

JBoss Communications JAIN SLEE TFTP Server Resource Adaptor User Guide

by Tom Uijldert and Eduardo Martins

Preface	v
1. Document Conventions	v
1.1. Typographic Conventions	v
1.2. Pull-quote Conventions	vii
1.3. Notes and Warnings	vii
2. Provide feedback to the authors!	viii
1. Introduction to JBoss Communications JAIN SLEE TFTP Server Resource Adaptor.	1
2. Resource Adaptor Type	3
2.1. Activities	3
2.2. Events	4
2.3. Activity Context Interface Factory	5
2.4. Resource Adaptor Interface	6
2.5. Restrictions	6
2.6. Sbb Code Examples	6
2.6.1. WRITE Request Event Handling	6
2.6.2. READ Request Event Handling with event suspending	7
3. Resource Adaptor Implementation	11
3.1. Configuration	11
3.2. Default Resource Adaptor Entities	11
3.3. Traces and Alarms	12
3.3.1. Tracers	12
3.3.2. Alarms	12
4. Setup	13
4.1. Pre-Install Requirements and Prerequisites	13
4.1.1. Hardware Requirements	13
4.1.2. Software Prerequisites	13
4.2. JBoss Communications JAIN SLEE TFTP Server Resource Adaptor Source Code.	13
4.2.1. Release Source Code Building	13
4.2.2. Development Trunk Source Building	14
4.3. Installing JBoss Communications JAIN SLEE TFTP Server Resource Adaptor.....	14
4.4. Uninstalling JBoss Communications JAIN SLEE TFTP Server Resource Adaptor..	14
5. Clustering	15
A. Revision History	17
Index	19

Preface

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the *Liberation Fonts* [<https://fedorahosted.org/liberation-fonts/>] set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight key caps and key-combinations. For example:

To see the contents of the file `my_next_bestselling_novel` in your current working directory, enter the `cat my_next_bestselling_novel` command at the shell prompt and press **Enter** to execute the command.

The above includes a file name, a shell command and a key cap, all presented in Mono-spaced Bold and all distinguishable thanks to context.

Key-combinations can be distinguished from key caps by the hyphen connecting each part of a key-combination. For example:

Press **Enter** to execute the command.

Press **Ctrl+Alt+F1** to switch to the first virtual terminal. Press **Ctrl+Alt+F7** to return to your X-Windows session.

The first sentence highlights the particular key cap to press. The second highlights two sets of three key caps, each set pressed simultaneously.

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in `Mono-spaced Bold`. For example:

File-related classes include `filesystem` for file systems, `file` for files, and `dir` for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialogue box text; labelled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

Choose **System > Preferences > Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, click the **Left-handed mouse** check box and click **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).

To insert a special character into a **gedit** file, choose **Applications > Accessories > Character Map** from the main menu bar. Next, choose **Search > Find** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit > Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in Proportional Bold and all distinguishable by context.

Note the **>** shorthand used to indicate traversal through a menu and its sub-menus. This is to avoid the difficult-to-follow 'Select **Mouse** from the **Preferences** sub-menu in the **System** menu of the main menu bar' approach.

Mono-spaced Bold Italic Of Proportional Bold Italic

Whether Mono-spaced Bold or Proportional Bold, the addition of Italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

To connect to a remote machine using ssh, type `ssh username@domain.name` at a shell prompt. If the remote machine is `example.com` and your username on that machine is john, type `ssh john@example.com`.

The `mount -o remount file-system` command remounts the named file system. For example, to remount the `/home` file system, the command is `mount -o remount /home`.

To see the version of a currently installed package, use the `rpm -q package` command. It will return a result as follows: `package-version-release`.

Note the words in bold italics above `username`, `domain.name`, `file-system`, `package`, `version` and `release`. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

When the Apache HTTP Server accepts requests, it dispatches child processes or threads to handle them. This group of child processes or threads is known as

a *server-pool*. Under Apache HTTP Server 2.0, the responsibility for creating and maintaining these server-pools has been abstracted to a group of modules called *Multi-Processing Modules (MPMs)*. Unlike other modules, only one module from the MPM group can be loaded by the Apache HTTP Server.

1.2. Pull-quote Conventions

Two, commonly multi-line, data types are set off visually from the surrounding text.

Output sent to a terminal is set in `Mono-spaced Roman` and presented thus:

```
books      Desktop  documentation  drafts  mss    photos  stuff  svn
books_tests Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in `Mono-spaced Roman` but are presented and highlighted as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
    public static void main(String args[])
        throws Exception
    {
        InitialContext iniCtx = new InitialContext();
        Object      ref  = iniCtx.lookup("EchoBean");
        EchoHome    home = (EchoHome) ref;
        Echo        echo = home.create();

        System.out.println("Created Echo");

        System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
    }
}
```

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.



Note

A note is a tip or shortcut or alternative approach to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.



Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring Important boxes won't cause data loss but may cause irritation and frustration.



Warning

A Warning should not be ignored. Ignoring warnings will most likely cause data loss.

2. Provide feedback to the authors!

If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you! Please submit a report in the the *Issue Tracker* [<http://bugzilla.redhat.com/bugzilla/>], against the product **JBoss Communications JAIN SLEE TFTP Server Resource Adaptor**, or contact the authors.

When submitting a bug report, be sure to mention the manual's identifier: `JAIN_SLEE_TftpServer_RA_User_Guide`

If you have a suggestion for improving the documentation, try to be as specific as possible when describing it. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.

Introduction to JBoss Communications JAIN SLEE TFTP Server Resource Adaptor

This Resource Adaptor enables an application - developed using a Service Building Block hosted in a SLEE container - to intercept TFTP requests and send or receive data using same TFTP protocol. Likewise, it enables that application to receive or send complete files. Thus, the application can combine serving static content through file transfer with serving dynamic content by relaying data from back-end systems.

Resource Adaptor Type

The Resource Adaptor Type is the interface which defines the contract between the RA implementations, the SLEE container, and the Applications running in it.

The name of the RA Type is `TftpServerResourceAdaptorType`, its vendor is `org.mobicients` and its version is `1.0`.

2.1. Activities

The Resource Adaptor Type defines one activity object, `net.java.slee.resource.tftp.TransferActivity`.

The `TransferActivity` represents a TFTP transfer. It is created on reception of a TFTP requests, and ends on end-of-file, closure of the datastream, or by an SBB invoking `sendError()` on the `TransferActivity` Object. The activity object interface is defined as follows:

```
package net.java.slee.resource.tftp;

public interface TransferActivity {

    public String getTransferId();

    public InputStream getInputStream() throws IOException;
    public OutputStream getOutputStream() throws IOException;

    public void receiveFile(String filename) throws FileNotFoundException, IOException;
    public void receiveFile(File file) throws FileNotFoundException, IOException;

    public void sendFile(String filename) throws FileNotFoundException, IOException;
    public void sendFile(File file) throws FileNotFoundException, IOException;

    public void sendError(int errorCode, String reason);
}
```

The `getTransferId()` method:
Retrieves the ID of the transfer.

The `getInputStream()` method:
Retrieves the data stream of the TFTP write request.

The `getOutputStream()` method:

Retrieves the data stream of the TFTP read request.

The `receiveFile(...)` methods:

Stores the data of the TFTP write request in the given file.

The `sendFile(...)` methods:

Sends the content of the given file to the TFTP client that initiated the TFTP read request.

The `sendError(int, String)` methods:

Sends an error to the TFTP client that initiated the TFTP read request.

2.2. Events

The Events fired by TFTP Server Resource Adaptor represent an incoming TFTP Request. The table below lists the Resource Adaptor Type event types.

Table 2.1. Events fired on the TransferActivity:

Name	Vendor	Version	Event Class	Description
<code>net.java.slee.resource.tftp.events.incoming.request.READ</code>	<code>net.java.slee</code>	1.0	<code>net.java.slee.resource.tftp.events.RequestEvent</code>	An incoming <code>READ</code> TFTP request.
<code>net.java.slee.resource.tftp.events.incoming.request.WRITE</code>	<code>net.java.slee</code>	1.0	<code>net.java.slee.resource.tftp.events.RequestEvent</code>	An incoming <code>WRITE</code> TFTP request.
<code>net.java.slee.resource.tftp.events.incoming.request.DATA</code>	<code>net.java.slee</code>	1.0	<code>net.java.slee.resource.tftp.events.RequestEvent</code>	An incoming TFTP <code>DATA</code> block.
<code>net.java.slee.resource.tftp.events.incoming.request.ACK</code>	<code>net.java.slee</code>	1.0	<code>net.java.slee.resource.tftp.events.RequestEvent</code>	An incoming TFTP <code>ACK</code> .
<code>net.java.slee.resource.tftp.events.incoming.request.ERROR</code>	<code>net.java.slee</code>	1.0	<code>net.java.slee.resource.tftp.events.RequestEvent</code>	An incoming TFTP <code>ERROR</code> block.



Important

Spaces were introduced in `Name` and `Event Class` column values, to correctly render the table. Please remove them when using copy/paste.

All event types use the same type `net.java.slee.resource.tftp.events.RequestEvent`. It's interface is as follows:

```
package net.java.slee.resource.tftp.events;

import org.apache.commons.net.tftp.TFTPPacket;

public interface RequestEvent {

    public TFTPPacket getRequest();

    public String getTypeDescr();

    public String getId();

}
```

The `getRequest()` method:

Retrieves the TFTP request packet which is associated with the event.

The `getTypeDescr()` method:

Retrieves the type description associated with the request.

The `getId()` method:

Retrieves unique event ID.

2.3. Activity Context Interface Factory

The Resource Adaptor's Activity Context Interface Factory is of type `net.java.slee.resource.tftp.TftpServerActivityContextInterfaceFactory`, it allows the SBB to retrieve the `ActivityContextInterface` related with an existing Resource Adaptor activity object. The interface is defined as follows:

```
package net.java.slee.resource.tftp;

import javax.slee.ActivityContextInterface;
import javax.slee.FactoryException;
import javax.slee.UnrecognizedActivityException;

public interface TftpServerActivityContextInterfaceFactory {
```

```
public ActivityContextInterface getActivityContextInterface(
    TransferActivity activity) throws NullPointerException,
    UnrecognizedActivityException, FactoryException;
}
```

2.4. Resource Adaptor Interface

The TFTP Server Resource Adaptor interface is currently not defined.

2.5. Restrictions

There are currently no known restrictions

2.6. Sbb Code Examples

The following code shows how to use the Resource Adaptor Type for common functionalities

2.6.1. WRITE Request Event Handling

The following code handles a TFTP WRITE request:

```
import net.java.slee.resource.tftp.TransferActivity;
import net.java.slee.resource.tftp.events.RequestEvent;

import org.apache.commons.net.tftp.TFTPWriteRequestPacket;
import org.apache.commons.net.tftp.TFTPRequestPacket;

public void onWrite(RequestEvent event, ActivityContextInterface aci) {

    TFTPWriteRequestPacket req = (TFTPWriteRequestPacket) event.getRequest();

    if (isSpecialRequest(req) {
        doSpecialThings(req);
    } else {
        // just receive data and create the requested file
        TransferActivity activity = (TransferActivity) aci.getActivity();

        try {
            activity.receiveFile(req.getFilename());
        } catch (FileNotFoundException e) {
            activity.sendError(TFTPErrPacket.FILE_NOT_FOUND, e.getMessage());
        }
    }
}
```

```

        return;
    } catch (Exception e) {
        activity.sendError(TFTPErrorPacket.UNDEFINED, e.getMessage());
        return;
    }
}

private boolean isSpecialRequest(TFTPRequestPacket packet) {
    // inspect packet and decide whether it needs special treatment
    return false;
}

private void doSpecialThings(TFTPRequestPacket packet) {
    // actually do the special things that need to be done.
    ...;
}

```

2.6.2. READ Request Event Handling with event suspending

The following code handles a TFTP READ request:

```

import net.java.slee.resource.ftp.TransferActivity;
import net.java.slee.resource.ftp.events.RequestEvent;

import org.apache.commons.net.ftp.TFTPReadRequestPacket;
import org.apache.commons.net.ftp.TFTPRequestPacket;

// CMP fields
public abstract void setSuspendedEventCmp(EventContext ctxt);
public abstract EventContext getSuspendedEventCmp();

public void onRead(RequestEvent event, ActivityContextInterface aci,
    EventContext ctxt) {

    TFTPReadRequestPacket req = (TFTPReadRequestPacket) event.getRequest();

    if (isSpecialRequest(req) {
        doSpecialThings(req);
    } else {

```

```
// just send the requested file
TransferActivity activity = (TransferActivity) aci.getActivity();

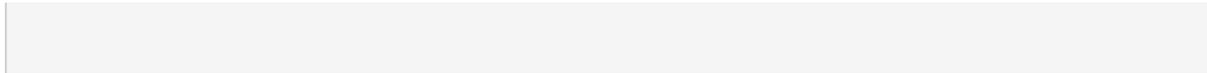
try {
    activity.sendFile(req.getFilename());
} catch (FileNotFoundException e) {
    activity.sendError(TFTPErrorPacket.FILE_NOT_FOUND, e.getMessage());
    return;
} catch (Exception e) {
    activity.sendError(TFTPErrorPacket.UNDEFINED, e.getMessage());
    return;
}
}

private boolean isSpecialRequest(TFTPRequestPacket packet) {
    // inspect packet and decide whether it needs special treatment
    return false;
}

private void doSpecialThings(TFTPRequestPacket packet, EventContext ctxt) {
    // actually do the special things that need to be done & suspend.
    ...;
    setSuspendedEventCmp(ctxt);
    ctxt.suspendDelivery();
}

public void onSomeOtherRelatedEvent(RelatedEvent event, ActivityContextInterface aci) {
    try {
        OutputStream os = getRequestActivity().getOutputStream();
        os.write("whatever data needs to be written".getBytes());
        os.flush(); os.close();
        getSuspendedEventCmp().resumeDelivery();
    } catch (Exception e) {
    }
}

private TransferActivity getRequestActivity() {
    for (ActivityContextInterface aci : sbbContext.getActivities())
        if (aci.getActivity() instanceof TransferActivity)
            return (TransferActivity) aci.getActivity();
    return null;
}
```



Resource Adaptor Implementation

This chapter documents the TFTP Server Resource Adaptor Implementation details, such as the configuration properties, the default Resource Adaptor entities, and the JAIN SLEE 1.1 Tracers and Alarms used.

The name of the RA is `TftpServerResourceAdaptor`, its vendor is `org.mobicens` and its version is `1.0`.

3.1. Configuration

The Resource Adaptor supports configuration only at Resource Adaptor Entity creation time, the following table enumerates the configuration properties:

Table 3.1. Resource Adaptor's Configuration Properties

Property Name	Description	Property Type	Default Value
<code>slee.resource.ftp.port</code>	port number to listen for ftp requests	<code>java.lang.Integer</code>	69

3.2. Default Resource Adaptor Entities

There is a single Resource Adaptor Entity created when deploying the Resource Adaptor, named `TftpServerRA`.

The `TftpServerRA` entity is also bound to Resource Adaptor Link Name `TftpServerRA`, to use it in an Sbb add the following XML to its descriptor:

```
<resource-adaptor-type-binding>
  <resource-adaptor-type-ref>
    <resource-adaptor-type-name>
      TftpServerResourceAdaptorType
    </resource-adaptor-type-name>
    <resource-adaptor-type-vendor>
      org.mobicens
    </resource-adaptor-type-vendor>
    <resource-adaptor-type-version>
      1.0
    </resource-adaptor-type-version>
  </resource-adaptor-type-ref>
  <activity-context-interface-factory-name>
    slee/resources/mobicens/tftpserver/acifactory
  </activity-context-interface-factory-name>
```

```
</resource-adaptor-type-binding>
```

3.3. Traces and Alarms

3.3.1. Tracers

Each Resource Adaptor Entity uses a single JAIN SLEE 1.1 Tracer, named `TftpServerResourceAdaptor`. The related Log4j Logger category, which can be used to change the Tracer level from Log4j configuration, is `javax.slee.RAEntityNotification[entity=TftpServerRA]`

3.3.2. Alarms

No alarms are set by this Resource Adaptor.

Setup

4.1. Pre-Install Requirements and Prerequisites

Ensure that the following requirements have been met before continuing with the install.

4.1.1. Hardware Requirements

The RA hardware requirements don't differ from the underlying JBoss Communications JAIN SLEE requirements, refer to its documentation for further information.

4.1.2. Software Prerequisites

The RA requires JBoss Communications JAIN SLEE properly set.

4.2. JBoss Communications JAIN SLEE TFTP Server Resource Adaptor Source Code

4.2.1. Release Source Code Building

1. Downloading the source code



Important

Subversion is used to manage its source code. Instructions for using Subversion, including install, can be found at <http://svnbook.red-bean.com>

Use SVN to checkout a specific release source, the base URL is ?, then add the specific release version, lets consider 1.0.0.BETA1.

```
[usr]$ svn co ?/1.0.0.BETA1 slee-ra-tftp-server-1.0.0.BETA1
```

2. Building the source code



Important

Maven 2.0.9 (or higher) is used to build the release. Instructions for using Maven2, including install, can be found at <http://maven.apache.org>

Use Maven to build the deployable unit binary.

```
[usr]$ cd slee-ra-tftp-server-1.0.0.BETA1
[usr]$ mvn install
```

Once the process finishes you should have the `deployable-unit` jar file in the `target` directory, if JBoss Communications JAIN SLEE is installed and environment variable `JBOSS_HOME` is pointing to its underlying JBoss Enterprise Application Platform directory, then the deployable unit jar will also be deployed in the container.

4.2.2. Development Trunk Source Building

Similar process as for [Section 4.2.1, "Release Source Code Building"](#), the only change is the SVN source code URL, which is NOT AVAILABLE.

4.3. Installing JBoss Communications JAIN SLEE TFTP Server Resource Adaptor

To install the Resource Adaptor simply execute provided ant script `build.xml` default target:

```
[usr]$ ant
```

The script will copy the RA deployable unit jar to the `default` JBoss Communications JAIN SLEE server profile deploy directory, to deploy to another server profile use the argument `-Dnode=`.

4.4. Uninstalling JBoss Communications JAIN SLEE TFTP Server Resource Adaptor

To uninstall the Resource Adaptor simply execute provided ant script `build.xml` `undeploy` target:

```
[usr]$ ant undeploy
```

The script will delete the RA deployable unit jar from the `default` JBoss Communications JAIN SLEE server profile deploy directory, to undeploy from another server profile use the argument `-Dnode=`.

Clustering

The TFTP Server Resource Adaptor is not cluster aware, which means there is no failover process for a cluster node's requests being handled once the node fails.

Appendix A. Revision History

Revision History

Revision 1.0

Fri Apr 1 2011

TomUijldert

Creation of the JBoss Communications JAIN SLEE TFTP Server RA User Guide.

Index

F

feedback, viii

