

## **Slim Install Project Transfer Module Functional Specification**

<b>Author</b>	<b>Date</b>	<b>Coments</b>
Moinak Ghosh	Sept 24 2007	Initial Draft

## Table of Contents

1 Requirements of Transfer Module.....	3
2 Transfer Module Tasks.....	3
3 Implementation.....	4
4 Platform Dependencies.....	4
5 Interfaces.....	4
6 Dependencies.....	5

## 1 Requirements of Transfer Module

The purpose of the Transfer Module is to lay out the OS bits onto the disk. In the scope of the Slim Install project this essentially means that the contents of the LiveCD are transferred onto the harddisk and some basic post-install configuration is done.

The actual partition and filesystem creation would already have been done by the Target Instantiation service. That service is also responsible for mounting the new filesystem for use by the Transfer Module.

The following requirements should be satisfied:

- The Transfer Module copies the OS image on the LiveCD in it's entirety. This forms the base installed system.
- Process of transferring should be interruptible without leaving the system in an unusable state
- The transfer process should provide feedback on the progress to assist the GUI in providing a progress bar
- The Transfer Module expects a mounted filesystem and the locale name to be passed in from the Orchestrator. The locale setup however is not intended for the initial prototype release.
- For the Slim Install initial delivery only a fresh install scenario is being considered at this time. Upgrade is left as a subsequent activity. However based on the approach of a LiveCD based installer, upgrades are typically handled via the packaging system and not directly via the LiveCD. So for the purposes of the Transfer Module upgrades are out of scope.

## 2 Transfer Module Tasks

The steps taken by this module can be broken down into the following tasks:

1. Copy the LiveCD contents to disk. This is essentially a straight cpio of the LiveCD contents from it's mounted location. The files of course get decompressed as they are read.
2. Copy some basic settings from the LiveCD environment to disk. These include Keyboard selection, Locale selection (on a multi-locale CD/DVD), Xorg config file if any, the etc/path\_to\_inst file, the SMF seed repository, the etc/devices/devic\_cache
3. Perform some specific actions needed due to the way the LiveCD is prepared. The contents of the archive file must be extracted.
4. Indicate a reconfiguration reboot by creating the file /reconfigure .

Additionally there are a bunch of post-install tasks that need to be done after the Transfer Module has finished. A tentative list is:

1. Install GRUB to MBR
2. Create Boot Archive

3. Setup root password and additional user
4. Mark Solaris Partition as Active
5. Update bootpath in bootenv.rc
6. Set hostname
7. Set date/time/timezone

These are either handled by the Orchestrator or via a postinstall service. It is important to note here that these final steps must happen only after the Transfer Module has completed. Marking the Solaris partition as active as the last step ensures that we do not make the system unbootable if install is aborted midway.

### 3 Implementation

The Transfer Module can be implemented as either a standalone binary or as a library. Keeping in view the current approach being following in Dwarf Caiman a shared library interface seems appropriate. The interaction of the Orchestrator with this module is quite simple. There will be a single exported API. The return code from the API call indicates success or failure. In case of a failure the logfile will contain relevant error messages.

Apart from getting the Alternate Root dir and locale name from Orchestrator the Transfer module also needs to get hold of the keyboard layout. This can be extracted from the Keyboard layout setup present in the LiveCD environment. This is a question that is asked while the LiveCD is booting up.

The action of transferring involves copying files from the LiveCD. Since the LiveCD is already in an installed state, this state is transferred to the harddisk. No additional package additions are performed, with the exception of locale setup. However as mentioned before setting up additional locales is not intended for the initial prototype release. One or more cpio operations are used to copy the files along with some additional steps as mentioned above. The cpio process is run with -V flag to get a feedback for every file copied. Since doing a cpio also requires running a find, the output of the find command is used to determine the total number of files to be copied before passing on to cpio.

### 4 Platform Dependencies

For the initial prototype delivery of Slim Install is intended to only support Intel/AMD platforms. Support for SPARC is planned for the March 2008 delivery of Slim Install. The Transfer module is also only relevant for initial install from the Slim Install LiveCD environment.

### 5 Interfaces

The following interfaces are exported:

int

TM\_perform\_transfer(nvlist\_t \*, callb\_t)

This performs the actual operation of copying the LiveCD to harddisk. This is a blocking call and returns a result code. Only one transfer operation can be active at any given time. This will be ensured via a global lock.

- Return values: TM\_SUCCESS or TM\_FAILURE
- `nvlist_t *` - A list of name-value pairs used to pass attributes like name of the alternate root directory and name of the locale.
- `callb_t` - A structure containing a pointer to the progress callback function and pointer to additional data to be passed to the function. The callback function is expected to accept a percent completion status and a message indicating the current action being performed.

void

TM\_abort\_operation(void)

This function will cause any currently executing transfer process to end. This is also a blocking call but returns quickly.

Apart from the above a additional project private environment variable is supported: TM\_DEBUG. This indicates a verbose debug level and is meant for debugging purposes.

## 6 Dependencies

The Transfer Module does not have any external dependencies apart from using cli utilities like `cpio` and `find`.