JBoss Transactions API 4.2

Installation Guide

JBTA-IG-3/23/06

Legal Notices

The information contained in this documentation is subject to change without notice.

Arjuna Technologies Limited makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Arjuna Technologies Limited shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Java™ and J2EE is a U.S. trademark of Sun Microsystems, Inc. Microsoft® and Windows NT® are registered trademarks of Microsoft Corporation. Oracle® is a registered U.S. trademark and Oracle9™, Oracle9 Server™ Oracle9 Enterprise Edition™ are trademarks of Oracle Corporation. Unix is used here as a generic term covering all versions of the UNIX® operating system. UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

Arjuna is a trademark of Hewlett-Packard Company.

Copyright

Software Version

JBoss Transactions API 4.2

Restricted Rights Legend

Contents

About This Guide

What This Guide Contains

The Installation Guide contains information on how to use JBoss Transactions API 4.2.

Audience

This guide is most relevant to engineers who are responsible for installing JBoss Transactions API 4.2 installations.

Prerequisites

None.

Organization

This guide contains the following chapters:

- **Chapter 1, Installing ArjunaTA:** contains information on the way to install ArjunaTA

Documentation Conventions

The following conventions are used in this guide:

Convention	Description
Italic	In paragraph text, italic identifies the titles of documents that are being referenced. When used in conjunction with the Code text described below, italics identify a variable that should be replaced by the user with an actual value.
Bold	Emphasizes items of particular importance.
`Code`	Text that represents programming code.
Function | **Function**	A path to a function or dialog box within an interface. For example, "Select File | Open." indicates that you should select the Open function from the File menu.
( ) and |	Parentheses enclose optional items in command syntax. The vertical

	bar separates syntax items in a list of choices. For example, any of the following three items can be entered in this syntax: `persistPolicy (Never | OnTimer | OnUpdate | NoMoreOftenThan)`
Note: **Caution**:	A note highlights important supplemental information. A caution highlights procedures or information that is necessary to avoid damage to equipment, damage to software, loss of data, or invalid test results.

Table 1 Formatting Conventions

Additional Documentation

In addition to this guide, the following guides are available in the JBoss Transactions API 4.2 documentation set:

- JBoss Transactions API 4.2 *Release Notes*: Provides late-breaking information about JBoss Transactions API 4.2.
- JBoss Transactions API 4.2 *Administration Guide*: This guide provides instructions for administering JBoss Transactions API 4.2.
- JBoss Transactions API 4.2 *Programmer's Guide*: Provides guidance for writing applications.

Contacting Us

Questions or comments about JBoss Transactions API 4.2should be directed to our support team. Send email to **`support@arjuna.com`**.

Chapter 0

Installing ArjunaTA

Preparing your system

Before installing the *ArjunaTA* software, we recommend the following administrative steps be taken, assuming a default configuration for *ArjunaTA*:

- Install the distribution into the required location.
- *ArjunaTA* requires a minimum object store for storing the outcome of transactions in the event of system crashes. The location of this should be specified in the properties file using the `com.arjuna.ats.arjuna.objectstore.objectStoreDir` environment variable, e.g., `java -Dcom.arjuna.ats.arjuna.objectstore.objectStoreDir=C:\temp foo`.
- By default, all object states will be stored within the `defaultStore` subdirectory of the object store root, e.g., `/usr/local/Arjuna/TransactionService/ObjectStore/defaultStore`. However, this subdirectory can be changed by setting the `com.arjuna.ats.arjuna.objectstore.localOSRoot` property variable accordingly.

ObjectStore management

Within the transaction service installation, the object store is updated regularly whenever transactions are created, or when *Transactional Objects for Java* is used. In a failure free environment, the only object states which should reside within the object store are those representing objects created with the *Transactional Objects for Java* API. However, if failures occur, transaction logs may remain in the object store until crash recovery facilities have resolved the transactions they represent. As such it is very important that the contents of the object store are not deleted without due care and attention, as this will make it impossible to resolve in doubt transactions. In addition, if multiple users share the same object store it is important that they realise this and do not simply delete the contents of the object store assuming it is an exclusive resource.

Operating System Services

It is possible to run the recovery manager and transaction server as operating system services.

Installing and Uninstalling

This section explains how to install and uninstall these operating system services for the two main types of operating system: Windows and UNIX (Solaris, Linux and HP-UX).

Windows

There are four scripts provided which will install/uninstall the recovery manager and transaction server services, these files can be found in `Services\bin\windows`:

- Recovery Manager Service
  - `InstallRecoveryManagerService-NT.bat` – running this script will install the Recovery Manager as a Windows service.
  - `UninstallRecoveryManagerService-NT.bat` – running this script will uninstall the Recovery Manager as a Windows service.
- Transaction Server
  - `InstallTransactionServerService-NT.bat` – running this script will install the Transaction Manager as a Windows service.
  - `UninstallTransactionServerService-NT.bat` – running this script will uninstall the Transaction Server as a Windows service.

Please note that you must either be logged on as an administrator or have administrator privileges to install/uninstall a Windows service.

Once you have run an install script you should see the following message:

```
wrapper  | Arjuna Transaction Service <service name> installed.
```

This indicates that the service has been installed successfully. The service should also be visible in the services list from the Control Panel (see Figure 1 Services list from the Control Panel)

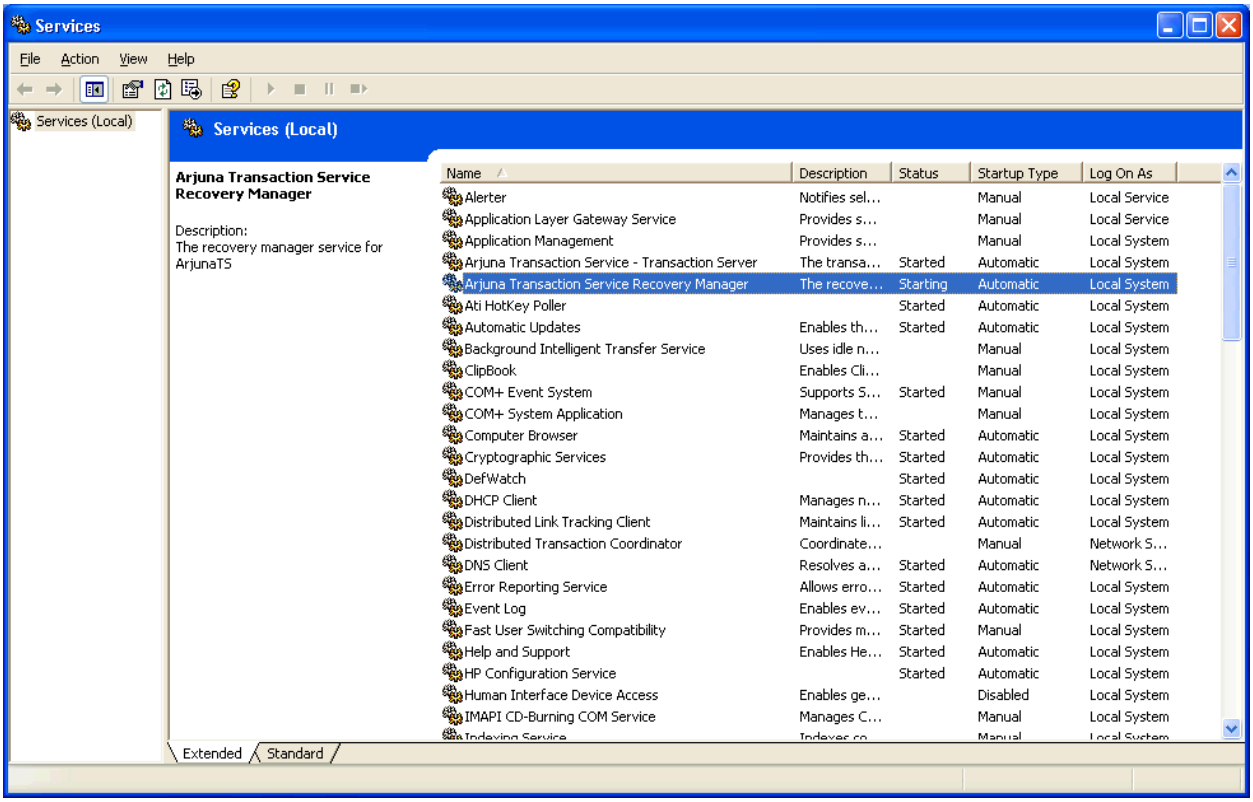

Figure 1 Services list from the Control Panel

Once you have run an uninstall script you should see the following message:

```
wrapper  | Arjuna Transaction Service <service name> removed.
```

This indicates that the service has been uninstalled successfully.

UNIX

To install the services on Unix machines perform the following steps:

1. Make sure that you are logged in as the user `root', as the installer needs to create files under the directory ``/etc''.

2. Change directory to `services/installer`.

3. Make sure the environment variable `JAVA_HOME` is set to the home directory of the JVM you wish the services to be run as. For example:

```
JAVA_HOME=/opt/java; export JAVA_HOME     (for sh)

setenv JAVA_HOME /opt/java                (for csh)
```

4. Run the installer script:

```
./install_services.sh
```

The following is example output from a computer running the Solaris operating system:

```
# ./install_services.sh

Adding $JAVA_HOME (/opt/java) to $PATH in
/opt/arjuna/ats-3.2/services/bin/solaris/recoverymanagerservice.sh
Adding $JAVA_HOME (/opt/java) to $PATH in
/opt/arjuna/ats-3.2/services/bin/solaris/transactionserverservice.sh
Installing shutdown scripts into /etc/rcS.d:
  K01recoverymanagerservice
  K00transactionserverservice
Installing shutdown scripts into /etc/rc0.d:
  K01recoverymanagerservice
  K00transactionserverservice
Installing shutdown scripts into /etc/rc1.d:
  K01recoverymanagerservice
  K00transactionserverservice
Installing shutdown scripts into /etc/rc2.d:
  K01recoverymanagerservice
  K00transactionserverservice
Installing startup scripts into /etc/rc3.d:
  S98recoverymanagerservice
  S99transactionserverservice
```

Upon restarting the computer the transaction server and recovery manager service should be started. To uninstall the services on a Unix machine simply perform the following steps:

1. Make sure that you are logged in as the user `root', as the installer needs to remove files under the directory `/etc'.

2. Change directory to `services/installer`.

3. Run the installer script with the "-u" argument:

```
./install_services.sh -u
```

The following is example output from a computer running the Solaris operating system:

```
# ./install_services.sh

Removing startup scripts from /etc/rc3.d:
  S98recoverymanagerservice
  S99transactionserverservice
Removing shutdown scripts from /etc/rcS.d:
  K01recoverymanagerservice
  K00transactionserverservice
Removing shutdown scripts from /etc/rc0.d:
  K01recoverymanagerservice
  K00transactionserverservice
```

```
Removing shutdown scripts from /etc/rc1.d:
  K01recoverymanagerservice
  K00transactionserverservice
Removing shutdown scripts from /etc/rc2.d:
  K01recoverymanagerservice
  K00transactionserverservice
```

Logging

The recovery manager and the transaction server services produce log files which are located in the `services/logs/` directory. There are two log files per service one called `<service name>-service.log` (e.g. `recovery-manager-service.log`), which contains information regarding the state of the service (e.g. started, stopped, restarted etc). The other is called `<services-name>.log` (e.g. `recovery-manager.log`) this contains information logged from the actual service (e.g. application level logging). To configure what information is logged in these files please edit the appropriate LOG4J configuration files which are located in `services/config/`.

ObjectStore management

Within the transaction service installation, the object store is updated regularly whenever transactions are created, or when *Transactional Objects for Java* is used. In a failure free environment, the only object states which should reside within the object store are those representing objects created with the *Transactional Objects for Java* API. However, if failures occur, transaction logs may remain in the object store until crash recovery facilities have resolved the transactions they represent. As such it is very important that the contents of the object store are not deleted without due care and attention, as this will make it impossible to resolve in doubt transactions. In addition, if multiple users share the same object store it is important that they realise this and do not simply delete the contents of the object store assuming it is an exclusive resource.

Additional jar requirements

In order to fully utilize all of the facilities available within *ArjunaTA*, it will be necessary to add all of the jar files contained in the lib directory of the distribution.

Setting properties

ArjunaTA has been designed to be highly configurable at runtime through the use of various property attributes, which will be described in subsequent sections. Although these attributes can be provided at runtime on the command line, it is possible (and may be more convenient) to specify them through a single properties file. At runtime *ArjunaTA* looks for the file `jbossjts-properties.xml` in the following order:

1. the current working directory, i.e., where the application was executed from.
2. the user's home directory.
3. the `etc` directory of the *ArjunaTA* installation; this must be placed in your CLASSPATH.

If found, all entries within this file will be added to the system properties. Obviously non-*ArjunaTA* specific properties can also be specified in this file. For example:

```
<property
  name="com.arjuna.ats.arjuna.coordinator.asyncCommit=YES"
  value="NO"/>
<property
  name="com.arjuna.ats.arjuna.objectstore.objectStoreDir=C"
  value="c:\temp\ObjectStore"/>
```

The name of the properties file can be overridden at runtime by specifying a new file using the `com.arjuna.ats.arjuna.common.propertiesFile` attribute variable.

Licensing

ArjunaTA requires a licence key in order to operate. A valid licence key must be provided to every *ArjunaTA* application at runtime as the property `ArjunaJTA_LicenceKey`

The *Transactional Objects for Java* framework is separately licenced, as is the complete two-phase commit protocol engine. As such, certain licences may only allow a single transactional resource to be registered with a transaction, or may not allow *Transactional Objects for Java* to be used.