

open



USE



IMPROVE



EVANGELIZE

Common installer modules

Jan Damborsky

開
放
的
열린
مفتوح
libre
मुक्त
ಮುಕ್ತ
livre
libero
ముక్త
开放的
açık
open
nyílt
:::
πικρ
オープン
livre
ανοικτό
offen
otevřený
öppen
открытый
வெளிப்படை



Common modules used by installer consumers

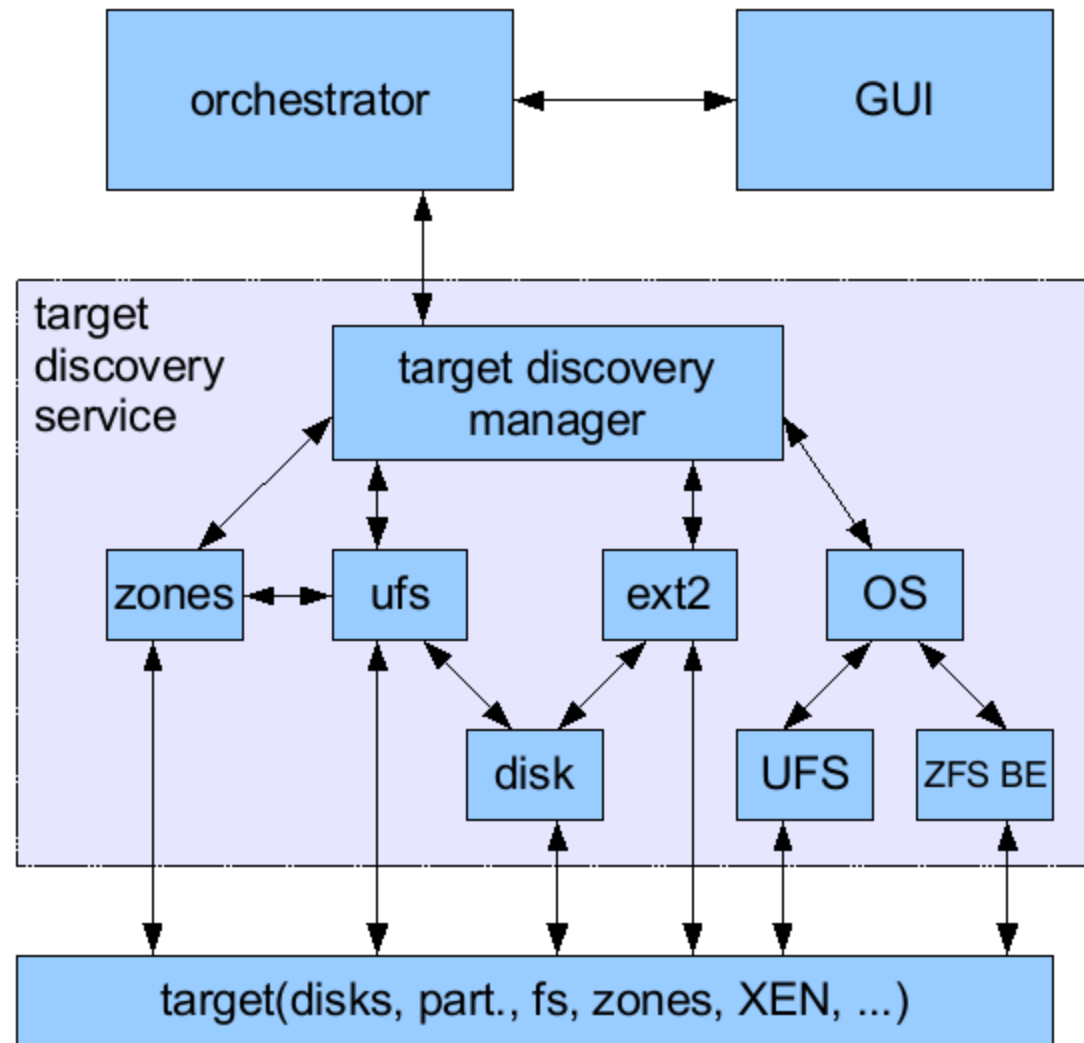
- Consumers
 - GUI installer
 - Automated Installer
 - Distribution Constructor
- Modules
 - Target Discovery
 - Target Instantiation
 - Transfer module
 - Orchestrator
 - ICT (Installation Completion Tasks)



Target Discovery - purpose

- Collects information about targets
- Information collected
 - Disk configuration - TD_OT_DISK
 - FDISK partition configuration (Linux swap is recognized) - TD_OT_PARTITION
 - UFS slices - TD_OT_SLICE
 - Solaris instances on UFS filesystem - TD_OT_OS

Target Discovery - structure





Target Discovery – disk discovery

- Disk name in ctd format (e.g. c0t0d0)
- Vendor name (e.g. FUJITSU)
 - Set to “unknown” for ATA drivers
- Controller type (SCSI, ATA, USB)
 - SATA not recognized for now (bug 6558646)
- Block size (now 512 bytes)
- Number of blocks (total size)
- Label type (FDISK, VTOC, EFI/GPT)
 - EFI not correctly recognized for x86 for now
- ...



Target Discovery – partition discovery

- Device name in ctdp format (e.g. c1t0d0p1)
- 'active' flag
- Partition ID (e.g. Solaris2 has 0xbf)
- Geometry - 1st sector and size in sectors
- Content type – only Linux swap is recognized for now
 - Check for Linux swap magic is done (“SWAP-SPACE” or “SWAPSPACE” string at the end of 1st page)



Target Discovery – slice discovery

- Device name in ctds format (e.g. c1t0d0s3)
- Slice number (e.g. 7 for c1t0d0s7)
- Geometry
 - 1st sector - relative to start of disk (Sparc) or Solaris partition (x86)
 - size in sectors
- Tag, flag
- ...



Target Discovery – OS discovery

- Only Solaris instances on UFS recognized for now
- “/” slice in ctds format (e.g. c0t0d0s0)
- Version (e.g. Solaris_11)
 - Taken from `/var/sadm/system/admin/INST_RELEASE`
- Check if Solaris instance is upgradeable
- ...



Target Discovery - implementation

- Standard C library – libtd + td_api.h
- Location in source tree
 - usr/src/lib/libtd/
- Modularized design
 - Manager module – td_mg.c
 - Disk module – td_dd.c
 - BE module - td_be.c
- Uses libdiskmgt library for most discovery tasks



Target Discovery - API

- Uses libnvpair(3LIB) for passing/obtaining data to/from caller
 - Data are passed as name-value pairs
 - Mitigates problems with compatibility
 - Simplifies implementation of enhancements
 - Allows defining complex data structures
- Two step process
 - Discovery – `td_discover(td_object_type_t, int *)`
 - Enumeration through the list of discovered objects and obtaining the data describing discovered objects
 - `td_get_next(td_object_type_t)`
 - `nvlist_t *td_attributes_get(td_object_type_t)`



Target Discovery example – disk discovery

```
# include <td_api.h>
# include <libnvpair.h>

td_errno_t  ret;
int         num;
nvlist_t   *attr_list;
char       *disk_vendor;

ret = td_discover(TD_OT_DISK, &num);
if (ret != TD_E_SUCCESS)
    /* something went wrong */

/* go through the list of disks and process attributes */
for (i = 0; i < num; i++) {
    ret = td_get_next(TD_OT_DISK);

    if (ret != TD_E_SUCCESS)
        /* something went wrong */

    attr_list = td_attributes_get(TD_OT_DISK);
    if (nvlist_lookup_string(attr_list, TD_DISK_ATTR_VENDOR, &disk_vendor) == 0)
        printf("Disk vendor: %s\n", disk_vendor);

    nvlist_free(attr_list);
}
```



Target Discovery - TODO

- Support for extended partitions
- Discovery of Snap Upgrade Boot Environments
- Implementing re-discovery feature
 - For now in order to refresh data structures when new disk is connected, installer has to be restarted



Target Discovery - dependencies

- libdiskmgt
- libnvpair(3LIB)



Target Discovery - debugging

- Using standard tools to compare collected data
 - Disk discovery – `iostat(1M)`, `format(1M)`, `rmformat(1)`, `cfgadm(1M)`
 - `# iostat -En`
 - `# cfgadm -al`
 - Slice discovery - `prtvtoc(1M)`
 - `# prtvtoc /dev/rdisk/c0t0d0s2`
 - Partition discovery – `fdisk(1M)`
 - `# fdisk -W - c0t0d0p`



Target Discovery - debugging

- Test drivers `test_td.c` & `tdmgtst.c`
 - They need to be built separately (bug 5644)
 - `test_td` prints formatted output (not OS discovery)
 - `tdmgtst` prints all provided attributes
- Disk discovery
 - `# test_td -dv`
- Partition discovery
 - `# test_td -v -p all`
- Slice discovery
 - `# test_td -v -s all`



Target Discovery - debugging

- OS discovery
 - # tdmgtst -d
- Obtain more detailed output
 - # export LS_DEST=3
 - # export LS_DBG_LVL=4
 - # test_td -dv



Target Discovery - resources

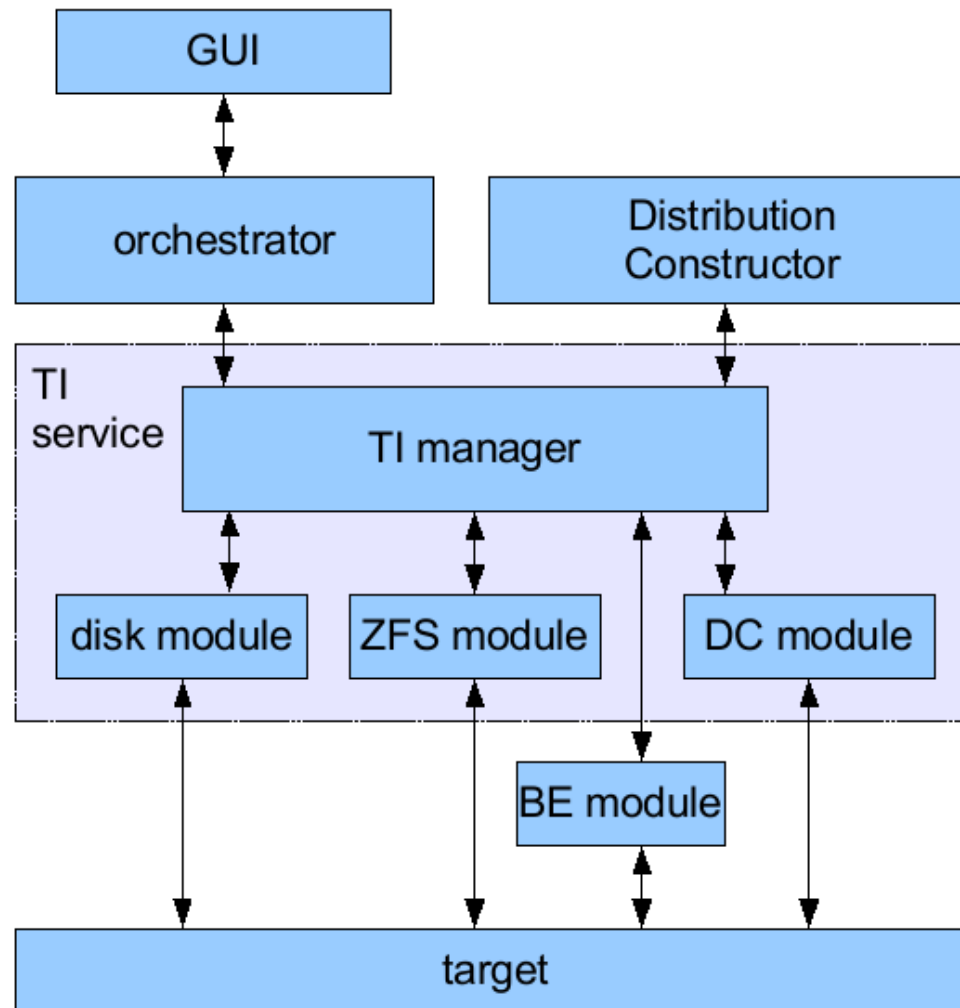
- Caiman project pages
 - <http://www.opensolaris.org/os/project/caiman/>
- Source code
 - hg clone `ssh://anon@hg.opensolaris.org/hg/caiman/slim_source`
 - http://src.opensolaris.org/source/xref/caiman/slim_source/usr/src/lib/libtd/
- Documentation
 - hg clone `ssh://anon@hg.opensolaris.org/hg/caiman/caiman-docs`
- Filing bugs
 - use bugzilla, not bugster - defect.opensolaris.org
 - category Development/installer/library
- Help
 - IRC channel - `#caiman-discuss` at chat.freenode.net
 - Mailing list - caiman-discuss@opensolaris.org



Target Instantiation - purpose

- Prepares target (disk(s), DC proto area, ...) for other modules (installer, Distro Constructor)
- Fully oriented on ZFS root – no support for UFS
- Allows creating of following targets
 - FDISK partition table
 - Solaris VTOC
 - ZFS pool, datasets, volumes
 - Boot Environment
 - Distribution Constructor boot archive area

Target Instantiation - structure





Target instantiation - features

- **Modularized**
 - Manager module
 - Disk module
 - ZFS module
 - BE module
 - Distro Constructor module
- **Stateless**
 - doesn't keep track of created targets
- **Supports both target creating and releasing**



Target Instantiation - implementation

- Standard C library – libti.so.1 + ti_api.h
 - usr/src/lib/libti/
- Wrapper for Python consumers exists
 - usr/src/lib/libti_pymod/libti.c
- Modularized design
 - Manager module – ti_mg.c
 - Disk module – ti_dm.c
 - ZFS module – ti_zfm.c
 - BE module – ti_bem.c
 - Distribution constructor module - ti_dcm.c



Target Instantiation – Disk module targets

- **FDISK partition table**
 - fdisk(1M) command with -B or -F options is consumed for this task
 - any partition configuration can be created
- **Solaris VTOC**
 - read_extvtoc(), write_extvtoc() from libadm(3LIB) are consumed
 - simple sanity checking is done (overlapping)
 - any slice configuration can be created
- **TODO**
 - Creating label on unlabeled Sparc disk (bug 3136)



Target Instantiation – ZFS module targets

- ZFS root pool
 - zpool(1M) command is consumed
 - For root pool, additional steps are carried out
 - # /usr/bin/mkdir <target_mntpoint>/<pool_name>/boot/grub
 - # /usr/sbin/zfs set org.opensolaris.caiman:install=busy
- ZFS filesystems
- ZFS volumes
 - Dedicated volume for swap and dump can be created



Target Instantiation – BE module targets

- Boot Environment
 - libbe Snap Upgrade library is utilized
 - allows creating BE in specified ZFS pool
 - ZFS filesystems to be created can be specified
 - mounts BE on provide mountpoint



Target Instantiation – DC module targets

- Distribution Constructor 'proto' area
 - Area used for populating with 'pkg install' command
 - It is ZFS filesystem - ZFS module is used for this
- RAMDISK for creating boot archive
 - # mkfile <size> <boot_archive>
 - # lofiadm -a <boot_archive>
 - # newfs <boot_archive>
 - # mount -o nologging <boot_archive> <mount_point>



Target Instantiation - API

- Uses libnvpair(3LIB) for passing/obtaining data to/from caller
 - Data are passed as name-value pairs
 - Mitigates problems with compatibility
 - Simplifies implementation of enhancements
- Allows creating or releasing target
 - Creating – `ti_create_target(nvlist_t *, ti_cbf_t)`
 - Releasing – `ti_release_target(nvlist_t *, ti_cbf_t)`
- Check, if target already exists
 - `boolean_t ti_target_exists(nvlist_t *)`

Target Instantiation example – creating Solaris fdisk partition using whole disk

```
# include <ti_api.h>
# include <libnvpair.h>

nvlist_t    *target_attributes;
char        *disk_name = "c0t0d0";

if (nvlist_alloc(&target_attributes, TI_TARGET_NVLIST_TYPE, 0) != 0)
    /* couldn't create nvlist for holding target attributes */

/* set target type to be created - FDISK */
nvlist_add_uint32(target_attributes, TI_ATTR_TARGET_TYPE, TI_TARGET_TYPE_FDISK);

/* add name of target disk */
nvlist_add_string(target_attributes, TI_ATTR_FDISK_DISK_NAME, disk_name);

/* create Solaris2 partition occupying whole disk */
nvlist_add_boolean_value(target_attributes, TI_ATTR_FDISK_WDISK_FL, B_TRUE);

if (ti_create_target(target_attributes, NULL) != TI_E_SUCCESS)
    /* couldn't create target described by provided list of attributes */

/* release previously created nvlist - it is no longer in use */
nvlist_free(target_attributes);
```



Target Instantiation - TODO

- Support for extended partitions
- Support for EFI/GPT labeled disks
- Fixing existing bugs



Target Instantiation - dependencies

- libadm(3LIB)
 - read_extvtoc(), write_extvtoc()
- fdisk(1M)
- zpool(1M), zfs(1M)
- libbe

Target Instantiation - debugging

- Test driver `test_ti.c`
 - needs to be built separately (bug 5644)
 - Allows 'dry run' – if not provided with '-c' option
- Create FDISK partition on whole disk `c0d0`
 - `# test_ti -c -t f -w -d c0d0`
- Create customized FDISK partition table on disk `c0t0d0`
 - `# test_ti -c -t f -f <configuration_file> -d c0t0d0`
 - Format of `<configuration_file>` is the same as produced by 'fdisk -W' or accepted by 'fdisk -F'



Target Instantiation - debugging

- Create default VTOC on disk c0d0
 - Slice 0 occupies all disk
 - Slices 2 and 8 (x86 only) are created
 - `# test_ti -c -t v -d c0d0`
- Create customized VTOC on disk c0t0d0
 - `# test_ti -c -t v -f <configuration_file> -d c0t0d0`
 - Format of `<configuration_file>` is the same as produced by `prtvtoc(1M)` or accepted by `fmthard(1M)`

Target Instantiation - debugging

- Create ZFS root pool on slice c0d0s0
 - # test_ti -c -t p -p rpool -d c0d0s0
- Release ZFS root pool
 - # test_ti -c -t P -p rpool -d c0d0s0
- Create 2GB swap on ZFS volume
 - # test_ti -c -t l -p rpool -z 2048 -n swap -u 1
- Create Boot Environment
 - # test_ti -c -t b -p rpool -n opensolaris
- Running in verbose mode
 - # export LS_DEST=3
 - # export LS_DBG_LVL=4



Target Instantiation - resources

- Source code

- hg clone `ssh://anon@hg.opensolaris.org/hg/caiman/slim_source`
- `usr/src/lib/libti` directory
- `http://src.opensolaris.org/source/xref/caiman/slim_source/usr/src/lib/libti/`

- Documentation

- hg clone `ssh://anon@hg.opensolaris.org/hg/caiman/caiman-docs`
- `target_instantiation` directory

- Filing bugs

- use bugzilla, not bugster -
`defect.opensolaris.org`
- category `Development/installer/library`



Transfer module - purpose

- Layout bits on the target
 - GUI installer – copy LiveCD content to the disk
 - Automated Installer – install from IPS repositories
 - Distribution Constructor – install from IPS repositories



Transfer module - implementation

- Python script – `transfermod.py`
- C library wrapper for consumers written in C language – `libtransfer.so.1` + `transfermod.h`
- Location in source tree
 - `usr/src/lib/libtransfer/` - Python implementation
 - `usr/src/lib/libtransfer_pymod/` - C wrapper



Transfer Module - API

- Python API
 - `tm_perform_transfer(args, callback = None)`
 - arguments passed as list of name-value pairs
 - Internally two Python classes defined
 - `Transfer_cpio()` - CPIO transfer method
 - `Transfer_ips()` - IPS transfer method
- C API
 - Uses `libnvpair(3LIB)` for passing/obtaining data to/from caller
 - `TM_perform_transfer(nvlist_t *nvl, tm_callback_t prog)`



Transfer Module – CPIO method

- Copies bits from 'source' to 'target' using `cpio(1)`
- Two modes implemented
 - All bits are transferred –
`TM_CPIO_ACTION=TM_CPIO_ENTIRE`
 - Only selected files are transferred -
`TM_CPIO_ACTION=TM_CPIO_LIST`
- Accepts following parameters
 - Source directory - `TM_CPIO_SRC_MNTPT`
 - Destination directory - `TM_CPIO_DST_MNTPT`
 - List of files to be transferred – `TM_CPIO_LIST_FILE`



Transfer Module – IPS method

- Installs IPS packages from IPS repository to 'target' using pkg(1)
- Two step process
 - Creating IPS 'image' – populates target with required IPS metadata – TM_IPS_INIT
 - # pkg image-create -F -a opensolaris.org=<http://pkg.opensolaris.org> /a
 - Requires IPS authority name, IPS repository location, target mountpoint
 - Installing packages to the target – TM_IPS_RETRIEVE
 - # pkg -R /a install slim_install
 - Requires target mountpoint, list of packages to be installed



Transfer Module – IPS method

- Other IPS operations supported
 - verifying content of IPS repository –
TM_IPS_REPO_CONTENTS_VERIFY
 - setting IPS authority – TM_IPS_SET_AUTH
 - unsetting IPS authority – TM_IPS_UNSET_AUTH
 - refreshing IPS catalogs – TM_IPS_REFRESH
 - uninstalling packages – TM_IPS_UNINSTALL



Transfer Module example 1 – copy liveCD content to the target

```
# include <transfermod.h>
# include <libnvpair.h>

nvlist_t    *transfer_attributes;

if (nvlist_alloc(&transfer_attributes, NV_UNIQUE_NAME, 0) != 0)
    /* couldn't create nvlist for holding transfer attributes */

/* transfer method to be used - CPIO */
nvlist_add_uint32(transfer_attributes, TM_ATTR_MECHANISM, TM_PERFORM_CPIO);

/* copy entire content */
nvlist_add_uint32(transfer_attributes, TM_CPIO_ACTION, TM_CPIO_ENTIRE);

/* copy from "/" to target mountpoint - "/a" in our case */
nvlist_add_string(transfer_attributes, TM_CPIO_SRC_MNTPT, "/");
nvlist_add_string(transfer_attributes, TM_CPIO_DST_MNTPT, "/a");

/* start transfer process */
if (TM_perform_transfer(transfer_attributes, NULL) != TM_SUCCESS)
    /* transfer process failed */

/* release previously created nvlist - it is no longer in use */
nvlist_free(transfer_attributes);
```



Transfer Module example 2 – installing packages on the target from IPS repository

```
from osol_install.transfer_mod import *

# step 1 - prepare IPS target - create IPS metadata
status = tm_perform_transfer([(TM_ATTR_MECHANISM, TM_PERFORM_IPS),
    (TM_IPS_ACTION, TM_IPS_INIT),
    (TM_IPS_PKG_URL, "http://pkg.opensolaris.org"),
    (TM_IPS_PKG_AUTH, "opensolaris.org"),
    (TM_IPS_INIT_MNTPT, "/a")])

if status:
    # something went wrong

# step 2 - install list of packages in package_list file on the target
status = tm_perform_transfer([(TM_ATTR_MECHANISM, TM_PERFORM_IPS),
    (TM_IPS_ACTION, TM_IPS_RETRIEVE),
    (TM_IPS_PKGS, "package_list"),
    (TM_IPS_INIT_MNTPT, "/a")])

if status:
    # something went wrong
```



Transfer module - dependencies

- cpio(1)
- pkg(1)



Transfer module - resources

- Source code
 - `hg clone ssh://anon@hg.opensolaris.org/hg/caiman/slim_source`
 - `usr/src/lib/libtransfer, usr/src/lib/libtransfer_pymod`
- Documentation
 - <http://www.opensolaris.org/os/project/caiman/files/transfermod.pdf>
- Filing bugs
 - use bugzilla, not bugster -
`defect.opensolaris.org`
 - category `Development/installer/library`



Orchestrator - purpose

- Glue which interconnects low level modules with installer (GUI, AI)
- Controls flow of installation process
 - Selects the right installation path according to the installation type
- Provides feedback to the main application about installation progress
- Provides common wrapper for
 - Target Discovery module
 - Transfer module
 - ICT



Orchestrator - purpose

- Calculates necessary disk size for the installation
- LiveCD
 - Taken from `/.image_info` file generated by Distribution Constructor – size of proto area in kilobytes
 - `IMAGE_SIZE=2048000`
- AI – hardcoded (bug 4546)
 - Minimum 4GB
 - Recommended 12GB



Orchestrator - purpose

- Calculates sizes for swap and dump devices
 - Those are created on ZFS volumes
 - $\text{swap_size} = \text{memory_size} / 2$ – min 512MB, max 64GB
 - $\text{dump_size} = \text{memory_size} / 2$ – min 512MB, max 32GB
 - Swap is created on UFS slice 1 for systems with low memory (less than 768 MB)



Orchestrator - implementation

- Standard C library – liborchestrator.so.1 + orchestrator_api.h
 - usr/src/lib/liborchestrator/



Orchestrator - API

- `om_perform_install(nvlist_t *uchoices, om_callback_t cb)`
 - Main entry point – controls flow of installation process
 - Now support for GUI installation and Automated Installation is implemented
 - AI is selected if `/.autoinstall` file is present (generated by Distribution Constructor when AI image is to be created)
 - `uchoices` – contains information about user choices
 - Target selected for the installation
 - Timezone
 - Default locale
 - Root password
 - User account – user name, login name, password
 - hostname



Orchestrator - dependencies

- Install libraries
 - Target Discovery – libtd
 - Target Instatiation – libti
 - Transfer Module – libtransfer
 - ICT – libict



Orchestrator - debugging

- Test drivers
 - needs to be built separately
 - they are not up to date – need to be accommodated for OpenSolaris (bug 3112)
 - disktest – tests target discovery
 - kbdtest – test of locale stuff
 - dummy_install – test of the installation



Orchestrator - resources

- Source code

- hg clone ssh://anon@hg.opensolaris.org/hg/caiman/slim_source
- usr/src/lib/liborchestrator

- Documentation

- http://www.opensolaris.org/os/project/caiman/files/dwarf_caiman_design.pdf

- Filing bugs

- use bugzilla, not bugster -
defect.opensolaris.org
- category Development/installer/library

open



USE



IMPROVE



EVANGELIZE

Thank you!

Jan Damborsky

“open” artwork and icons by chandan:
<http://blogs.sun.com/chandan>

開
放
的
열린
مفتوح
libre
मुक्त
ಮುಕ್ತ
livre
libero
ముక్త
开放的
açık
open
nyílt
ᄒᆞᆫ
オープン
livre
ανοικτό
offen
otevřený
öppen
открытый
வெளிப்படை