WIND RIVER

Use Cases for Target Management

Eclipse DSDP-Target Management Project

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

Version 1.1
June 22, 2005
Status: Draft – Public Review

Document Information

Project ID	Eclipse DSDP-Target Management Project
Version	1.1
Status	Draft – Public Review
Author	Martin Oberhuber

Approvals

DSDP-PMC	

Reviewers

Rudolf Frauenschuh	WindRiver	DSDP Sponsor
Michael Scharf	WindRiver	Architectural Overseer

Document History

Date	Version	Author	Summary
03 June 2005	Rev 0.9	Martin Oberhuber	Initial draft
06 June 2005	Rev 1.0	Martin Oberhuber	Added “Initial Startup” UC, status public review
22 June 2005	Rev 1.1	Martin Oberhuber	Added “Remote Build” and “Discovery” UC Incorporated Comments from TM Conf.Call

Table of Contents

About this Document

There is a lot interest in creating a unified component for Target Management in Eclipse. Part of this interest comes from CDT, where a Bugzilla Entry currently documents initial work on a Remote System Framework:

- https://bugs.eclipse.org/bugs/show_bug.cgi?id=65471

The Device Software Development Project (DSDP) has taken on the charter of developing a framework for Target Management that can be extended by interested parties to fill their needs for driving the interaction with remote systems.

This document summarizes the Use Cases for Target Management that we know now, in order to

- Ensure that interested parties have an understanding what Target Management is, what it can do and what it cannot (or will not) do.
- Create a common terminology for Target Management.
- Make sure that we keep all known desired uses of Target Management in mind when designing the basic framework.

This document is not a "pure" Use Case Specification; it contains ideas for implementation or data structures where we felt it was appropriate to clarify the issues.

Not all Use Cases described in this document will be implemented in a first version of the Target Management system. Some of them will be implemented as extensions (and not in the framework).

The goal of this document is to convey the Big Picture of target management, to ensure that the frameworks we are going to discuss are versatile enough.

What is Target Management?

When a software developer writes programs that are intended to run on a different computer system than the local one (we will call it the *target* system), there is a need to interact with various remote systems during development. Such interaction may be needed to

- Build the program (remote build)
- Connect, and get status information from remote (target) systems
- Deploy the program (download)
- Run or Debug the program
- Test the program (e.g. automated tests, performance tests)
- Get data from remote systems (upload)

Target Management shall provide a consistent user interface for these actions, even though different Eclipse extensions may implement the corresponding functionality. In order to do that, the Target Management framework shall also maintain a registry of remote system (target) definitions, their properties and communication parameters.

Some of the remote systems (targets) involved may be shared by a team of software developers, or located far away. For such *shared targets*, the Target Management framework shall provide interfaces for

- Target Discovery (find a suitable remote system),
- Exchange of target definitions between users, and
- Reservation to ensure exclusive access to shared targets.

For reservation of remote targets, it shall be possible to define a *target group* that can be reserved or unreserved as a whole, in order to run or test a software system that requires interaction between multiple remote targets.

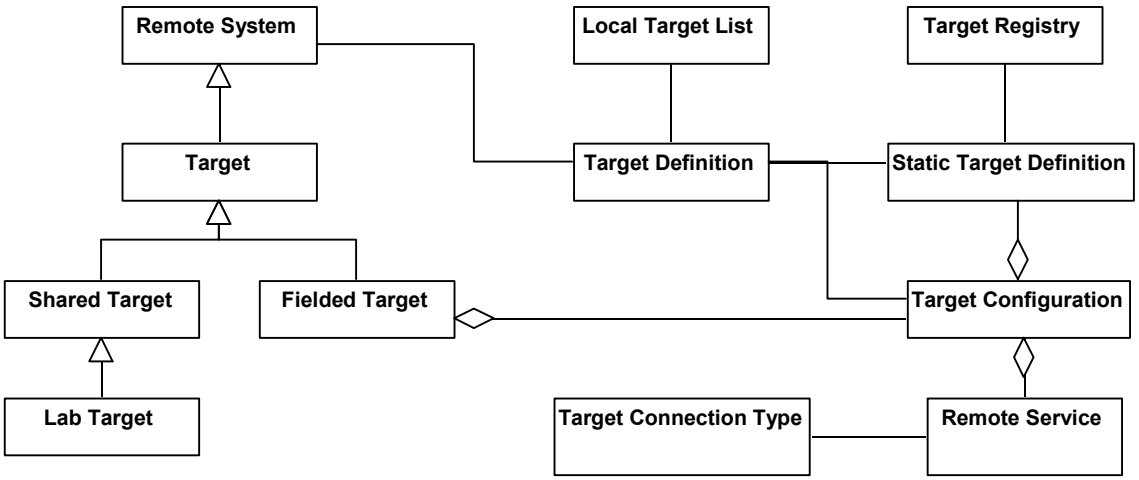
The Target Management framework shall allow for extensions to provide *secure communication* to remote systems, as well as *secure authorization* with a target reservation system.

Scope: What is Target Management Not?

The scope of Target Management ends where registered tools like a debugger or test automation framework take over. The border between target management and its clients may be fuzzy, especially in terms of downloading or reboot for hardware debugging.

Basically, Target Management should prepare target connections, and select a target context such that other tools (like a debugger) can take over debugging the context.

Definition of Terms



We shall call the remote computer systems where software is being run *target,* while we shall keep the term *remote system* for other remote computers involved in the software development process, like build machines or database hosts.

Yet, since the same communication protocols may apply to both types of remote system and the same target registry may hold references to both types of system, we will not keep this separation too strict and allow mixing these two terms as appropriate.

Build systems like “make” also often use the term *target* to refer to the result or goal of a build run. If we refer to such a build output, we shall call it *build-target* in order to keep it separate from the target systems we mean otherwise.

Remote System	A computer system that is accessed by some sort of communication protocol, and provides some services to remote users. Remote Systems may also be virtual, i.e. software processes that simulate or emulate a system on a host computer.
Remote Host	A remote system that allow concurrent access of multiple users at the same time. Typically, remote hosts are used for running software tools, whereas remote targets are used for running and testing the system under development.
Target	A remote system that is used for running or debugging software under development. Typically, the user of a target has exclusive (single user) access to the target, whereas on a host, multiple users may work in parallel.
Single Context Target	A target that only provides a single context, e.g. a single core system connected to a hardware debugging probe.
Multi Context Target	A target that provides multiple contexts, e.g. when an OS is running on the target and it is required to select individual contexts for operations.
Development Target	The primary target that a developer works with, as opposed to (shared) targets that are used only rarely for testing. Development targets should be fast and easily accessible in order to ensure a good workflow.
Shared Target	A target that may be used by multiple remote users (typically one at a time, so the target is reserved for exclusive access for one user at a time. We use the term “Remote Host” for remote systems that allow concurrent access of multiple users at the same time).
Lab Target	A shared target that cannot be accessed directly, so that operations like power on/off, reset or console access must be handled through a lab management system.
Fielded Target	A target that has been deployed to a “very remote” site, such that less support for software development is provided and secure communication to that target is desired. Fielded targets run a fixed, deployed target configuration.
Build-target	The result or goal of running a build system (like *make*).
Target Definition	The properties of a target that are required to identify it and open communication channels to services provided by the target.
Static Target	The parts of a target definition that never change, even if the target boots a different operating system, for instance: CPU type, memory

Definition	size, location, boot loader, peripherals connected.
Target Configuration	Settings loaded on a target for running software. For instance: Kernel image of OS loaded, IP addresses assigned to hardware interfaces, JTAG register files, remote services running on the target. The target configuration is part of dynamic target definition.
Target Registry	A shared database of target definitions, maybe integrated with a lab management system.
Lab Management System	A combination of software and hardware that allows remote access to lab targets, including remote power off and on as well as serial console redirection via a terminal server.
Target Group	A set of targets that are used together.
Local Target List	The list of targets that a software developer actively works with. It is a subset of the targets registered with various target registries, plus additional local target definitions that the software developer uses only locally. Members of the local target list are the primary candidates for operations like connect, download, debug and run.
Remote Service	Targets can provide a variety of services like a console prompt, data transfer, reboot, inventory or other specific services provided by agents running on the target. Remote services for a target can also be provided by extra hardware that is connected to a target, like an ICE box connected to a JTAG connector. For each remote service that is to be used by the Target Management system, a local service adapter must make the service available.
Communication Channel	A communication channel gives access to a remote service. Communication channels may be protocols like plain TCP/IP (with host + port specified), telnet or secure shell (ssh). The same remote service may be reachable via different communication channels.
Remote Service Adapter	A local extension to the Target Management system, that makes a remote service available to the user via GUI actions.
Target Connection Type	The primary means of connecting a target, e.g. “Linux kgdb connection”. Connection types may use a single remote service, or multiple remote services to accomplish different tasks. Additional services may be used as provided by the target configuration that is referenced by the connection type. The main use of the connection type is to make it easier for users to create initial connections to a target. They define a wizard that helps in setting up the initial connection with all required properties.

Actors

Developer

Software Developers want to easily setup target connections, so that they can build, download, run, debug and test their software. They will typically not want to enter administrative information about their targets but have a quick way to get the target connection.

Lab Administrator

Lab administrators are responsible for placing new targets into a shared target lab. They enter target properties as well as administrative information like access permissions for the new target, so that developers can find, reserve and connect the target. Lab administrators can also mark targets as being offline or broken.

Power User

Power Users are like Developers, but may have additional rights to perform administrative tasks in the target registry, e.g. change access permissions on targets or forcedly unreserved targets that are reserved by developers who went on vacation. There may be different roles of power users with different access permissions on the target registry.

Target Registry

The target registry is mostly passive in that it holds target information entered by developers or lab administrators. Yet, specific implementations of the target registry may also be able to send events to connected Target Manager clients when information in the registry changes, like a target is being added, removed, reserved or unreserved.

Lab Management System

Like the target registry, the lab management system can potentially also send events to the Target Manager clients, e.g. in case of hardware failure or administrative system shutdown.

Status Update Timer

A local status update timer may trigger the Target Management system to periodically poll connected targets for status information, like a UNIX “top” command.

Target

The target is not an actor by itself, but it may offer remote services that trigger events in the Target Management system through the local adapters for these services.

Constraints

Extensibility

The Target Management System should be extensible in terms of target registry types, communication channels and services or tools operating on targets.

Simplicity

It should be **simple** to access a local target, i.e. there should be no need for overly complex user interfaces due to using the Target Management Framework when only basic functionality is required.

Security

Extensions of the Target Management Framework may implement security critical features like authentication with a target registry or secure communications. The framework must not compromise these.

Scalability

When using shared target registries, working with these registries must scale to thousands of targets and thousands of concurrent users. This is mostly for target discovery and reservation.

The local target list does not need to scale so much in terms of the number of connected targets, but individual targets (or target groups) may comprise of thousands of nodes (CPUs, components, OS objects like processes running). Also it may be necessary to process a large number of events from the target(s) per time slot. Therefore, update of any views on target state must be decoupled from the actual state being modified by events.

Slow Connections

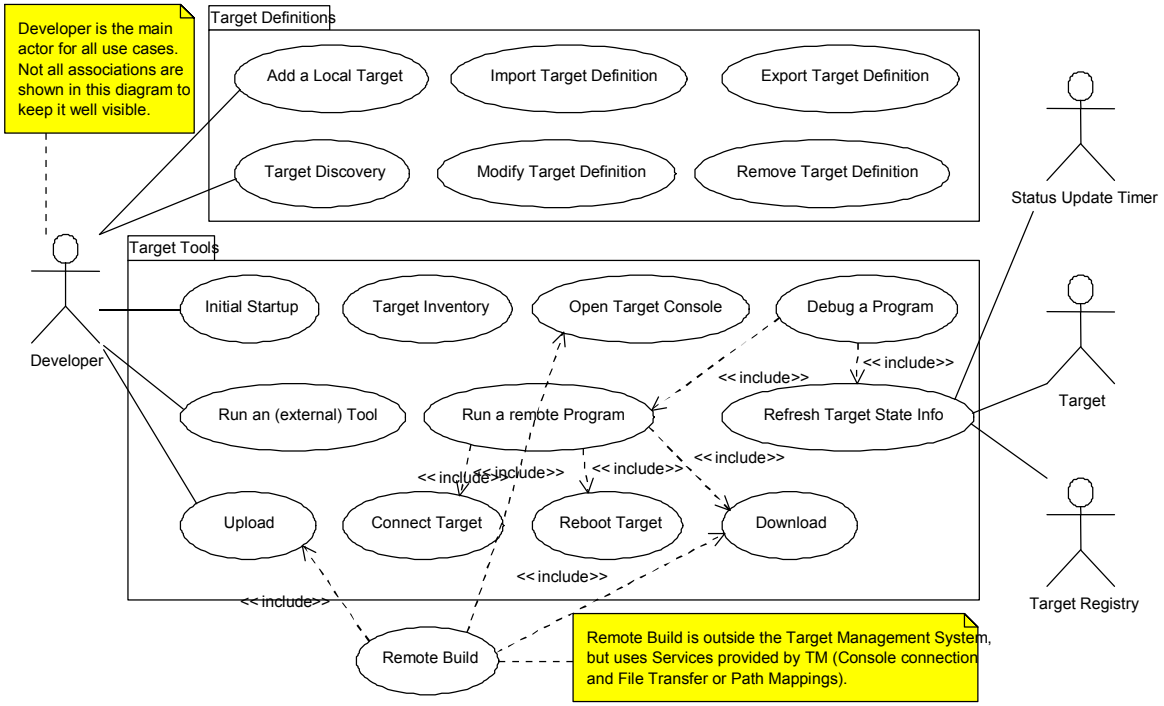
The system must be able to handle very slow target connections, or very large communication delays for very remote targets. Wherever services registered by the target manager need some communications, it must be possible to limit these to the absolutely necessary. Target State Information should be updated only when necessary, i.e. when viewed. It should be possible to toggle views showing target state between “continuous update” and “snapshot on demand” mode.

Documented APIs

All functionality should be available for programmatic use via APIs in order to allow Unit testing.

System Use Cases

Working with Local Targets



Initial Startup

Actor: Developer

Summary: Prepare the Target Management System for first-time use in a Workspace.

Description: Allow Eclipse plug-ins to automatically perform operations in the Target Management System when Eclipse is started for the first time (i.e. a workspace is created). At this time, registered services should be called to create default connections in the local target list, search for target registries to connect (see target discovery), or prepare a set of pre-canned configurations to start with.

Result: Workspace is prepared (e.g. by automatically adding target definitions).

Add a Local Target

Actor: Developer

Summary: Define the properties for a local target, and add it to the local target list, so that it becomes available for connect, run, download and status info queries.

Description: For the developer, just adding a local target alone is useless. The local target will always be needed in conjunction with some other operation, like retrieving status information, download or run.

Developers do not want to enter any information that is not required for the action they want to perform, so it should only be necessary to enter the properties required for an initial connection. The user should not be bothered with static target definition

or target configuration data.
This could be implemented by Target Connection Types, which register themselves as an Eclipse "New Wizard" in the File > New > Target menu.
Validity of the information entered should be checked in the wizard to make sure that connecting the target will actually be possible with the information entered.
It can make sense to add a local target definition based on the settings of an existing target definition, so that only some settings are modified ("clone a target definition").

Result: Target information is added.

Notes: See also "Add a Shared Target" for targets being added by a Lab Administrator.

Export Target Definition

Actor: Developer

Summary: Export a working target definition, such that other developers can use it.

Description: Select a target definition to be exported, and export it to an external source.
External sources can be files (to be given to other users), or target registries such that a local target is made "shared".
Exporting might be implemented via an Eclipse "export" wizard, or the implementation of the local target list might be similar to Eclipse Launch Configurations, such that target definitions can be declared "local" or "shared" and they can be exported / imported by means of a version control system when they are shared.

Result: Target information is exported.

Import Target Definition

Actor: Developer

Summary: Import target definition from an external source.

Description: External sources of target definitions can be files, or registries of shared targets. When a target definition is imported from a shared target registry, a reference to the target definition in the registry is kept so that reserve/unreserve operations can be performed.

Result: Target information is added to the local target list.

Exceptions: Target definition of the name to be imported already exists. In this case, a new target definition with a name like "new target (1)" should be created.

Target Discovery

Actor: Developer

Summary: Search the LAN and local serial connections for any usable targets or target registries, such that they can be used immediately.

Description: User starts target discovery, and chooses connection channels to check for targets (serial ports, network). Any targets or target registries that are found are presented to the user in a list such that they can be added to the known local services.

Result: Targets and/or target registries are added.

Modify Target Definition

Actor: Developer, Lab Administrator

Summary: Modify settings of a target definition.

Description: Modify settings of a target definition, for example: change the System (Kernel) image, change options for registered services (e.g. enable logging for the console service), and add or remove remote services.
For shared targets, the local developer may only be allowed to modify dynamic settings (like load a different system image) but not static information like the CPU type.
For shared targets, modifications remain local until they are exported to the shared target registry.

Result: Target definition is modified.

Remove Local Target Definition

Actor: Developer

Summary: Remove a target definition from the local target list.

Description: Remove (unreferenced) a target definition from the local target list.
In case the target was reserved, the user is asked if he would like to unreserved the target (it might still be used by the same user in a different session).

Result: Target definition is removed from the local target list.

Notes: In case of a shared target, removing it from the local list does not remove it from the shared target registry (it can be re-imported). See “Remove Lab Target” for removing shared targets.

Connect Target

Actor: Developer

Summary: Fire up all programs and services required to connect to a target, and do some initial protocol handshake to verify that the connection is valid.

Description: The meaning of “connect” may depend on the services provided by the target. The idea is that after connecting, the user knows that a target is available (switched on and connected properly). When there are services registered that continuously show updated target information, these services are started.
Target connections may require additional programs to be launched. This is particularly true when dealing with a virtual target, i.e. simulating a system by software. In that case, at least the simulator may need to be launched. Other programs to be launched may include software for secure communication channels (e.g. ssh), network file system servers, agent controllers or message brokers.

Result: Target is made available for operations like download, run/debug, status information retrieval and reset.

Disconnect Target

Actor: Developer

Summary: Disconnect all target services.

Description: Disconnect all connections to target services, close programs and tools depending on these connections, and terminate programs that have been started in order to support the target connection but are no longer needed now.

Result: Target is disconnected.

Refresh Target State Information

Actor: Developer, Status Update Timer, Target

Summary: Query a target to get dynamic status information like a process list, running/suspended status, system load etc.

Description: The Target Management System should show target state information like a process list, available memory, resource usage etc.
Some of this state information may be visible permanently (shown by one or several Eclipse views), while other information may be updated on demand only (to show in dialogs). There may be different adapters for bringing state information from the target into the Target Management system.
Some state information may be updated automatically by receiving events from an agent running on the target, other might need to be queried explicitly. Some of these explicit queries may be configured to run automatically at given time intervals.
The Target Management System needs to maintain internal data structures to hold those parts of target state information that are updated by events or by timers, or that should be shown permanently.

Result: Dynamic status information is shown.

Open Target Console

Actor: Developer

Summary: Connect a target and show a command line prompt.

Description: Availability of the console action depends on whether the target provides a console service.

Result: Target console is opened.

Remote Build

Actor: Developer

Summary: Build a workspace or individual project(s) on a remote system.

Description: This use case will not be handled by the target management framework itself. Instead, external build service may use services provided by the target management framework to accomplish this task.

Availability of the console action depends on whether the target provides a console service.

Result: Target console is opened.

Target Inventory

Actor: Developer

Summary: Query the target to automatically obtain information about static target properties like MAC addresses, memory size, CPU type and speed.

Description: Availability of this action depends on the services provided by the target. When an inventory service is available, it queries static information from the target and presents it to the user in a dialog. In the dialog, information items may be accepted to add to the target definition or rejected. All accepted information is then added to the static target properties.

Result: Static target properties are updated.

Download

Actor: Developer

Summary: Select one or more files and download them to a target in the local target list.

Description: There may be different kinds of download:

- Load to an absolute address in target memory (e.g. via JTAG debuggers)
- Load into target flash memory ("special load")
- Load a kernel module on the target.
- Download to a target file system

Depending on the type of download, different properties may be needed like a target file system location, a memory location or a type of flash device. For downloading into a target file system, a file system browser view might be desired.

Download can also be done implicitly by having a known file system location that can be accessed by both the host and the target; in this case, a mapping of path names must be done.

For each of these types of download, a remote service with a corresponding remote service adapter must be registered to handle the case. Remote Service Adapters may need to do data transformations (ascii-ebcdic, windows-UNIX line ends) depending on the type of remote system.

Download operations may be triggered directly, or they may be part of a Launch Configuration "script".

Result: Data is downloaded to the target.

Notes: See also "Download to Target Group" for loading software onto multiple targets. It needs to be discussed whether "download to absolute location" or "flash programming" should really be extensions of the Target Management system, or handled by a debugger.

Upload

Actor: Developer

Summary: Upload file(s) from a target to the development host.

Description: Select file(s) from a target file system, select a location in the host file system, and start uploading. Upload should only be allowed from file systems, and not from other parts of target memory (that is supposed to be part of a debugger).

Result: Data is uploaded from the target.

Notes: See also "Upload from Target Group" for loading data from multiple targets.

Run a remote Program

Actor: Developer

Summary: Run a program on the target.

Description: The program to run can be an entry point in target memory (either symbolic or an absolute address), or it can be a path in the target file system. A remote service adapter must contribute the specifics of running.
Run operations may be triggered directly, or they may be part of a Launch Configuration "script". Launch configuration "scripts" may be configured to perform actions like run required local programs, download data, reset the target and execute commands on the target.

Result: A program is run.

Debug a remote Program

Actor: Developer

Summary: Attach a debugger to a context on the target.

Preconditions: A debug service adapter must be registered for the target.

Description: Debug operations may be triggered directly, or they may be part of a Launch Configuration "script".
When starting the debugger directly, first get up-to-date status information from the target and present a list of valid contexts to attach (processes, tasks, cores). Allow the user to choose a context and attach the debugger. Optionally, it may be necessary to also load symbol information about the selected debug context into the debugger.
When the debugger is started as part of a Launch Configuration "script", the script should do the same as in Use Case "Run a Program" and then attach the debugger to a specified context. Optionally, the "run" script can also be empty such that the debugger attaches to a pre-specified context.

Result: The debugger opens with the context to debug on the remote target.

Reboot target

Actor: Developer

Summary: Initiate reboot on a remote target.

Preconditions: The target, or the connected lab management system, must provide a service for programmatic reboot.

Description: In case there are options for reboot, open a dialog to specify the options (JTAG debuggers, for instance, may provide either cold or warm reboot). Otherwise, initiate a plain reboot on the target.
Rebooting might lead to loading a different system image (Kernel) on the target, so it might lead to a change in the Target Configuration.

Result: A target is rebooted.

Run an (external) tool for a target

Actor: Developer

Summary: Run an (external) tool and give it parameters taken from a target definition.

Preconditions: A target (or target group) must be selected to run the tool on. An external tool must be registered so that it is know which parameters are to be passed, and how.

Description: Select a target (or target group) from the local target list and choose the tool to run from a menu.

Result: A target tool is launched.

Working with Shared Targets

There are no interrelations between the various use-cases for shared targets. Also, the main actor is always a developer or lab administrator. We therefore do not show an extra use case diagram for the use-cases dealing shared targets.

Connect to Target Registry

Actor: Developer, Lab Administrator

Summary: Connect to a target registry, such that information about available shared targets can be retrieved.

Summary: contact the remote target registry, and enter login information to authenticate (if required by the registry service).

Result: A target registry is added, and target discovery is enabled for that registry.

Add a Shared Target

Actor: Lab Administrator

Summary: Like adding a local target, but enter more static information about the target that can be used for target discovery. More information may be added automatically later through target inventory.
Also, set up initial authorizations for the lab target to specify who is allowed to use the shared target.
Adding a shared target might be done in two steps: first add a local target, then "export" it into the target registry.
When the Shared Target Registry system supports this feature, it should notify connected clients that a target has been added.

Result: A shared target definition is added. Connected clients are notified.

Modify Shared Target Definition

Actor: Power User, Lab Administrator

Summary: Modify properties of a lab target, e.g. when hardware was changed or it was decided that access permissions for a lab target should change.
Different roles of users may be allowed to make different modifications, e.g. some users may only change access permissions and notes while others may change actual hardware information.
Other modifications of shared target definition may include marking the target as broken or unavailable.
When the Shared Target Registry system supports this feature, it should notify connected clients on change of a target definition they have referenced in their local target list.

Result: Target definition is modified in the target registry. Connected clients are notified.

Remove Shared Target Definition

Actor: Lab Administrator

Summary: Remove a Target Definition from the Target Registry (because the target is removed from the lab).

Result: Target definition is removed from the target registry. Connected clients are notified.

Revoke Access Permissions

Actor: Lab Administrator

Summary: Revoke access permissions for a user or group of users on all the targets in a lab.

Result: Access Permissions are revoked.

Lab Target Inventory

Actor: Lab Administrator

Summary: Like Target Inventory for local targets. Probably allow a batch mode to do periodic checks of target availability and services provided.

Result: Static target information is updated.

Search for a Target

Actor: Developer

Summary: Search all connected target registries for a target description that matches given search criteria.

Description: Some search criteria may be fixed (like CPU type, memory size and reserved/unreserved status), while others may depend on the target registry types available. Target Registries can define what search criteria they allow.
The user is presented a mask to enter search criteria. Then, the search is submitted to present a list of targets matching the given criteria. The list of matches may be sorted by criteria like CPU utilization or memory size.

Result: A list of matching targets is presented, and the user may choose to add selected elements of the match list to the local target list. At the time a target is added to the local target list, it may be reserved automatically.

Reserve a Target

Actor: Developer

Summary: Mark a shared target as reserved by the local user.

Result: The target is marked reserved. Depending on target registry policies, the target may be unreserved automatically after some time of inactivity.

Unreserve a Target

Actor: Developer, Power User, Lab Administrator

Summary: Select a target from the local target list and mark it unreserved. Power Users and Lab administrators may also unreserve targets that are reserved by users other than themselves.

Result: The target is marked unreserved.

Power Cycle / Reboot a Lab Target

Actor: Developer

Summary: Like rebooting a local target, but done through the lab management system.

Result: The target is rebooted / power cycled.

Send a message to target users

Actor: Developer, Power User, Lab Administrator

Summary: Send a message to the persons that have a target (or a list of targets) reserved.

Description: The message can ask users to free the target, or tell them that the system is about to be taken offline. Messages can be sent through the remote registry service, or sent as E-Mail.

Result: A message is sent.

Working with Target Groups

Target groups are administrative entities that can be reserved, unreserved or used for download and upload as a whole. The use cases are similar to working with single targets:

- Create Target Group
- Add Target(s) to Group
- Remove Target(s) from Group
- Reserve Target Group
- Unreserve Target Group
- Load System on Target Group
- Upload data from Target Group

Software Use Cases

Software Use Cases are initiated by extensions of the Target Management framework. They include registration of the various services that are to be added to the framework by extensions.

- Register Target Registry Adapter
- Register Target Connection Type
- Register Remote Service Adapter
- Register Target Tool or Action

Sample Data Structures

This section lists some examples of data structures. In a real world system, not all attributes will be filled out, or some additional attributes may be needed. The sample is just here to clarify what could be in the various data structures.

Sample Static Target Definition

The static target definition is mostly handled by the external target registry or lab management system. The external system determines what attributes are stored, and which of them are searchable for target discovery. The target registry adaptor determines how available attributes are integrated with further target management services like reserve/unreserved, reboot, console access etc.

Attribute	Notes
Target Number (key)	Unique identifier of the target, e.g. a barcode
Target Name	
Location	Physical location of the target
Owner	Contact information in case of problems
Zone	Reference to administrative zone (group of targets) holding permission info
CPU Family (architecture)	Only the main CPU in case of a multi core system. It should be possible to match the target CPU architecture against compiler switches for building software for that target.
CPU Type	
BSP	Board Support Package, i.e. name of the hardware system e.g. “Sun Ultra 4-MP”, “Motorola mv5300”
Number of CPUs	
CPU clock speed	Physical clock speed of the main CPU
Performance index	Performance number, and name of the tool to measure the index
Memory Size	
Peripherals connected	List of important peripherals connected, e.g. video cards, …
Installed Software	List of software that is installed “fixed” on the target, i.e. in system ROM or Flash like a boot loader, ROM monitor or even target OS if it is known that the OS cannot change. The software list need not be complete, only as relevant for target connections.
Reserved by	User id of person reserving the target, or empty if unreserved
Reserved since	Date + Time
Reserved until	Date + Time
Network Interfaces	List of network interfaces, each holding • Device name • MAC Address • IP Address suggested by the lab admin to avoid conflicts • Network switch port that the interface is connected to
Gateway	IP address of gateway in case the target is in a separate subnet
Hardware Services	List of hardware services connected to the target, e.g. • Serial console connected to terminal server host: port • JTAG debugger connected to IP address: port

	• Trace tool connected
Comments	Free-form comments for the target

Sample Target Configuration

Target	Reference to the Static Target Definition for this configuration
System Image	Information about the kernel (system) image. May contain an absolute network file system path to the image, version control and release information, build time. Several variants may be supplied (with and without debug info for instance).
Software Services	List of services available in the booted target system, e.g. telnetd, ftpd, sshd, gdbserver, TPTP agent. Each of the services holds property information as needed, e.g. the port where it is listening.
Mounted file systems	List of file systems mounted from remote hosts, each with remote host identification and local mount point.
Comments	Free-form comments for the target configuration

Questions

- Do we need more special handling for multi-core systems, i.e. data structures to identify each core individually?
- Do we need framework code for dealing with hardware debugging related things like scan chain setup?
- The Eclipse Communication Framework (ECF) has some infrastructure for dealing with communications over multiple adaptable protocols as well as discovery of communication services. There might be some overlap. This needs to be checked.

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 Systems Explorer (RSE), http://www.developer.ibm.com/isv/rational/remote_system_explorer.html