Profiling Remote WebSphere Applications using Rational Application Developer

Timothy C. Fanelli
WebSphere Integration Test
IBM Poughkeepsie
July, 2009

This article guides you through the configuration of IBM® Rational® tools to enable profiling of JEE applications hosted on a remote WebSphere Application Server. Throughout the article you will see pointers to the documentation that helped complete these tasks. This article is intended for users performing the JEE roles application developer and/or deployer; responsible for the monitoring and optimizing the performance of applications hosted on IBM® WebSphere® Software.

Introduction

Overview

This article guides you through the configuration of Rational tools to enable profiling of JEE applications hosted on a remote WebSphere Application Server. Throughout the article you will see pointers to the documentation that helped complete these tasks. This article is intended for users performing the JEE roles application developer and/or deployer; responsible for the monitoring and optimizing the performance of applications hosted on IBM® WebSphere® Software.

The article will describe configuration of the products necessary to successfully profile JEE applications on remote WebSphere Application Server instances.

- IBM WebSphere Application Server Network Deployment® Version 6.1.0.21 for Linux, Unix or Solaris platforms
- IBM Rational Application Developer® Version 7.5
- IBM Rational Agent Controller® Version 8.2.*x*

Audience

You should have a good understanding of Linux, Unix or Solaris operation including the execution of shell scripts within the UNIX shell environment, and the commands associated with administering a WebSphere Application Server cell.

This article assumes that you already have a running installation of WebSphere Application Server Version 6.1.0.21 or later on a Unix, Linux or Solaris environment. It covers topics related to both standalone application servers, and Network Deployment environments.

Section 1: Rational Agent Controller 8.2

Agent Controller Overview

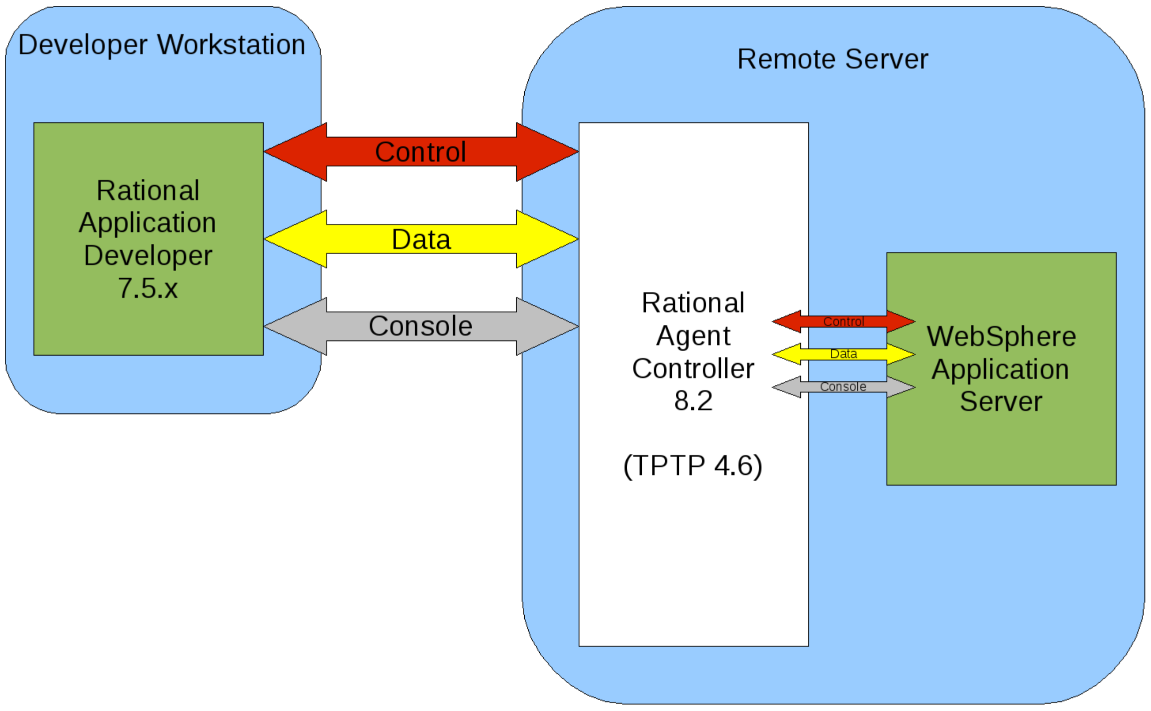
Rational Agent Controller is a test and performance tooling platform that allows developers to profile Java applications running on remote servers, using the Java Virtual Machine Tool Interface (JVMTI). JVMTI was introduced in J2SE 5.0, and allows a program to inspect the state, and control execution, of applications running in a JVM.

There are three "players" that must be involved in order to gather profiling data from remote applications:

1. Client – An application that allows the user of a service to interact with the provider. In this article, the client is the RAD workspace.
2. Agent – An application that exposes services through an agent controller. In this article, the agent is a WAS server.
3. Agent Controller – A process that resides on the same system as the Agent, and manages details of communication with the Agent.

An more detailed overview of these can be found on the Rational Application Developer Info Center, here: http://publib.boulder.ibm.com/infocenter/radhelp/v7r5/topic/org.eclipse.tptp.platform.agentcontroller.doc.user/concepts/ac/c_ac_ovr.html

As you can see in the illustration below, our setup will consist of these same three components. Here, the developer's RAD workbench acts as the client, and communicates with a RACv82 installation on the remote server. RAC, in turn, communicates with WebSphere Application Server, which is the agent.


Developer Workstation
Rational Application Developer 7.5.x
Control
Data
Console
Remote Server
Rational Agent Controller 8.2
(TPTP 4.6)
Control
Data
Console
WebSphere Application Server

There are three logical communication channels used to pass information between the components: control, data and console. The actual connections used between components is not a one-to-one mapping, and it is configurable.

The control channel between the client, RAD and the agent controller allows the exchange of command-formatted messages. The client makes requests and the agent controller sends responses. A separate control channel exists between the agent controller and the agent, WAS. RAD can exchange control messages with WAS, but the Agent Controller is in the middle of the exchange when using the control channel. The control channels are set up when a Client or an Agent initiates contact with the Agent Controller through a pre-configured port or named pipe.

The data channel can be either a uni-directional or bi-directional channel over which data of any form can be sent. Typically, the Client establishes half of the channel and then sends a request (along with its own handle) to the Agent that it wants to exchange data with, and telling it to create the other half of the channel. The Agent then asks the Agent Controller to connect the two halves, thus establishing a Client-to-Agent Data Channel.

The console channel is simply a bi-directional data channel that is used to support an application that requires input, output, or both to a console display.

Installing Rational Agent Controller 8.2

Overview

To profile an application on a remote WebSphere Application Server, you must first install IBM's Rational Agent Controller (RAC) on the target machine. RAC enables two way communication between independent processes to allow monitoring and control.

Rational Application Developer (RAD) uses a built-in internal version of the Agent Controller, referred to as the Internal Agent Controller (IAC) for use in profiling application in RAD's Universal Test Environment (UTE). If you defined a new local server in your RAD workspace, and started it in profiling mode to profile you application, then RAD is communicating with the server in the UTE via the IAC.

The process for profiling a remote server is very similar, though slightly more complicated. The IAC may only be used to profile local UTE servers, so we must install RAC on the same machine as our target server. Then, we must configure the target machine's environment variables to properly support starting a Java Virtual Machine (JVM) with profiling enabled. Finally, we must configure the target server's JVM options to enable profiling.

Obtaining Rational Agent Controller 8.2

You will find instructions for downloading and installing Rational Agent Controller on IBM's support site. Use Rational Agent Controller 8.2.0 for Solaris SPARC or Linux on System Z, and 8.2.0.1 for AIX.

RAC 8.2.0	http://www-01.ibm.com/support/docview.wss?rs=2042&context=SSRTLW&dc=D400&uid=swg24023836&loc=en_US&cs=UTF-8&lang=en&rss=ct2042rational
RAC 8.2.0.1	http://www-01.ibm.com/support/docview.wss?rs=180&context=SSEQTP&dc=D400&uid=swg24023999&loc=en_US&cs=UTF-8&lang=en&rss=ct180websphere

Installing Rational Agent Controller 8.2

You must install RAC on the same server as your WebSphere Application Server. If you do not have physical access to the server hosting your WebSphere Application Server, you have the option of performing a silent installation of RAC. An example response file is shown below.

There are several parts of this response file which must be customized to suit your particular environment. Specifically:

1. Under the server element, update the `location` attribute of the `repository` element so that it specifies the full path to the `diskTag.inf` file on the RAC 82 installation media. This value should be specified using URL syntax, as shown.
2. Update the `installLocation` attribute of the `profile` element to point to your desired installation directory. Note, when using interactive mode, the default installation location would be `/opt/IBM/SDP`
3. Update the value of the `data` element with `key="eclipseLocation"` to match the installation location specified in (2).
4. Update the `offering` element's `feature` attribute.
   - Set the `id` attribute to match the `diskSetOfferingId` value found in the `diskTag.inf` file on the installation media.
   - Set the `version` attribute to match the `diskSetOfferingVersion` value found in the `diskTag.inf` file on the installation media.
   - Change the `feature` attribute to specify whether you wish to install 32-bit or 64-bit Rational Agent Controller. Note: this must coincide with your WebSphere Application Server installation. Use 32-bit if you run 32-bit WAS, and 64-bit if you run 64-bit WAS.
   - Change the value of the preference `com.ibm.cic.common.core.preferences.eclipseCache` to point to the shared installation directory. Note, when using interactive mode, the default shared installation location would be `/opt/IBM/SDPShared`

Once you have created your response file, you can execute it using the command:

```
./IBMIM --launcher.ini ./silent-install.ini
        -input responseFile.txt -showVerboseProgress
```

Sample response file for silent installation

```
<?xml version="1.0" encoding="UTF-8"?>
<agent-input acceptLicense='true'>

<server>
       <repository location='file:///mnt/rac_v82/diskTag.inf'/>
</server>

<profile installLocation='/opt/IBM/RAC/' id='IBM Rational Agent Controller'>
       <data key='eclipseLocation' value='/opt/IBM/RAC/'/>
       <data key='cic.selector.nl' value='en'/>
       <data key='user.RAC_DefaultType,com.ibm.rational.agent.controller.solarissparc'
             value='DEFAULT'/>
       <data key='user.RAC_ALLOW,com.ibm.rational.agent.controller.solarissparc'
             value='ALL'/>
       <data key='user.RAC_HOSTS,com.ibm.rational.agent.controller.solarissparc'
             value=''/>
       <data key='user.RAC_SECURITY,com.ibm.rational.agent.controller.solarissparc'
             value='false'/>
       <data key='user.RAC_USERS,com.ibm.rational.agent.controller.solarissparc'
             value=''/>
       <data key='user.RAC_USERTYPE,com.ibm.rational.agent.controller.solarissparc'
             value='ANY'/>
</profile>

<install modify='false'>
       <offering profile='IBM Rational Agent Controller'
                 version='8.2.0.20090611_1150'
                 features='rac feature,Select 64bit RAC to install'
                 id='com.ibm.rational.agent.controller.solarissparc'/>
</install>

<preference value='true' name='com.ibm.cic.common.core.preferences.searchForUpdates'/>
<preference value='/opt/IBM/RACShared'
            name='com.ibm.cic.common.core.preferences.eclipseCache'/>
<preference value='30' name='com.ibm.cic.common.core.preferences.connectTimeout'/>
<preference value='30' name='com.ibm.cic.common.core.preferences.readTimeout'/>
<preference value='0' name='com.ibm.cic.common.core.preferences.downloadAutoRetryCount'/>
<preference value='true' name='offering.service.repositories.areUsed'/>
<preference value='false' name='com.ibm.cic.common.core.preferences.ssl.nonsecureMode'/>
<preference value='false'
       name='com.ibm.cic.common.core.preferences.http.disablePreemptiveAuthentication'/>
<preference value='true'
            name='com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts'/>
<preference value='false' name='PassportAdvantageIsEnabled'/>
</agent-input>
```

Detailed output will be generated to the console, which you should inspect for errors. Additional logs will also be generated in the `/var/ibm/InstallationManager/logs` directory.

Verify RAC Installation

After installation, there are several ways you can verify your installation. This section assumes you used the installation location shown in the sample response file: `/opt/IBM/RAC`.

Set up the command line environment for library linking, by executing:

```
LD_LIBRARY_PATH=/opt/IBM/RAC/AgentController/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH
```

Verify RAC Server Version is 4.6.0, using the command:

```
/opt/IBM/RAC/AgentController/bin/ACServer -v
```

Finally, start Rational Agent Controller, and execute the sample application, using the commands:

```
/opt/IBM/RAC/AgentController/bin/ACStart.sh
/opt/IBM/RAC/AgentController/bin/SampleClient
```

If your installation was successful, you should see output similar to the following:

```
Connected to the Agent Controller on "localhost" at port number #####

The Time Collector Agent ID: ###

Established a data channel with the agent.

Sending 5 Hello messages over data channel to TimeCollector ...

Start the TimeCollector ...

Incoming data: Hello from Time Collector Agent - Count 0

Incoming data: Hello from Time Collector Agent - Count 1

Incoming data: Hello from Time Collector Agent - Count 2

Incoming data: Hello from Time Collector Agent - Count 3

Incoming data: Hello from Time Collector Agent - Count 4


Stop the TimeCollector ...

Incoming data: Hello from Time Collector Agent - Count 5

Incoming data: Hello from Time Collector Agent - Count 6

Incoming data: Hello from Time Collector Agent - Count 7

Incoming data: Hello from Time Collector Agent - Count 8

Incoming data: Hello from Time Collector Agent - Count 9

Incoming data: Hello from Time Collector Agent - Count 10

All finished
Press any key to exit...
```

Configuring the Operating System's Environment Variables

You must configure 5 environment variables before attempting to start a JVM instance with profiling enabled. Each of these variables, and a description of their value is shown in the table below.

Variable Name	Value Description
`TPTP_AC_HOME`	The installation directory where Rational Agent Controller installed to. The default location on Solaris and Linux is `/opt/IBM/SDP/AgentController`
`JAVA_PROFILER_HOME`	The location of the org.eclipse.tptp.javaprofiler plugin directory, under `$TPTP_AC_HOME/plugins/`

`PROBEKIT_HOME`	The location of the org.eclipse.hyades.probekit plugin, under `$TPTP_AC_HOME/plugins/`

Linux Solaris
`LD_LIBRARY_PATH`
 	
AIX
`LIBPATH`
	Specifies the load library path used by the Java Virtual Machine for loading native libraries. It must include:
	`$JAVA_PROFILER_HOME`
	`$TPTP_AC_HOME/bin`
	`$TPTP_AC_HOME/lib`

`PATH`	The executable search path, also used by some JVMs to load native libraries. Must include:
	`$JAVA_PROFILER_HOME`
	`$TPTP_AC_HOME/bin`
	`$TPTP_AC_HOME/lib`
	`$PROBEKIT_HOME/lib`

As an example, on a Linux or UNIX operating system, the default RAC installation directory would be /opt/IBM/SDP/AgentController. The following statements, then, would properly configure the necessary environment variables using a BASH shell:

```
TPTP_AC_HOME=/opt/IBM/SDP/AgentController
JAVA_PROFILER_HOME=$TPTP_AC_HOME/plugins/org.eclipse.tptp.javaprofiler
PROBEKIT_HOME=$TPTP_AC_HOME/plugins/org.eclipse.hyades.probekit
LD_LIBRARY_PATH=$JAVA_PROFILER_HOME:$TPTP_AC_HOME/bin:$TPTP_AC_HOME/lib
LIBPATH=$LD_LIBRARY_PATH
PATH=$JAVA_PROFILER_HOME:$TPTP_AC_HOME/bin:$TPTP_AC_HOME/lib:$PROBEKIT_HOME/lib:$PATH

export TPTP_AC_HOME JAVA_PROFILER_HOME PROBEKIT_HOME LD_LIBRARY_PATH LIBPATH PATH
```

Note that in this example, both `LD_LIBRARY_PATH` and `LIBPATH` are set. This is done for compatibility. `LD_LIBRARY_PATH` must be set on Linux or Solaris platforms, where AIX uses `LIBPATH` instead.

Starting a JVM in Profiling Mode

Once you have successfully installed RAC and configured the environment variables, you can start any JVM using the following command line arguments to launch it with JVM profiling enabled:

```
java -agentlib:JPIBootLoader=JPIAgent:server=enabled;<profiler> <program>
```

Where <profiler> is one of:

1. CGProf – For execution time analysis, find out where your CPU time is going and zero in on performance bottlenecks.
2. HeapProf – for object allocation analysis. Keep track of your objects and find out where the problem spots are, or find application memory leaks.
3. ThreadProf – for threading analysis. Check the activity of your threads, resolve deadlocks and get detailed information on your application's monitor usage.

Section 2: WebSphere Application Server

WebSphere Application Server Configuration

You must now update the WebSphere Application Server configuration so that the WAS JVM can be started in profiling mode. This involves updating the setupCmdLine.sh script with the proper environment variables, as shown above. For standalone application servers, you should include these changes in the setupCmdLine.sh file found in the bin directory under the installation location for WAS. In a network deployment environment, you will want to make these changes to the setupCmdLine.sh scripts found under the node's bin directory, for each node that contains servers you will want to profile.

Starting the Server with Profiling Enabled

WebSphere Network Deployment

When using servers in a network deployment environment, you will have to manage the server's state manually. This includes modifying the server's JVM arguments, and starting and stopping the serverusing the Integrated Solutions Console, or via the command line.

Using the Integrated Solutions Console

To start a server with profiling enabled, you must set the appropriate JVM arguments for that server, as described above in Starting a JVM in Profiling Mode. You can configure a server's JVM arguments in the Integrated Solutions Console by going to:

Servers->Server Types->Application Servers

Then, click on the appropriate server name, and browse to:

Server Infrastructure->Java and Process Management->Process Definition

Under “Additional Properties,” click “Java Virtual Machine.” In here, you will see a field labelled “Generic JVM Arguments.” Add the appropriate JVM arguments here. Save your configuration, and synchronize the changes to the nodes.

The next time you start the server, it will launch with the new JVM arguments. To disable profiling, simply remove the JVM arguments from this same property.

Manually using the Command Line

You can choose to manage the JVM arguments manually using the command line, instead of using the Integrated Solutions Console as described above. The server's configuration file, server.xml, is located under your WebSphere installation directory, in:

```
profiles/profileName/config/cells/cellName/nodes
      /nodeName/servers/serverName/server.xml
```

Open the file in a text editor, and locate the `jvmEntries` element. Set the JVM arguments by editing the genericJvmArguments attribute value.

Alternatively, the listing below gives a shell script that can be used to start a WebSphere Application Server with profiling enabled. The script automates the task of editing the server.xml file, and is useful if you will be enabling and disabling profiling of the server frequently.

Once the script has started the server, it waits for 60 seconds and then reverts the server's configuration back to the way it was originally. The 60 second wait is to allow any RAD instances that may be using this server to properly detect the “profiling” state. This is discussed further in the next section.

```
#!/bin/bash
#
# Enables profiling in the specified server's server.xml configuration file, then
# starts the server.
#
# Usage:
#    ./profileServer.sh <server_name>
#
# Note:
#    By default this script enables the CGProf profiler. To change this, modify
#    the value of the profilingArgs variable, below.
#
#      Where <profiler> is either:
#         CGProf - for execution analysis
#         HeapProf - for object allocation and heap analysis
#         ThreadProf - for thread analysis

binDir=`dirname $0`
. $binDir/setupCmdLine.sh

profilingArgs='-agentlib:JPIBootLoader=JPIAgent:server=enabled;CGProf'
serverCfgPath=$USER_INSTALL_ROOT/config/cells/$WAS_CELL/nodes/$WAS_NODE/servers/$1

if [ -e $serverCfgPath/server.xml ]; then
        cp $serverCfgPath/server.xml $serverCfgPath/server.xml.backup_$$
        currentJvmArgs=`grep 'genericJvmArguments' $serverCfgPath/server.xml |    \
             sed 's|.*genericJvmArguments=\"\([^\"]*\)\".*|\1|'`

        echo "Current genericJvmArguments: $currentJvmArgs"
        echo $currentJvmArgs | grep "\\$profilingArgs" > /dev/null

        if [ $? -eq 0 ]; then
                echo "Server configuration already setup for profiling... not changing"
        else
                sed "s/genericJvmArguments=\"/genericJvmArguments=\"$profilingArgs /" \
                     $serverCfgPath/server.xml.backup_$$ > $serverCfgPath/server.xml
                echo "Added $profilingArgs to server configuration file"
        fi
else
        echo "Could not locate server.xml in path $serverCfgPath."
        exit
fi

$binDir/startServer.sh $1

echo "Waiting 60 seconds before restoring configuration so RAD can query server
status..."
for i in 0 1 2 3 4 5 6 7 8 9
do
        for i in 0 2 4 6 8
        do
```

```
                sleep 2
                echo -n '.'
        done
done


echo -e "\nReverting server configuation back to original state..."
cp $serverCfgPath/server.xml.backup_$$ $serverCfgPath/server.xml
rm $serverCfgPath/server.xml.backup_$$
```

Standalone WebSphere Application Server

When using a standalone WebSphere Application Server, Rational Application Developer has the capability to manage the server's JVM arguments for you. In the event you need to edit them manually later, refer back to the previous section for WebSphere Application Server ND. Those steps are applicable to a standalone WAS server, but not necessary.

You should have already added the server to your RAD workspace. For specific information on how to do this, see Creating a WebSphere Application Server in the RAD 7.5 Info Center.

You will need to start the server manually, since RAD does not have the ability to start a stopped remote application server.

To enable profiling on the started server, simply right click the server in your RAD workspace, and select “Restart in Profiling.” You will be prompted with a dialog where you can choose the profiler to use, corresponding to the three JVM profiler arguments discussed above. RAD will update the server's configuration, and restart it.

Section 3: Profiling WebSphere Applications using Rational Application Developer

Once your application server is running with profiling enabled, you can use Rational Application Developer (RAD) to gather profiling data from the remote JVM.

Connecting RAD to Your WebSphere Application Server

WebSphere Network Deployment

To profile an application hosted on a managed server, you must manually connect your RAD workspace to the remote agent. This process can be used to gather profiling data from any JVM running in profiling mode.

Be sure you have followed the directions above to start your managed server with profiling enabled. Then, in your RAD workspace, switch to the Profilng and Logging perspective and follow these steps:

1. Click on the Profile drop-down menu, and select **Profile...** The Profile wizard opens.

2. Double click on **Attach to Agent**. A new configuration is created.
3. Under the **Host** tab, specify the **Host name or IP address** and the **Agent Controller** port (if different from the default) of the agent controller on the remote server.
4. Switch to the Agents tab to view a list of available agents that can be attached to (Note: if the list is empty then no agents are available to attach to on the selected host. You can always use the **Refresh** button to get the list of available agents)
5. Check the analysis types that you wish to use to attach to the available agent.
6. Configure the profiling filters as required. Refer to the Specifying profiling criteria topic for more information on configuring the profiling filters.
7. Click **Apply** to apply the changes.
8. Click **Profile** to attach to the application.
9. The Profiling Monitor view is refreshed displaying the agent representing your application.

Once you are connected to the remote agent, RAD will prompt you to switch to the "Profiling and Logging Perspective" where you can gather profiling data.

Standalone WebSphere Application Server

Follow the instructions above for starting your standalone WebSphere Application Server in profiling mode. After RAD has detected that the server has started, it detects that the server's JVM is running in profiling mode.

Once RAD detects this, it will automatically prompt you to switch to the "Profiling and Logging Perspective" where you can gather profiling data.

Using the Profiling and Logging Perspective

Once you have connected your workspace to the agent controller, you can begin gather and viewing data regarding your application. You can find further information on this in the RAD Info Center, under Monitoring and Profiling Applications.

Conclusion

This article guided you through the configuration of IBM® Rational® tools for profiling JEE applications on remote WebSphere Application Server. You were given all the information found necessary to successfully profile your enterprise applications in QA or Production WebSphere environments, allowing you to tune and customize your applications for maximum performance.

Resources

Installation Instructions for IBM Rational Agent Controller
http://www-01.ibm.com/support/docview.wss?rs=2042&uid=swg27013420

WebSphere Application Server Information Centers
http://www-01.ibm.com/software/webservers/appserv/was/library/

Rational Application Developer 7.5 Information Center
http://publib.boulder.ibm.com/infocenter/radhelp/v7r5/index.jsp

About the author

Timothy C. Fanelli is a software engineer for IBM Poughkeepsie working from Clarkson University, New York. You can reach Timothy at tfanelli@us.ibm.com.