

Troubleshooting

1. Installation

1.1. Error Page appears with message "java.security.AccessControlException: access denied (java.net.SocketPermission <serverName> resolve)"

The security policy is being loaded before the system properties required for [Jini](#) have been set. Set these properties in the command line which starts the application server, rather than loading them from the EAR file.

1.2. Error Page appears with message "Authentication failed"

It is likely that you have not set up the JNDI properties correctly for your application server, and the servlet container cannot access the EJBs in the EJB container.

1.3. WebSphere Application Server Fails To Start

After installing RADRunner on WebSphere, the application server(s) concerned may fail to start, giving a "Start Sub-Command Error". This appears to be a characteristic behaviour of WebSphere 4.0.3 on setting a security manager in the JVM of the application server, particularly when creating a number of servers in a server group. It is most easily cured by starting and stopping the WebSphere Admin Server on the machine(s) concerned, after which the application server(s) should start as normal. If not, a machine reboot may be necessary to solve the problem.

1.4. "Page Not Found" On WebSphere

If you cannot access the RADRunner login page after installing RADRunner on WebSphere, it may be because the default installation package references one of the example Data Sources (*SampleDataSource*). Although this Data Source is not actually used by the application, some versions of WebSphere will attempt to connect to it before starting RADRunner, and the username/password combination specified in the RADRunner installation package (*wsdemo/wsdemo1*) is unlikely to be valid on your machine.

The solution is to uninstall RADRunner, and install it again, this time specifying either:

- a valid username/password combination for *SampleDataSource*
or
- a different Data Source, with a username/password combination that you know to be valid.

1.5. Caught Java exception: org.exolab.castor.xml.MarshalException with message "Parsing Error: Content is not allowed in prolog."

This occurs when the Playwright XML text stored in the database is being retrieved as *binary data*, rather than as *ascii characters*. The cause is generally that a CLOB (Character Large Object) column is being used to store data, but the JDBC driver used to access the database is bringing back CLOB values as BLOB (Binary Large Object) values.

The solutions are:

- Switch to a database and JDBC driver combination known to work. For example, DB2 and Oracle will not give this problem.
- Use a different column type to store Playwright text. For example, with the free open source database MySQL, a LONGTEXT column can be used.

For details of how to implement these solutions, see [Choosing a Database](#).

1.6. Jini Services Fail To Start On A Standalone Windows Machine

There is a known problem with running Jini services on a *standalone* Windows machine (i.e., one that is not connected to a network). The services seem to require the presence of a physical network connection in order to run.

This can be solved on Windows 95/98/ME via the procedure below, which installs a "loopback adapter" - a virtual network connection. However, this does not seem to work on 2000/XP. If you are using one of these operating systems, it is necessary either to:

- connect your RMI server to a network (a hub or switch is not necessary - connection to one other machine via a crossover cable is sufficient)
- switch to a database persistence solution for RADRunner.

Note that installing a loopback adapter may prevent your machine from connecting normally to a network. If you wish to establish a network connection at a later date and experience problems, open *Start/Settings/Control Panel/Network* and remove the loopback adapter.

Installing a loopback adapter on a Windows 95/98/ME machine

1. Go to the Start Menu and select *Settings/Control Panel*.
2. Select the icon *Network*.

Troubleshooting

3. Choose the tab *Identification*, and check that the text field *Computer name* is populated - if not, enter a name.
4. Choose the tab *Configuration*, and click on *Add*.
5. In the dialog box that pops up, select *Adapter* and click on *Add* again. Select *Manufacturer of Microsoft*, and click on *Microsoft Virtual Private Networking Adapter*.
6. Click on *OK* to save the new Adapter.
7. With the new Adapter selected, click on *Add*.
8. In the dialog box that pops up, select *Protocol* and click on *Add* again. Select *Manufacturer of Microsoft*, and click on *TCP/IP*. Click on *OK*.
9. You will be prompted to configure the TCP/IP protocol:
 - Select "Determine IP address", and enter *169.254.0.1*.
 - Enter subnet mask *255.255.0.0*.
 - Deactivate WINS.
 - Activate DNS. The text field *hosts* should contain your computer name - if not, enter it. Under the DNS server search sequence, type *169.254.0.1*.
10. Click on *OK* to save the new Protocol.
11. Click on *OK* again to exit the Network dialog. You will be prompted to restart your PC - do so.

1.7. RADRunner is unable to create Boss user or 42 Role Instance

The most likely cause is that RADRunner cannot connect to the data store.

If you are using a relational database,

- Check that the system properties specified to RADRunner reflect the required database connection. If you are using a container Data Source rather than a direct connection, check that the Data Source is specified correctly in the application server - the application server may allow you to test the configuration by connecting to the database.
- Make sure that the application server has the JDBC driver for your database in its classpath. In particular, ensure that the driver *.jar file is placed directly under the appropriate *lib* or *lib/ext* directory, and not in a sub-directory (created, for example, by an unzip utility).
- If the machine used for RADRunner has a software firewall installed (for example, *Zone Alarm* or *Norton Internet Security*), turn it off and try again. If RADRunner works this time, the software firewall is preventing RADRunner from accessing the port on which the database is listening. The solution is to allow programs running on the local network full access to this port - consult your database documentation to determine the port number (for MySQL, for example, it is 3306).
- If you are using DB2, and DB2 exception CLI0616E shows up in the RADRunner log file, you need to run "db2jstrt" on the DB2 host to initialize the JDBC listener.
- if you are using a database which does not support the CLOB data type (such as

), RADRunner will not be able to create its table and will fail to initialize. The solution is either to switch to a database which does support this data type, to configure RADRunner to [use a different column type to store Playwright XML text](#).

If you are using JBoss, note that some versions (3.2.4 and 3.2.5) of JBoss have a bug that prevents RADRunner from executing (technical details). JBoss versions 3.2.3 and earlier are OK, but those are longer available on the JBoss web site. Hence we recommend that you return to an earlier version of JBoss, 3.0.8, which we currently use for production installations, or a version higher than 3.2.6RC2.

If you are using a Jini JavaSpace, check that:

- The RMI daemon is running on the Jini host
- The HTTP server used to serve the downloadable JARs is functioning correctly
- The HTTP proxy (if you use one) is functioning correctly
- Your [JINI](#) settings are correct
- There is no mismatch between the Java Virtual Machine (JVM) used to run the application server and the JVM used to run the Jini services. Although in principle they should be independent, in practice the way the JVMs calculate class serial ids may differ and this can cause errors when downloading classes dynamically via RMI. As described in [JINI](#) with regard to the setting of RM_JINI_JAVA_HOME, it is best to ensure that both JVMs are not only supplied by the same vendor (e.g., Sun or IBM) but also that they have exactly the same version number.

If none of these appears to be the cause, try restarting your application server. Once it has tried to talk to a database or an RMI server, it may maintain a persistent socket connection to it - if the call failed, this connection will now be invalid. Restarting the application server should clear the connection. If not, try the solution that the experienced IT technician turns to when all else fails: reboot the computer.

2. Usage

2.1. Pages appear strangely in browser

RADRunner makes uses of some HTML features which are standard, yet not fully supported by all browsers - for instance, the clock in the banner updates every second via dynamic HTML, and the Cascading Style Sheet attribute *overflow: auto* is used to present messages in a scrolling text box.

In Mozilla neither of these features works, and in Konqueror, the clock works but the text messages do not scroll. Other browsers may have similar problems. All features work correctly in Internet Explorer, which currently provides the most comprehensive support for standard HTML of any web browser.

Troubleshooting

If you need to support multiple browsers, the solution is to edit the RADRunner user interface appropriately, which is a simple matter for anyone who knows HTML - no Java or Javascript knowledge is required. For example, remove the clock code from the banner by deleting the appropriate lines in *RMBanner.jsp*, and present messages in a non-scrolling text box by deleting the overflow attribute from *rm.css*. The RADRunner user interface is designed for full customization prior to installation, so this sort of activity is usual when configuring a new RADRunner system.

You can experiment with such changes by expanding the *radrunner.ear* archive file into a directory called *radrunner.ear* (first renaming the archive to *radrunner.zip* to avoid a name conflict), and within it expanding the *radrunner.war* archive file into a directory called *radrunner.war* (again renaming the archive to *radrunner.zip* first). In some application servers (e.g., WebSphere) this expansion is done automatically on installing the application. In others (e.g., JBoss) you will need to do it by hand.

The sub-directory *radrunner.war* will contain all the HTML code in files of form *RM*.jsp*, with the Cascading Style Sheet *rm.css* in a further sub-directory *css*. In general, changes made to files in *radrunner.war* and/or its sub-directory *css* will take effect immediately without restarting the application server. However, if you change *RMBanner.jsp*, *RMNavigation.jsp*, or *RMFooter.jsp* it is necessary to signal to the application server that they have changed by *touching* the files which include them - *RMLogin.jsp*, *RMActor.jsp*, *RMRole.jsp*, and so on. On UNIX-based operating systems, use the *touch* command to do this. On other operating systems, open the files in an editor, make a trivial change to each one (e.g., add a space somewhere then delete it), and save them all.

2.2. Role resources do not change as activities are carried out

You may have a process which runs apparently normally, yet when you examine Role resources via a snapshot window, no change appears to be made to the initial values.

This is usually because of file permissions on the server. When you invoke a snapshot window, the server creates a disk file containing the XML data, and then displays it in a web page. Each time you do this for a particular Role, the same filename is used, in order to conserve disk space on the server (on which thousands of Roles may be running). However, after the first time that the file is generated for a particular Role, the operating system may refuse to let it be over-written by subsequent snapshots - particularly if snapshots are being made available by *http://* rather than *file://*, since in this case the web server may take out its own lock on the files concerned.

The solution is to set file permissions appropriately for the server operating system and (if necessary) web server.

2.3. Strange behaviour in the Resource and Role Snapshot Windows

If you find that the Resource and Role Snapshot windows do not perform as expected, it may be because your system is set up to open files of type *.xml in a program other than your current web browser. In this case, depending on your operating system and browser configuration, the snapshot will invoke this other program rather than displaying the XML in a browser window.

If you do not like this behaviour, the solution is to configure the browser you use for RADRunner to open *.xml files itself, rather than by invoking another program. If this is not possible, configure the operating system on your client so that *.xml files are always opened in the browser you use for RADRunner.

2.4. Activities cannot cancel

You may find that it is impossible to cancel an activity. In other words, after carrying out one or more tasks, if the user clicks *cancel*, or an error condition is met on save, the tasks are saved anyway.

If so, the cause is likely to be that your database and/or RADRunner table are not [enabled for transaction support](#).

2.5. Uploaded files are not visible from Role instances

If uploaded files are not visible from within Role instances, check that you have [made the in directory visible via HTTP](#).

2.6. Import All Playwright Files In Specified Directory has no effect

If the directory used for import has the same name as a zip file at the parent location, on some operating systems with built-in zip support (e.g., Windows XP) RADRunner may try to read from the zip file instead of the directory and no files will be imported.

The solution is to rename or move either the zip file or the directory.

2.7. SOAP Errors - Client

RADRunner uses Apache SOAP to invoke web services as a client. There is a known problem with old versions of Apache SOAP (for example, version 2.2) in the case when the service being invoked is implemented using a Delphi 6 SOAP server. This results in a *StringIndexOutOfBoundsException: String index out of range: -1*, and is caused by Delphi

Troubleshooting

sending a SOAP response of the following form:

```
***** RESPONSE CONTENT:
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: 516
Content:

  <?xml version="1.0" encoding='UTF-8'?>
...
***** END RESPONSE
```

Old versions of Apache Soap throw an exception in HTTPUtils.Post when parsing the "Content:" line, because they assume that a line will always have a value:

```
// Remove trailing ; to prevent ContextType from throwing exception
if (valuebuf.charAt(valuebuf.length()-1) == ';') {
    valuebuf.deleteCharAt(valuebuf.length()-1);
}
```

With Delphi's response the valuebuf is empty and this causes the exception.

RADRunner is packaged with Apache SOAP version 2.3.1, and the bug is fixed in this version. However, depending on the configuration of your application server, it may be that an old version of SOAP is being loaded instead for use with RADRunner - in which case the problem will manifest itself within RADRunner.

Note that a Delphi-implemented service call will in fact be successfully made by RADRunner - the error is only detected on receiving the response. However, the Activity which contains the service call will roll back on detection of the error, and no changes internal to the current Role will be saved as a result of the web service call task, or any other tasks previously carried out under the same transaction.

The fix to this problem is *either* to upgrade your application server to use the latest version of Apache SOAP, *or* to configure your application server to ensure that RADRunner loads its own bundled version of Apache SOAP.

2.8. SOAP Errors - Server

RADRunner is a SOAP server as well as a SOAP client. Hence it is distributed with the *rpcrouter* servlet from the open source Apache SOAP implementation.

In order for this to function correctly, it is necessary to ensure that *xercesImpl.jar* (which supports the DOM level 2 interfaces) is loaded before any native DOM library available in your application server classpath which contains the DOM level 1 interfaces (*xml4j.jar*, for example, is supplied with WebSphere, and *xml.jar* with Tomcat). RADRunner's internal classpath is correct, but depending on your application server configuration, it may be necessary to ensure that the application server classpath also follows this rule. This can be done by editing the batch file or shell script that starts the application server, to place a reference to a copy of *xercesImpl.jar* at the start of the classpath.

2.9. The RMI daemon running Jini crashes or shows errors

The daemon may be running out of memory, when many concurrent service requests are made. See [Configuration Tuning](#) for details of how to increase the size of the heap allocated to the JVM running the RMI daemon. It may be necessary to use a heap with a maximum size of hundreds of MB, depending on your user base.

If the crashes still occur after increasing the RMI heap size, it may be caused by a [known problem with Sun's Jini Software Kit \(JSK\) in versions up to 1.2.1](#), which can cause the JavaSpace and related services to fail in situations of exceptionally high and sustained load (Sun say that this problem will be rectified in future releases of the JSK). At present our testing demonstrates the JSK version 1.2.1 to be unstable when loads of over 1500 requests/minute are sustained for several minutes in a row.

We were only able to simulate this amount of stress using either an application server cluster, or a single application server on a 4-CPU machine with no bound on the number of threads [*figures taken during stress and load testing at IBM Solution Partnership Centre, Hursley*]. If you are using a single application server container you are unlikely to reach these load levels, unless you allow a very high (or infinite) limit on the number of threads created by the application server JVM.

If you find that the JSK crashes in situations of high and sustained load, we recommend that you switch to using either the [database persistence option](#) or [JSpaces](#) with RADRunner. You can preserve existing data via Playwright export and import.

Monitoring Request Load Level

You may decide, however, to continue using the free JSK in a particular high-load environment, for example if high and sustained throughput occurs only occasionally.

To support this configuration, RADRunner is able to prevent the problem with the JSK from

Troubleshooting

impacting users by monitoring request load level, and dynamically applying a limit to the number of JavaSpace requests/second when it deems necessary.

This feature can be enabled by setting the following system properties in the JVM of each application server container (see [System Properties](#)):

1. `com.rolemodellers.jini.stressCheckInterval` to the interval in seconds at which to check how many JavaSpace read/take operations have been carried out (for example, 10 - the default value of 0 disables the feature)
2. `com.rolemodellers.jini.stressLimit` to the maximum number of JavaSpace read/take operations permitted within the interval - note that this is per application server container, so the maximum desired load should be divided by the number of containers (for example, 100 with one container, 25 with 4 containers)
3. `com.rolemodellers.jini.StressDelay` to the delay in milliseconds to apply to each JavaSpace read/take operation in a stress situation (for example, 200)

If a limit is placed on Jini requests due to high load, it will be removed automatically by RADRunner as soon as the load returns to an appropriate level.

Note that this feature will only operate correctly if the maximum number of threads created by each application server container JVM is set to a reasonable amount (for example, 50). Otherwise, the delay will still be applied to each read/take operation, but since there may effectively be an infinite number of requests during any period, it will not result in any limit to the load on the JSK. Consult your application server documentation for details of how to ensure that it will not allow a high or unlimited number of threads to be created, although it is likely that this is the default configuration.

3. Technical Support

When using RADRunner, there will be times when the system does not act as expected. In decreasing order of usual frequency, the likely causes are:

- Incorrect user input
- A bug in the process definition
- A misconception about how the software operates
- Configuration matters
- A bug in RADRunner
- A bug in open source software used by RADRunner.

In order for us to provide the best possible technical support to all users, we encourage users to attempt to resolve issues themselves before contacting us. This not only reduces the load on our technicians, but helps users to become proficient at rectifying problems on their own - which speeds things up greatly for them as well as others in the end.

Hence, please follow the procedure below to resolve problems encountered with RADRunner:

1. Look as carefully as possible through the messages presented on screen when the undesirable behaviour occurs. Look for things like the wrong object being specified in an activity, the wrong activity being carried out, attempting to access an object before it has been created, and so on.
If this does not help,
2. Look thoroughly through this document and all other RADRunner documentation, both by visual inspection and by a web search, to see if the issue you are encountering is described.
If so, instructions for resolving it will be given in the document concerned. We are careful to maintain detailed records of all problems encountered by RADRunner users, and incorporate into the documentation any which may affect others.
If this does not help,
3. Export the objects concerned to files as Playwright. Open the files in a text editor (or a specialised XML editor, if you have one available) and examine them for clues as to what is going on.
If this does not help,
4. Regenerate the error, noting the time at which you did so, and ask your system administrator to copy and paste the relevant log file section into an extract file. Tell your system administrator the time at which you regenerated the error, so that they can find the log file section more easily. Examine the log extract to see if you can work out from this what the problem is.
If this does not help, and the log extract contained no messages of priority DEBUG,
5. Ask your system administrator to set the logging level to DEBUG - this may involve restarting the application server. Then regenerate the error again, noting the time at which you did so, and ask your system administrator to copy and paste the relevant log file section into an extract file. Tell your system administrator the time at which you regenerated the error, so that they can find the log file section more easily. Examine the log extract to see if you can work out from this what the problem is.
If this does not help,
6. Send both the Playwright files and the DEBUG log extract to [Role Modellers support](#), together with a full description of the undesirable behaviour. We will attempt to resolve the problem as quickly as possible. This step should be treated as a last resort - the heavier the load on our technical support, the harder it will be for us to resolve problems that are caused by the software itself, as opposed to problems of process definition and usage.