JBoss Communications JAIN SLEE HTTP Servlet Resource Adaptor User Guide

by Amit Bhayani and Eduardo Martins

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` or ***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 HTTP Servlet Resource Adaptor**, or contact the authors.

When submitting a bug report, be sure to mention the manual's identifier: JAIN_SLEE_HttpServlet_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 HTTP Servlet Resource Adaptor

An HTTP Servlet is used to extend the capabilities of servers that host applications access via a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers. The aim of the HTTP Servlet Resource Adaptor is to provide the ease of same request-response programming model in SLEE environment. HTTP Servlet Resource Adaptor is not an replacement for HTTP Servlets but is suppose to work in close proximity with Servlet to gain the smooth integration between web application hosted in Web Server and application developed using Service Building Block hosted in the SLEE container.

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 `HttpServletResourceAdaptorType`, its vendor is `org.mobicents` and its version is `1.0`.

2.1. Activities

The Resource Adaptor Type defines two activity objects, the types `net.java.slee.resource.http.HttpServletRequestActivity` and `net.java.slee.resource.http.HttpSessionActivity`.

The `HttpServletRequestActivity` represents a specific incoming request. It is created by the Resource Adaptor when processing the incoming event, unless the request is received with an `HttpSession`. The activity ending is done by the Resource Adaptor once the related request event is unreferenced inside the SLEE Container. The activity object interface is defined as follows:

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

public interface HttpServletRequestActivity {

   /**
    * Method to fetch the Request ID for this request.
    *
    * @return
    */
   public Object getRequestID();

}
```

The `getRequestID()` method:
Retrieves the Request's ID.

The `HttpSessionActivity` represents an `HttpSession`. It is created on demand by an SBB, through the RA SBB Interface, and ends on timeout, managed by the underlying HTTP Servlet framework, or by an SBB, invoking the `invalidate()` on the related `HttpSession` Object. The activity object interface is defined as follows:

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

public interface HttpSessionActivity {

    public String getSessionId();

}
```

The `getSessionId()` method:
Retrieves the Session's ID.

2.2. Events

The Events fired by HTTP Servlet Resource Adaptor represent an incoming HTTP Request, and for each HTTP Request a different event type is used for each activity type. The table below lists the Resource Adaptor Type event types.

Table 2.1. Events fired on the HttpServletRequestActivity:

Name	Vendor	Versi	Event Class	Description
net.java. slee.resource. http.events. incoming.request. HEAD	net.java.slee	1.0	net.java. slee.resource. http.events. HttpServletRequestEvent	An incoming `HEAD` HTTP request.
net.java. slee.resource. http.events. incoming.request. GET	net.java.slee	1.0	net.java. slee.resource. http.events. HttpServletRequestEvent	An incoming `GET` HTTP request.
net.java. slee.resource. http.events. incoming.request. POST	net.java.slee	1.0	net.java. slee.resource. http.events. HttpServletRequestEvent	An incoming `POST` HTTP request.
net.java. slee.resource. http.events. incoming.request. PUT	net.java.slee	1.0	net.java. slee.resource. http.events. HttpServletRequestEvent	An incoming `PUT` HTTP request.
net.java. slee.resource. http.events. incoming.request. DELETE	net.java.slee	1.0	net.java. slee.resource. http.events. HttpServletRequestEvent	An incoming `DELETE` HTTP request.
net.java. slee.resource. http.events.	net.java.slee	1.0	net.java. slee.resource. http.events. HttpServletRequestEvent	An incoming `OPTIONS` HTTP request.

Name	Vendor	Versi	Event Class	Description
incoming.request. OPTIONS				
net.java. slee.resource. http.events. incoming.request. TRACE	net.java.slee	1.0	net.java. slee.resource. http.events. HttpServletRequestEvent	An incoming `TRACE` HTTP request.

Important

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

Table 2.2. Events fired on the HttpSessionActivity:

Name	Vendor	Versi	Event Class	Description
net.java. slee.resource. http.events. incoming.session. HEAD	net.java.slee	1.0	net.java. slee.resource. http.events. HttpServletRequestEvent	An incoming `HEAD` HTTP request.
net.java. slee.resource. http.events. incoming.session. GET	net.java.slee	1.0	net.java. slee.resource. http.events. HttpServletRequestEvent	An incoming `GET` HTTP request.
net.java. slee.resource. http.events. incoming.session. POST	net.java.slee	1.0	net.java. slee.resource. http.events. HttpServletRequestEvent	An incoming `POST` HTTP request.
net.java. slee.resource. http.events. incoming.session. PUT	net.java. slee	1.0	net.java. slee.resource. http.events. HttpServletRequestEvent	An incoming `PUT` HTTP request.
net.java. slee.resource. http.events. incoming.session. DELETE	net.java.slee	1.0	net.java. slee.resource. http.events. HttpServletRequestEvent	An incoming `DELETE` HTTP request.
net.java. slee.resource. http.events. incoming.session. OPTIONS	net.java.slee	1.0	net.java. slee.resource. http.events. HttpServletRequestEvent	An incoming `OPTIONS` HTTP request.
net.java. slee.resource. http.events. incoming.session. TRACE	net.java.slee	1.0	net.java. slee.resource. http.events. HttpServletRequestEvent	An incoming `TRACE` HTTP request.

Important

Spaces where 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.http.events.HttpServletRequestEvent`. It's interface is as follows:

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

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public interface HttpServletRequestEvent {

    public HttpServletRequest getRequest();

    public HttpServletResponse getResponse();

    public String getId();

}
```

The `getRequest()` method:
: Retrieves the HttpServletRequest which is associated with the event.

The `getResponse()` method:
: Retrieves the HttpServletResponse which is associated with the request, which can be used to reply to the incoming 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.http.HttpServletRaActivityContextInterfaceFactory`, 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.http;

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

public interface HttpServletRaActivityContextInterfaceFactory {

    public ActivityContextInterface getActivityContextInterface(
            HttpSessionActivity activity) throws NullPointerException,
            UnrecognizedActivityException, FactoryException;

    public ActivityContextInterface getActivityContextInterface(
            HttpServletRequestActivity activity) throws NullPointerException,
            UnrecognizedActivityException, FactoryException;

}
```

2.4. Resource Adaptor Interface

The HTTP Servlet Resource Adaptor interface, of type `net.java.slee.resource.http.HttpServletRaSbbInterface` , which an SBB uses to create HtppSessionActivity instances, is defined as follows:

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

import javax.servlet.http.HttpSession;
import javax.slee.SLEEException;
import javax.slee.resource.ActivityAlreadyExistsException;
import javax.slee.resource.StartActivityException;

public interface HttpServletRaSbbInterface {

    public HttpSessionActivity getHttpSessionActivity(HttpSession httpSession)
            throws  NullPointerException, IllegalArgumentException, IllegalStateException,
            ActivityAlreadyExistsException, StartActivityException,
            SLEEException;
```

```
}
```

The `getHttpSessionActivity(HttpSession)` method:
Retrieves an `HttpSessionActivity` for the specified `HttpSession`. If the activity does not exist a new one is created.

2.5. Restrictions

The HTTP Servlet does imposes some restrictions on the usage of the `HTTP Servlet` API java objects:

The `getAttribute(String)` method of the `HttpSession` object, retrieved from a request:
The `_ENTRY_POINT` attribute is reserved for internal usage, if such parameter value is used a SecurityException is thrown.

The `getValue(String)` method of the `HttpSession` object, retrieved from a request:
The `_ENTRY_POINT` attribute is reserved for internal usage, if such parameter value is used a SecurityException is thrown.

The `putValue(String,Object)` method of the `HttpSession` object, retrieved from a request:
The `_ENTRY_POINT` attribute is reserved for internal usage, if such value is used on the first parameter a SecurityException is thrown.

The `removeAttribute(String)` method of the `HttpSession` object, retrieved from a request:
The `_ENTRY_POINT` attribute is reserved for internal usage, if such parameter value is used a SecurityException is thrown.

The `removeValue(String)` method of the `HttpSession` object, retrieved from a request:
The `_ENTRY_POINT` attribute is reserved for internal usage, if such parameter value is used a SecurityException is thrown.

The `setAttribute(String,Object)` method of the `HttpSession` object, retrieved from a request:
The `_ENTRY_POINT` attribute is reserved for internal usage, if such value is used on the first parameter a SecurityException is thrown.

2.6. Sbb Code Examples

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

2.6.1. GET Request Event Handling

The following code examples the handling of an HTTP GET request:

```
public void onGet(HttpServletRequestEvent event,
        ActivityContextInterface aci) {

    // detach from HttpServletRequestActivity
    aci.detach(sbbContext.getSbbLocalObject());

    HttpServletResponse response = event.getResponse();
    try {
        PrintWriter w = response.getWriter();
        w.print("onGet OK! Served by SBB = " + getSbbId());
        w.flush();
        response.flushBuffer();
        log
                .info("HttpServletRAExampleSbb: GET Request received and OK! response sent.");
    } catch (Exception e) {
        log.error(e);
    }

}
```

2.6.2. PUT Request Event Handling With Session Creation

The following code examples the handling of an HTTP PUT request, and the creation of an HttpSessionActivity:

```
public void onPut(HttpServletRequestEvent event,
        ActivityContextInterface aci) {

    SbbLocalObject sbbLocalObject = sbbContext.getSbbLocalObject();
    // detach from HttpServletRequestActivity
    aci.detach(sbbLocalObject);
    try {
        // here we will setup a session activity before sending the response back
        if (httpServletRaSbbInterface == null) {
            Context myEnv = (Context) new InitialContext().lookup("java:comp/env");
            httpServletRaSbbInterface = (HttpServletRaSbbInterface) myEnv.lookup(
                "slee/resources/mobicents/httpservlet/sbbrainterface");
```

```
                httpServletRaActivityContextInterfaceFactory =
                    (HttpServletRaActivityContextInterfaceFactory) myEnv.lookup(
                        "slee/resources/mobicents/httpservlet/acifactory");
            }
            HttpSession httpSession = event.getRequest().getSession();
            HttpSessionActivity httpSessionActivity = httpServletRaSbbInterface
                    .getHttpSessionActivity(httpSession);
            ActivityContextInterface httpSessionActivityContextInterface =
                httpServletRaActivityContextInterfaceFactory
                    .getActivityContextInterface(httpSessionActivity);
            httpSessionActivityContextInterface.attach(sbbLocalObject);
            HttpServletResponse response = event.getResponse();
            PrintWriter w = response.getWriter();
            w.print("onPut OK! Served by SBB = " + getSbbId());
            w.flush();
            response.flushBuffer();
            log.info("HttpServletRAExampleSbb: PUT Request received and OK! response sent.");
        } catch (Exception e) {
            log.error(e);
        }
    }
```

Resource Adaptor Implementation

This chapter documents the HTTP Servlet 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 `HttpServletResourceAdaptor`, its vendor is `org.mobicents` 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
name	the servlet name which the RA entity should use	java.lang.String	mobicents

Important

JAIN SLEE 1.1 Specification requires values set for properties without a default value, which means the configuration for those properties are mandatory, otherwise the Resource Adaptor Entity creation will fail!

3.2. Default Resource Adaptor Entities

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

The `HttpServletRA` entity is also bound to Resource Adaptor Link Name `HttpServletRA`, 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>
         HttpServletResourceAdaptorType
      </resource-adaptor-type-name>
      <resource-adaptor-type-vendor>
         org.mobicents
```

```
            </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/mobicents/httpservlet/acifactory
         </activity-context-interface-factory-name>
         <resource-adaptor-entity-binding>
            <resource-adaptor-object-name>
               slee/resources/mobicents/httpservlet/sbbrainterface
            </resource-adaptor-object-name>
            <resource-adaptor-entity-link>
               HttpServletRA
            </resource-adaptor-entity-link>
         </resource-adaptor-entity-binding>
      </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 `HttpServletResourceAdaptor`. The related Log4j Logger category, which can be used to change the Tracer level from Log4j configuration, is `javax.slee.RAEntityNotification[entity=HttpServletRA]`

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 HTTP Servlet 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 2.4.0.FINAL.

```
[usr]$ svn co ?/2.4.0.FINAL slee-ra-http-servlet-2.4.0.FINAL
```

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-http-servlet-2.4.0.FINAL
[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 HTTP Servlet 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 HTTP Servlet 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 HTTP Servlet 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 Tue Dec 30 2009 EduardoMartins
Creation of the JBoss Communications JAIN SLEE HTTP Servlet RA User Guide.

Index

F