

Prep1: Preparation for PaP demonstrations

Generic terms used here and in specifications of each of the demonstrations:

<PaPSupportRoot>: An URL root location at web-server where PaP Support software is located (i.e. "http://<some web-location>").

<PaPAppRoot>: An URL root location at web-server where some PaP application is located (i.e. "http://<some web-location>").

<PaPBoot>: A directory location where the "PaP boot software" for a PaP server is located (e.g. "c:\temp\PaPboot").

<Node1>: A network node capable to be a web-server (e.g. "nodeNameXXX" or "129.241.238.255").

<Node3>, **<Node4>**: Network nodes capable to serve as PaP Server nodes. A PaP Server node is a network connected PC running Windows NT 4.

What to do	How to do it	What should happen
- Determine the values to be used for the generic terms defined in the previous, and referred within this document. - Remember to substitute the real values applied for the generic values where applied in the demo specifications.	Determine real values for: <Node1> = <Node3> = <Node4> = <PaPSupportRoot> = <PaPAppRoot> = <Node3>; <PaPBoot> = <Node4>; <PaPBoot> =	If three nodes are not available, it is possible to run all demos with only one node, but that will of-course reduce the effect of the demos.
Preparation of <Node 1>	- Install PaP Support software and defined plays "School" and "Watcher" (i.e. make all the software available from internet). - The PaP software may be taken from CD-ROM, floppy disk set, or internet - depending on what is available. - Note that the PaP Support and different PaP applications can all be located at different nodes, or they can be located at some PaP Server node if that node also is a web-server.	

What to do	How to do it	What should happen
If not already done, install the PaP Support software at the web-server.	Copy and unpack the file "PaPSupport.zip" at the selected "<PaPRoot>" location.	
If not already done, install the PaP application software at the web-server.	Copy and unpack the file "PaPApplications.zip" at the selected "<PaPAppRoot>" location.	
Preparation of <Node3> and <Node4> Install necessary software at <Node3> and <Node4> so that the two nodes are capable to serve as "PaP Server nodes".	- PaP specific software may be taken from CD-ROM, floppy disk set, or internet - depending on what is available. - Java software is downloaded from web.	
If not already installed, install Java Runtime Environments, version 1.2	- Usually JRE is not installed if command "java" from a "DOS-window" gives message "The name specified is not recognised as an internal or external command, operable program or batch file". - If JRE has already been installed, the path normally will be "C:\Program Files\JavaSoft\JRE\1.2\bin". - PaP unspecific software is the Java Runtime Environments, version 1.2 (JRE 1.2) that may be downloaded from internet address "http://www.sun.com/".	
If not already installed, install the PaP bootstrap.	Copy and unpack the file "PaPboot.zip" at the selected "<PaPboot>" location>	All files related to PaP bootstrap will be located at the <PaPboot> directory when installed, and the installation do not modify any other parts of the node.

What to do	How to do it	What should happen
PaP configuration of <Node3>.	- Use a text editor (e.g. Notepad) and modify "pap.cnf" in the <PaPboot> directory according to what is specified below. - The content of the file should be (remember to substitute for generic values specified in <...>): <code>codebase = <PaPSupportRoot></code> <code>policy = <PaPSupportRoot>policy</code> <code>homeinterface =</code> <code>Actor://localhost/pas1/PaP.Director1</code> <code>debugserver = localhost</code> <code>nodeprofile = profDefault</code>	- Director will execute at own node. - If application configuration is used (as for "School"), the "profDefault" configuration is applied.
PaP configuration of <Node4>.	- Use a text editor (e.g. Notepad) and modify "pap.cnf" in the <PaPboot> directory according to what is specified below. - The content of the file should be: <code>codebase = <PaPSupportRoot></code> <code>policy = <PaPSupportRoot>policy</code> <code>homeinterface =</code> <code>Actor://<Node3>/pas1/PaP.Director1¹</code> <code>debugserver = localhost</code> <code>nodeprofile = profAlt1</code>	- Director will execute at <Node3>. - If application configuration is used (as for "School"), the "profAlt1" configuration is applied. - Note that default location of director is at <Node3>. This requires that running PaP applications at <Node4> also implies that PaP Server at <Node3> is running.
Start PaP Server at <Node3> and <Node4>		It is possible for PaP applications to execute on specified PaP Servers.

¹ Note that default location of director is at <Node3>. This requires that running PaP applications at <Node4> also implies that PaP Server at <Node3> is running.

What to do	How to do it	What should happen
- Start a PNES instance at each of the PaP Server nodes involved in running PaP demos (i.e. <Node3> and <Node4>). - Only one PNES instance shall exist at each PaP Server node.	From a DOS window perform the commands: <pre>cd <PaPBoot> java PaP.boot PaP.PNES</pre>	A debug window (from now on called DW) indicating the PNES instance running will appear on the node where the command was executed. - The window is divided into two parts: The operator command section (OCS) (upper part), and the trace output window (TOS) (lower part). - OCS will extensively be used when running the tests for specifying operator controlled PaP commands. Commands in OCS may either be written, or selected by using arrow up/down keys. - Written commands are executed when pushing the "Return" key.
Run any Pap define demo (Demo 1 - Demo 4) Continue from here when running a PaP demo.	From now on <Node3> and <Node4> are ready to perform PaP applications. This can be demonstrated by running one or more of the defined demos (i.e. Demo 1 - Demo 4).	
... Stop PaP Server Shutdown of all PaP applications on a PaP Server, and do not longer allow to run PaP applications on that server.		

What to do	How to do it	What should happen
Initiate shutdown of a PaP Server.	- From PNES debug window perform command: Unbind. - An alternative is to stop the PNES process from the Windows NT task manager. In that case there will be an uncontrolled stop of each of the Java VMs that constitutes the PaP system.	All eventual performing PaP applications on that node will terminate, and also all PAS instances and the PNES instance itself.

Demo1: Flexibility in use of the distributed architecture

What to do	How to do it	What should happen
Do all necessary preparation for a demonstration.	<See specification of "Prep1: Preparations for PaP demonstrations" in the previous>	All involved nodes ready for demonstration execution.
Plug-in of involved plays		A director actor, and the PAS instance where it executes will appear, and the involved plays are plugged-in to the Repertoire base. Demonstrates - Plays can be installed from different actors managed by the same director.
Plug-in the "Watcher" play from <Node3>	<Node3> ² : <i>PlayPlugIn Watcher v1_1</i> <PaPApplRoot>	- After some time delay a new DW (i.e. debug window) indicating that PAS instance "pas1" is running at <Node3>. - A new DW indicating that director actor named "Director1" is running within "pas1" at <Node3>. - Response of command should indicate plug-in Ok in PNES DW (i.e. "COMMAND result: OK(...)).

² When nothing else is specified this means an operator command specified in the PNES debug window (i.e. PNES OCS).

What to do	How to do it	What should happen
Plug-in the "School" play from <Node 4>	<Node4>: <i>PlayPlugIn School v1_1 <PaPApp/Root></i>	- Response of command should indicate plug-in Ok in PNES DW. - Both <Node3> and <Node4> have the same Home Interface (i.e. they are attached to the same Director. - You can verify that both plays are installed at the same director by executing the command "ListPlays" at the director's DW.
Control where an application executes		The use of the "Watcher" and "School" applications is demonstrated. Demonstrates: - Applications running at different nodes can be started from same node. - Applications running at the same node can be started at different nodes. - Termination of applications from the application itself or another application. - PAS instances terminate automatically when they have no actors to control.

What to do	How to do it	What should happen
Start an actor with identifier "WS1", playing the role "WatchServer" and running within PAS instance "pas1" at node "<Node3>".	<Node3>: ActorPlugIn Actor://<Node3>/pas1/WS1 WatchServer	- A DW indicating that the "WS1" actor instance is running appears at "<Node3>". - The Watcher application specific window that presents subscription information appears at "<Node3>".
Start an actor, playing role "WatchServer" with identifier "WS1" running within PAS instance "pas1" at node "<Node4>".	<Node3>: ActorPlugIn Actor://<Node4>/pas1/WS1 WatchServer	- A DW indication a new PAS instance "pas1" is running at <Node4>. - A DW and a Watcher specific window at <Node4> indicate a WatchServer instance is running. - PaP specific actions are registered in the Watcher window at <Node3>.
Start an actor, playing role "WatchServer" with identifier "WS2" running within PAS instance "pas1" at node "<Node3>".	<Node4>: ActorPlugIn Actor://<Node3>/pas1/WS2 WatchServer	- A new DW for actor WS2 and a new watcher window appears at "<Node3>".
Initiate the termination of an application by performing a "ActorPlugOut" action.	- <Node4>, WS2³: Status <note the identifier for the only role session <rs>. - <Node4>, WS2: ActorPlugOut <rs>	- Actor "WS2" will terminate, and all its owned windows will disappear. - The actor is terminated because the mode "auto termination when no role sessions" has been used.

³ Initiate command from debug window of actor "WS2"

What to do	How to do it	What should happen
Initiate the termination of an application from the application itself.	<Node4>: <Push the "X" button in the Watcher application window>	- Actor "WS1" at node "<Node4>" will terminate, and its DW. - The PAS instance "pas1" at "<Node4>" will terminate, because no more actors is controlled by that PAS instance.
Demonstrate two alternatives for configuration of the School application. Alternative1: Locate SchoolServer and SchoolRTLServer actors at same node and PAS instance as SchoolClient that caused its creation. Alternative2: Locate SchoolServer and SchoolRTLServer actors at same node and PAS instance as director for the SchoolClient that caused its creation.	Preconditions: a) Use a Watcher application, as used in the previous to verify where different parts of "Scool" executes. b) The "School" play has already been plugged-in. c) The demonstration is based on the fact that "<Node3>" and "<Node4>" has different configurations - defined in file "<PaPboot>/pap.cnf" (done during demo preparations). At "<Node3>" the element "nodeprofile" has value "profDefault", while at "<Node4>" it has the value "profAlt1". This implies that the School application will select among the two defined alternatives when configuring.	Demonstrates: - "SchoolServer" and "RTLServer" actors will be located at the same node and PAS instance as the actor that caused their creation. - "SchoolServer" and "RTLServer" actors will be located at the same node and PAS instance as the director for the actor that caused their creations.
Alternative1: Locate SchoolServer and SchoolRTLServer actors at same node and PAS instance as SchoolClient that caused its creation.		

What to do	How to do it	What should happen
Start a school client (i.e. teacher or student) by plug-in and actor playing the role "SchoolClient" at PAS instance "pas1" at node "<Node3>". Because an actor with that name and identifier do not already exists it will be created.	<i>ActorPlugIn Actor://<Node3>/pas1/sc1 SchoolClient</i>	- Two DWs indicating that the "sc1" actor instance and its co-operating user interface is running at "<Node3>". - The SchoolClient "School: Logon" window should appear at "<Node3>". - The Watcher window is updated indicating the two created School application actors.
Log on the user "u" at school "s". The user's password is "p".	In "School: Logon" window, set values: School=s, User=u, Password=p, then push "Ok" button	- A new DW indication a "SchoolServer" instance is running appears. - The "School:Logon" window disappears, and the new window "School: Work to do" appears. Demonstrates For this configuration the "School Server" instance appears at the same node and PAS instance where the SchoolClient that caused its creation is located.
Select courses and lectures at the work type to perform	In "School: Work to do", select "CoursesAndLectures" and push "Ok" button	- The "School: Work to do" window disappears, and the new window "School: Courses and Lectures" appears.

What to do	How to do it	What should happen
Select to do "Real time lecture" for "Course1" and "Lecture11". The "Perform: ..." window that will appear consists of two parts. The upper part where operator can write text (i.e. messages), which will be displayed in the lower (i.e. the yellow field) for all users performing real time lecture for the same course and lecture.	In "School: Courses and Lectures", select "Course1", "Lecture11" and "Teacher", and then push "Real time lecture" button.	- A new DW indicating a "RTLServer" serving "Course1" and "Lecture11" appears. - The "School: Courses and Lectures" window disappears and the window "School: Real time lecture" appears. - The window "Perform: ..." appears. Demonstrates For this configuration the "RTLServer" instance, as for the "SchoolServer" instance, appears at the same PAS instance as the actor that caused its creation.
Plug-in and actor playing the role "SchoolClient" at PAS instance "pas1" at node "<Node4>". Because an actor with that identifier do not already exists it will be created.	<Node3>: ActorPlugIn Actor://<Node4>/pas1/sc1 SchoolClient	- Two DWs indicating that a PAS instance and the "sc1" actor instance is running at "<Node4>". - The SchoolClient "School: Logon" window should appear at "<Node4>". - The Watcher window is updated.

What to do	How to do it	What should happen
Log on the same user as done previously - at the same school, course and lecture. The two logged on SchoolClients may now send messages to each other by typing in the upper field in the "Perform:..." window.	<Logon the same user at same school, course and lecture for the user at <Node4> as previously done at <Node3>>.	Ends up with a "School: Real time lecture" and a "Perform: ..." window at <Node4>. Demonstrates - Two logged on SchoolClients are served by the same "SchoolServer" and "RTLServer" instances. - This shows that the SchoolClient logged on at <Node4> is capable of finding the SchoolServer and RTLServer created by the logon performed at <Node3>, even if the application (i.e. the play "School") have now knowledge about any node <Node3>, and/or that there has been logged on some user there.

What to do	How to do it	What should happen
Log-off both SchoolClients.	At <Node3> and <Node4>: - At window "School: Real time lecture", push "Terminate" button. - At window "School: Work to do", select "Terminate" and push "Ok" button.	- All the School application specific windows disappear. - All the DW for actors disappears, except for the window serving the "Director" located at <Node3>. - The PAS instance window at <Node4> disappears. Demonstrates Actor and PAS instances are discarded when no needs for them any more.
Alternative2: Locate SchoolServer and SchoolRTLServer actors at same node and PAS instance as director for the SchoolClient that caused its creation. Plug-in and actor playing the role "SchoolClient" at PAS instance "pas1" at node "<Node4>". Because an actor with that name and identifier do not already exists it will be created.	<Node3>: <i>ActorPlugIn Actor://<Node4>/pas1/sc1 SchoolClient</i>	- Two DWs indicating a PAS instance and the "sc1" actor instance is running at "<Node4>". - The SchoolClient "School: Logon" window should appear at "<Node4>".

What to do	How to do it	What should happen
Start real time lecture for user "u" at school "s", course "Course1" and lecture "Lecture11". Note that in this situation the "SchoolServer" and "RTLServer" instances appears at <Node3>, which is NOT the node where the SchoolClient related actors are located. This is because the School application is configured to run the common servers at the node where director is located. Start "real time lecture" for user "u" at school "s" for course "Course1" and lecture "Lecture11".	<Complete the SchoolClient logon at <Node4> as previously done, until the "Perform: ..." window appear at <Node4>	Different DW and School application windows appears and disappears.
	<Start-up and logon a SchoolClient at <Node3> in the same way as shown in the previous, until the "Perform: ..." window appears> <Send messages between the two logged on users as demonstrated in the previous>	-Different DW and School application windows appears and disappears. Demonstrates - Two logged on SchoolClients are served by the same "SchoolServer" and "RTLServer" instances. - This shows that the SchoolClient logged on at <Node3> is capable of finding the SchoolServer and RTLServer created by the logon performed at <Node4>, even if the application (i.e. the play "School") have now knowledge about any node <Node4>.

What to do	How to do it	What should happen
Terminate the School application Log-off School users.	At <Node3> and <Node4>: <Log-off the two logged on users the same way as demonstrated earlier by using the School application windows> <Push the "X" button in the Watcher application window>	Different DWs disappears, and School application windows appears and disappears. - Watcher application window disappears - Watcher actor DW window disappears
Terminate Watcher instance		
Plug-out the earlier plugged-in plays "Watcher" and "School".	<Node3>: PlayPlugOut Watcher <Node4>: PlayPlugOut School	- DW indication Director at <Node3> terminates. - DW indicating "pas1" at <Node3> terminates. Demonstrates Actor and PAS allocated resources are released when not used any more.

Demo2: Different versions of the same play

What to do	How to do it	What should happen
Do all necessary preparation for a demonstration. NOTE: Only one PaP Server is needed for this demonstration. In the specifications below <Node3> has been used. If you chose to use <Node4> in stead, remember that PaP Server at <Node3> must be started, because the director for <Node4> runs there. Demonstrate use of different versions of the same play	<See specification of "Prep 1" in the previous> The "Watcher" play exits in two versions - versions v1_1 and v1_2. The difference between these two versions is indicated for the user by displaying the version identifier in the WatchServer application specific window header.	Involved nodes ready for demonstration execution. Demonstrates - Multiple versions of the same play may exist at the same director. - If multiple versions of the same play exist, the last plugged in version is used.⁴ - When last plugged in version is plugged out, the previous plugged in version will be used.

⁴ Note that this is not according to the procedure defined in the specifications.

What to do	How to do it	What should happen
Plug-in of the "Watcher" play, version "v1_1" is done from <Node3>	<Node3> ⁵ : <i>PlayPlugIn Watcher v1_1</i> <PaPAppRoot>	- A new DW indicating that PAS instance "pas1" is running at <Node3>. - A new DW indicating that director actor named "Director1" is running within "pas1" at <Node3>. - Response of command should indicate play plug-in Ok in PNES DW.
- Start an actor performing the role "WatchServer" defined in the play "Watcher" with version "v1_1". - The command "ListPlays" in the director actor DW will confirm that version "v1_1" of "Watcher" is the only plugged-in play.	<Node3>: <i>ActorPlugIn Actor://<Node3>/pas1/WS1 WatchServer</i>	- A new DW indicating an actor instance playing role "WatchServer" is running. - A "WatchServer" specific window used for presentation of subscription information shows up. - The play version identifier "v1_1" is shown in the watcher window header.
Plug-in the "Watcher" play, version "v1_2" from <Node3>	<Node3>: <i>PlayPlugIn Watcher v1_2</i> <PaPAppRoot>	- Response of command should indicate plug-in Ok in PNES DW.

⁵ When nothing else is specified this means an operator command specified in the PNES debug window (i.e. PNES OCS).

What to do	How to do it	What should happen
Start an actor performing the role "WatchServer".	<Node3>: <i>ActorPlugIn Actor:://<Node3>/pas1/WS2 WatchServer</i>	- A new DW window and a watcher window appear. - The play version identifier "v1_2" is shown in the watcher window header. Demonstrates - The last plugged in play version is used
Plug-out version "v1_2" of "Watcher" play.	<Node3>: <i>PlayPlugOut Watcher v1_2</i>	- Response of command should indicate plug-out Ok in PNES DW.
- Start an actor performing the role "WatchServer". - The command "ListPlays" in the director actor DW will confirm that version "v1_1" of "Watcher" is the only plugged-in play.	<Node3>: <i>ActorPlugIn Actor:://<Node3>/pas1/WS3 WatchServer</i>	- A new DW window and a watcher window appears. - The play version identifier "v1_1" is shown in the watcher window header. Demonstrates - When a play version is removed, the previous plugged-in version is used.
Clean-up from the demo by terminating the running watcher instance, and plug-out of all versions of the Watcher play.	- <Node3>: Push the "X" button on the remaining Watcher instance window. - <Node3>: <i>PlayPlugOut Watcher</i>	- All windows terminate, except for the PNES instance window.

Demo3: Dynamic change of behaviour for existing actors

What to do	How to do it	What should happen
Do all necessary preparation for a demonstration. NOTE: Only one PaP Server is needed for this demonstration. In the specifications below <Node3> has been used.	<See specification of "Prep 1" in the previous>	All involved nodes ready for demonstration execution.

What to do	How to do it	What should happen
Demonstrate dynamic change of behaviour for existing actors	- The "Watcher" play has defined two almost identical roles - roles "WatchServer" and "WatchServerNew". The difference between these two roles is indicated for the user by displaying the role identifier in the WatchServer application specific window header. - An "ActorChangeBehaviour" may be executed for an actor playing the "WatchServer" role, and the actor behaviour will change to play the role "WatchServerNew". This change of actor behaviour is initiated by sending the message "UpgradeBehaviour" to the actor.	Demonstrates - Use of the ActorChangeBehaviour function
Plug-in of the "Watcher" play, version "v1_1" from <Node3>	<Node3> ⁶ : <i>PlayPlugIn Watcher v1_1</i> <PaPAppRoot>	- A new DW indicating that PAS instance "pas1" is running at <Node3>. - A new DW indicating that director actor named "Director1" is running within "pas1" at <Node3>. - Response of command should indicate plug-in Ok in PNES DW.

⁶ When nothing else is specified this means an operator command specified in the PNES debug window (i.e. PNES OCS).

What to do	How to do it	What should happen
- Start an actor performing the role "WatchServer" defined in the play "Watcher" with version "v1_1". - The command "Status" in the actor's DW indicates that the actor perform the "WatchServer" role.	<Node3>: ActorPlugIn Actor://<Node3>/pas1/WS1 WatchServer	- A new DW indicating an actor instance playing role "WatchServer" is running. - A "WatchServer" specific window used for presentation of subscription information shows up. - The play version identifier "v1_1" is shown in the watcher window header.
NOTE: Before initiating next command, observe the following from the existing watcher window header: 1. "WatchServer" as current role. 2. The time when the actor was started.		

What to do	How to do it	What should happen
Initiate "ActorChangeBehaviour" by sending the message "UpgradeServer" to the actor playing the "WatchServer" role. The new role becomes "WatchServerNew" (Can be verifier by executing the "Status" command in the actor's DW.	<pre><Node3>: RoleSessionAction Actor://<Node3>/pas1/WS1 UpgradeBehaviour</pre>	- The DW and watcher window for the existing Watcher actor disappears. - A new DW and watcher window appears. - In the new watcher window the following information indicates actor upgrade: a) Role changed to "WatchServerNew" b) Watcher start time is the same as before upgrade of behaviour. c) A counter indicates the number of upgrades performed. Demonstrates - Change of role for an existing actor - AI so keep state information after role change
Perform clean-up after demonstration by terminating the watcher actor, and to plug-out the "Watcher" play.	- <Node3>: Push "X" button in watcher window to terminate watcher actor. - <Node3>: <i>PlayPlugOut Watcher</i>	- DW and watcher window terminates. - PAS instance "pas1" terminates

Demo 4: Use of actor capabilities

What to do	How to do it	What should happen
Do all necessary preparation for a demonstration.	<See specification of "Prep 1" in the previous>	All involved nodes ready for demonstration execution.
Demonstrate use of actor capabilities	Actor capabilities have been used in the "School" play. When a SchoolClient request for a real time lecture server to connect to, it request for an actor playing the role "RTLServer", and which has the capability set that corresponds to a specific course and lecture. This can be verified through: a) Trace list of ActorPlugIn from "SchoolUserInterface" actor to "RTLServer" actor specifies capabilities as required. b) Using "Status" command at "RTLServer" actors list the current capabilities for the actors. Use "Demo 1" as a guideline for how to run "School" application.	