

Step-by-step ProActive debugging

Arnaud Contes: *arnaud.contes@activeeon.com*

Maxime Menant: *maxime.menant@activeeon.com*

Etienne Vallette d'Osia: *etienne.vallette_d_osia@sophia.inria.fr*

Laurent Vanni: *laurent.vanni@sophia.inria.fr*



Debugging a ProActive application :

Outline

- Context
- Solution:
 - StepByStep
 - Connecting debuggers to JVMs
- Demonstration
- Thereafter



Context



Context

1 - How are you debugging your ProActive application ?

`System.out.println()` ?

2 - How to connect a debugger to a remote Runtime ?

Ssh tunneling, port forwarding, ... Complex, isn't it ?



Solution: Step-by-step



Step-by-step: controlling application at service level

The screenshot displays the ProActive Parallel Suite monitoring interface. The main window shows a hierarchical view of virtual nodes and active objects (Ao) across three hosts. A legend window titled "Active Object State Information" is open, showing details for Ao2#7, including its ID and Host URL. A "Set Step by Step motion Delay" dialog box is also visible, allowing the user to select breakpoint types to stop at, with "SendRequest" and "NewService" selected. The console window at the bottom shows the application's execution logs.

Legend - Active Object State Information

Click on an Active Object to see its informations.
To go to the next breakpoint, click on the breakpoint type.

Active Object	Value
Ao2#7	
Ao2#1	
id	33bae1a5-11c517e0ae1--7fe8--053e28c6a96aa54e-33bae1a5-11
Host URL	rmi://eon1.inria.fr:6659/PA_JVM1126560748_GCMNode-0
StepByStep	Enabled
Not blocked on a breakpoint...	
List of Breakpoints enabled	End Immediate Service, New Immediate Service, New Service, S
Ao1#5	

Set Step by Step motion Delay

Select the breakpoint type where you want to stop :

- SendRequest
- NewService
- EndService
- NewImmediateService
- EndImmediateService

OK Cancel

Console

```
Monitoring
16:22:31 => Exploring eon2.inria.fr:6659:Linux(OS version: 2.6.23.15-80.1
16:22:41 => Exploring eon3.inria.fr:6659:Linux(OS version: 2.6.23.15-80.1
16:22:41 => Exploring eon1:6659:Linux(OS version: 2.6.23.15-80.fc7) with
16:22:41 => Exploring eon2.inria.fr:6659:Linux(OS version: 2.6.23.15-80.1
16:22:52 => Exploring eon3.inria.fr:6659:Linux(OS version: 2.6.23.15-80.1
16:22:52 => Exploring eon1:6659:Linux(OS version: 2.6.23.15-80.fc7) with
16:22:52 => Exploring eon2.inria.fr:6659:Linux(OS version: 2.6.23.15-80.1
```



Step-by-step: Controls

Different levels of selection for Active Object present in:

world / host / jvm / node

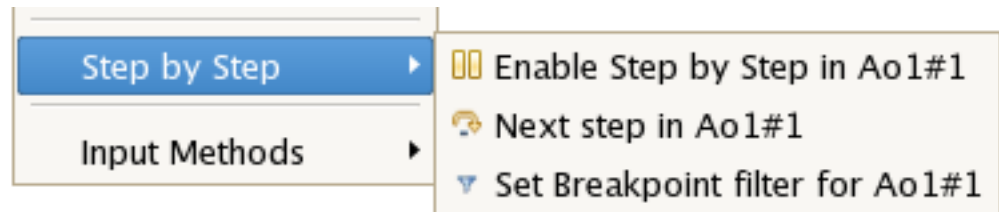
With the following features:

- Pause / Resume
- Next Step
- Slow Motion
- Filters

Toolbar for controlling application at world level:

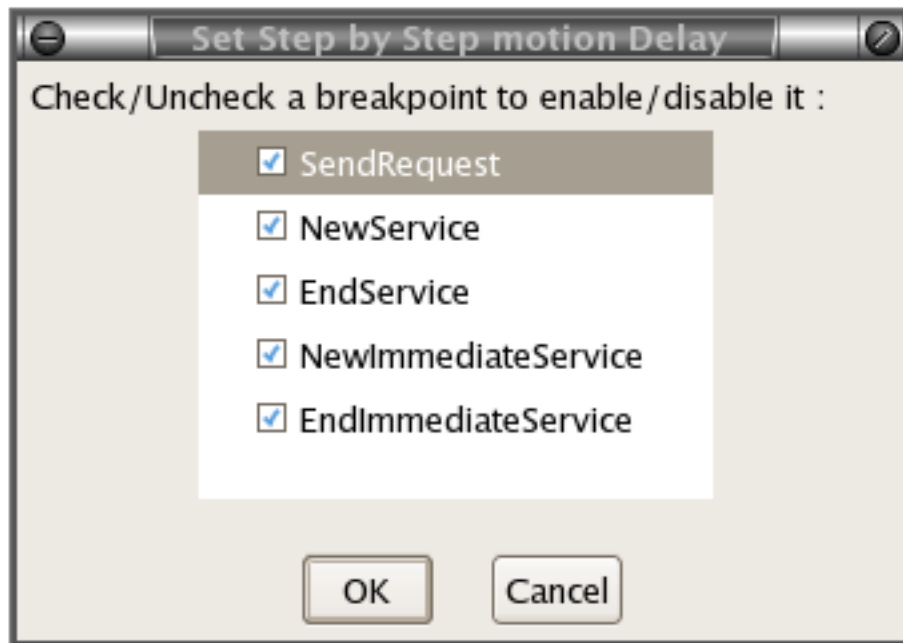


Contextual menu for the others:



Step-by-step: Breakpoints overview

There are different kinds of breakpoints at service level.



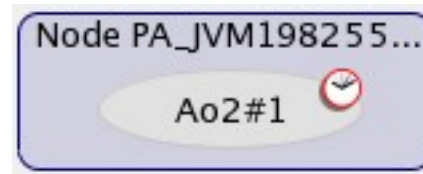
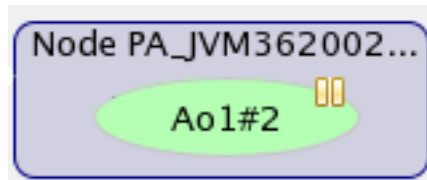
You can enable or disable each breakpoint through this filters view.



Step-by-step: Visual feedback

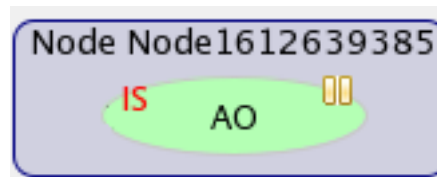
Some flags are used to show the state of an Active Object during the step by step:

Pause



Slow Motion

Immediate Service



Step-by-step: Informations

Active Object State Information ⓘ

Clear View Click on an Active Object to see its informations.
To go to the next breakpoint, click on the breakpoint type.

Active Object	Value
ImmediateHello#1	
Id	7b165f79-11d057a6b13--7fe6--c66d431d4f9350b4-7b165f79-11d057a6b13--8000
Host URL	rmi://puravida.inria.fr:6659/Node755625977
StepByStep	Enabled
Breakpoint #0	
Blocked on breakpoint	End Immediate Service (IS) (click here to go to the next step)
On method	totoS (IS)
Breakpoint #1	
Blocked on breakpoint	End Immediate Service (IS) (click here to go to the next step)
On method	toto (IS)
List of Breakpoints enabled	End Service, New Service, End Immediate Service, New Immediate Service, Send Request

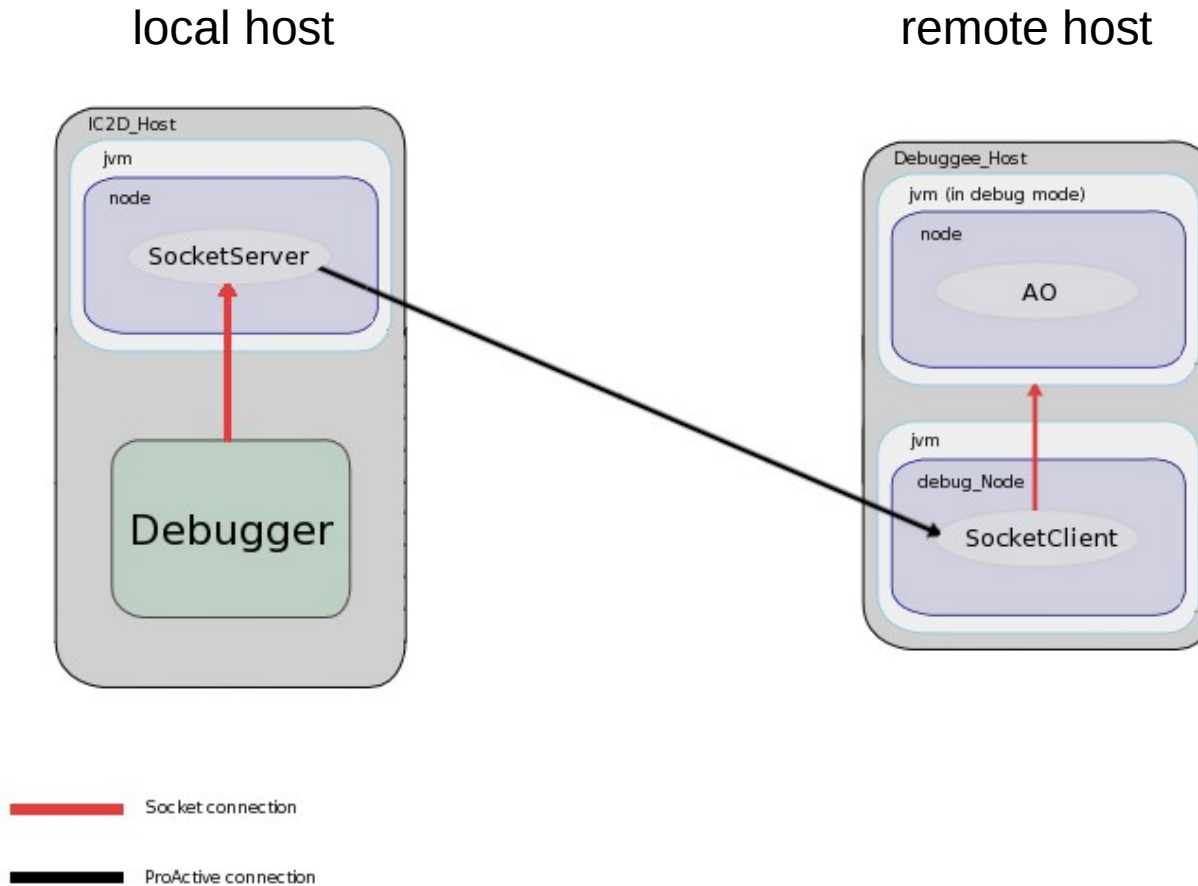


Solution: Debugger connection



Debugger connection:

Connecting remote JVM to a local debugger using ProActive connection



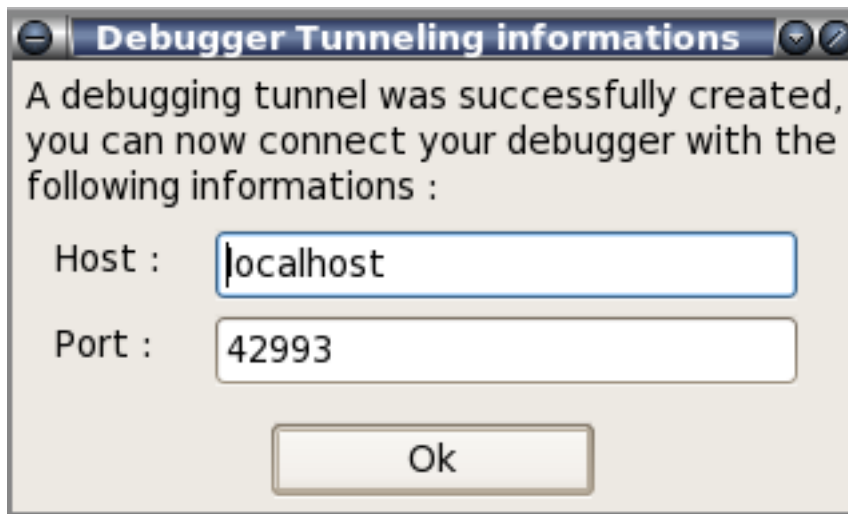
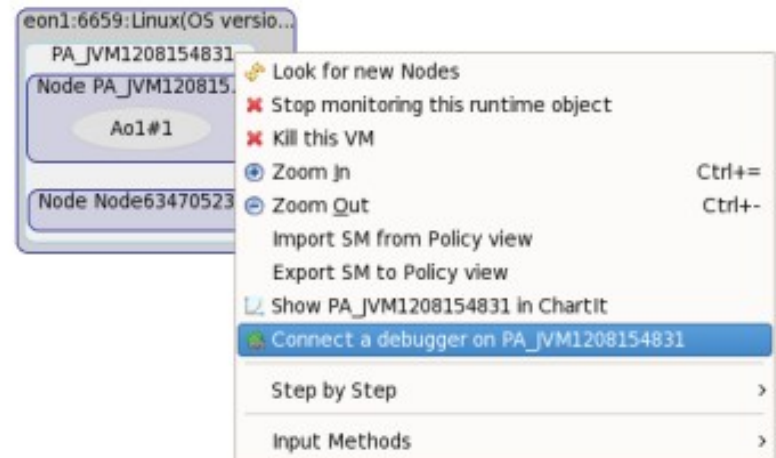
Debugger connection: Usage

First, enable the debug mode for all the JVMs used in the application within the XML deployment file by adding one line in the configuration of the application.



Debugger connection: Usage

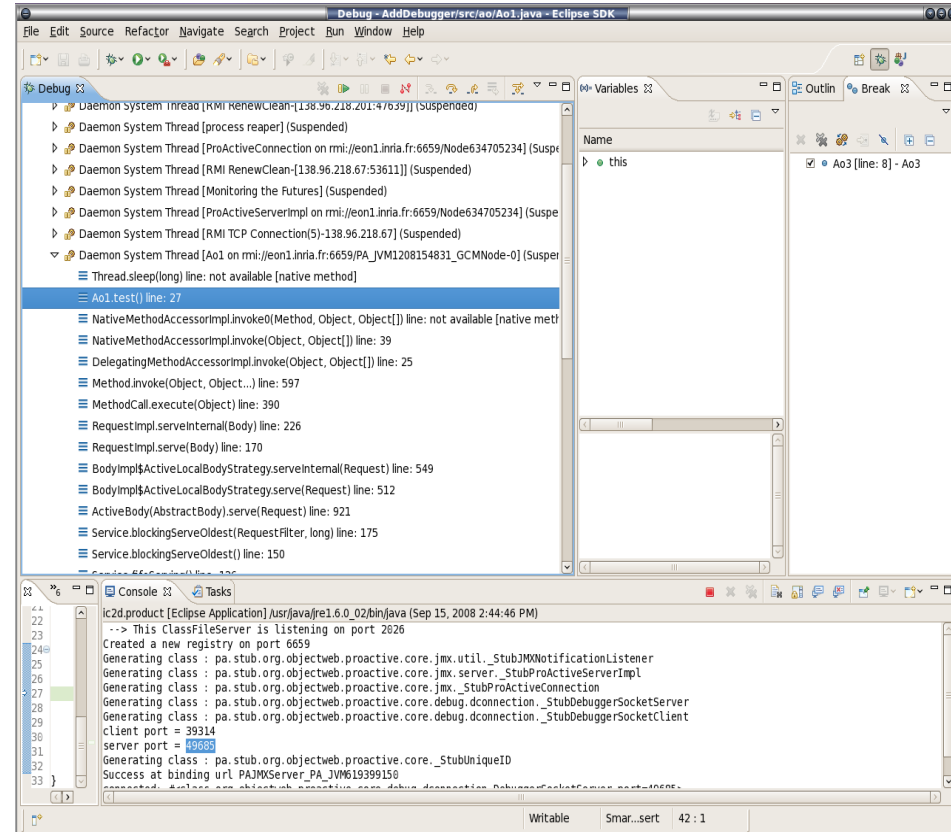
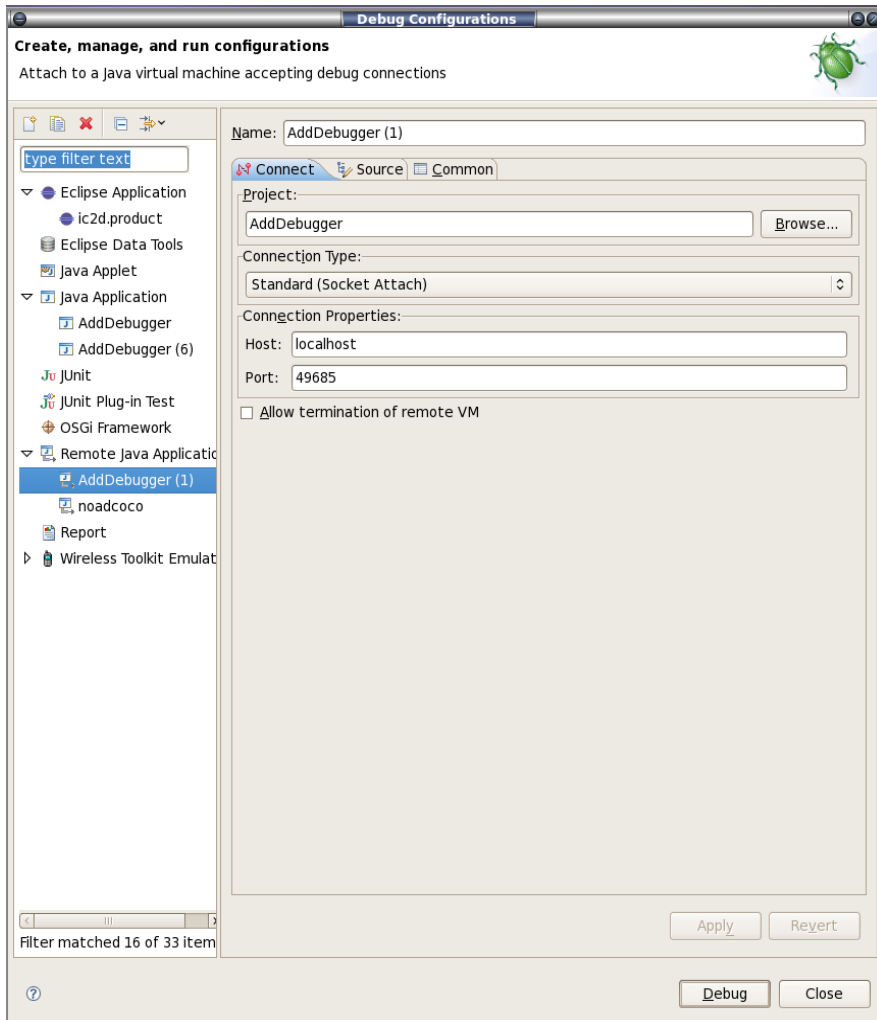
Then, select the JVM that you want to debug



And connect your debugger to localhost with the given port...



Debugger connection: Example



Demonstration



Conclusion: Actual limitations

- Step-by-step
 - Breakpoints are parts of code
 - Partial control of the application (act only with services)
- Debugger connection
 - One debugger per JVM at the same time
 - Local binding performed by the user



Conclusion: Thereafter

The enhancement of ProActive debugging features :

- Distributed Debugging
 - A single interface to debug distributed JVMs (place breakpoint, etc.)
- Replay From Checkpointing
 - Including some work to upgrade if necessary the Fault Tolerance mechanism: Checkpoint Server etc. (stable memory, redundancy).
- Distributed Service Debugging
 - Capacity to view and breakpoint the flow of a given service, and consequent asynchronous calls.



Thanks for your attention!

Any Questions...

