### Tutorial: Tools for mobile Linux (TmL) Exercises

#### Setting Up the Environment for the Hands-on Sessions

You can get all the software you need for the TmL tutorial from <u>http://wiki.eclipse.org/DSDP/TML/EclipseCon2009\_Tutorial</u>. Alternatively, you can copy it from the pen drives that are going to be passed around at the beginning of the tutorial at EclipseCon.

This is a check list of the components you need:

- Eclipse 3.5 Galileo Classic (with JDT and PDE) with the TmL plug-ins;
- □ The puppy Linux emulator with the echo server;
- □ A pre-configured workspace with the tutorial projects (optional).

Run Eclipse and select the pre-configured workspace.

# Hands-on Session #1: Using the Device Framework to integrate an emulator into Eclipse

In this session, we are going to use the Device Framework component from TmL to integrate the puppy Linux emulator into the Eclipse platform. This will allow you to start and stop the emulator from Eclipse, as well as to visualize the state of the emulator, i.e. whether it has been started or stopped.

In order to do this, we are going to create extensions using the following extension points:

- org.eclipse.tml.deviceTypes, to define a type for the emulator;
- **org.eclipse.tml.serviceDefinition**, to define operations or services for the emulator;
- **org.eclipse.tml.device.ui.newDeviceWizardPages**, to define a wizard to create instances of the emulator.

You will also learn how to use the following extension points:

- org.eclipse.tml.status, to define states for a state machine;
- **org.eclipse.tml.service**, to define operations or services to transition from a state to another state.
- 1. Open the project **org.eclipsecon.tml.session1**, or create a plug-in project (in this case you will need to copy the code yourself).



🖨 Java - Eclipse SDK	
File Edit Source Refactor Navigate Search Project Run Window Help	
New Alt+Shift+N V 🖄 Java Project	
Open File	
New Project	
Select a wizard Create a Plug-in Project	
<u>₩izards:</u> type filter text	E New Plug-in Project
Cyperind conce     Java Project     Java Project from Existing Ant Build     Pugain Project	Plug-in Content Enter the data required to generate the plug-in.
B ≧ General B ≧ CVS B ≧ Java B ≧ Plug-in Development	Plug-in Properties       Plug-in [D:       org.eclipsecon.tml.session1       Plug-in Version:       1.0.0       Plug-in Name:       Session1 Plug-in
Redk Mext >	Plug-in Provider:         Execution Environment:         12SE-1.5         Plug-in Options         © generate an activator, a Java class that controls the plug-in's life cycle
	Activator: org.eclipsecon.tml.session1.Activator  ✓ Tijs plug-in will make contributions to the UI  Enable API Analygis  Rich Clent Application  Would you like to create a rich client application? ○ Yes ④ Ng
	? <back next=""> Einish Cancel</back>

- 2. Switch to the **Plug-in Development** perspective.
- 3. Open the **META-INF/MANIFEST.MF** file.
- 4. Select the **Overview** tab, check the option **This plug-in is a singleton** and save the file.

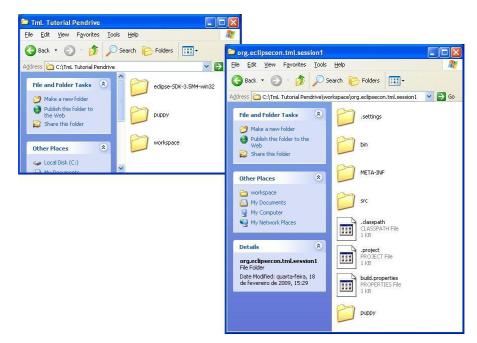
Overvi	ew					0 🌣 🎏 🤇
i <b>eneral Info</b> This section d	escribes general inform	nation about this	plug-in.	Plug-in Co	ntent t of the plug-in is made	
D:	org.eclipsecon.tml	.session1				up or two sections: iq-ins required on this pluq-in's
ersion:	1.0.0				ath to compile and run.	
lame:	Session1 Plug-in			Runtin Runtin		at make up this plug-in's
Provider:						
atform Filter				Extension	/ Extension Point Co	ontent
Activator:	org.eclipsecon.tml	carcion1 Active	Browse	This plug-in	may define extensions	and extension points:
ACCIVACOL.						tions this plug-in makes to the
Activate th	his nlug-in when one of	its classes is loar	fed	platfo	m.	
	nis plug-in when one of n is a singleton	its classes is load	ded	platfo Exten adds t		w function points this plug-in
This plug-ir xecution En				Exten adds t	sion Points: declares ne o the platform.	w function points this plug-in
This plug-ir xecution En	n is a singleton Invironments Inimum execution envir			Test this plue	<u>sion Points</u> : declares ne o the platform. ug-in by launching a sep <u>n an Eclipse application</u>	parate Eclipse application:
This plug-ir xecution En Specify the mi blug-in.	n is a singleton Invironments Inimum execution envir		d to run this	Test this plue	<u>sion Points</u> : declares ne o the platform. ug-in by launching a sep	parate Eclipse application:
This plug-ir xecution En Specify the mi blug-in.	n is a singleton Invironments Inimum execution envir		d to run this	Test this plue	<u>sion Points</u> : declares ne o the platform. ug-in by launching a sep <u>n an Eclipse application</u>	parate Eclipse application:
This plug-ir xecution En Specify the mi blug-in.	n is a singleton Invironments Inimum execution envir		d to run this Add Remove	Exten adds t Testing Test this plu Launc Launc Exporting	<u>sion Points</u> : declares ne o the platform. ug-in by launching a sep <u>n an Eclipse application</u>	arate Eclipse application: in Debug mode
This plug-ir xecution Er Specify the m plug-in. 32SE-1.5	n is a singleton Invironments Inimum execution envir		d to run this Add Remove	Testing Test this plu Launc Manuel Exporting To package	sion Points: declares ne o the platform. ug-in by launching a sep n an Eclipse application n an Eclipse application and export the plug-in	arate Eclipse application: in Debug mode
This plug-ir xecution Er Specify the m plug-in. 32SE-1.5 Configure JRE	n is a singleton <b>ivironments</b> inimum execution envir		d to run this Add Remove	Exten adds to Test this pli Launc Launc Launc Exporting To package 1. Organ 2. Exter	sion Points: declares ne o the platform. ug-in by launching a sep n an Eclipse application n an Eclipse application and export the plug-in	parate Eclipse application: In Debug mode : : : : : : :
This plug-ir xecution Er Specify the m plug-in. 32SE-1.5 Configure JRE	nis a singleton ivironments inimum execution envir associations		d to run this Add Remove	✓ Exten adds to Testing Test this pli ▲ Launc ★ Launc ★ Launc To package 1. Organ 2. Extern Extern 3. Specif	aion Points: declares ne othe platform. an Eclipse application n an Eclipse application n an Eclipse application and export the plug-in using the values the strings within alize the strings within	arate Eclipse application: in Debug mode : 9 <u>Organize Manifests Wizard</u> the plug-in using the , kaged in the deployable plug-

 Select the Dependencies tab and add the plug-ins org.eclipse.tml.common.utilities and org.eclipse.tml.framework.device to the list of dependencies.



Required Plug-ins Specify the list of plug- plug-in.	ins required for the operation of this Specif	rted Packages y packages on which this ying their originating plu
org.eclipse.ui	Add	ying choir ongineering pe
	🖨 Plug-in Selection	
	Select a Plug-in:	
	I org.eclipse.tm.tcf.rse (0.2.0) org.eclipse.tm.tcf.rse (0.2.0) org.eclipse.tml.common.utilities (0.3.0) org.eclipse.tml.device.qemu (0.2.0) org.eclipse.tml.device.qemureact (0.3.0) org.eclipse.tml.device.qemureact (0.3.0) org.eclipse.tml.tranework.device (0.3.0) org.eclipse.tml.framework.device.dvl.c.ul (0.3.0) org.eclipse.tml.framework.device.ul (0.3.0) org.eclipse.tml.framework.device.lipse.tml.framework.	
		ancel

6. Copy the entire **puppy** directory from where you copied it into your plug-in project directory.



7. Select the **Extensions** tab from **MANIFEST.MF** in the editor and add an extension of type **org.eclipse.tml.deviceTypes**.



All Extensions	lªz ⊡
Define extensions for this plug-in type filter text	in the following section.
	Extension Point Selection Create a new Device Types Definition extension.
	Extension Points Extension Wizards Extension Point filter:
verview Dependencies Runtime	

8. Select the extension and fill in the **ID** field with **puppyEmulator**.

Extensions		D 🗱 🎄 O
All Extensions Define extensions for this plug-in in the following section.  type filter text  General org.eclipse.tml.deviceTypes  General org.eclipsecon.tml.session1.deviceType1 (deviceType)	J <sup>a</sup> <sub>Z</sub> E Add Remove	Extension Details         Set the properties of the selected extension. Required fields are denoted by "*".         ID:       puppyEmulator         Name:         Show extension point description         Open extension point schema         % Find declaring extension point

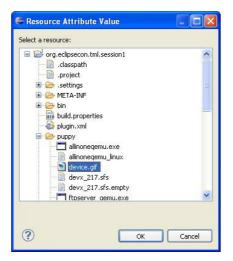
9. Select the **deviceType** element that was automatically created under the extension and fill in the detail fields as follows:



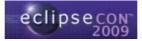
Extensions			(1) (1) (1) (1) (1) (1) (1) (1) (1) (1)
Il Extensions Define extensions for this plug-in in the following section.	↓ª ⊡		Element Details perties of "deviceType". Required fields are denoted by "*".
type filter text		id*:	org.eclipsecon.tml.session1.puppyEmulator
org.eclipse.tml.deviceTypes	Add	name*:	Puppy Emulator
org.eclipse.tml.deviceTypes     org.eclipsecon.tml.session1.deviceType1 (deviceType)	Remove	label*:	Puppy Emulator
	Kellove	handler*:	Browse.
	Up	description: isAbstract:	false
	Down	superClass:	
		icon:	Browse.

*Note:* the **id** field must contain the plug-in id followed by the **ID** from the previous step, e.g. **org.eclipsecon.tml.session1.puppyEmulator**.

10. Click the **Browse...** button next to the **icon** field and select **puppy/device.gif** from the plug-in project directory. This icon is going to represent the puppy Linux emulator in the GUI.



11. Click the handler label to open the New Java Class dialog. The dialog shows that the handler class must implement the interface org.eclipse.tml.framework.device.model.handler.IDeviceHandler. Fill in the Package field with org.eclipsecon.tml.session1 and the Name field with PuppyHandler. Alternatively, you can click Browse... and select the pre-defined class provided with the project in the pre-configured workspace.



Package: [ Enclosing type: ] Name: [	org.eclipsecon.tml.session1/src org.eclipsecon.tml.session1	otected	Browse Browse Browse
Package: [ Enclosing type: ] Name: [	PuppyHandler  Dgublic Odefault Oprivate Opri	otected	Bro <u>w</u> se
Enclosing type:	PuppyHandler  Deublic Odefault Oprivate Opri	otected	
Name:	public Odefault Oprivate Opri	otected	Bro <u>w</u> se
	public Odefault Oprivate Opri	otected	
Modifiers: 🤇		otected	
Superclass:	iava.lang.Object		Browse
Interfaces:	🛈 org.eclipse.tml.framework.device.model.handl	er.ID	<u>A</u> dd
			<u>R</u> emove
Which method stubs w	ould you like to create?		
C	] public static void main(String[] args)		
5	<u> <u> <u> </u> <u> </u></u></u>		
	In <u>h</u> erited abstract methods		
Do you want to add co	mments? (Configure templates and default value	<u>here</u> )	
Ľ	<u>G</u> enerate comments		

12. Click **Finish** to create the class and open it automatically in the editor. The new class contains three auto-generated stubs: the no-arg constructor, the **createDeviceInstance()** method and the **createDeviceLauncher()** method.

org.eclipsecon.tml.session1	🖸 PuppyHandler.java 🛛	-
package org.eclip	secon.tml.session1;	^
import org.eclips	e.tml.framework.device.model.IDeviceLauncher;[]	
<b>publi</b> c class Pupp	yHandler implements IDeviceHandler (	
⊖ public PuppyH // TODO ≩ }	andler() { uto-generated constructor stub	
	nce createDeviceInstance(String argO) { uto-generated method stub 11;	
	eLauncher createDeviceLauncher(IInstance arg0) ( uto-generated method stub 11;	_
3		>

13. Replace the auto-generated method stub in the **createDeviceInstance()** method with the following line of code:

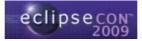


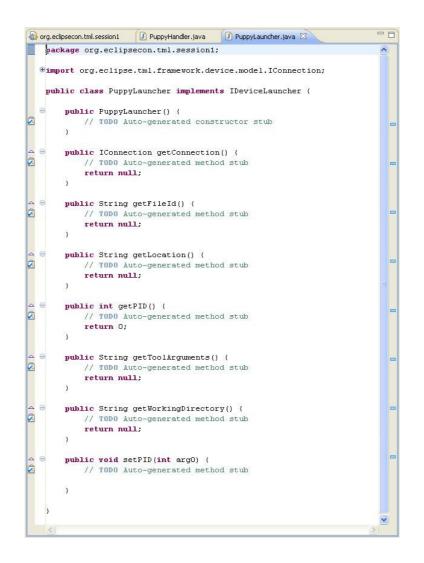


14. The createDeviceLauncher() method is going to create an instance of the launcher class, which is responsible for invoking the emulator executable in this case. To create the launcher class, create a new Java class called PuppyLauncher that implements org.eclipse.tml.framework.device.IDeviceLauncher. Alternatively, you can select the pre-defined class provided with the project in the pre-configured workspace.

<b>Java Class</b> Create a new Java	class.	0
Source folder:	org.eclipsecon.tml.session1/src	Browse
Pac <u>k</u> age:	org.eclipsecon.tml.session1	Browse
Enclosing type:	org.eclipsecon.tml.session1.PuppyHandler	Bro <u>w</u> se
Na <u>m</u> e:	PuppyLauncher	
Modifiers:	O public         O default         O private         O protected           abstract         final         static	
Superclass:	java.lang.Object	Browse
Interfaces:	O org.eclipse.tml.framework.device.model.IDeviceLauncher	Add
Which method stubs	s would you like to create?	
	public static void main(String[] args)	
	Constructors from superclass	
	Inherited abstract methods	
Do you want to add	comments? (Configure templates and default value <u>here</u> ) — Generate comments	
1	Finish	Cancel

15. Click **Finish** to create the class and open it automatically in the editor. The class contains a number of auto-generated stubs.





16. First, add the following code to handle the process id (pid) of the emulator instance:

-	org.ed	lipsecon.tml.session1	🚺 PuppyHandler.java	🖸 PuppyLauncher.java 🛛	- 0
	•imj	port org.eclipse alic class Puppy	/Launcher <b>implemen</b>	; vice.model.IConnection;[] ts IDeviceLauncher (	~
	Θ	<pre>public int pic public PuppyLa pid = 0; }</pre>			
	Ð	<b>public</b> IConnec	tion getConnectio	n() (	
~	Ð	public String	getFileId() { .		
	Ð	public String	getLocation() {		
	⊜	<pre>public int get     return pic }</pre>			
	Ð	public String	getToolArguments(	) (.	
	٠	<b>publi</b> c String	getWorkingDirecto	ry() (	
4	•	public void se this.pid = )	etPID(int pid) { = pid;		
	<				2



17. Now, add the following code to handle the connection along with the corresponding **import** statements:

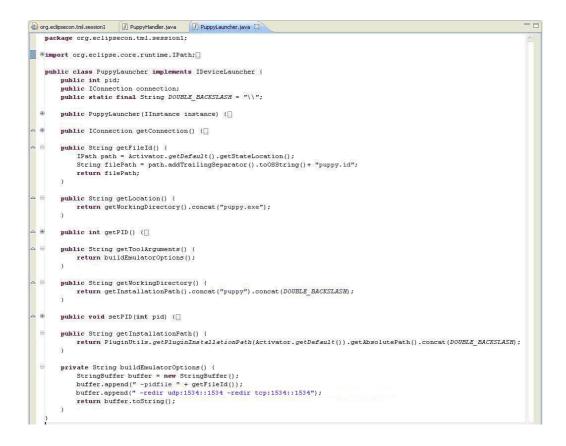


18. In order to fix the **org.eclipse.tml.service.start.launcher.DefaultConnection** import, select the **Dependencies** tab of **MANIFEST.MF** and add the plug-in **org.eclipse.tml.service.start** to the list of dependencies.

equired Plug-ins	Ja ≥	Imported Packages Specify packages on which this plug-in dep
apeorg.eclipse.ui	Add	identifying their originating plug-in.
org.eclipse.core.runtime		
prog.eclipse.tml.common.utilities (0.3.0)	Plug-in Selection	
org.eclipse.tml.framework.device (0.3.0)	e r tug in selection	
	Select a Plug-in:	
	🐌 org.eclipse.tml.fran	mework.device.wizard (0.3.0) 🛛 🔼
	🐌 org.eclipse.tml.linu:	
	🐌 org.eclipse.tml.linu:	
	org.eclipse.tml.proi	
	org.eclipse.tml.serv	
	org.eclipse.tml.serv	
		vice.vncviewer (0.3.0)
	🐌 org.eclipse.tml.vnc	
		viewer.vncviews (0.5.0)
		vser (3.2.300.v20081201) 🔤
		vser.source (3.2.300.v20081201)
		tsheets (3.3.200.v20081205)

19. Finally, we must enable the launcher class to invoke the emulator executable by adding the code below, which sets the path to the emulator executable as well as the options with which to invoke it.

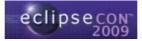


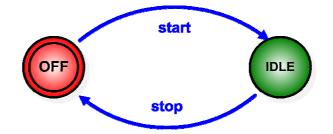


20. Now that the launcher class is complete, add the following line of code to the createDeviceLauncher() method in class PuppyHandler along with the corresponding import statement:

🚯 org.	eclipsecon.tml.session1 🛛 PuppyHandler.java 🛛 🚺 PuppyLauncher.java	
pa	ackage org.eclipsecon.tml.session1;	<u>_</u>
ינ ע ע ע ע	<pre>mport org.eclipse.tml.framework.device.internal.model.MobileInstance; mport org.eclipse.tml.framework.device.model.IDeviceLauncher; mport org.eclipse.tml.framework.device.model.IInstance; mport org.eclipse.tml.framework.device.model.handler.IDeviceHandler; mport org.eclipse.tml.sessionl.PuppyLauncher; mblic class PuppyHandler implements IDeviceHandler ( public PuppyHandler() {</pre>	
<b>△ ⊖</b>	<pre>) public IInstance createDeviceInstance(String id) (     return new MobileInstance(id); )</pre>	
≏ ⊜ } 	<pre>public IDeviceLauncher createDeviceLauncher(IInstance instance) {     return new PuppyLauncher(instance); }</pre>	
<	ai de la companya de	2

21. In the next few steps we are going to implement the state machine for this emulator, which is defined as follows:





In this case we are going to use the **OFF** and **IDLE** states which have been predefined in the framework, together with the **start** and **stop** operations provided with the reference implementation in plug-ins **org.eclipse.tml.service.start** and **org.eclipse.tml.service.stop**.

What if we did not have pre-defined states, or if those states were not suitable for our implementation? In this case, we would use the extension point **org.eclipse.tml.status** to define each state identifier:

LExtensions	Extension Elemen Set the properties of "*" id*: name*: image: canDeleteInstance: canEditProperties:	t Details f "status". Required fie MYSTATE My State true true	elds are denoted by
-------------	--	--	---------------------

And what if we did not have suitable operations in the reference implementation? We would need to use the extension point **org.eclipse.tml.service** to define them.

22. Select the **Dependencies** tab from **MANIFEST.MF** and add the plug-in **org.eclipse.tml.service.stop** to the list of dependencies (remember that we added **org.eclipse.tml.service.start** when defining the launcher class).

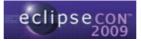


and plan to a		Y	
equired Plug-ins	Jªz	Imported Packages	
becify the list of plug-ins required for the oper ug-in.	ation of this	Specify packages on which this plug- identifying their originating plug-in.	in depends without explicitl
porg.eclipse.ui	Add	Plug-in Selection	
> org.eclipse.core.runtime > org.eclipse.tml.common.utilities (0.3.0)	Remove	Select a Plug-in:	
p⊷org.eclipse.tml.framework.device (0.3.0) p⊷org.eclipse.tml.service.start (0.3.0)	Up Down Properties	<ul> <li>org.eclipse.tml.linuxtools (0.1.0)</li> <li>org.eclipse.tml.linuxtools (0.1.1)</li> <li>org.eclipse.tml.inuxtools (0.2.0)</li> <li>org.eclipse.tml.service.vncviewer (0.5.0)</li> <li>org.eclipse.tml.vncviewer (0.5.0)</li> <li>org.eclipse.tml.vncviewer (0.5.0)</li> <li>org.eclipse.tml.vncviewer (3.2.300.v2)</li> <li>org.eclipse.ul.nowser (3.2.300.v2)</li> </ul>	0.3.0) 5(0.5.0) 0081201) 300.v20081201) 0.v20081205) 3.3.200.v2008120( 31208)
	Total: 5		

23. Select the **Extensions** tab from **MANIFEST.MF** and add an extension of type org.eclipse.tml.serviceDefinition. Select the extension (not the service element) and fill in the **ID** field with the device type ID concatenated to the plugin id, e.g. org.eclipsecon.tml.session1.puppyEmulator. Note: the ID must correspond to the plug-in id + the device type ID.

Extensions       Image: Construct of the selected extension.         All Extensions       Jage: Construct of the selected extension.         Define extensions for this plug-in in the following section.       Set the properties of the selected extension. Required field are denoted by "*".         type filter text       ID: org.eclipse.tml.deviceType)         Image: Construction org.eclipse.tml.deviceType)       Add         Image: Construction org.eclipse.tml.serviceDefinition       Remove         Image: Construction org.eclipse.tml.serviceDefinition       Show extension point description         Image: Construction org.eclipse.tml.serviceDefinition       Show extension point description         Image: Construction org.eclipse.tml.serviceDefinition       Image: Construction org.eclipse.tml.serviceDefinition         Image: Construction org.eclipse.tml.serviceDefinition       Image: Constructio	org.eclipsecon.tml.session1 🖾 🗾 PuppyHandler.java	D PuppyLauncher.java
Image: Set the properties of the selected extension. Required field are denoted by "*".         Set the properties of the selected extension. Required field are denoted by "*".         Image: Set the properties of the selected extension. Required field are denoted by "*".         Image: Set the properties of the selected extension. Required field are denoted by "*".         Image: Set the properties of the selected extension. Required field are denoted by "*".         Image: Set the properties of the selected extension. Required field are denoted by "*".         Image: Set the properties of the selected extension. Required field are denoted by "*".         Image: Set the properties of the selected extension. Required field are denoted by "*".         Image: Set the properties of the selected extension. Required field are denoted by "*".         Image: Set the properties of the selected extension. Required field are denoted by "*".         Image: Set the properties of the selected extension. Required field are denoted by "*".         Image: Set the properties of the selected extension. Required field are denoted by "*".         Image: Set the properties of the selected extension. Required field are denoted by "*".         Image: Set the properties of the selected extension. Required field are denoted by "*".         Image: Set the properties of the selected extension. Required field are denoted by "*".         Image: Set the properties of the selected extension. Required field are denoted by "*".         Image: Set the properties of the selected extension. Required field ar	Extensions	2) 🎄 🌾 🚺
	type filter text	Set the properties of the selected extension. Required fields are denoted by "*".  ID: org.eclipsecon.tml.session1.puppyEmulator Name:  Show extension point description Open extension point schema
	<	

24. Select the **service** element from the extension you have just created and fill in the **id** field with the id of the start service from the reference implementation, i.e. **org.eclipse.tml.service.start.startService**. We don't need to define a handler class here, since the service is already defined in the reference implementation.

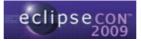


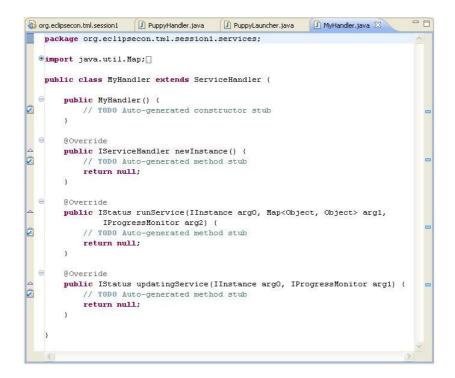
					莽 🎏 🤅
I <mark>l Extensions</mark> Jefine extensions for this plug-in in the followir	↓ <mark>a</mark>	20 2020	n Element Details ted element has no pro	operties to set.	
type filter text		id:	org.eclipse.tml.servi	ce.start.startSer	vice
org.eclipse.tml.deviceTypes     Section 2	Add Remove	<u>handler:</u>			Browse
	Up Down				

What if the start service were not defined in the reference implementation? In this case, we would need to create a handler class that extends **org.eclipse.tml.framework.device.model.handler.ServiceHandler**:

🖶 New Java Cla	35	
Java Class		
Create a new Java	class.	
Source fol <u>d</u> er:	org.eclipsecon.tml.session1/src	Browse
Pac <u>k</u> age:	org.eclipsecon.tml.session1.services	Bro <u>w</u> se
Enclosing type:		Browse
Na <u>m</u> e:	MyHandler	
Modifiers:	public Odefault Oprivate Oprotected     abstract I final I static	
Superclass:	org.eclipse.tml.framework.device.model.handler.ServiceHandler	Browse
Interfaces:		<u>A</u> dd
		Remove
Which method stubs	would you like to create?	
	public static void main(String[] args)	
	Constructors from superclass	
Do you want to add	✓ Inherited abstract methods comments? (Configure templates and default value here)	
Do you waik to add	Generate comments	
?	Einish	Cancel

The service handler class can override the methods below, where **runService()** is the method that performs the actual operations.

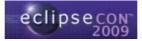




25. Select the **service** element from the **serviceDefinition** extension and add an element of type **status**:

All Extensions		↓ª <sub>z</sub> ⊟	Extension Elec
Define extensions for this	plug-in in the follow	ing section.	Set the propert
type filter text			id: org.
🖃 🐲 org.eclipse.tml.e	deviceTypes	Add	handler:
Puppy Emula		Remove	
🖻 🖙 org.eclipse.tml.s			
X org.edinse.	pol corvico chart chi	artSal	
[X] org.eclipse.	t <u>pl cervice start sta</u> New		X status
·····X org.eclipse.			X status
ing,eclipse.۱ کا	New		X status
<b>X</b> org.eclipse.	New Delete	<u> </u>	X status
uii arg,eclipse.	New Delete	Ctrl+X	X) status
∟ X org.edipse.	Delete	Ctrl+X Ctrl+C	X status
L. [X] org.edipse.	Delete → Cut Copy Paste	Ctrl+X Ctrl+C	X status

26. Fill in the **startId**, **endId** and **haltId** fields with the values **OFF**, **IDLE** and **OFF**, respectively. They correspond to the start state, end state and error state for the start operation, where the error state is the state of the emulator if the operation fails.



II Extensions	Jaz B	Extensio	n Element Details	
efine extensions for this plug-in in the follow			roperties of "status". Req	uired fields are denoted by
type filter text	L	startId*;	þff	
🖃 🚥 org.eclipse.tml.deviceTypes	Add	endId*:	IDLE	
Puppy Emulator (deviceType)	Remove	haltId*;	OFF	
Greek org.eclipse.tml.service.start.st	Up			
	Down			

27. The previous steps complete the definition of the start service, which transitions from the OFF state to the IDLE state. To define the stop service, create another extension of type **org.eclipse.tml.serviceDefinition** with the same emulator ID:

Extensions	0 🌣 ≉ 🛈
All Extensions Define extensions for this plug-in in the following section. Type filter text	Extension Details         Set the properties of the selected extension. Required fields are denoted by "*".         ID:       org.eclipsecon.tml.session1.puppyEmulator         Name:         Show extension point description         Image:         Show extension point description         Open extension point schema         Find declaring extension point

28. Select the **service** element and fill in the **id** field with the id of the stop service, i.e. **org.eclipse.tml.service.stop.stopService**.

Extensions	● 莽 (2)
Extensions     ↓ª₂ □       efine extensions for this plug-in in the following section.     type filter text       ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓	Extension Element Details         The selected element has no properties to set.         id:       _brg.eclipse.tml.service.stop.stopService         handler:



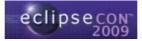
29. Add an element of type **status** to the **service** element and fill in the **startId**, **endId** and **haltId** fields with the values **IDLE**, **OFF** and **IDLE**, respectively.

Extensions			◎ 莽 🎏 (
Il Extensions $\downarrow^a_Z$		n Element Details roperties of "status". R	equired fields are denoted by
type filter text	startId*:	IDLE	
e	endId*;	OFF	
Puppy Emulator (deviceType)     Remove       Government     Gradingeneration       K     (status)       Government     Gradingeneration       K     (status)       Government     Gradingeneration       K     (status)	haltId*:	IDLE	
< >			

30. This completes the definition of the stop operation. The only thing we need to do now in order to be able to run an instance of the puppy Linux emulator is to define the wizard to create an instance of this device type. Select the **Dependencies** tab from **MANIFEST.MF** and add the plug-in **org.eclipse.tml.framework.device.ui** to the list of dependencies.

equired Plug-ins becify the list of plug-ins required for the operation of t	ja e Plug-in Selecti	on 🔳 🗖	
<ul> <li>org.eclipse.ui</li> <li>org.eclipse.tml.common.utilities (0.3.0)</li> <li>org.eclipse.tml.framework.device (0.3.0)</li> <li>org.eclipse.tml.service.start (0.3.0)</li> <li>org.eclipse.tml.service.stop (0.3.0)</li> </ul>	Add     Select a Plug-in:       Remove     The org.eclipse.tml.f       Up     The org.eclipse.tml.f       Down     The org.eclipse.tml.f       Properties     The org.eclipse.tml.f       The org.eclipse.tml.f     The org.eclipse.tml.f       Down     The org.eclipse.tml.f       The org.eclipse.tml.f     The org.eclipse.tml.f       The org.eclipse.tml.f     The org.eclipse.tml.f       The org.eclipse.tml.f     The org.eclipse.tml.f       The org.eclipse.tml.f     The org.eclipse.tml.f	ramework.device.sdk (0.3.0) ramework.device.ui (0.3.0) ramework.device.wizard (0.3.0) nuxtools (0.1.0) nuxtools.base (0.1.0) rotocol (0.2.0) ervice.vncviewer (0.3.0)	······································
Automated Management of Dependencies	من م	owser (3.2.300.v20081201) owser.source (3.2.300.v20081201) ostsboots (3.3.300.v20081201)	~

31. Select the **Extensions** tab from **MANIFEST.MF** and add an extension of type **org.eclipse.tml.device.ui.newDeviceWizardPages**.



*org.eclipsecon.tml.session1 🛛 🗍 PuppyHandler.java	PuppyLauncher.jav	a ) — E
Extensions		0 🎄 🌾 🛈
II Extensions Define extensions for this plug-in in the following section. type filter text	↓Z ⊟	Received and the selected extension. Required fields are denoted by "*".
org.eclipse.tml.deviceTypes     org.eclipse.tml.deviceType)     org.eclipse.tml.serviceDefinition     R org.eclipse.tml.serviceLastric.startService (service)     R (status)     org.eclipse.tml.serviceDefinition     R (status)     org.eclipse.tml.service.stop.stopService (service)     R (status)	Remove Up	ame: Show extension point description Open extension point schema Find declaring extension point

32. Select the **wizardPage** element from the extension you have just created. Again, we are going to use a default wizard page provided with the framework. The default wizard page contains three fields: host, port and display. To use the default page, click **Browse...** next to the **pageClass** field and select **org.eclipse.tml.framework.device.ui.DefaultConnectionInfoWizardPage** from the list.

*org.eclipsecon.tml.session1 🛛 🚺 PuppyHandler.java	PuppyLauncher.j	r.java 🗌 🗖 🗖	
Extensions			
Il Extensions efine extensions for this plug-in in the following section.	↓ <sup>a</sup> ₂ ⊡	Extension Element Details Set the properties of "wizardPage". Required fields are denoted by "**". id*: org.eclipsecon.tml.session1.puppvWizardPage	
type filter text	Add Remove	DiscretionClass:     Organization Class:	
	Down	Select Type	
❀ ☑ org.eclipsecon.tml.session1.wizardPage1 (wizardPag	e)	Matching items:	ard
erview Dependencies Runtime Extensions Extension Points B	uild MANIFEST.MF p	bin.org.eclipse.tml.framework.device.ui.wizard\org.eclipse.tml.framework.device.ui_	_0.3.0.
		?	Cancel

33. Select the **wizardPage** element and add an element of type **deviceType** to it. Fill in the **deviceTypeld** field with the same id you used in the service definition extensions, i.e. the plug-in id followed by the device type id, e.g. **org.eclipsecon.tml.session1.puppyEmulator**.



org.eclipsecon.tml.session1 🛛 🕖 PuppyHandler.java	PuppyLauncher.java	•
Extensions		0 🎄 🕸 🔾
II Extensions Define extensions for this plug-in in the following section. Type filter text	Jªz ⊟	Extension Element Details Set the properties of "deviceType". Required fields are denoted by "**". deviceTypeId*: bra.ecliosecon.tml.session1.puppYEmulator
org.eclipse.tml.deviceTypes     org.eclipse.tml.deviceType)     org.eclipse.tml.serviceDefinition     X org.eclipse.tml.serviceDefinition		
<	>	

34. This completes the code required to integrate the puppy Linux emulator to Eclipse using the Device Framework. Make sure you saved everything and run the plug-in as an Eclipse application.

) 🔁 org.eclipsecon.t i 🔁 🛋 JRE System i 🖻 🛋 Plug-in Depa	New	tensions	↓ ↓ªz
Generation      Generation	Show In	=4 Alt+Shift+W	
plugin.xml	Copy Copy Copy Copy Copy Copy Copy Copy	Itrl+C Itrl+V Delete Alt+Shift+S	nl.deviceTypes     Add       ulator (deviceType)     Rem       nl.serviceDefinition     e.tml.service.start.startService (service)       us)     nl.serviceDefinition       us)     e.tml.service.stop.stopService (service)       us)     us)
	Alt+Shift+T	<ul> <li>Inl.device.ui.newDeviceWizardPages</li> <li>econ.tml.session1.puppyWizardPage (wizardPage)</li> <li>clipsecon.tml.session1.puppyEmulator (deviceType)</li> </ul>	
	Refresh Close Project Assign Working Sets	-5	w
	Run As		General Application Alt+Shift+X, E MANIFEST.f
	Debug As Team Compare With Restore from Local History		Image: Solution of the soluti
	PDE Tools		Run Configurations

35. Select Window > Open Perspective > Other... and choose TmL perspective from the list. The TmL perspective is going to show you the Instance Management view, which lists the available device types along with their instances.

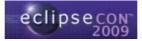


➡ TmL perspective - Eclipse SDK File Edit Navigate Search Project Run Wi	odow Holo		
i 📑 🔹 🔜 👜 i start client start server			😰 🖪 TmL perspective ,
Ra-Navigator 😫 📃 🗖	Ta Instance Management		🛍 • 🗸 🗆 🗖
0 0 0 0 0 5	Instance name	Status	
	🖃 🥎 Generic QEMU device		
	<none></none>		
	🖃 🥎 Puppy Emulator		
	<none></none>		
	🖃 🧐 QEmuARM Mobile Device		
	<none></none>		
	🖃 🧐 QEmuReact Mobile Device		
	<none></none>		
	Console		🛃 🖳 - 📑 🕶 🗖
	No consoles to display at this time.		
	No consoles to display at this time.		
i 0*			

36. Right-click on the Puppy Emulator device type and select **New...** In the first wizard page, type a name for the emulator instance and click **Next**.

¢			- 🗆 🛛
	ew Device Instance e name and select a device		
Name: Device types:	Puppy Emulator	bnbbÅ	
?	< <u>B</u> ack	Next > Einish	Cancel

37. In the second wizard page, fill in the **Host**, **Port** and **Display** fields and click **Finish**.

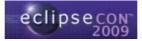


¢	
and second second	Connection Information the information required for device connection
Host:	127.0.0.1
Port:	5900
Display:	:0.0
?	<pre></pre>

38. The emulator instance has now been created and the corresponding services are shown, as well as the initial state. Click **Start Instance** to start the emulator instance.

Finl. perspective - Eclipse SDK			
<u> E</u> ile <u>E</u> dit <u>N</u> avigate Se <u>a</u> rch <u>P</u> roject <u>R</u> un <u>W</u> ind	low <u>H</u> elp		
🗂 • 🖃 🖆   💁 •   🛷 •   🖉 • 🤅			TmL perspective
🔁 Navigator 🛛 🗖 🗖	🖪 Instance Management 🛛		🛍 • 🏱 🗖 E
		Status	
	Services           Start Instan           Image: Stop Instanc		
	Console CPU Load Memory Start   End   Region name	y Map 🛛	ලි 23 😭 🗖
: <b>0</b> *	15		1

39. The emulator instance is started and the state is updated in the GUI. You can click **Stop Instance** to stop the instance.



Finl perspective - Eclipse SDK			
<u>File Edit N</u> avigate Se <u>a</u> rch Project <u>R</u> un <u>W</u> inde	ow <u>H</u> elp		
📬 • 🔛 🗁   💁 •   🛷 •   🖉 - 🖗	]-*> +++		TmL perspective
😵 Navigator 🕱 📃 🗖	🕅 Instance Management 🛛		🛍 • 🍸 🗖 🗖
	Instance name	Status	
÷ ↔ @   <b>E</b> & ▼	Instance name Generic QEMU device Cone> Qenov Come> QemuARM Mobile Device Cone> QemuReact Mobile Device Cone> Services Services Services Concole CPU Load Memor Start End Region name		▼ \\$ \$ □ □ \$ \$
1. 🕈			

40. This concludes the first hands-on session, where you integrated a puppy Linux emulator to Eclipse by defining the device type and the state machine and using states and operations provided by the framework and reference implementation. In the next session, you are going to use the VNC Viewer to enable visualization of the emulator display in an Eclipse view.

# Hands-on Session #2: Using the VNC Viewer to visualize the emulator display in Eclipse

In this session, we are going to take the puppy Linux emulator that we integrated into Eclipse in the previous session and use the VNC Viewer component from TmL to visualize the emulator display in an Eclipse view. A state will be added to the state machine to indicate that the emulator is connected to the VNC Viewer, and an operation will support the transition from the IDLE state to the new state.

In order to do this, we are going to create an extension using the extension point **org.eclipse.tml.serviceDefinition** to indicate the new operation along with the corresponding state transitions. Both the state and the connect operation are included in the **org.eclipse.tml.vncviewer** plug-in, so it is not necessary to define them.

1. Open the project from the previous session, or use the pre-defined project **org.eclipsecon.tml.session2.** This exercise assumes that you have the completed

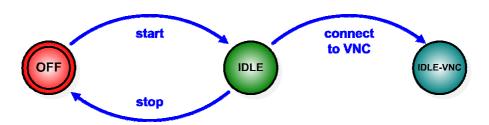


project from the previous session as a starting point, and the pre-defined project is provided in case you haven't completed it.

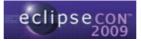
- 2. Switch to the **Plug-in Development** perspective.
- 3. Open the **META-INF/MANIFEST.MF** file.
- 4. Select the **Dependencies** tab from **MANIFEST.MF** and add the plug-in **org.eclipse.tml.service.vncviewer** to the list of dependencies.

quired Plug-ins	Jªz	Imported Packages
ecify the list of plug-ins required for the operation		🗧 Plug-in Selection
p≈org.eclipse.ui p≈org.eclipse.core.runtime p≈org.eclipse.tml.common.utilities (0.3.0)	Add	Select a Plug-in: dd nove
<ul> <li>org.eclipse.tml.framework.device (0.3.0)</li> <li>org.eclipse.tml.service.start (0.3.0)</li> <li>org.eclipse.tml.service.stop (0.3.0)</li> <li>org.eclipse.tml.framework.device.ui (0.3.0)</li> </ul>	Up Down	Image: Solution of the second seco
	Properties	<ul> <li>org.eclipse.tml.linuxtools.base (0.1.0)</li> <li>org.eclipse.tml.protocol (0.2.0)</li> <li>org.eclipse.tml.service.vncviewer (0.3.0)</li> </ul>
	Total: 7	Corg.eclipse.tml.vncviewer (0.5.0)     Corg.eclipse.tml.vncviewer (0.5.0)     Corg.eclipse.ui.browser (3.2.300.v20081201)     Corg.eclipse.ui.browser source (3.2.300.v20081201)     Corg.eclipse.ui.browser source (3.2.300.v20081201)
Automated Management of Dependencies	↓ªz	org.eclipse.ui.cheatsheets (3.3.200,v20081205)     org.eclipse.ui.cheatsheets (3.3.200,v20081205)     org.eclipse.ui.cheatsheets.source (3.3.200,v20081208)

5. Now, we are going to modify the state machine by adding the **IDLE-VNC** state and operations that connect the VNC client in the VNC Viewer to the VNC server in the emulator and disconnect the VNC client from the VNC Viewer. These operations correspond to the transitions from the **IDLE** state to the **IDLE-VNC** state:



The **IDLE-VNC** state and the operations that we are going to use to connect and disconnect VNC client and server are included in the **org.eclipse.tml.vncviewer** plug-in, which was added as a dependency, so it is not necessary to define either.



To add the new state and the connect transition to the state machine, select the **Extensions** tab from **MANIFEST.MF** and add an extension of type **org.eclipse.tml.serviceDefinition**.

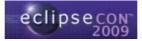
6. Fill in the **ID** field from the **serviceDefinition** extension with the plug-in id followed by the emulator id, e.g. **org.eclipsecon.tml.session2.puppyEmulator**.

Extensions		🖸 🎄 🔇
All Extensions Define extensions for this plug-in in the following section. type filter text	↓ <sup>4</sup> <sub>Z</sub> ⊨ Add Remove Up Down	Extension Details         Set the properties of the selected extension. Required fields are denoted by "*".         ID:       org.eclipsecon.tml.session2.puppyEmulator         Name:         Show extension point description         Open extension point schema         Pind declaring extension point

7. Select the **service** element from the extension you have just created and fill in the id field with **org.eclipse.tml.service.vncviewer.vncViewerService**, which identifies the VNC connection operation from the VNC Viewer plug-in. Again, it is not necessary to create a handler class, since it is already defined in the VNC Viewer plug-in.

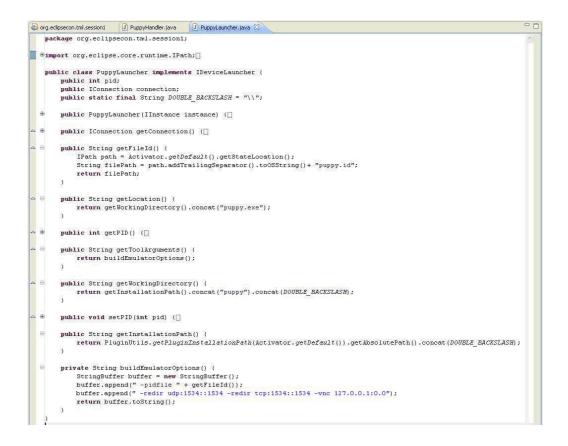
🔂 *org.eclipsecon.tml.session2 🛛						- 6
😓 Extensions						> 莽 ≉ 💿
All Extensions Define extensions for this plug-in in the following section.	↓ª ⊟			n <b>ent Details</b> ment has no properti	es to set.	
type filter text		id:	org.e	eclipse.tml.service.vn	cviewer.vncViewerS	ervice
Correctly and the service of th	Add Remove	handle	<u>a</u>			Browse
Overview Dependencies Runtime Extensions Extension Poir	its Build MAN	IFEST.MF p	lugin.xml	build.properties		

8. Right-click on the **service** element and add an element of type **status**. Fill in the **startId**, **endId** and **haltId** fields with **IDLE**, **IDLE-VNC** and **IDLE**, respectively.



Extensions			O 🌣 ≉ O
I Extensions efine extensions for this plug-in in the following section.	↓ <sup>a</sup> z ⊡	ana and	n Element Details roperties of "status". Required fields are denoted by "*".
type filter text		startId*:	IDLE
	Add	endId*:	IDLE-VNC
org.eclipse.tml.serviceDefinition     org.eclipse.tml.serviceDefinition     org.eclipse.tml.serviceDefinition     org.eclipse.tml.serviceDefinition     org.eclipse.tml.vncviewer.vncViewerService (s	Remove Up Down	haltId*:	IDLE
< >			

- Follow the same steps above to define an operation to disconnect the VNC client from the emulator, using the service org.eclipse.tml.service.vncviewer.unplugVncViewerService and the states IDLE-VNC, IDLE and IDLE-VNC as startId, endId and haltId, respectively.
- 10. Open the launcher class and add an option for the VNC server in the emulator as follows.



- 11. Make sure you saved everything and run the plug-in as an Eclipse application.
- 12. Create the emulator instance just as you did in the previous exercise.



13. Click **Start Instance** to start the emulator instance.

TmL perspective - Eclipse SDK			
ijle Edit Navigate Se <u>a</u> rch <u>P</u> roject <u>R</u> un <u>W</u> ind	(1) (1) (1) (1) (1) (1) (1) (1) (1) (1)		
📫 • 🔛 🖄   💁 •   🖋 •   🖉 •			
🖫 Navigator 🛛 🗖 🗖	🕼 Instance Management 🛛		🕅 • 🏹 🗖 🗖
(→ → @   🖻 🔩 🎽		Status	
	Puppy Emulator     puppy     QEmuARM Mobile Device	* OFF	
	<ul> <li>         QEmuReact Mobile Device <none> </none></li> </ul>		
	Services Start Instan Stop Instanc Connect VNC		
	Disconnect V		
	[INFO][Status: id=INAC] [INFO][Status: id=OFF;: [INFO][Status: id=IDLE] [INFO][Status: id=IDLE] [DEBUG]Disposing SWT Re	name=OFF] -VNC;name=IDLE-VNC] ;name=IDLE]	
	<		2

14. Click **Connect VNC** to connect the VNC Viewer to the emulator instance.

Finl. perspective - Eclipse SDK			
<u> Eile Edit Navigate Search Project Run Wind</u>	ow <u>H</u> elp		
। 🗈 • 🗟 🖄 • । 🛷 • । 🖉 - 🖗	] - 🍫 🗇 - 🗇 -		🖺 🖪 TmL perspective ), 🎽
Ra-Navigator 🛛 🗖 🗖	🔝 Instance Management 🛛		🛍 • 🗸 🗆 🗖
승 수 👰 📄 🔩 🏹	Instance name	Status	
	<ul> <li>Puppy Emulator</li> </ul>		
	₽ puppy	" IDLE	
	<ul> <li>QEmuARM Mobile Device</li> </ul>		
	<none></none>		
	<ul> <li>Section 2 Construction 2 Constructina Construction 2 Construction 2 Construction 2 Construction 2</li></ul>		
	<none></none>		
	Services		
	11		
	Start Instan		
	Stop Instanc		
	Connect VNC		
	Disconnect V		
	Console X		■ × 💥 🔩 🔐 🖉 🖉 • 😁 • 🗖 •
	-		
	puppy [Emulator Instance] C:\Docum	ents and Settings\frp	743\Desktop\TmLTutorial_FR\workspace\org.eclipsecon.tml.session2\puppy\puppy.exe
			2
	<		>
: D*			

15. You should now see the VNC Viewer view with the emulator display:



🖨 TmL perspective - Eclipse SDK				
<u>Eile E</u> dit <u>N</u> avigate Se <u>a</u> rch <u>P</u> roject <u>R</u> un <u>W</u> ind	ow <u>H</u> elp			
। 📬 • 🔛 👜 । 💁 • । 🛷 • । 🖄 • 🤅	] - 🍫 🔶 - 🔶 -			😭 🔲 TmL perspective , 🎽
	🔝 Instance Management 🛛		<b>*</b> • • •	🖾 VNC Viewer 🛛 🔍 🔍 🔳 🕵 🖓 🗖
수 수 👰 🗏 🔄 🏹	Instance name	Status		Detailed ongoing messages are 🔥
	<ul> <li>Puppy Emulator</li> </ul>	A		/tmp/xerrs.log (X) and /var/lo
	puppy     OEmuARM Mobile Device	* IDLE-VNC		Now executing 'init' script in
	< QEMUARM Mobile Device <none></none>			(Note: initial-ramdisk is reta
	OEmuReact Mobile Device			Loading kernel modules
	<none></none>			Mounting /dev/hda
				Mounting /dev/hdb
				Mounting /dev/hdc
				Mounting /dev/hdd
				Mounting tmpfs Creating unionfs
				creating anitonis
				Now executing 'rc.sysinit' scr
				Making the filesystem usable
				Checking if version update
	Services		*	Loading kernel modules
			->I'	Loading "us" keyboard map
	Start Instan			puppyserialdetect is running 1
				Detecting keyboard: ps/2 Mouse Setting up network interfaces.
	📕 Stop Instanc			Testing if eth0 network is ali
				Trying to get IP address fro
	Connect VNC			
	💹 Disconnect V			×
				<u>&lt;</u>
	Console			I 🗙 💥 📴 💭 🕑 🖻 • 😁 • 🗖 🗋
	puppy [Emulator Instance] C:\Docum	ents and Settings\frp743\Desi	ktop\TmLTutorial_FR\worl	<pre>space\org.eclipsecon.tml.session2\puppy\puppy.exe</pre>
				~
	<u> </u>			>
i 0*				

16. To disconnect from the VNC Viewer, click **Disconnect VNC**, then click **Stop Instance** to stop the emulator.

🖨 TmL perspective - Eclipse S						
<u>Eile Edit N</u> avigate Se <u>a</u> rch <u>P</u> rojec						
📫 • 🔛 🗁   💁 •   🛷						TmL perspective
🔝 Instance Management 🛛	🛍 • `	VNC Viewer	23			् ् 🔍 🔳 🎇 🗖 🗖
Instance name	Status	200				<u>^</u>
<ul> <li>Puppy Emulator</li> </ul>	* IDLE-VNC	10-				_ 🗆 ×
<ul> <li>puppy</li> <li>QEmuARM Mobile Device</li> </ul>	IDEC-WWC	h 1 🗟 -	•%			_ 0 >
<none></none>			and the second second second			
		_ U	🚡 轮 🔞 🤆	x @ 🔲 🏽	🖞 🛷 📃 🔯 12 item	is (40 hidden)
		br pu	noices ghttpd		my- applications corrections echoServer tcf_debu.tar	iy-roxapps org.eclipse.t m.tcf.agent
Services Start Instan Stop Instanc		calendar co	ontacts		JUY	
Connect VNC		play		U		
						×
Console						
puppy [Emulator Instance] C:\Docume	nts and Settings/fro74	Decktop) Trol Tutorial ED w	orkenace)org edincecon h	al session?\nunnvlni		
bobby fermination trustratice ] c. (Docrime	nts and setungs(ITP/4)	/wosk.op/microcondl_FR(W	orrapace (org. eclipsecon. d	m. session is (puppy (pu	appy.oxe	~
						~
<						>
i ∎•						



17. You can also create an instance of the QEMU ARM Linux emulator and/or the QEMU ReactOS emulator provided, start them and connect each one to the VNC Viewer.

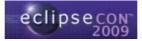
# Hands-on Session #3: Using the Protocol Framework to implement an echo protocol

In this session, we are going to use the Protocol Framework to implement an echo client that communicates with the echo server included in the puppy Linux emulator that we integrated into Eclipse in the previous sessions.

- 1. Open the project **org.eclipsecon.tml.session3**, or create a plug-in project (in this case you will need to copy the code yourself).
- 2. Switch to the **Plug-in Development** perspective.
- 3. Open the **META-INF/MANIFEST.MF** file. Select the **Overview** tab and check the option **This plug-in is a singleton**.
- 4. Select the **Dependencies** tab and add the plug-in **org.eclipse.tml.protocol** to the list of dependencies.

© Dependencies			◎ 蓉 \$0
Required Plug-ins Specify the list of plug-ins required for the operati plug-in.	↓ <mark>a</mark> on of this	Imported Packages Specify packages on which this plug-in identifying their originating plug-in.	depends without explicitly
org.eclipse.ui org.eclipse.core.runtime org.eclipse.tml.protocol (0.2.0)	Add Remove Up Down Properties		Add Remove Properties,,
Automated Management of Dependenci	Total: 3		Total: 0

5. Select the **Extensions** tab and add an extension of type **org.eclipse.tml.protocol.protocolDefinition**.



Extensions	O 🎄 ≉ 🔇
All Extensions Define extensions for this plug-in in the following section.    type filter text	Extension Details         Set the properties of the selected extension. Required fields are denoted by "*".         ID:         Name:         Show extension point description         Open extension point schema         Find declaring extension point
erview Dependencies Runtime Extensions Extension Points Bui	

6. Right-click on the extension you have just created and add an element of type **protocol**. Fill in the detail fields as follows:

org.eclipsecon.tml.session3 🛛				
Extensions				0 🅸 🎘 🤇
II Extensions Define extensions for this plug-in in the followin	Jª <sub>Z</sub> ⊫ Ig section.	Extension Element		l fields are denoted by
type filter text		protocolId*:	org.eclipsecon.tml	.session3.echo
<ul> <li>Image: state of the state of th</li></ul>		<u>handshake*:</u> parentProtocol:		Browse
	Up	isBigEndianProtocol:	true	2

7. Click on the **handshake** label to create the class with the handshaking code, or click **Browse...** to use the pre-defined class provided with the project. The class must implement the interface **org.eclipse.tml.protocol.lib.lProtocolHandshake**.



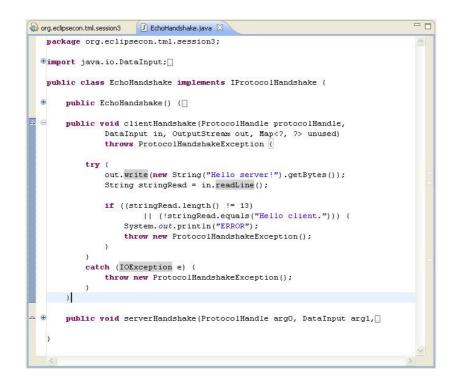
🛢 New Java Cla	55	
<b>Java Class</b> Create a new Java	class.	
Source fol <u>d</u> er:	org.eclipsecon.tml.session3/src	Browse
Pac <u>k</u> age:	org.eclipsecon.tml.session3	Browse
Enclosing type:		Browse
Na <u>m</u> e:	EchoHandshake	
Modifiers:	O gublic         O default         O private         O protecter           abstract         final         static	d
<u>S</u> uperclass:	java.lang.Object	Browse
Interfaces:	🛿 org.eclipse.tml.protocol.lib.IProtocolHandshake	<u>A</u> dd
		Remove
Which method stub:	; would you like to create?	
	public static void main(String[] args)	
	Constructors from superclass	
	Inherited abstract methods	
Do you want to add	comments? (Configure templates and default value <u>here</u> )	
	Generate comments	
0	Finish	Cancel

8. The new class contains two methods, **clientHandshake()** and **serverHandshake()**. We are going to write code for the former but there is no need to write any code for the latter, since it is already implemented on the emulator.

-	org.ecli	ipsecon.tml.	session3	🖸 EchoHandshake.java 🛛	- 0
	pac	kage or	g.eclips	econ.tml.session3;	^
	🕀 imp	ort java	a.io.Dat	aInput;[]	
	pub.	lic cla	ss Echol	andshake implements IProtocolHandshake (	
8	Θ			dshake() ( to-generated constructor stub	
4	Đ		Output throws	ientHandshake(ProtocolHandle argO, DataInput arg Stream arg2, Map , ? arg3) ProtocolHandshakeException ( to-generated method stub	1,
		}			
~	Θ	public		rverHandshake(ProtocolHandle argO, DataInput arg Stream arg2, Map , ? arg3)	1,
\$		11		ProtocolHandshakeException { to-generated method stub	
		}			
	}				
	<				2

9. Modify the **clientHandshake()** method stub as follows:





10. Select the Extensions tab and add an extension of type

org.eclipse.tml.protocol.protocolMessage to define the request that the client is going to send to the server, then right-click on it and add an element of type message. Fill in the detail fields as follows:

Extensions			◎ 蓉 \$\$(
<b>VII Extensions</b> Define extensions for this plug-in in the following section. Type filter text	Jªz ⊡	Extension Element Detail Set the properties of "messa protocolId*:	Is ige". Required fields are denoted by "*". org.eclipsecon.tml.session3.echo
	Add Remove	messageId*: messageCode*: isMessageCodeSigned*: messageCodeSizeInBytes*: messageName: messageHandler:	org.eclipsecon.tml.session3.request 0x600 false 2 request Erowse.
<u>()</u>			

The **protocolld** here must correspond to the id that was given to the protocol in the protocol definition, since this indicates that the message belongs to that protocol.

We don't need to create a handler for the request message because it is going to be handled by the server on the emulator, so just leave the **messageHandler** field empty.



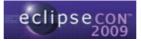
11. Right-click on the **message** element and add an element of type **variableSizeData** to specify the field that contains the string that the client is going to send to the server.

Extensio	ns			
All Extensions		Jª₂ ⊟	Extension	Elemer
Define extension	s for this plug-in in the following section.	-	Set the pro	operties
type filter text			protocolId*	k.
🖃 🐲 org.ed	ipse.tml.protocol.protocolDefinition	Add	messageId	
10000000	a.eclipsecon.tml.session3.echo (protoc ipse.tml.protocol.protocolMessage	Remove	messageCo	ode*:
X red	uest (peccade)		ichlocencod	- Sigr
I rec	uest (neccare) New	Fixeds	icMoscoco iizeData	
in R		🖹 iterat	5izeData ableBlock	
in R	New	X iterat	SizeData ableBlock ataHandler	jizeI
i rec	New Delete	X iterat	5izeData ableBlock	
E Inco	New           Delete           of: Cut         Ctrl+X	X iterat	SizeData ableBlock ataHandler	jizeI
IX red	New Delete & Cut Ctrl+X Copy Ctrl+C	X iterat	SizeData ableBlock ataHandler	JizeI
	New       Delete       of Cut     Ctrl+X       in Copy     Ctrl+C       in Paste     Ctrl+V	X iterat	SizeData ableBlock ataHandler	jizeI

12. Fill in the detail fields as follows:

Extensions			0 莽 \$\$(
Il Extensions efine extensions for this plug-in in the following section. type filter text Group org.eclipse.tml.protocol.protocolDefinition	↓ 12	<b>ision Element D</b> re properties of "v FieldSigned*: eldSizeInBytes*:	etails variableSizeData". Required fields are denoted b false
ora eclipsecon tral session3 echo (protocol)	Remove value sizeFie	FieldName*: eldName: etName:	L echoData
< >			

13. Select the **Extensions** tab and add an extension of type org.eclipse.tml.protocol.protocolMessage to define the response that the server is going to send back to the client. Right-click on it and add an element of type message, filling in the detail fields as follows:



Extensions		◎ 蓉 急(
III Extensions       ↓ d <sup>2</sup> <sub>KZ</sub> ⊂         Define extensions for this plug-in in the following section.       type filter text         Image: text = 0 = 0 = 0 = 0 = 0 = 0 = 0 = 0 = 0 =	Set the properties of "messa protocolId": messageId": messageCode":	Is age". Required fields are denoted by "*". org.eclipsecon.tml.session3.echo org.eclipsecon.tml.session3.response Dx800 false 2 response Browse Browse
Sependencies Runtime Extensions Extension Points Build		

14. Click on the **messageHandler** label to create the class containing the code that the client is going to use to handle the response message. You can also click **Browse...** to use the pre-defined class provided for this tutorial. Note that the class must implement **org.eclipse.tml.protocol.lib.IMessageHandler**:

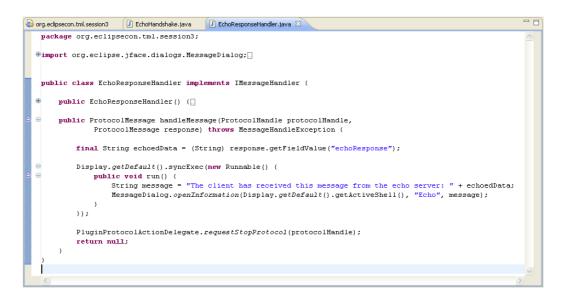
🖶 New Java Clas	55	
Java Class Create a new Java	class.	
Source folder:	org.eclipsecon.tml.session3/src	Browse
Pac <u>k</u> age:	org.eclipsecon.tml.session3	Browse
Enclosing type:		Browse
Name:	EchoResponseHandler	
Modifiers:	O gublic         O default         O private         O protected           abstract         final         static	
<u>S</u> uperclass:	java.lang.Object	Browse
Interfaces:	Org.eclipse.tml.protocol.lib.IMessageHandler	Add
	swould you like to create? public static void main(String[] args) Constructors from superclass Inherited abstract methods comments? (Configure templates and default value <u>here</u> ) Generate comments	
?	Einish	Cancel

15. Right-click on the **message** element and add an element of type **variableSizeData** to specify the field that contains the string that the server is going to send back to the client, and fill in the detail fields as follows:

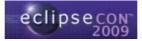


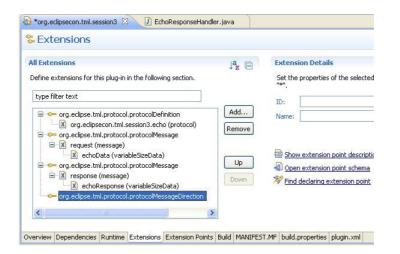
Extensions				0 🎋 🎏 🛈
Il Extensions Use filter text	Jª <sub>Z</sub> E Add Remove Up Down	Extension Element D Set the properties of "" """" isSizeFieldSigned*: sizeFieldSizeInBytes*: valueFieldName*; sizeFieldName: charsetName: value;	Details variableSizeData". Required fie false 1 echoResponse US-ASCII 1	
	Down	value:		

16. Modify the code in the **handleMessage()** method stub in the message handler class you created before as follows:

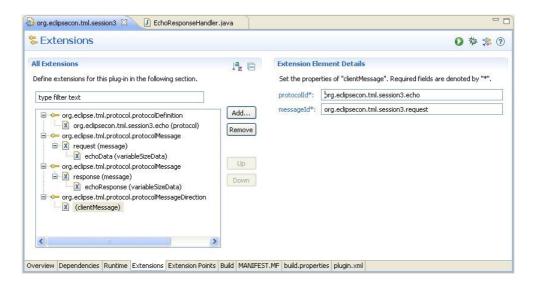


17. Now we need to define the message directions, specifying that the request message goes from the client to the server (i.e. it is a client message) and that the response message goes from the server to the client (thus being a server message). Select the **Extensions** tab from **MANIFEST.MF** and add an extension of type **org.eclipse.tml.protocol.protocolMessageDirection**.





18. Right-click on the extension you have just created and add an element of type **clientMessage.** Fill in the details with the protocol id and the id of the request message that you defined before.



19. Right-click on the extension again and add an element of type **serverMessage**. Fill in the detail fields with the protocol id and the id of the response message that you defined before.



Extensions			0 🎄 ≉ 🔘
Il Extensions efine extensions for this plug-in in the following section.	↓ª <sub>z</sub> ⊡		ement Details rties of "serverMessage". Required fields are denoted by "*". orq.eclipsecon.tml.session3.echo
ype filter text   ype filter text   ype filter text   you grading a state of the st	Add Remove Up Down	messageId*:	org.eclipsecon.tml.session3.response
X     (client/Message)       X     (server/Message)			

20. Open the launcher class **from the emulator project** and add the following option:



21. The protocol is now ready, but we need to create a way to start the client. We are going to add a button to the toolbar to do this. Select the **Extensions** tab from **MANIFEST.MF** and add an extension of type **org.eclipse.ui.menus**. Right-click on the extension and add an element of type **menuContribution**, filling in the **locationURI** field with **toolbar:org.eclipse.ui.main.toolbar** to add a command to the toolbar.

Extensions			0 🌣 🅸 🕽
III Extensions	↓ <sup>a</sup> z ⊡	Extension Ele	ement Details
Define extensions for this plug-in in the following section.		Set the proper	rties of "menuContribution". Required fields are denoted by "*".
type filter text		locationURI*:	toolbar:org.eclipse.ui.main.toolbar
	Add Remove	<u>class:</u>	Browse
🖳 🕅 toolbar;org.eclipse.ui.main.toolbar (menuContribution)	Up Down		
< >			

22. Right-click on the **menuContribution** element and add another element of type **command**, filling in the **commandId** field with **org.eclipsecon.tml.session3.startEchoClient**. This should be the same id used in the next steps, where we are going to create the actual command.



Extensions				0 🅸 🏂
I Extensions efine extensions for this plug-in in the following section. :ype filter text		Extension Elem Set the properti commandId*:	nent Details les of "command". Required fields are denoted by org.eclipsecon.tml.session3.startEchoClient	"*".
Constant Series Constant	Add Remove	label: id: mnemonic: icon: disabledIcon: hoverIcon:		Browse Browse
		<u>tooltip:</u> helpContextId: style: mode:	push	×

23. Add an extension of type org.eclipse.ui.commands. Right-click on the extension and add an element of type command, filling in the id field with the same command id you used in the previous step and filling in the name field with the text to be displayed in the UI.

Extensions			🖸 🕸 🚺
All Extensions Define extensions for this plug-in in the following section. Eype filter text	Jª <sub>Z</sub> ₪ Add Remove	Extension Elem Set the propertie id*: Dame*: category: description: categoryId: defaultHandler: returnTypeId: helpContextId:	nent Details ses of "command". Required fields are denoted by "*". org.eclipsecon.tml.session3.startEchoClient Start Echo Client Browse Browse Browse Browse Browse

- 24. Make sure you saved everything and run the plug-in as an Eclipse application.
- 25. Start the emulator instance and connect it to the VNC Viewer.
- 26. Start the echo server on the emulator by typing xxxx in the command line.
- 27. Start the echo client in Eclipse by clicking on the toolbar button.
- 28. You can see the string that the client sends to the server and that the server sends back to the client.



# Hands-on Session #4: Using the /proc tools to display system information in Eclipse views

In this session, we are going to take the puppy Linux emulator that we integrated into Eclipse in the first and second sessions, and use the /proc tools to get system information.

- 1. Open the project you created in the second hands-on session, or use the predefined project **org.eclipsecon.tml.session4**.
- 2. Run the plug-in as an Eclipse application and start the emulator.
- 3. Start the /proc tools server on the emulator (/root/tcf\_Debug/agent).
- 4. Select **Window > Show View > Other...** and choose the **CPU Load** view from the **Linux Tools** category.



5. Click the **Connect** button from the **CPU Load** view toolbar.



6. In the dialog box, fill in the **Host** and **Port** fields with 127.0.0.1 and 1534, respectively, and select **TM TCF Agent as** the **Protocol**. Click **OK** to connect.



e		
2012/02/02/02/02	nnection alues for the connection parameters	P
Host:	127.0.0.1	
Port:	1534	
Protocol:	TM TCF Agent 💌	
?	ОК	ancel

- 7. The view shows processor load information from the emulator.
- 8. Select **Window > Show View > Other...** and choose the **Memory Map** view from the **Linux Tools** category.
- 9. Click the **Connect** button from the **Memory Map** view toolbar.



- 10. In the dialog box, fill in the **Host** and **Port** fields with 127.0.0.1 and 1534, respectively, and select **TM TCF Agent as** the **Protocol**. Click **OK** to connect.
- 11. The view shows memory map information from the emulator.

