MOBILE DEVICES AND EMULATORS WITH TML

SUMMARY

This document describes the components that were implemented so far in the scope of the Tools for mobile Linux project. It reviews the project goal of providing tools and frameworks to support development of mobile applications and describes how the two existing components contribute to it. The first component is the Device Framework, which supports integration of a mobile device or emulator to the Eclipse IDE. The second component is the VNC viewer, which allows an Eclipse view to display the contents of a device or emulator display using the well-known VNC (or RFB, Remote Frame Buffer) protocol. The two components will be often used together to support a given device or emulator.

By TmL Team, Eclipse Foundation.

March 15, 2008

INTRODUCTION

Mobile devices are gaining strong acceptance as an essential component in the end-user daily life. In order to harness the potential of mobile devices more powerful software development environments and tools are needed to support developers to create sophisticated applications and make the bridge between robust IDE and mobile devices.

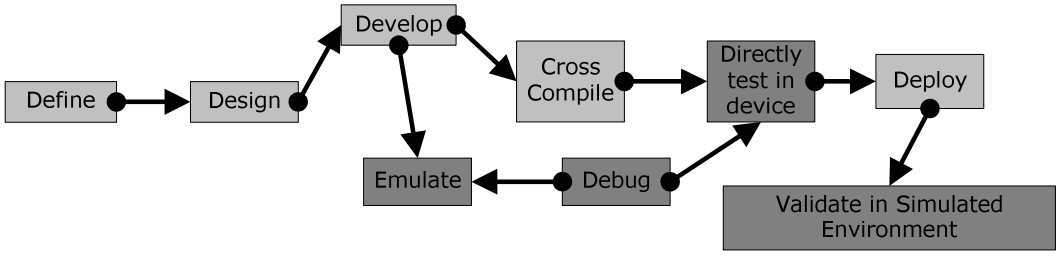
Eclipse is a natural choice of a base platform for such development environments and tools. Eclipse has a proven track-record as an IDE for desktop applications in Java, C/C++ and various other programming languages. By using Eclipse's powerful and flexible plug-in and extension point mechanism for extensibility, one can leverage those existing capabilities for embedded software development.

The most significant contribution of TmL will be support to integration of mobile device emulation into the Eclipse IDE and simulation of end-to-end enterprise environments for testing of enterprise applications (backend and communication simulation).

Device emulation is important because hardware prototypes are often not yet available when developers start creating the applications for a new mobile device. Simulation of end-to-end enterprise environments is important because such environments in reality may involve several devices and more expensive equipment such as servers. Device emulation and simulation of end-to-end enterprise environments were thus chosen to be the initial focus of the TmL project.

MOBILE DEVELOPMENT AND TML

The bottom line of mobile development is providing tools which spanning the entire software development lifecycle adding support to run, debug and validate applications developed directly into mobile devices or using emulated environments.



The TmL framework addresses integration between development and the end-user mobile device. Using the framework it is possible help the developer providing an ease way to run the application created using emulator, simulator or the real device including debug, deploy and testing with simulated mobile infrastructure.

Most existing Eclipse projects have a more general, horizontal focus. It is important batting around existing Eclipse projects to vertically extend Eclipse into a powerful development environment for the mobile device market. Therefore it is natural that TmL will interact, re-use and extend other existing projects

It is part of this environment the creation of a framework to cover the end-to-end mobile services development. It should be capable to address a simulated mobile infrastructure to support messaging, location and several other mobile based services.

TOOLS FOR MOBILE LINUX (TML)

► OVERVIEW

Applications developed for mobile devices cannot easily be executed on a developer's PC. Execution on actual device hardware is often hindered by limited availability of hardware, esp. in the early development phase.

A critical piece of the TmL development environment is a Device Framework. TmL seeks to define an extensible framework for a real device or emulator. The design goal is to keep the device generic enough to accommodate implementations using different technologies like: virtualization servers, operating system, communication protocols and others.

The framework shall provide for ways to:

- Support different launching parameters and configurations
- Infrastructure for Eclipse control components to communicate with devices executing in the real or emulated context
- Provide Properties pages to define device configurations and arguments, including arguments to be passed on to the device system
- Support launching of and connection to remote devices

To implement parts of the emulator framework we are looking at the possibility of using existing DSDP/TM features, as well as open source protocols like the "Remote Frame buffer Protocol" (RFB).

Part of the development environment for mobile device applications is the simulation of the interaction with a mobile infrastructure. This infrastructure may include components like:

- Simulated messaging servers
- Deployment servers

Simulation or capture/replay of location based data (Cell information, GPS, etc.)

► COMPONENTS

Addressing these capabilities the TmL scope was split in three components:

- Device Framework, responsible for providing an extensible platform to support mobile devices and their aggregate services.
- VNC Protocol, a scratch implementation of RFB protocol to support visualize the frame buffer content. It can also promoting the protocol communication interface between device and TmL framework.
- Simulated end-to-end environment, responsible for supporting a fully infrastructure to connect mobile devices and it has also providing the integration among all devices connected into this network.

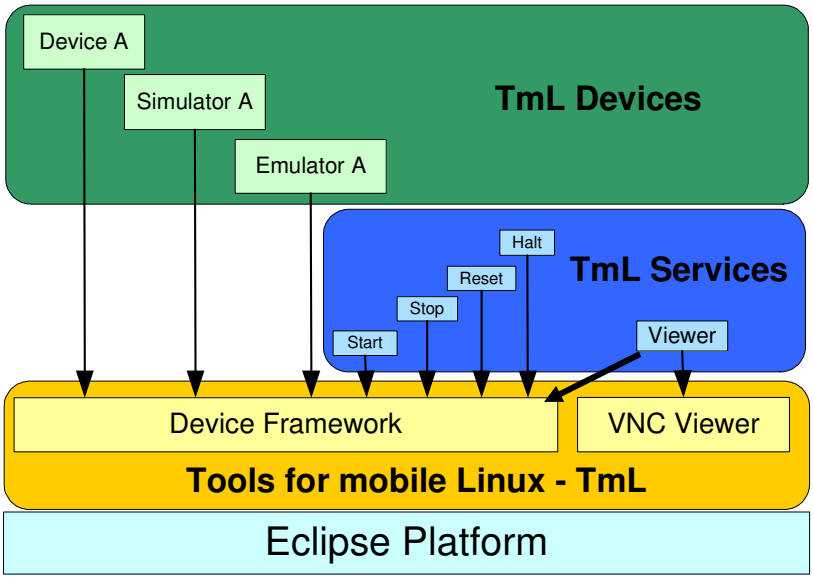
► ARCHITECTURE

The architecture of the TmL project is based on the Eclipse extension point and plug-in mechanism..

The TmL Device Framework comprises components that are useful for handling mobile devices and emulators, such as the VNC Viewer component. It also has an adapter layer for TM interoperability.

Examples of services are:

- Starting and stopping devices or emulators.
- Deploying and launching applications on a device or emulator.
- A frame buffer service to visualize the contents of a remote display using e.g. the VNC Viewer component.



DEVICE FRAMEWORK

The TmL Device Framework is designed as a generic framework that can be used as the basis to create Eclipse plug-in that connects devices and emulators to the Eclipse integrated development environment (IDE).

Device manufactures can implement plug-ins based on this framework to support their devices, thus allowing third party application developers and independent software vendors (ISV's) to run, test and debug applications on the corresponding devices and/or emulators.

The Device Framework makes directly access from the Eclipse IDE, enabling the user to manage the services provided by each one. An example of such a service is display visualization, which can be provided by a mobile phone emulator through a VNC server and used on the IDE by means of the TmL VNC Viewer embedded in an Eclipse View.

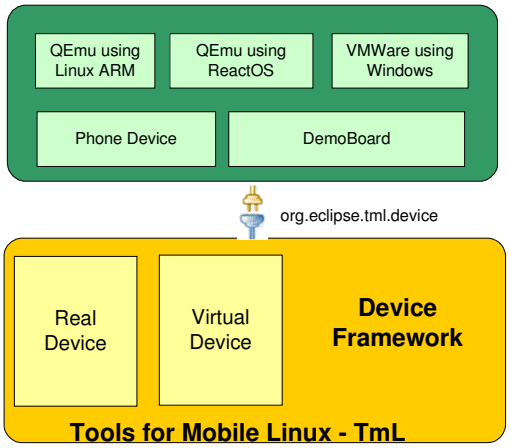
The Device Framework supports common services, such as ftp or telnet, which can be shared by devices and emulators with minimal customization. In addition to this, other services provided by the framework can be extended and customized for each different plug-in.

► DEFINITIONS

The main purpose of the TmL Device Framework is to integrate devices and emulators on the IDE. The framework defining set of actions called **services** and associates them to the device.

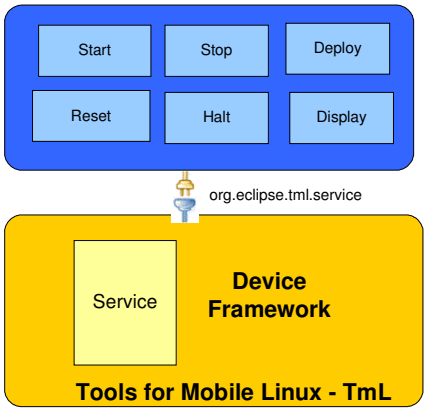
♦ SERVICE

A service represents a number of related functionalities and is implemented as a plug-in. The framework attempts to provide implementations that are either general enough (e.g. the VNC viewer) or contain partial implementations that must be extended by specific plug-ins (e.g. the Status state machine).



♦ MOBILE DEVICE

A mobile device represents an abstract description of a real device or a device emulator. It must be implemented as a plug-in that extends the *device extension point defined by the TmL Device Framework..*



The device object typically contains binary executables that emulate the mobile device or a collection of scripts that manage the connection between the host computer and the mobile device itself.

The device plug-in also contains configurable properties and extends those extension points from the TmL framework corresponding to the services available from the mobile device or emulator.

Prior to using a mobile device, the user must create an *instance* of it. Several *instances* of the same device should be created as long as they have non-conflicting properties (e.g. different ports). The TmL framework provides the code for creation of mobile device instances.

Examples of mobile devices include:

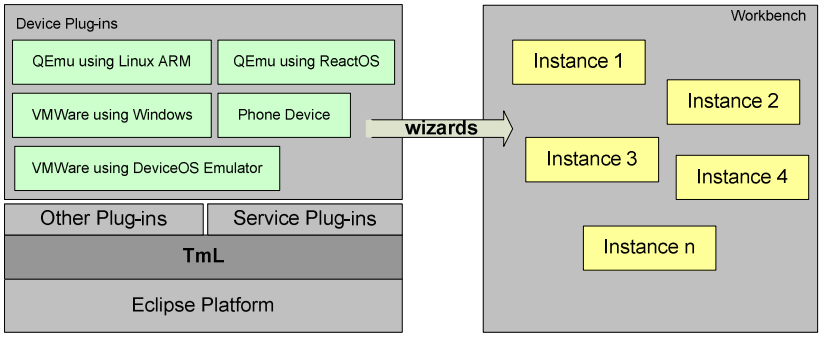
- QEmu using Linux ARM
- QEmu using React OS
- VMWare using Linux
- Phone Device
- Demo board

A device must indicate the location of the binaries in the filesystem, the startup command line and the correct parameters required to run different services for that device or emulator (e.g. startup command is different for QEmu-based emulators and VMware-based emulators).

Besides, each mobile device should support a set of services. This support in some cases demands an additional code and extra configuration to work with those mobile device associated.

♦ MOBILE DEVICE INSTANCES

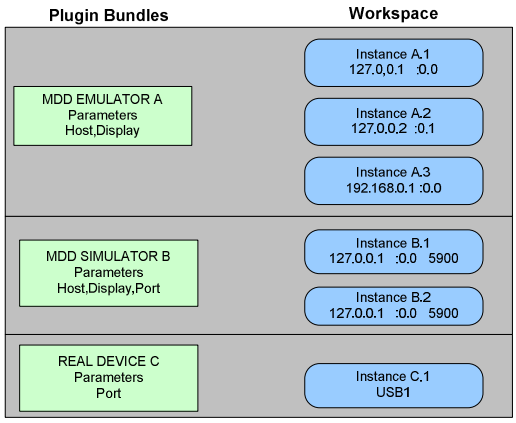
One can think of a TmL-based mobile device plug-in as a "class" that contains static information required to create and run instances of the corresponding device, whereas the dynamic information associated with each individual instance is stored in the Eclipse workspace and linked to the corresponding instance.



The diagram above shows the difference between ***mobile device plug-ins*** **and** ***mobile device instances***.

Each ***mobile device plug-in*** provides features such as a New Device Instance Wizard to create a new device instance, property edition and other functionalities.

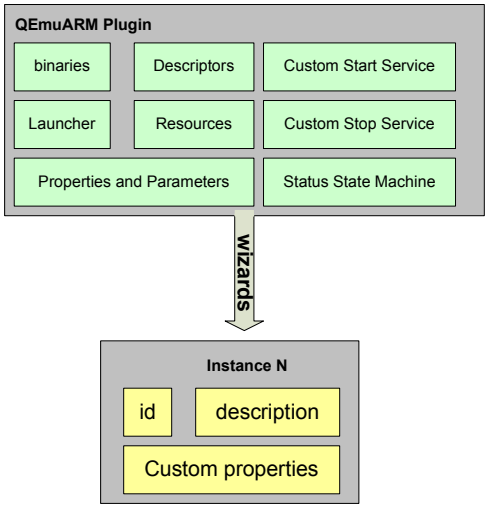
A mobile device instance assigns values for the mobile device properties defined in the corresponding plug-in. These property values are stored in metadata from the user workspace.



♦ MOBILE DEVICE PLUG-IN EXAMPLE

The diagram below clarifies the files that each plug-in contains and is part of a ***mobile device plug-in*** or a mobile ***device instance.***

The example is a QEmu ARM emulator running Linux. The *QEmu ARM plug-in* is represented as a ***device*** with binaries to start the emulator, descriptions of the device, a properties.config file and service configurations.



The properties are described in *properties.config* located in the *QEmuARM plug-in.* This file describes which properties are needed to mobile device, services and device instances. It is composed by:

- *Instance properties* - that will be replaced by the properties inside properties.config in the instance. (note that some properties can be read-only)

```
Properties.config
<host>127.0.0.1</host>
<port>5900</port>
```

- *Emulator properties* – that will be replaced by the user using a properties page and they are properties valid for all instances of this plug-in type.

```
Properties.config
<instance>
    <host>127.0.0.1</host>
    <port>5900</port>
</instance>
<emulator>
  <parameters>
     <param id="1" name="-L" value="." />
     <param id="1" name="-m" value="256" />
     <param id="1" name="-vnc" host="y"  />
  </parameters>
  <location read-only="y">
     <path>/qemu/bin</path>
     <bin>run.bat</bin>
  </location>
</emulator>
```

The properties.config for each instance has the properties values associated with the properties described in *properties.config* located in the *QEmuARM.*

The *Start/Stop Service* is configured to start, stop and refresh status using the specific commands for this device.

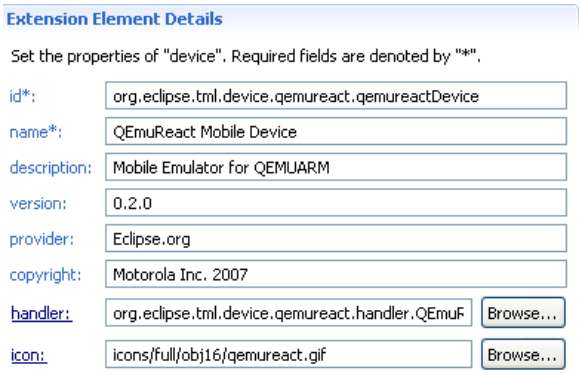
The *Display service* is also configured to provide protocol setting and connection parameters specific for this device.

► EXTENSIBILITY

This section describes the extension points available and it shows which functionality each one implements or extends.

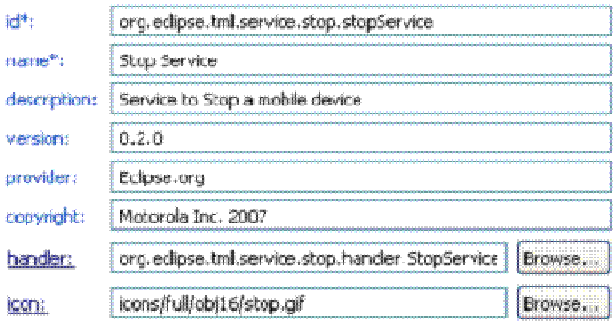
♦ DEVICE

The purpose of this extension is define a new device allowing developer setup id, name , description, version, provider, copyright, icon and class handler that framework will load when any information about this device was requested.



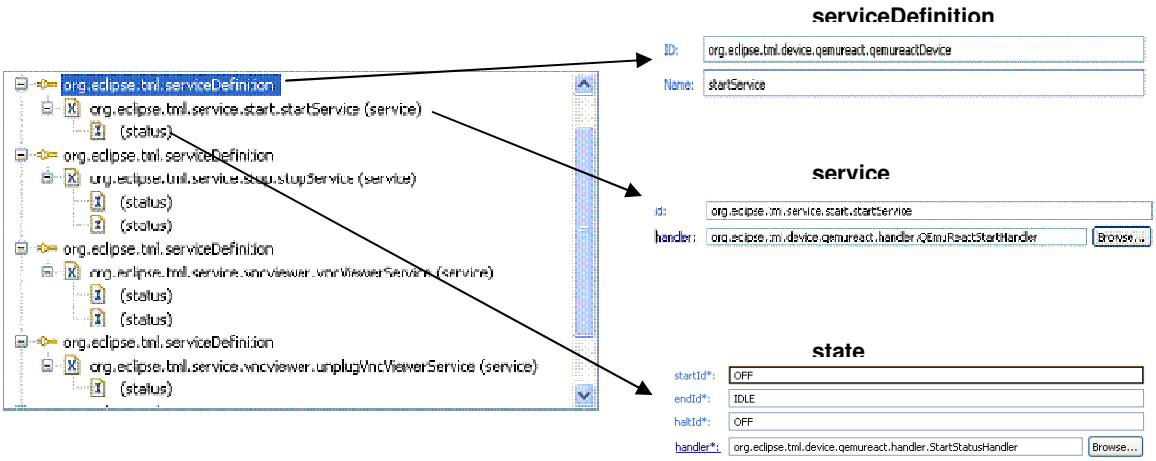
♦ SERVICE

This extension defines a new service and it associate information data for this service. Each service could be linked with devices providing a set of common functionalities related to maintain this kind of service.



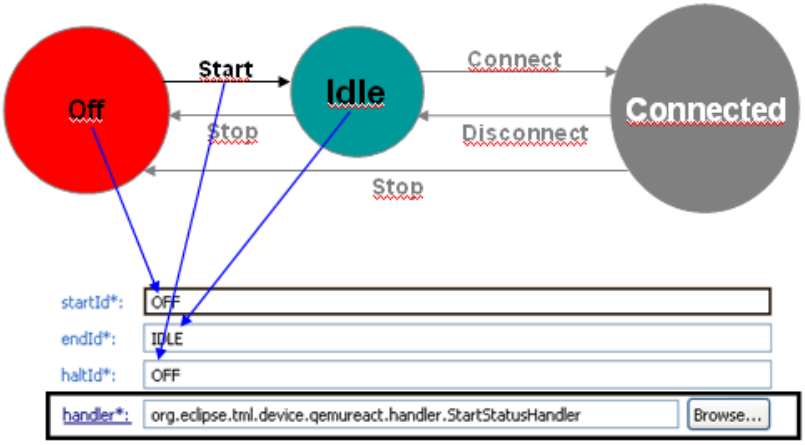
♦ SERVICE DEFINITIONS

This extension is used by devices to link services and devices. It is responsible for provide the status transition using valid values and manage the service execution.



♦ STATUS

This extension is used to provide the state machine for each device. It is possible uses a set of default states or define additional status specific for a device.



► TARGET MANAGEMENT (TM) INTEGRATION

The TmL framework will provide a set of adapters to support TM Platform. It will be possible choose run TmL default UI or making use of TM Interface totally transparent for the user.

The plug-in developer just need implements the customized adapters provided by framework and some additional extension points, then the plug-in automatically will be integrated with TM Platform.

► PLAN AND FUTURE IDEAS

This section describes the current plan for TmL and promotes discussion about new Ideas for the future.

- Automatic wizards to create device plug-ins
- Skins
- Keyboard mapping
- Multiple displays

VNC VIEWER COMPONENT

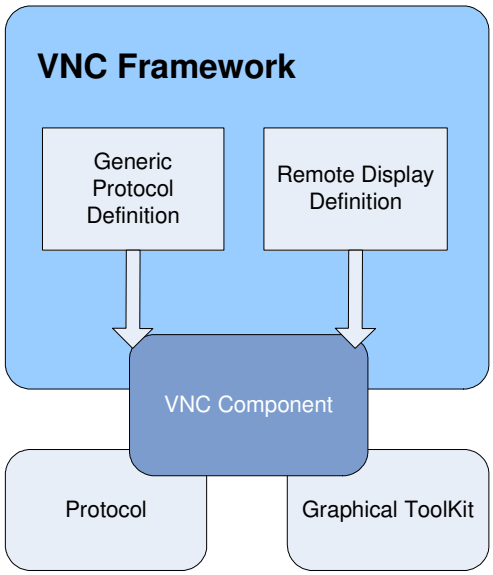
The VNC Viewer Component is designed as a library to support visualizes frame buffer content directly inside SWT components.

► USING VNC VIEWER

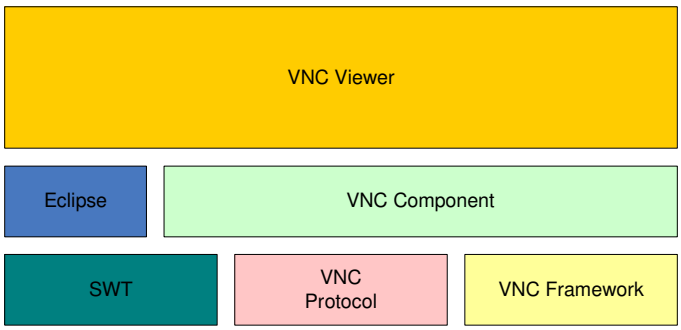
The VNC viewer is a standalone library that implements RFB/VNC protocol in SWT Components. It is totally possible uses this component without load all classes of TmL.

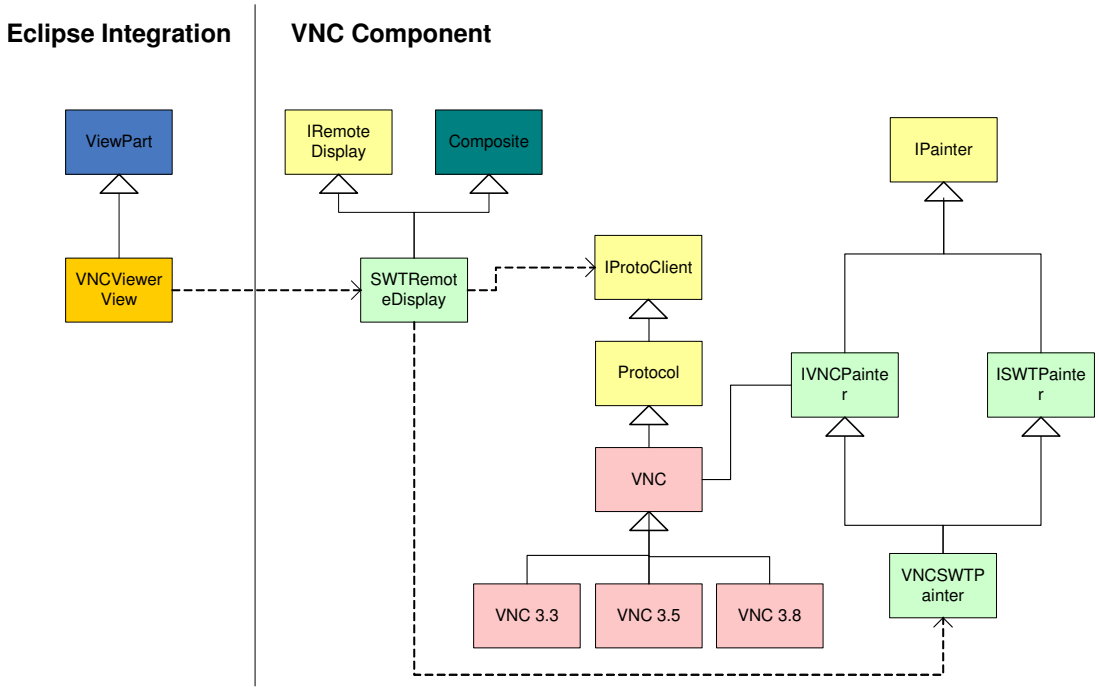
TmL also implements a plug-in that uses this library and makes connection between the RFB protocol and Eclipse Views.

► ARCHITECTURE



VNC Viewer View - Eclipse





REFERENCES

[1] Eclipse Portal (http://www.eclipse.org)

[2] Tml Site (http://www.eclipse.org/dsdp/tml)

[3] Tml Wiki (http://wiki.eclipse.org/DSDP/TML)

[4] Motorola (http://www.motorola.com)

[5] Eldorado (http://www.eldorado.org.br