WIND RIVER

DSDP-TM Design Proposal

Eclipse DSDP-Target Management Project

Martin Oberhuber, Wind River Systems
martin.oberhuber@windriver.com

Version 0.3
July 18, 2005
Status: Draft – Public Review

Document Information

Project ID	Eclipse DSDP-Target Management Project
Version	0.3
Status	Draft – Public Review
Author	Martin Oberhuber

Approvals

DSDP-PMC	

Reviewers

Document History

Date	Version	Author	Summary
22 June 2005	Rev 0.1	Martin Oberhuber	Initial draft
30 June 2005	Rev 0.2	Martin Oberhuber	Incorporate ideas from DSDP-TM phone conf.
18 July 2005	Rev 0.3	Martin Oberhuber	Simplified and stripped down

Table of Contents

About this Document

The DSDP Target Management Project has taken on the charter of developing a framework for interacting with remote systems, that can be extended by interested parties to fill their needs. Use Cases as well as terms and definitions for this project have been collected and posted on the eclipse.dsdp news group.

This document proposes a high level design for the target management framework to foster discussion among interested parties. The goal of this document is to convey some important design decisions, and help to get an understanding how target management will be integrated in Eclipse.

This document covers the framework for target management only. Therefore, for instance, no Preference Pages are listed as Eclipse extension. We assume that extensions of the target management framework for specific connection types, for instance, will register such preference pages, but we do not anticipate the framework to do so.

As the title indicates, this is a proposal meant to be the base for discussions. Feedback is welcome on the eclipse.dsdp newsgroup.

Basic Decisions

Separate Data Models for Static and Dynamic Data

The Target Management Framework works on two distinct types of data:

- Static properties of targets, target configurations and target connections that are used to describe how a connection to services on the target is made. These static properties do not change unless edited by the user. They can be stored in a file, exported, and shared among users.
- Dynamic data obtained from various services on the target, like a remote file system or the list of processes. Various update policies may be in place to refresh dynamic data.

Although static and dynamic target properties may be shown in a single target manager view, we want to keep the static and dynamic models internally separate. This is to ensure that static data can be made persistent at any time without problems.

All data model elements should implement the Eclipse **IAdaptable** interface for maximum flexibility.

Dynamic Model Update Policies

It is expected that different services on the target require different update policies to the common model.

- Events from the Target
- Automatic Polling at Time Intervals
- Manual Polling
- Lazy Evaluation (only query the data that is shown to the user)
    - When a Filter is applied on the current view, services may or may not allow to apply the filter to their queries already.

For all model updates, we have to keep in mind that target connections may be slow. In general, we expect that various remote services are asynchronously called to request some data, which is later on pushed into the common model when the asynchronous callback returns.

Persistent Storage of Connection Data in Projects

Persistent Data for Target Connections will be stored as resources (XML files) in the local file system. The default project to store these connections in will be a project named "Remote Connections" that can be created when it does not exist. Users can choose to store their connection data in different projects if they want. But we will not store connection data outside project / resource scope.

Eclipse Extensions

Remote Connections View

org.eclipse.ui.views

The Target Management Framework shall register an Eclipse View that shows the local target list as well as dynamic target information obtained from services running on the target.

Given the fact that the Target Management Framework may be used in different contexts where the terminology for remote systems is different ("target" vs. "remote host" for instance), we propose to name the view "**Remote Connections**" view.

The Remote Connections View shall be a Tree View with following levels:

- (Optional) Target Group
    - Target Connection
        - Subsystem
            - Filter on Subsystem
                - Dynamic data as obtained from Subsystem

Subsystems and below are only shown when a target is connected.

The user can define Target Groups to perform common operations on a set of targets. In case such target groups exist, the Target Manager View may provide different view modes for either showing the targets grouped, showing a plain list of all targets, and showing connected targets only.

Important Decision:

For a single physical target, different configurations may be defined (which may be active at different times). For instance, a user may boot his board to run VxWorks on one occasion, or boot Linux on another occasion; on a third occasion, a JTAG connection may be used to connect the same board without any OS running.

Our experience showed that users tend think in terms of these target connections rather than the physical targets connected. They create a target connection and do not want to enter too much data or properties for the physical target.

We do need to maintain the information about what target connections refer to what physical targets internally (for instance in order to allow proper reservations of shared lab targets). But we propose not to show the physical targets in the remote connections view by default, because

- In most cases, there will be only one connection below each physical target.
- Even if there are multiple connections below the physical target, only one of them can be active at any time.

Thus we think that adding the "physical target" level would lead to confusion rather than help. It might be desired, however, to allow another view mode "group by physical target" in order to facilitate management of the association between physical targets and target connections.

Actions on the Remote Connections View

org.eclipse.ui.action, org.eclipse.ui.viewActions, org.eclipse.ui.commands

The Remote Connections View will register Eclipse Commands for "Add Connection", "Delete", "Refresh", "Properties", "Connect", and "Disconnect". Clients and extenders of the remote connections view may register more actions or commands.

New Wizards

org.eclipse.ui.newWizards

The core target management framework will offer a single connection type "Standard TCP/IP connection" which offers services for remote shell (via telnet) and remote file system (via FTP). This connection type will be registered as a "New" wizard. More Services (e.g. Ssh, rlogin, sftp, ...) may be added to this connection type via extensions.

It should be very simple to create a basic connection to a remote system (e.g. New > Remote Connection > Standard TCP/IP connection).

Import and Export Wizards

Org.eclipse.ui.importWizards, org.eclipse.ui.exportWizards

The Core Target Management Framework will not register any import wizards, since persistent connection data will be stored in readable and exchangeable XML files anyway. Clients of the framework may register import wizards for their use.

Dynamic Variables

Org.eclipse.core.variables.dynamicVariables

The framework will provide dynamic variables to get the name of a selected connection, the IP address of a selected connection, and other arbitrary properties of the selected connection. These may be used in "External Tool Launchers" to launch tools for the selected connection.

Extension Points

The following section lists extension points, where clients of the Target Management Framework can register their extensions.

Target Connection Type

E.g. "Wind River VxWorks Target Server connection"

Like an RSE Profile: a set of services

Connection type is bound to a "New Connection" wizard.

Subsystem Type

E.g. "Target Contexts", "Kernel", "Cores", "Files"

Remote Service

E.g. FTP, Telnet, ssh, rlogin, …

May require a particular subsystem (e.g. FTP requires the Files subsystem)

May be added to any connection type

Other Extensions (Actions, Icons, Property Pages,...)

We expect that other extensions can be provided to the system via standard Eclipse extension points.

Workflow for Important Use Cases

Connect a local target

File > New > Connection: Standard TCP/IP → Enter target name and IP Address, press next, then select from installed service types (FTP, telnet, ssh, rlogin… depending on services)

Creates an entry in the Remote Connections view, and fills subsystems as defined by the services connected.

Additional services / connection types can be added by selecting the same target and choosing "New Connection" from there, e.g. adding a serial line console. Thus all connections to the same target are under that target.

Differences to Existing Projects

Extension Points as they are implemented in currently known implementations:

Bugzilla 65471 RSF	DSDP-TM	IBM RSE
CommunicationInterface e.g. IP, Serial		
ServiceType e.g. IRemoteFSManager	SubsystemType	
Service e.g. FTPRemoteFS	RemoteService	Subsystem
RemoteSystemTemplate e.g. MVLinuxTarget	**TargetConnectionType** brings its own Wizard	SystemType
RemoteSystemTemplateProvider		

Questions

- Will we need some API for secure authentication (for communication protocols like ssh), or can that be handled by the various services?

References

- DSDP project proposal, http://www.eclipse.org/proposals/eclipse-dsdp/index.html
- Remote System (Target) Definitions, https://bugs.eclipse.org/bugs/show_bug.cgi?id=65471
- Eclipse Communication Framework, http://www.eclipse.org/ecf/
- IBM Remote System Explorer (RSE), http://www.developer.ibm.com/isv/rational/remote_system_explorer.html