being added cannot already exist on this node.

The administrator executing this command will be prompted for the new user's password, unless the deferredpassword parameter is set to true.

```
add security username=auser roles=switchmonitor
add security username=auser roles=switchmonitor passwordexpirationdays=10
trustedhostuser=true allowemptypassword=false
add security username=auser roles=switchmonitor passwordrequired=true deferredpassword=true
```

Table 7.74. add command parameters

Name	Description	Required
username	User name.	Yes.
roles	Comma separated list of roles.	Yes.
passwordexpira-tiondays	Password expiration time, in days. A value of 0 means that the password does not expire.	No. Default value is 0.
trustedhostuser	Boolean indicating whether the user may only be authenticated when connecting from a trusted host.	No. Default value is false.
allowemptypass-word	Boolean indicating whether to allow an empty password for this user.	No. Default value is true.
passwordrequired	Boolean indicating whether a password is always required. If true, the user must always present a password during authentication, and cannot use the trusted host facility.	No. Default value is false.
deferredpassword	If true, password definition is deferred until the initial authentication event.	No. Default value is false.

audit command

Audit security policy configuration for administration targets.

If the name parameter is not specified, auditing is performed on all installed administration targets.

```
audit security
audit security name=configuration
```

Table 7.75. audit command parameters

Name	Description	Required
`name`	Target name.	No. No default value.

display command

Display security information.

Table 7.76. Display types

Type	Description	Name	Detailed
`accesscontrol`	Access control rules.	Fully scoped type name.	Yes.
`audit`	Audit rules.	Not used.	No.
`authenticationsources`	Authentication sources.	Authentication source name.	No.
`hosts`	Trusted hosts.	Host name.	No.
`principals`	Defined principals.	Principal name.	Yes.

```
display security type=accesscontrol
display security type=audit
display security type=authenticationsources name=Local
display security type=hosts
display security type=principals detailed=true
```

Table 7.77. display command parameters

Name	Description	Required
`type`	Type of information to display. See Table 7.27 on page 136 for supported values.	Yes.
`name`	Filter output. See Table 7.27 on page 136 for details on name filtering.	No. No default value.
`detailed`	Display detailed output. See Table 7.27 on page 136 for details on which display types support detailed output. It is not an error to specify a value of `true` for types that do not support detailed output.	No. Default value is `false`.

export command

Export user configuration.

The exported configuration can be saved as a file and reloaded as configuration data to restore all user definitions for a node.

The `users` parameter can be used to restrict the export to a subset of the users defined on the node.

```
export security name=myUsers version="1.0"
export security name=myUsers version="1.0" users="auser,anotheruser"
```

Table 7.78. export command parameters

Name	Description	Required
`name`	Configuration name.	Yes.
`version`	Configuration version.	Yes.
`users`	Comma separated list of users to include in the configuration.	No. No default value.

remove command

Remove a user.

```
remove security username=auser
```

Table 7.79. remove command parameters

Name	Description	Required
`username`	User name.	Yes.

reset command

Reset a user's password.

The administrator executing this command will be prompted for the new password, unless the `deferredpassword` parameter is set to true.

```
reset security username=auser
reset security username=auser deferredpassword=true
```

Table 7.80. reset command parameters

Name	Description	Required
`username`	User name.	Yes.
`deferredpassword`	If `true`, password definition is deferred until the initial authentication event.	No. Default value is `false`.

update command

Update a user's security parameters.

The parameter values specified in this command update the current values for the user.

This command cannot be used to change a user's password. Use the `reset` command to change a user's password.

```
update security username=auser roles="switchmonitor,switchadmin"
update security username=auser roles=switchmonitor passwordexpirationdays=20
trustedhostuser=false allowemptypassword=true
update security username=auser roles=switchmonitor passwordrequired=false
```

Table 7.81. update command parameters

Name	Description	Required
`username`	User name.	Yes.

Name	Description	Required
`roles`	Comma separated list of roles.	Yes.
`passwordexpira-tiondays`	Password expiration time, in days. A value of 0 means that the password does not expire.	No. Default value is 0.
`trustedhostuser`	Boolean indicating whether the user may only be authenticated when connecting from a trusted host.	No. Default value is `false`.
`allowemptypass-word`	Boolean indicating whether to allow an empty password for this user.	No. Default value is `true`.
`passwordrequired`	Boolean indicating whether a password is always required. If `true`, the user must always present a password during authentication, and cannot use the trusted host facility.	No. Default value is `false`.

Statistics

Access and control of runtime statistics collection is done using the `statistics` target.

Details on the statistics data gathered, and their meaning, can be found in the TIBCO ActiveSpaces® Transactions **Performance Tuning Guide**.

`statistics` target Access and control runtime statistics collection.

Supported statistics are defined in Table 7.82 on page 163. This columns in the table have these meanings:

- **Statistic** - Statistic name
- **Description** - Brief description.
- **Enable / Disable** - Yes indicates that the statistic must be explicitly enabled to start collection and disabled to stop collection.
- **Clear** - Yes indicates that the statistic value can be cleared.
- **Detailed** - Yes indicates that detailed output is available for the statistic.
- **Filter** - Indicates whether filtering is support for the statistic. For statistics that support filtering, the valid filter values are `java` or `none`. A value of `java` indicates that statistics are only reported for Java managed objects. A value of `none` reports statistics for all managed objects.

Table 7.82. Supported statistics

Statistic	Description	Enable / Disable	Clear	Detailed	Filter
`activetransac-tions`	Active Transactions	No	No	No	No
`allocationsummary`	Allocator Allocation Summary	No	No	No	No
`allocatorbuckets`	Allocator Buckets	No	No	Yes	No

Statistic	Description	Enable / Disable	Clear	Detailed	Filter
allocatorsummary	Allocator Summary	No	No	No	No
blockedtransactions	Blocked Transactions	No	No	No	No
businessstatemachine	Business State Machine	No	Yes	No	No
cpu	CPU Utilization	Yes	No	No	No
deadlock	Deadlock	No	Yes	Yes	No
disk	Disk	Yes	No	No	No
distributionchannel	Distribution Network	Yes	Yes	Yes	No
distributionnode	Distribution	No	Yes	No	No
engine	Node Processes	No	No	No	No
eventbus	Shared Memory IPC	No	Yes	Yes	No
eventlog	Shared Memory IPC Detailed	Yes	Yes	No	No
files	Files	No	No	No	No
hash	Shared Memory Hashing	No	No	Yes	No
host	Kernel Information	No	No	No	No
javacache	JNI Cache	No	Yes	No	No
javatransaction	Transaction	Yes	Yes	Yes	No
jni	Runtime JNI Calls	Yes	Yes	No	No
ktsevents	Operator Events	No	Yes	No	No
localmutex	Local Mutex	Yes	Yes	Yes	No
memoryusage	Memory Usage	No	No	No	No
mutex	Shared Memory Mutex	Yes	Yes	Yes	No
namedcache	Named Caches	No	Yes	No	No
native	Native Runtime Calls	No	Yes	No	No
navigation	Query	No	Yes	No	java or none. Default is java.
network	Network	Yes	No	No	No
object	Objects	No	Yes	No	java or none. Default is java.
partition	Partition	No	Yes	No	No
sar	System Activity Reporter	Yes	No	No	No
systeminfo	System Information	No	No	No	No
threads	System Threads	No	No	No	No
timer	Timers	No	No	No	No
transaction	Transaction Locking	Yes	Yes	Yes	java or none. Default is java.

Statistic	Description	Enable / Disable	Clear	Detailed	Filter
`transactioncontention`	Transaction Contention	Yes	Yes	No	`java` or `none`. Default is `java`.
`transactionpromotion`	Transaction Promotion	Yes	Yes	No	`java` or `none`. Default is `java`.
`vmstat`	Virtual Memory	Yes	No	No	No

clear command

Clear current statistic values. Clearing a statistic that does not support clear has no effect. See Table 7.82 on page 163 for a list of statistics that support clearing.

```
clear statistics statistics="transaction,mutex"
```

Table 7.83. clear command parameters

Name	Description	Required
`statistics`	Comma separated list of statistics. See Table 7.82 on page 163 for a complete list of valid statistics.	Yes.

disable command

Disable collection of statistics. Disabling a statistic that does not support disable has no effect. See Table 7.82 on page 163 for a list of statistics that support disable.

```
disable statistics statistics="transaction,mutex"
```

Table 7.84. disable command parameters

Name	Description	Required
`statistics`	Comma separated list of statistics. See Table 7.82 on page 163 for a complete list of valid statistics.	Yes.

display command

Display statistic values.

Displaying a statistic that has no data available will generate empty output.

```
display statistics statistics="transaction,mutex"
display statistics statistics="transaction,mutex" detailed=true
display statistics statistics="object,transaction" filter=none
display statistics statistics="transaction,mutex" label="Transaction & Mutex Report"
```

Table 7.85. display command parameters

Name	Description	Required
`statistics`	Comma separated list of statistics. See Table 7.82 on page 163 for a complete list of valid statistics.	Yes.
`detailed`	Boolean indicating whether detailed values should be displayed. See Table 7.82 on page 163 for statistics that support detailed output.	No. Default value is `false`.

Name	Description	Required
filter	Filter statistics output. See Table 7.82 on page 163 for statistics that support filtering and valid filter values. Only a single filter value may be specified for a list of statistics.	No. Default value is statistics specific.
label	A value to add to the header in non-delimited output.	No. No default.

enable command

Enable collection of statistics. Enabling a statistic that does not support enable has no effect. See Table 7.82 on page 163 for a list of statistics that support enabling.

```
enable statistics statistics="transaction,mutex"
```

Table 7.86. enable command parameters

Name	Description	Required
statistics	Comma separated list of statistics. See Table 7.82 on page 163 for a complete list of valid statistics.	Yes.

snapshot command

Display a timed snapshot of statistic values.

Internally the snapshot command performs these steps in order - *warm up sleep*, disable, clear, enable, *time sleep*, disable, display. The *warm up sleep duration* is controlled by the warmupseconds parameter. The *time sleep* duration is controlled by the time parameter.

```
snapshot statistics statistics="transaction,mutex" seconds=10
snapshot statistics statistics="transaction,mutex" seconds=10 warmupseconds=10
detailed=true
snapshot statistics statistics="object,transaction" seconds=10 filter=none
snapshot statistics statistics="transaction,mutex" seconds=10 warmupseconds=10
serialize=true
```

Table 7.87. snapshot command parameters

Name	Description	Required
statistics	Comma separated list of statistics. See Table 7.82 on page 163 for a complete list of valid statistics.	Yes.
seconds	The number of seconds to collect data.	Yes.
warmupseconds	The number of seconds to wait before starting the snapshot collection cycle.	No . Default value is 0.
detailed	Value passed to the display command in the snapshot collection cycle.	No. No default value.
filter	Value passed to the display command in the snapshot collection cycle.	No. No default value.
serialize	Serialize collection of statistics if true. A separate collection cycle is executed for each statistic being collected, starting with the *warm up sleep*.	No. Default value is false.

status command

Display current collection status of statistics that can be enabled and disabled.

```
status statistics
```

supported command

Display supported statistics.

```
supported statistics
```

Configuration file syntax

This section describes the syntax of TIBCO ActiveSpaces® Transactions configuration. TIBCO ActiveSpaces® Transactions configuration files must have a suffix of `.kcs` (Kabira Configuration Service).

Conventions

Words that are not white space, within comments, or formed with capital letters and underscores, are literal within the examples in this section. Examples include the words `configuration` and `version` and the characters `{ }`.

Syntax

The TIBCO ActiveSpaces® Transactions configuration syntax is shown below in an EBNF-like syntax. Capital letters are grammar rules, lower case strings are literal, and punctuation is literal with the following exceptions:

- parentheses denote an optional grammar element. It may appear zero or one time.
- parentheses followed by a plus sign denote a mandatory grammar element that may appear one or more times.
- parentheses followed by an asterisk denote an optional grammar element that may appear zero or more times.

Example 7.7. Configuration file syntax

```
CONFIGURATION
        configuration STRING
        version STRING
        type STRING
        {
                (CONFIG_BLOCK)+
        };
CONFIG_BLOCK
        configure IDENTIFIER
        {
                (CONFIG_CLASS)+ | (CONFIG_BLOCK)*
        }
CONFIG_CLASS
        IDENTIFIER
        {
                (FIELD)*
        };
```

```
FIELD
        IDENTIFIER = VALUE;
VALUE
        PRIMITIVE_VALUE
        | ARRAY_VALUE
        | OBJECT_VALUE
ARRAY_VALUE
        {
                (VALUE (, VALUE)*)*
        }
OBJECT_VALUE
        {
                (IDENTIFIER = VALUE;)*
        }
```

Be aware of the following specific syntax features:

- outside of quoted strings, white space (spaces, tabs, newlines, etc.) are ignored.
- white space is preserved within quoted strings.
- Java comments are supported.
- a configuration file may only contain a single configuration specification.

Strings The only string character that needs to be escaped is a double-quote - "". An un-escaped double-quote terminates a literal string. To add a double quote in a string, escape it with a back-slashes - "\". For example:

```
myString = "this is \"a quoted\" string";
```

All other back-slashes are treated as character literals.

For example:

```
//
//    Configuration file syntax
//
myString = "this is single quoted \'c\' entry";
myOtherString = "this is \\"a quoted\\" string";

//
//    Configuration data in memory
//
this is single quoted \'c\' entry   // myString
this is \"a quoted\" string     // myOtherString
```

String values are 8-bit clean, but do not support embedded null values, or multi-byte character values with embedded null values.

Definitions

These definitions are used in the section called “Syntax” on page 167.

STRING A literal string enclosed within leading and trailing double quote characters. A string may span multiple lines.

IDENTIFIER An unquoted string containing no white space, or a *STRING*. Follows the same rules as Java identifiers. *IDENTIFIER* is used for scope, class, field and object element names.

CONFIG_BLOCK *CONFIG_BLOCKs* define the scope for the *CONFIG_CLASS* identifiers. There must be a `configure` *IDENTIFIER* block to define the package scope for each *CONFIG_CLASS*.

CONFIG_CLASS `INDENTIFIERs` for a configuration class are an unscoped `IDENTIFIER`. A configuration class may contain zero or more *FIELDs*.

FIELD An *IDENTIFIER* which is the unscoped name of the configuration value. Fields may appear in any order within a configuration class. It is an error to specify the same field more than once. A field value may be a simple value, an array value or an object value.

OBJECT_VALUE An *IDENTIFIER* which is the unscoped name of an object element.

VALUE Initialization data for a field. The value must match the type of the field it is being applied to.

Values fall into three categories, primitive values, array values, and object values. A primitive value is one of the following:

- **String** - multi-byte clean, string literals that are loaded without any conversion.
- **Long** - Decimal, hexadecimal, and octal representations using Java conventions. A negative number must be indicated with a leading "-".
- **Boolean** - `true/false` and `TRUE/FALSE`.
- **Enumerations** - Scoped name of an enumeration value. The scope is either a fully scoped package name or the enumeration name.

```
package my.enum.test;

import com.kabira.platform.kcs.Configuration;

enum Numbers
{
    ONE,
    TWO,
    THREE
}

public class SomeClass extends Configuration
{
    public Numbers      myNumbers;
}


configuration "myconfiguration" version "1.0" type "enumtest"
{
        configure my.enum.test
        {
                SomeClass
                {
                        //
                        // Package scoped enumeration
                        //
                        myNumbers = my.enum.test.Numbers.ONE;
                };
                SomeClass
                {
                        //
                        // Enumeration name scoping
                        //
                        myNumbers = Numbers.ONE;
```

```
                };
        };
};
```

- Date - A string with a format of *Y-M-D T* where:
  - `Y` is the year, including a century, expressed as a decimal number, e.g. `2011`.
  - `M` is the month expressed as a decimal number from 01 to 12, e.g. `12` for December.
  - `D` is the day of the month expressed as a decimal number from 01 to 31, e.g. `04`.
  - `T` is the time expressed as `H:M:S` where `H` is a decimal number between representing a 24 hour clock from 00 to 23, `M` is the minute expressed as a decimal number from 00 to 59, and `S` is the second expressed as a decimal number from 00-60, e.g. `11:05:23`.
- **Byte** - A single character represented using supported Java character constant syntax (e.g. 'a' or '\n'). Or an unsigned value between 0 and 255.

An array value is a comma-separated list, within curly braces. The final element of the list does not have a trailing comma.

```
FIELD = { “red”, “green”, “yellow” };
```

Arrays may be of primitive types, objects, or arrays.

An object value is enclosed in curly braces, and has individual object element names.

```
FIELD =
{
        ELEMENT_NAME = VALUE;
        ELEMENT_NAME = VALUE;
};
```

Objects may contain elements of primitive types, arrays, or objects.

Example

Here is a simple sample configuration file:

Example 7.8. Configuration file example

```
configuration "simple config" version "1.2A" type "simple"
{
    // a CONFIG_BLOCK
    configure simple.name
    {
        // a CONFIG_CLASS
        Person
        {
            // string FIELD
            name = "Sir Raleigh Walter";

            // Long FIELD
            age = 42;

            //  Boolean FIELD
            lactoseIntolerant = false;
        };
    };
```

```
    // another CONFIG_BLOCK
    configure popmusic.bands
    {
        // a CONFIG_CLASS
        Band
        {
            // String FIELD
            bandName = "The Beatles";

            // A ARRAY_VALUE of OBJECT_VALUEs
            bandMembers =
            {
                // A OBJECT_VALUE
                {
                    name = "John Lennon";
                    instrument = Guitar;
                },

                // Another OBJECT_VALUE
                {
                    name = "Paul McCartney";
                    instrument = Bass;
                },

                // Yet another OBJECT_VALUE
                {
                    name = "George Harrison";
                    instrument = Guitar;
                },

                // The last OBJECT_VALUE
                {
                    name = "Ringo Starr";
                    instrument = Drums;
                }
            };

            // A ARRAY_VALUE of Strings
            songList =
            {
                "Hey Bulldog",
                "Only a Northern Song",
                "All Together Now"
            };
        };
    };
};
```

Default distribution configuration

This is the default distribution configuration.

Example 7.9. Default distribution configuration

```
// Name
//    $RCSfile: distribution.kcs,v $
//
// History
//    $Revision: 1.1.2.2.4.1 $ $Date: 2015/05/14 14:09:17 $
//
// COPYRIGHT
//    Copyright 2010-2015 TIBCO Software Inc. ALL RIGHTS RESERVED.
//    TIBCO Software Inc. Confidential Information
```

```
//
// Description
//    Distribution configuration
//

configuration "distribution" version "1.0" type "distribution"
{
    configure switchadmin
    {
        configure Distribution
        {
            //
            // High-Availability configuration data.
            //
            // Only one instance of this configuration can be
            // specified.
            //
            HighAvailability
            {
                //
                // Enable or disable node quorum behavior
                //
                nodeQuorum = Disable;

                //
                // Number of remote nodes required for a quorum.
                // This is ignored if nodeQuorum is Disable.
                //
                minimumNumberQuorumNodes = 0;

                //
                // Keep-alive send interval
                //
                keepAliveSendIntervalSeconds = 1;

                //
                 // Keep-alive non-response timeout interval
                // following a send.
                //
                nonResponseTimeoutSeconds = 2;
            };

            //
            // Transaction configuration data.
            //
            // Only one instance of this configuration can be
            // specified.
            //
            Transaction
            {
                //
                // The number of seconds to wait for a
                // distributed lock.  If this time value
                // expires, a deadlock is thrown and the
                // transaction is retried.
                //
                timeoutSeconds = 60;

                //
                // The maximum number of committed or aborted
                // transactions to retain for each remote node.
                // This is used for recovery processing to
                // determine the outcome of global transactions
                // if a remote node crashes and is restarted
                // with a loss of shared memory.
                //
                numberCompletedTransactions = 1000;
```

```
};


//
// Dynamic discovery configuration data.
//
// Only one instance of this configuration can be
// specified.
//
DynamicDiscovery
{
    //
    // Enable or disable dynamic discovery
    //
    enabled = true;
};

//
// Transport configuration data.
//
// Only a single instance of this interface can be
// specified.
//
Transport
{
    //
    // Listener address list.
    //
    // A list of TCP network addresses on which to
    // start a listener.  Remote nodes communicate
    // to this node using these listeners.
    //
    // The format of each specified network
    // listener address is:
    //
    //    IPv4|IPv6:<host address>:<port number>
    //
    // host address - default value is all
    //           interfaces
    // port number - default value is 5557
    //
    // For example to listen on all IPv4 interfaces
    // on port number 5557 would require the
    // following entry:
    //
    //    { IPv4:: };
    //
    // To listen on all IPv6 interfaces on port 5558
    // would require the following entry:
    //
    //     { IPv6::5558 };
    //
    // To listen on both of these requires the
    // following entry:
    //    {
    //        IPv4::,
    //        IPv6::5558
    //    };
    //
    // This value is audited at configuration
    // activation.
    //
    listenerAddressList =
    {
        "IPv4::"
    };
```

```
                //
                // The number of ports to search before
                // reporting a listener failure.  The search
                // is started at the configured network
                // listener port number and then incremented
                // by one on each failure up to this value.
                //
                // A value of 0 disables port search.
                //
                numberSearchPorts = 20;

                //
                // The amount of time to wait for a remote
                // node to move into the Active state before
                // a resource unavailable exception is raised.
                //
                // The wait is done for a partition's active
                // node when enabling a partition, or
                // for each remote node when a discover
                // cluster command is executed.
                //
                // This value must be > 0.
                //
                nodeActiveTimeoutSeconds = 60;

                //
                // Control enabling of TCP_NODELAY socket
                // option on connection establishment.
                //
                tcpNoDelayEnabled = true;

                //
                // Control maximum PDU size used for
                // distributed communications.
                //
                // This value must be > 4000.
                //
                maximumPDUSizeBytes = 1000000;
            };
        };
    };
};
```

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

T

U