

# IGNITE-UX COLD INSTALLATIONS



Steve Bennett, Hewlett-Packard Company  
3404 East Harmony Road MS 99:2UR9  
Fort Collins, CO 80525  
970-229-6679  
bennett@fc.hp.com

Ignite-UX is a new tool supplied with HP-UX to do initial configuration of HP-UX systems. It replaces the existing **HP-UX Install** program (also known as **Cold Install**) as well as the **CHAMP** program (which was used by manufacturers to create HP-UX systems). Ignite-UX supports all HP-UX releases starting with 10.01. It is provided as a bundled (free) product on the DART applications media which is currently scheduled to be released in mid-July of 1997. It is also available via the web at <http://www.software.hp.com>.

Ignite-UX provides several advantages over the existing Cold Install product:

- True Client/Server Model  
The install sessions for multiple targets can be controlled from a single server. A new user interface is provided to run on the server and manage multiple simultaneous install sessions. Alternatively, a single install session can be controlled from the target machine if that is more convenient for the user. A single install server can serve multiple HP-UX releases. The install server itself can be running any HP-UX release, 10.01 or later.
- Enhanced User Interface  
The user interface has been modernized and now has a Windows95-like “tabbed dialog” feel. This allows the tool to present more configuration capabilities without overwhelming the casual user. In addition, a wizard mode is available for the novice user.
- Support for Multiple Software Sources  
Loads can occur from multiple software sources in a single install session. For example, you could install your base OS from one SD depot, a set of patches from another depot, and the applications you want from a third depot – all in one session.
- Archive Installation

In addition to the continued support of SD software sources, Ignite-UX also supports tar and cpio archives. Tools are provided to help you create a “golden system image” if you wish to install from an archive.

- **Easy Customizability**  
It is easy to create a system that is ready to go as soon as the install session completes. Many of the tasks which are typically done as separate steps after an install have been incorporated into the installation process. Ignite-UX allows you to specify what kernel parameters you want set and what user-supplied scripts you would like to run as part of the session. Many different script hooks are provided so you can do your own customizations (during the installation and after it). Also, the host and networking information which must normally be supplied at first boot can alternatively be specified at install-time.
- **Save Customized Configurations for Future Use**  
It is possible to create a configuration for your particular needs, save it away, and then quickly apply that configuration to multiple install targets.
- **Non-interactive Installs**  
Ignite-UX allows you to set up a configuration and then install it on a target machine with no further user interaction. This can be done in both the initial installation and the re-installation cases.
- **Generate a Manifest**  
Ignite-UX provides a tool to scan a system and produce a report detailing what hardware is present, how the disks are used, what kernel modifications have been made, and what software has been loaded. This report can be customized to meet your needs.

---

## BASIC USE MODELS

---

Ignite-UX is a very flexible toolset and can be used to solve many different customer problems. This section describes some of the primary customer use models and how Ignite-UX handles them.

### COLD INSTALLATION FROM MEDIA

The most basic use model is the installation of a single new machine from some sort of media. No server needs to be involved in this scenario; everything which is needed is contained on the media. The user controls the install session via a terminal user interface running on the target machine’s console. This use model was supported by Cold Install and is also supported by Ignite-UX. The only real difference between Cold Install and Ignite-UX in this case is the user interface. The user is given the choice of either a terminal version of the tabbed dialog or the novice task wizard interface.

### COLD INSTALLATION OVER THE NETWORK FROM TARGET

Cold Install allowed you to set up a server machine, do a network boot on the target machine, and control the installation from the target machine via a terminal user interface. Ignite-UX has this capability as well. There are also some hybrids that are possible. For example, you can boot the install kernel from media on the target machine, enable networking, and then install software from a network depot which was set up previously.

## COLD INSTALLATION OVER THE NETWORK FROM SERVER

It is often convenient to control/monitor remote installations from a central point. This is especially true if you are doing multiple installs simultaneously. After you do a network boot on the target machine, all further interactions can take place on the server. Information about each installation is kept on the server in a per-target directory identified by the link-level address of the target machine. This data allows you to re-install the same target with the same configuration at a later time if that should prove necessary.

Another advantage of this approach is that you can use the graphical user interface instead of the terminal user interface to perform the installation. A great deal of emphasis has been placed on making the graphical user interface both powerful and easy to use. Our methodology has included rapid prototyping with continuous customer reviews and formal usability testing led by our Human Factors engineer.

You may also chose to run a non-interactive install. You can define the configuration you want applied ahead of time and set an option to not run the UI. In this case, the target system will start its installation process as soon as it contacts the server. You are able to monitor the target's process from the server, if you choose.

## RE-DEPLOY OVER THE NETWORK FROM SERVER

For a variety of reasons, it is sometimes necessary to re-install an existing HP-UX machine:

- You have a large number of machines which are running HP-UX 9.something. You have decided to make the switch to 10.something, and you have decided that a re-install is the way to do this. [Note that Joe Grim (HP) has written an Interworks paper titled *Ignite-UX Case Studies* which deals more thoroughly with this case.]
- You have a large number of similarly-configured, dataless workstations (i.e. all volatile data is on a separate file server). When a hard to diagnose or hard to fix problem occurs on one of the nodes, you re-install to save time (the phrase we often heard from customers was "Don't troubleshoot, just re-install").
- You have an existing machine in place, which needs to be set up for a different use. It will require different software or a different configuration.

The main difference between these cases and the cold installation use models is that these scenarios already have a running system. Ignite-UX can take advantage of this fact and allow re-installs without having to physically go to the target machine and boot it. The *bootsys(1m)* command is used to push an install kernel and file system to the target machine's disk, set it up to boot from the install kernel, and then reboot the system. When the target machine boots, it connects with the install server machine and you can then direct the install from the central server as explained earlier (either interactively or non-interactively).

---

## GRAPHICAL USER INTERFACE

---

This section shows a couple of screen shots from the graphical user interface to try to give you a feel for what it is like. This is a very incomplete treatment of what the UI can do - mostly just a teaser!

Figure 1 shows the basic server screen. Each of the target machines which is ready to install, in the process of being installed, or recently complete is represented by an icon. The icons are color-coded to represent basic status (ok, warning, and error). The thermometer next to each icon represents how far along the installation is. By selecting a particular target machine, you can:

- start an install (either a new install or a re-install based on a saved install session)

- see the high-level status of the install or the low-level install log
- see a detailed hardware description of the target machine
- save the install session away for future use or remove it

In addition, many of the Ignite-UX server configuration (setup) tasks can be done from this interface.

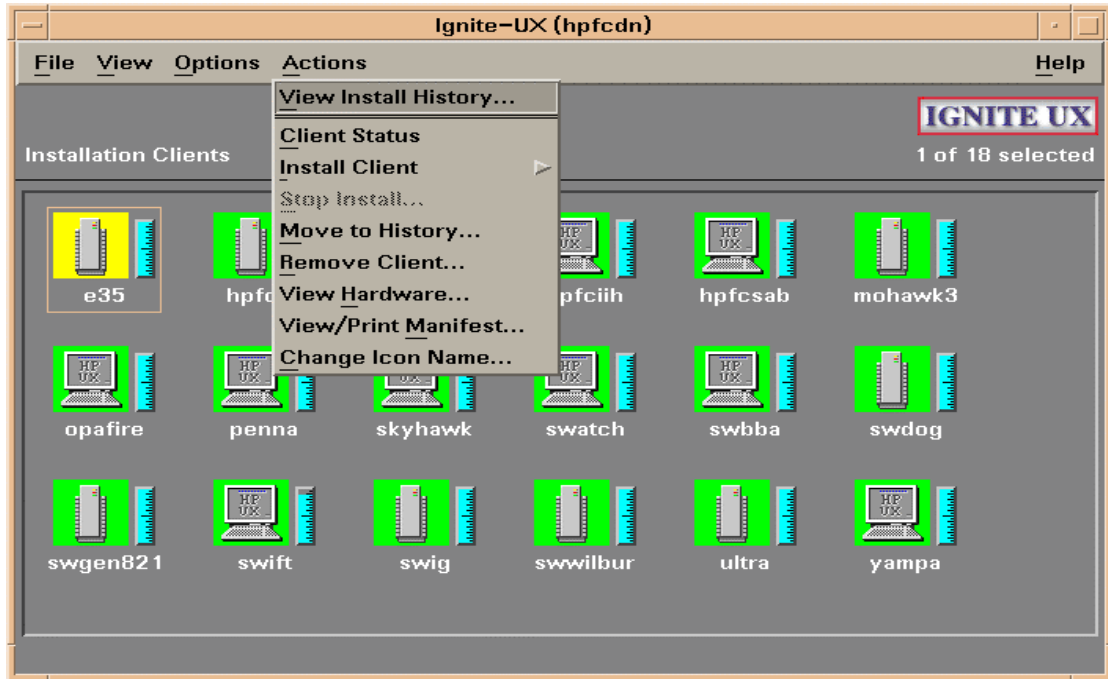


Figure 1: Ignite-UX Server

Figure 2 shows the first screen you see once you launch an install on a particular target machine. Notice that there are five tabs across the top of the screen. Selecting each tab brings up a different screen:

- Basic (shown)
 

This screen lets you select the starting configuration for your install. These configurations include the defaults shipped by HP as well as the configurations you have customized and saved via the *Save As* button in the user interface. This screen also lets you do some basic layout of your system including what disk to use for root, how much swap space to allocate, what language to use, etc.
- Software (not shown)
 

This screen allows you to select which software packages you wish to install.
- System (not shown)
 

This screen allows you to specify system-specific parameters normally set during the first boot including hostname, IP address, timezone, root password, and other networking information.
- File System (not shown)
 

This screen lets you lay out your disks and file systems. It supports a rich set of configuration options.
- Advanced (not shown)

This screen lets you select which scripts you want to run as part of the installation process.

Note that you do not need to visit all of the screens. You may hit the *Go!* button in any screen at any time.

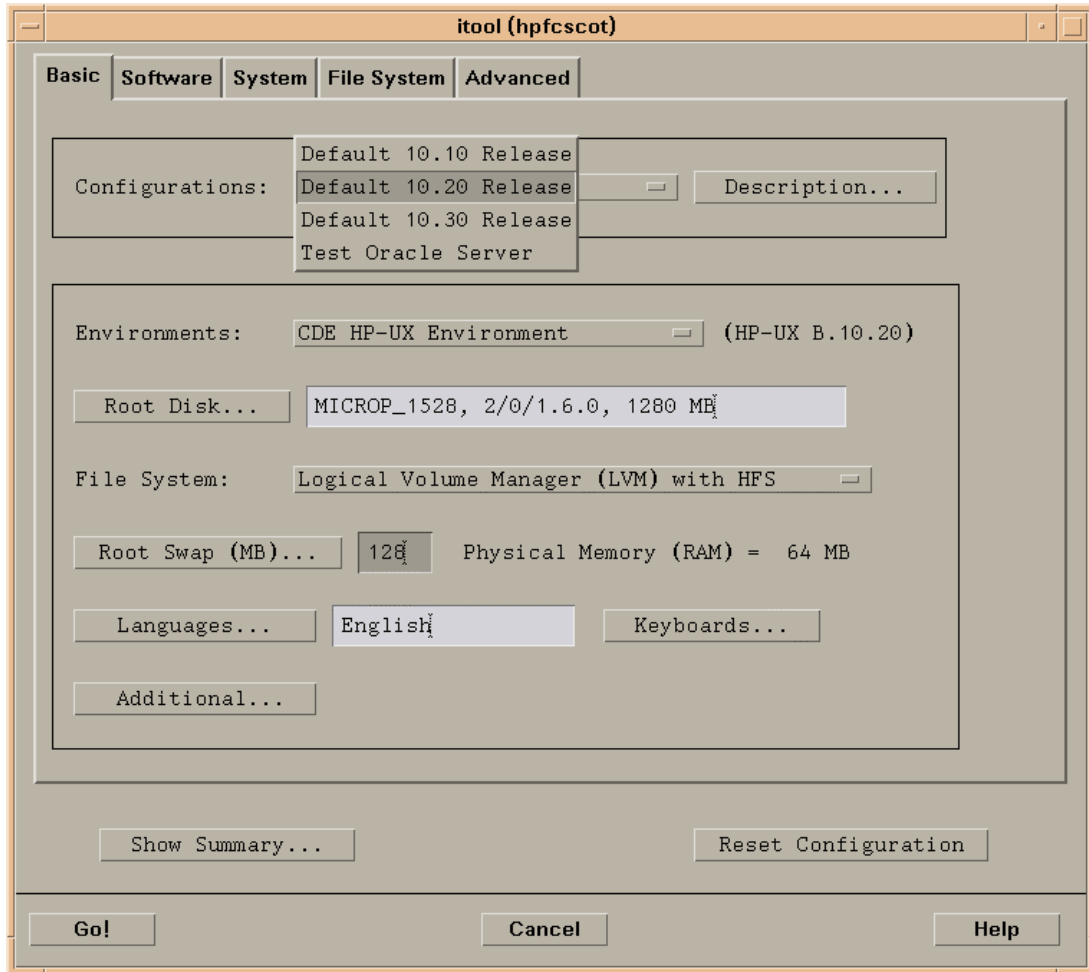


Figure 2: The Basic Tab

---

## BASIC FLOW

---

The basic mechanics of Ignite-UX are very similar to the way Cold Install worked. The overall goal is to boot a system from an independent device (network, CD-ROM, tape) in a self-contained environment such that it can configure the file systems on a disk and lay down a fresh copy of the HP-UX operating system.

The process of booting the system to be installed (usually referred to as the client or target) involves booting an HP-UX kernel (the *install kernel*) which in turn loads into memory a small RAM-disk file system (*INSTALLFS*). This file system is mounted as the root file system and the kernel continues to boot by running *init* from within the file system. The *init* command turns out to be the first phase of the

Ignite-UX application. Note that the install kernel and *INSTALLFS* will be pulled from the media if this is a media install and from the Ignite-UX server machine if this is a network install.

When the target machine is booted from an Ignite-UX server, two different communication mechanisms are used between the target and the server. The *tftp(1)* protocol is used by the target machine to transfer files from the server such as configuration files and user-supplied scripts which must run on the target machine.

In addition, the target and server may communicate with each other via NFS. The target will NFS mount the server's **/var/opt/ignite/clients** directory (if it has been exported). The server and target will communicate via files in the per-client directory **/var/opt/ignite/clients/0x{target-LLA}/**:

- the target will provide status information that the server can read
- the server will in turn write a file that contains instructions for the target
- the target will also generate a manifest and log file in this directory which may be printed and analyzed on the server

The NFS communication is only necessary when you run the UI from the server machine or if you want the log file and manifest to be accessible on the server machine. If the server does not export the **/var/opt/ignite/clients** directory, the target machine will proceed without it.

Once booted, Ignite-UX must acquire a configuration file (config file) which describes how the target machine should be installed. If a config file for this machine is already present on the install server and specifies a non-interactive install, the process will continue without any user interaction. Otherwise, Ignite-UX will allow the user to construct or modify a config file by running the user interface. The Ignite-UX target is booted in such a state that it waits for the server to instruct it how and when to do the installations.

Once a suitable config file for the target machine has been acquired, Ignite-UX uses it to direct the process in constructing the target machine.

---

## THE CONFIG FILE

---

Ignite-UX's central data store is called a config file. A config file can be thought of as a recipe for how to construct a target system. The config file is expressed in a language designed for this purpose. The language is fully defined in the *instl\_adm(4)* manual page. The syntax is human-readable; config files may be created directly by a user or via the Ignite-UX graphical user interface. The config file language is much like many programming languages in that it supports the use of variables and conditional expressions.

Most of the important elements which make up an installed system can be described in the config file:

- disk and file system layout
- software to be installed
- the target machine's system identity and network configuration
- kernel modifications (additional drivers or tunable parameter settings)
- user-defined scripts which will run at various points in the installation process to further customize the target machine

## TYPES OF CONFIG FILES

For maintenance convenience, the configuration information is split into several different config files. These config files fall into these basic classes:

- Default disk and file system layout

Because the capabilities of each operating system release differ somewhat, HP supplies a different set of defaults for each release. These are located in **/opt/ignite/data/Rel\_{release}/config** where *release* is the result of the **uname -r** command. For example, **/opt/ignite/data/Rel\_B.10.20/config** contains the default disk layouts for the HP-UX 10.20 release.
- Software description of a single SD depot

Config files which describe software available from SD depots can be automatically generated via an Ignite-UX tool called *make\_config(1m)*. This tool produces one config file per SD depot. Software description config files are located in **/var/opt/ignite/data/Rel\_{release}/\***.
- Software description of an archive

Config files can be hand built to allow access to non-SD archives (templates are provided with the Ignite-UX product in **/opt/ignite/data/examples/** to give you a good starting point). Archives may be in either tar or cpio format. Archive software description config files are also located in **/var/opt/ignite/data/Rel\_{release}/\***.
- Local configuration overrides that apply globally

It is often convenient to specify defaults which will be applied to every machine installed from a particular server. For example, you might want to specify the same NIS domain for all machines. Such overrides should be placed in the **/var/opt/ignite/config.local** file.
- Boot control parameters and networking information

It is possible to specify defaults for attributes like the IP address of the Ignite-UX server and whether to run a UI to install a new target. These can be specified in the first 8 Kb of the install file system (**/opt/ignite/boot/INSTALLFS**). This information is added/deleted with the *instl\_adm(1m)* command.
- Client-specific configuration files

Each client which is to be installed has a configuration file which is peculiar to it located at **/var/opt/ignite/clients/0x{LLA}/config**, where LLA is the link-level address of the client. This file is typically created as a result of running the user interface to specify the target machine configuration.

This file usually refers to other config files mentioned above. It also contains specific directives to override what may have been defined in the other files. For example, you may wish to customize the disk layout beyond what the defaults in **/opt/ignite/data/Rel\_{release}/config** allow. The customizations end up in **/var/opt/ignite/clients/0x{LLA}/config**.
- Named configurations created by saving a configuration via the UI

You can create your own default configurations via the UI and save them for future use. For example, you might have a large number of users with similar machines who all run CAD tools. You could build a configuration which matches what they need and save it in a configuration called "CAD System". When you need to install a new system of this type, you can select "CAD System" from the UI and you're done (or you could customize it further using

"CAD System" as a starting point). Saved configurations are located in **/var/opt/ignite/saved\_cfgs/\***.

- Other customized building blocks

The next section describes how multiple config files can be combined to define a single configuration. You can build your own config files to specify a particular building block you are interested in, and then combine them in arbitrary ways. These building block config files should be located in **/var/opt/ignite/data/Rel\_{release}/\***.

## COMBINING CONFIG FILES VIA INDEX ENTRIES

The grouping of config files into useful configurations is accomplished in the INDEX file (**/var/opt/ignite/INDEX**). This file contains a list of valid configurations, each of which is made up of one or more config files. The list of these configurations is presented in the UI as the basic starting point (this list can be seen in the user interface at the top of the screen in Figure 2). For example, the INDEX file might contain:

```
cfg "HP-UX B.10.20 Default" {
    description "This selection supplies the default system
                configuration that HP supplies for the B.10.20
                release."
    "/opt/ignite/data/Rel_B.10.20/config"
    "/var/opt/ignite/data/Rel_B.10.20/core_700_cfg"
    "/var/opt/ignite/data/Rel_B.10.20/core_800_cfg"
    "/var/opt/ignite/data/Rel_B.10.20/apps_700_cfg"
    "/var/opt/ignite/data/Rel_B.10.20/apps_800_cfg"
    "/var/opt/ignite/data/Rel_B.10.20/patches_700_cfg"
    "/var/opt/ignite/data/Rel_B.10.20/patches_800_cfg"
    "/var/opt/ignite/config.local"
}

cfg "CAD System - 10.10" {
    description "This selection is the typical CAD system installation
                for HP-UX B.10.10"
    "/opt/ignite/data/Rel_B.10.10/config"
    "/var/opt/ignite/data/Rel_B.10.10/core_700_archive_cfg"
    "/var/opt/ignite/data/Rel_B.10.10/apps_700_cfg"
    "/var/opt/ignite/data/Rel_B.10.10/patches_700_cfg"
    "/var/opt/ignite/config.local"
} = TRUE
```

With this INDEX file, the UI would present two different configurations: "HP-UX B.10.20 Default" and "CAD System - 10.10". The "CAD System - 10.10" configuration is the default (it is marked TRUE). Once you choose one of these base configurations, you can do further customizations with the UI if necessary, or just accept the defaults the configuration provided and do the install immediately.

If you selected "CAD System - 10.10", you would get the combination of the five config files listed for that clause. The order of the config files is significant; attributes specified in a later config file can override the same attributes specified in an earlier config file. There are also two config files which are implicitly used every time. Any information stored in the first 8 Kb of **/opt/ignite/boot/INSTALLFS** is



implicitly appended to each configuration. The client-specific configuration file (**/var/opt/ignite/clients/0x/LLA/config**) is implicitly added as the last config file for each configuration.

A default **cfg** clause for each release is shipped as part of the Ignite-UX product. Additional **cfg** clauses are added when:

- you save a named configuration from the graphical user interface via the *Save As* button
- you wish to create a configuration by modifying the INDEX file directly

## A FLAVOR OF CONFIG FILES

This section shows a few example config files to give you an idea of their look and capabilities. It does not pretend to fully cover the subject. See the *instl\_adm(4)* manual page for a complete description.

This example shows how a disk might be defined. Here, the disk is located at hardware address 2/0/1.6.0 and does not use LVM. The disk contains the “/” file system and a swap area. The swap area takes up 64 Mb, and the file system takes up whatever space is left over:

```
partitioned_disk {
    physical_volume disk[2/0/1.6.0] {
    }
    fs_partition {
        mount_point = "/"
        usage=HFS
        size=remaining
        file_length=long
    }
    swap_partition {
        usage=SWAP
        size=64Mb
    }
}
```

In this example, two disks are put together to form a single LVM volume group. Two file systems are defined; both are striped across both disks. The first file system (“/apps1”) is sized by calculating the amount of space required by the software which is to be loaded, and then adding a 30% free space cushion. The second file system (“/apps2”) gets all of the remaining space on the disks.

```

volume_group "appsvol" {
  physical_volume disk[2/0/1.5.0] {
  }
  physical_volume disk[2/0/1.4.0] {
  }
  logical_volume "apps1" {
    mount_point= "/apps1"
    usage=VxFS
    size=30%free
    minfree=5
    stripes=2
  }
  logical_volume "apps2" {
    mount_point= "/apps2"
    usage=VxFS
    size=remaining
    minfree=5
    stripes=2
  }
}

```

This example defines a few of the network parameters which will be assigned to the machine after it has been installed:

```

final system_name = "acorn1"
final ip_addr["lan0"] = "15.99.45.123"
final netmask["lan0"] = "255.255.248.0"
final nis_domain = "nis1"
final route_gateway[0] = "15.99.45.1"

```

This example defines a single SD depot from which software can be installed. Two different pieces of software are defined for the SD depot. Each can be selected independently for installation. The **impacts** lines tell Ignite-UX how much space this software requires in a given directory. This information is used to size the file systems correctly. Another interesting construct is **sw\_category**. This allows you to group the software so that the user interface can present it in chunks which make sense to you. Since this example references an SD depot, it would have been created by the *make\_config(1m)* command:

```

sw_source "ee_apps_depot" {
    description = "Electrical Engineering Application"
    source_format = SD
    source_type = "NET"
    sd_server = "15.23.45.6"
    sd_depot_dir = "/var/opt/ignite/depots/Rel_B.10.20/ee_apps"
}
sw_category "Applications" {
    description = "User Applications"
}
sw_sel "EE CAD Package" {
    sw_source = "ee_apps_depot"
    sw_category = "Applications"
    sd_software_list = "EECad,r=1.2,a=HP-UX_B.10.20_700"
    impacts = "/var" 90524Kb
    impacts = "/sbin" 1248Kb
}
sw_sel "EE Routing Package" {
    sw_source = "ee_apps_depot"
    sw_category = "Applications"
    sd_software_list = "EERoute,r=2.4,a=HP-UX_B.10.20_700"
    impacts = "/usr" 12568Kb
    impacts = "/var" 26788Kb
}

```

## CUSTOMIZATIONS BASED ON THE TARGET MACHINE

The config file syntax provides a large number of system attribute keywords which describe the target system. Some examples are:

- **disk[*hw-path*].size:** the size of the disk at the specified hw-path
- **memory:** the amount of memory present on the target system
- **hardware\_model:** the string returned from the **uname -m** command
- **lla:** the link-level address of the target machine

System attribute keywords can be used in expressions in config files so that a particular clause is only included in specific target situations. The basic format of these clauses is:

```
(x) {y}
```

which translates roughly to “if the expression *x* is true, then do *y*”.

For example, the following clause increases the size of some kernel tunable parameters if the target machine has more than 256 Mb of memory:

```

(memory > 256Mb) {
    mod_kernel += "nproc (20+12*MAXUSERS)"
    mod_kernel += "maxuprc 1000"
}

```

As another example, if you want to run a script to do some particular graphics customizations, but you only want to do so when the target machine has the appropriate hardware:

```
(graphics[0].planes > 0) {
    post_config_script +=
        "/var/opt/ignite/scripts/multi_plane_graphics"
}
```

You can also specify multiple conditions. This example installs a particular piece of (previously defined) application software if the target machine is a workstation (700) and there are at least 2 disks on the system. It also displays a message to let the user know why it is happening:

```
( (hardware_model ~ "9000/7.*") & (num_disks >= 2) ) {
    note += "Installed apps1 because this is a big series 700."
    init sw_sel "apps1" = TRUE
}
```

It is also possible to add an else clause. This example uses a generic variable capability and mathematical expressions which are new for Ignite-UX. This example sets the primary swap size based on the amount of memory in the target system:

```
(memory > 512Mb) {
    init _hp_pri_swap = 512Mb
}
else {
    init _hp_pri_swap = memory * 2
}
```

## CUSTOMIZATIONS BASED ON USER SELECTION

It is sometimes advantageous to be able to select particular customizations independent of the target machine's hardware setup. For example, you might have some machines which you intend to use as NFS file servers. You would like the ability to select NFS server capability from the UI when you are configuring the target system.

You have found that NFS file servers are more efficient if some of their kernel parameters are tweaked. NFS file servers also require some changes to the `/etc/rc.config.d/nfsconf` file via the `ch_rc(1m)` command.

The solution is to define a custom software selection (**sw\_sel**). This **sw\_sel** will be presented in the UI under the *Software* tab. Here's the example:

```
sw_source "special configs" {
    source_format = cmd
}
sw_sel "NFS Server" {
    sw_category = "Machine Uses"
    sw_source = "special configs"
    mod_kernel += "dbc_min_pct 35"
    mod_kernel += "dbc_max_pct 65"
    post_config_cmd += "
        /usr/sbin/ch_rc -a -p NFS_SERVER=1
        /usr/sbin/ch_rc -a -p NFS_CLIENT=1
        /usr/sbin/ch_rc -a -p NUM_NFSD=8"
}
```

Figure 3 shows the software tab in the user interface when the NFS server config file is used. As shown, the selected category is “Machine Uses” as defined in the config file. Choosing a different category would display a different set of software. If you were to select “NFS Server” from this screen, the kernel modifications specified in the config file would be applied during the installation. Likewise, the *ch\_rc* commands specified in the config file will be run as part of the installation.

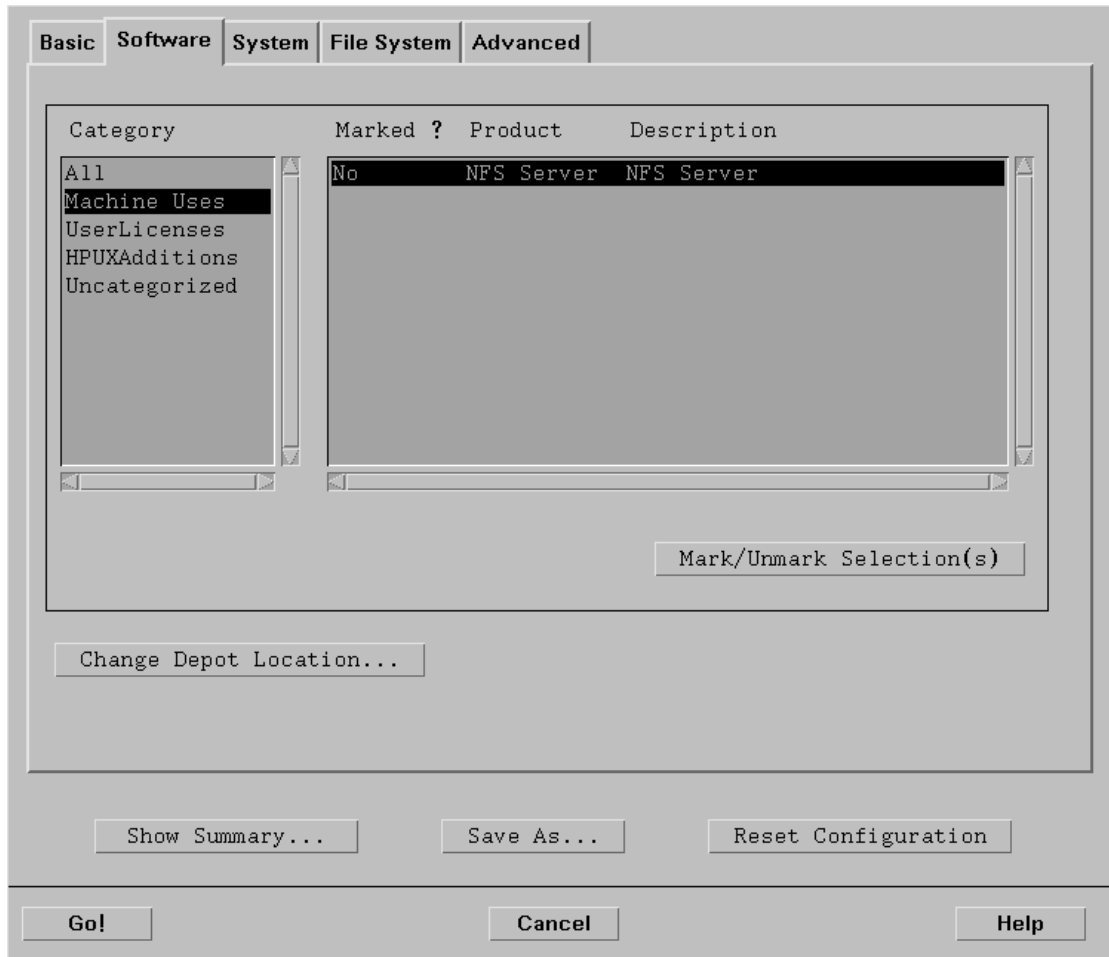


Figure 3 - Software Tab with NFS Server

## SYSTEM IMAGE INSTALLATION

In addition to supporting the standard OS installations from an SD depot, Ignite-UX supports a new customer use model called *System Image Installation*. This model recognizes that many, if not all, target nodes in a network may be identical (or mostly identical) to each other. It is possible to take advantage of this fact by building an archive which contains all of the files you want installed on each of the targets and then using Ignite-UX to install them.

This approach can have several advantages:

- Because the compressed system image is unpacked directly to disk over the network, the installation process can be much faster than an equivalent process using SD. The time savings will depend on the size of the installation being done and the capacity of

the network, but a typical system image can be extracted in about 20 minutes compared to about an hour for an SD install.

- Instead of troubleshooting a target, it is often more cost-effective to completely re-install with a known-good system image. Customers who have followed this approach say “Don’t troubleshoot, just re-install”.
- When coupled with *dataless* nodes (i.e. all volatile data is on a separate file server), system replacement time or move time is drastically reduced.
- Once a system image has been created, it is simple to apply it to multiple target machines. Very little or no user interaction is required during these subsequent installs, reducing the chance of error.

Building this *golden system image* is done by setting up a single machine the way that you want all of your machines to look, and then creating an archive of that machine. To set up the first machine:

- 1) install the base operating system you want via Ignite-UX
- 2) install whatever patches you want via SD (could also be done as part of the first step via Ignite-UX)
- 3) install whatever applications you want on every system (could also be done as part of the first step via Ignite-UX)

Ignite-UX supplies a tool called *make\_sys\_image(1m)* to actually create the image. This tool allows you to specify any of the archive methods (*tar* and *cpio*) and compression methods (*gzip* and *compress*) which Ignite-UX supports. It also provides the ability to produce an identity-neutral system image. Files which contain information that is specific to this machine (instead of generic to all machines which will use the archive) are cleansed. For example, the */etc/rc.config.d/netconf* file contains the machine’s hostname. Since you certainly don’t want all of the machines in your installation to have the same hostname, this file is cleaned up before the archive is created. Since *make\_sys\_image(1m)* is a shell script, you can modify it if your idea of which files need to be cleansed is different from ours.

Once the archive has been created, a simple config file is created to reference it. The config file would be located in */var/opt/ignite/data/Rel\_{release}/*. For example:

```
sw_source "core archive" {
    load_order = 0
    source_format = archive
    source_type = "NET"
    post_load_script = "/opt/ignite/data/scripts/os_arch_post_1"
    post_config_script = "/opt/ignite/data/scripts/os_arch_post_c"
    nfs_source = "15.1.23.45:/var/opt/ignite/archives"
    # could also use ftp or remsh instead of nfs
}
sw_sel "golden image 1" {
    description = "image1: English HP-UX 10.20 CDE"
    sw_source = "core archive"
    archive_type = gzip tar
    archive_path = "image1.gz"
    impacts = "/" 27Kb
    impacts = "/var" 5703Kb
    impacts = "/opt" 100456Kb
    impacts = "/etc" 2346Kb
}
```

Some example archive config files are shipped with Ignite-UX under */opt/ignite/data/examples/*. There are two scripts associated with the OS archive. These scripts can be modified to further

customize which files are used as is from the archive and which files are further modified by the Ignite-UX installation process.

The biggest decision in all of this is to figure out how much you want to put into your golden system image. If you put everything a target system needs into the image, the installation will be fast, but the maintenance of the image may be high:

- You will have to roll the image whenever a change occurs to anything contained in the image; for example, a new patch.
- If you have multiple classes of machines/users, you may need to maintain multiple images. For example, if a group of your users need EE tools and another group needs ME tools, you would need to maintain two full archives.

An alternative approach is to take advantage of Ignite-UX's ability to install from multiple sources in a single install session. You might want to have the following sources:

- basic OS archive (common to all of your target machines)
- current patches which are generic to all machines
  - these could be in another archive or in SD format
  - these could also be in the basic OS archive
- EE tools (could be in SD format or in an archive)
- ME tools (could be in SD format or in an archive)

You could then set up a couple of configurations - one for what is needed for EE machines and one for ME machines. When you need to make changes to individual sources, they are smaller in scale. There is no absolute right or wrong way to do any of this; the key is that Ignite-UX gives you the ability to make the tradeoffs which are correct for your situation.

---

#### THANKS!

---

Many people have worked together to make the Ignite-UX product a reality. I'd especially like to thank the other members of the Ignite-UX development team: David Arko, Bruce Bigler, Carlos Bonilla, Paul Christofanelli, Barb Flahive, Jeff Finz, Scot Greenidge, Jim Heaton, Bobbie Keever, Michael Roberts, Chris Rockwell, and Bruce Rodean.

More special thanks go to John Agosta and Ed Struzynski who have vigorously applied Ignite-UX to the HP-UX 9.\* to 10.\* transition strategy. Thanks also to Rob Lucke for freely sharing his insights regarding system image installation.

This development team has worked closely with many customers throughout the entire life cycle of the project. Our customer contacts have been frequent and invaluable. We owe a special debt of gratitude to all of the customers who took the time to show us how they did their jobs, bravely volunteered to be Beta sites, and worked with us to make Ignite-UX a useful tool.