# Product Control

Controlling installation and Product Features

Anas Nashif, SUSE Linux AG

Revision History

| | | |
|---|---|---|
| Revision 0.1 | $Date:$ | |
| | Initial Release as PDF | |

## Table of Contents

## 1. Functionality

The product control enables customization of the installation makes it possible to enable and disable features during installation in the final installed product. It controls the workflow and what is really shown to the user during installation.

Beside workflow configuration, other system variables are configurable and can be predefined by the system administrator,to name a fre the software selection, environment settings such as language, time zone, keyboard can be configured and would override default variables provided with shipped products.

The idea of having a pre-defined installation workflow and pre-defined system settings is the middle ground between manual installation and automated installation.

The product configuration file is provided in text on the installation media and defines various settings needed during installation. The following is a list of supported configuration options:

- Workflow

  Replaces the static workflow list with a configurable list using the product configuration file. Entire sections of the workflow can be skipped.

  For example, it will be possible to set the language variable in the configuration file and if the installation language is to be forced for some reason, i.e. IT department wants to force French installations in Quebec, Canada, then the entire dialogue is skipped. If the IT department is to recommend some settings but still give the user the choice to change the default settings, the language dialogue will be shown with Frenchy preselected.

  If non of the above options is used, the default dialogue settings is shown.

- Proposals

  As with the workflow, proposals are also be configurable. For example, certain products would skip some proposals. In the proposal screen the pre-configured settings can be shown with the possibility to change them or with inactive links if the configuration is to be forced.

- System Variables

  Lets the user define system variables like language, keyboard, time zone, window manager, display manager etc. The defined variables will be used as defaults in the respective dialogues.

- Package Selections and additional individual packages

  Define what base package selection and add-on selections should be used for the installation. Additionally provide the possibility to define a list of additional packages. All packages and selections can be selected depending on the architecture using a special architecture attribute in the configuration file.

- Partitioning

  Integrates flexible partitioning into configuration file, instead of the separate file currently used.

- Scripting and Hooks

  To customize installation further more, hooks and special slots can be defined where the user can execute scripts. For example, scripts can be executed at the very beginning of the installation (After processing the configuration file), in the installation system before initial boot, in the chroot-ed environment and after initial boot and before/after every step in the workflow. Scripting languages supported during installation are currently Shell, Perl.

## 2. Implementation

The control file is implemented in simple structured XML syntax which so far has been used for automated installation . The XML structure used can be mapped easily to YaST data structures and all data types available in YaST are supported for easy data access and manipulation.

The primary use of the control file is to configure the workflow of the installation and it offers the possibility to predefine a certain setup, but it also defines product installation features and other product related variables.

### Note

Note that the control file is not an optional tool to help customize installation, it is required during installation and without the file, installation may fail or lead to unexpected results. YaST provides a default and general control file which is always available in the system. The general and product independent control files is installed by the package *yast2-installation* in `/usr/share/YaST2/control/control.xml`.

During installation, *linuxrc* searches for the a file named `control.xml` on the installation medium (CD, NFS, FTP..) and copies the file into the installation system and makes the file available to YaST. YaST then starts and looks for the control file in 3 location before it starts with the installation workflow:

- `/control.xml`

  Usually the file is in top directory after it has been copied by linuxrc and during initial installation phase.

- `/var/lib/YaST2/control.xml`

  After reading the file, and before second installation phase, the control file is copies from the top directory to `/var/lib/YaST2/control.xml`

- `/usr/share/YaST2/control/control.xml`

  This is the location where *yast2-installation* installs the file in all products. The file is the same on all products.

## 2.1. Product Specific Control Files

One of the main reasons for using the control is to provide non YaST developers to change the installation behavior and customize various settings without the need to change YaST packages.

# 3. Configuration

## 3.1. workflows

Using the control file, multiple workflows can be defined for different modes and installation stages. Thus, the element *workflows* in the control file evaluates to a list of workflows.

Beside defining what YaST clients should be executed during installation, the workflow configuration also let you specify the wizard steps and how they should appear during graphical installation.

A workflow list element is a map with the following elements:

- label

  The label of the workflow as it appears on the left side of the wizard. For example *Base Installation*

- defaults

  The default arguments to the clients. This is a map element.

- stage

  This options defines the stage or phase of installation.. Possible values are *initial* for the initial stage and *continue* for the workflow of the installation after reboot

- mode

  Defines installation mode. Several modes are available, most important modes are:

  - installation
  - update
  - autoinst

- modules

This is the actual workflow and is a list of elements describing the order in which the installation should proceed.

A module element is a map with the following configuration options:

- name: The name of the module. All installation clients and modules have a unified prefix (inst_) which can be ommited here. For example, if the YaST file for the module is called *inst_test*, then the name in the control file is *test*
- label: The label of the module in the step dialog. This is an optional element. If it is not set, the label of the previous module is used.
- arguments: The arguments for the module is a comma separated list which can accept booleans and symbols.

The following listing shows a typical installation workflow:

```
    <workflows config:type="list">
      <workflow>
          <!-- 'label' is what the user will see -->
          <label>Base Installation</label>
          <!-- default settings for all modules -->
          <defaults>
              <!-- arguments for the clients -->
              <arguments>false,false</arguments>
              <!-- allowed architectures "all", "i386", "i386,ia64,x86_64"  -->
              <archs>all</archs>
          </defaults>
          <stage>initial</stage>
          <mode>installation,update</mode>
          <modules  config:type="list">
              <module>
                  <name>info</name>
                  <arguments>false,true</arguments>
              </module>
              <module>
                  <name>proposal</name>
                  <arguments>true,true,`ini</arguments>
                  <label>Installation Settings</label>
              </module>
              <module>
                  <name>do_resize</name>
                  <update config:type="boolean">false</update>
                  <archs>i386,x86_64,ia64</archs>
                  <label>Perform Installation</label>
              </module>
              <module>
                  <name>prepdisk</name>
                  <!-- Multiple modules with the same 'label' will be
                       collapsed to one single user-visible step.
                       The step is considered finished when the last module
                       with the same 'label' is finished.  -->
                  <label>Perform Installation</label>
              </module>
              <module>
                  <name>kickoff</name>
                  <label>Perform Installation</label>
              </module>
              <module>
                  <name>rpmcopy</name>
                  <label>Perform Installation</label>
              </module>
              <module>
                  <name>finish</name>
                  <label>Perform Installation</label>
              </module>
          </modules>
      </workflow>
```

## 3.2. Proposals

FIXME

## 3.3. Installation and Product Variables

It is possible to define some installation variables (language, timezone, keyboard,.. ) and force them in the proposal. User will still be able to change them however.

The following variables can be set:

- Timzeone
- Language
- Keyboard
- Auto Login (not recommended for multi-user environments and server installations)
- IO Scheduler

  Default is *as*.
- Desktop Scheduler

the following example shows all options above

```
<globals>
    <enable_autologin config:type="boolean">true</enable_autologin>
    <language>de_DE</language>
    <timezone>Canada/Eastern</timezone>
    <use_desktop_scheduler config:type="boolean">true</use_desktop_scheduler>
    <io_scheduler>as</io_scheduler>
</globals>
```

## 3.4. Software

Using this section in the control file you can change the software proposal during installation.

By default, the software proposal is generated depending on the available space in the system. If enough space is available, a fairly large selection of packages and add-on package groups is automatically selected. This behavior is controlled by the *selection_type* element in the control file.

The default value for the above element is *auto*. To force a selection which can not be changed by the user, the value *fixed* has to be used.

If a fixed software selection is desired, then the fixed selection name has to be specifed in the control file. This selection must be a base selection. To specify the name of the base selectiont to be forced, use the *base_selection* element.

The list of base selections can be found on the first CD of the product in the directory `suse/setup/descr` or by using the following command (The example shows the list of selections for *SLES 9*):

To switch from *desktop* based proposals to the normal behavior of software proposals in YaST , the *software_proposal* element has been introduced. Setting the value to *desktop* will make YaST show a dialog with Desktops to select from, i.e. KDE or *Gnome*. If this element is not defined, default behavior is assumed.

```
for i in `grep -l baseconf SUSE-SLES-Version-9/CD1/suse/setup/descr/* `; do
      basename $i .sel;
done
```

The above will have the following output:

```
Full-Installation
Minimal+X11
Minimal
default
```

which matches the base selections in *SLES 9*.

Additionally, you can configure how updating of packages should be performed. The following options are available:

- delete_old_packages

  Do not delete old RPMs when updating.

- only_update_selected

  One can update (only update packages already installed) or upgrade (also install new packages with new functionality). For example, SLES should do "update", not "upgrade" by default

## 3.5. Partitioning

If present, the partition proposal will be based on the data provided in the control file.

### 3.5.1. Algorithm for space allocation

Space allocation on a disk happens in the following order. First all partition get the size allocated that is determined by the size parameter of the partition description. If a disk cannot hold the sum of these sizes this disk is not considered for installation. If all demands by the size parameter are fulfilled and there is still space available on the disk, the partitions which have a parameter "percent" specified are increased until the size demanded by by "percent" is fulfilled. If there is still available space on the disk (this normally only can happen if the sum of all percent values are below 100), all partitions that are specified with a size of zero are enlarged as far as possible. If a "maxsize" is specified for a partition, all enlargement are only done up to the specified maxsize.

If more than one of the available disks is eligible to hold a certain partition set, the disk is selected as follows. If there is a partition allocated on that disk that has its size specified by keywords "percent" or by "size=0" and does not have a "maxsize" value set then the desired size for this partition is considered to be unlimited. If a partition group contains a partition which an unlimited desired size, the disk that maximizes the partition size for the unlimited partitions is selected. If all partitions in a partition group are limited in size then the smallest disk that can hold the desired sizes of all partitions is selected for that partition group.

If there are multiple partition groups the the partition group with the lowest number (means highest priority) get assigned its disk first. Afterward the partition group with the next priority gets assigned a the optimal disk from the so far unassigned disks.

### 3.5.2. Configuration Options

The following elements are global to all disks and partitions:

#### prefer_remove

| | |
|---|---|
| Possible values | true\|false |
| Default value | true |
| Description | If set to false the partition suggestion tries to use gaps on the disks or to re-use existing partitions. If set to true then the partition suggestion prefers removal of existing partitions. |

#### remove_special_partitions

| | |
|---|---|
| Possible values | true\|false |
| Default value | false |
| Description | If set to false YaST2 will not remove some special partitions (e.g. 0x12 Compaq diagnostics, 0xde Dell Utility) if they exists on the disk even if prefer_remove is set to true. If set to true YaST2 will remove even those special partitions. |

**Caution**

Caution: Since some machines are not even bootable any more when these partitions are removed one should really know what he does when setting this to true

#### keep_partition_fsys

| | |
|---|---|
| Possible values | comma separated list of reiser, xfs, fat, vfat, ext2, ext3, jfs, ntfs, swap |
| Default value | Empty list |
| Description | Partitions that contain filesystems in that list are not deleted even if prefer_remove is set to true. |

#### keep_partition_id

| | |
|---|---|
| Possible values | comma separated list of possible partition ids |
| Default value | Empty list |
| Description | Partitions that have a partition id that is contained in the list are not deleted even if prefer_remove is set to true. |

### keep_partition_num

| | |
|---|---|
| Possible values | comma separated list of possible partition numbers |
| Default value | Empty list |
| Description | Partitions that have a partition number that is contained in the list are not deleted even if prefer_remove is set to true. |

To configure individual partitions and disks, a list element is used with its items describing how should the partitions be created and configured

The attributes of such a partition are determined by several elements. These elements are described in more detail later.

## General remarks to all option values

If there is a blank or a equal sign (=) contained in an option value, the values has to be surrounded by double quotes ("). Values that describe sizes can be followed by the letters K, M, G. (K means Kilobytes, M Megabytes and G Gigabytes).

### mount

| | |
|---|---|
| Example | <mount>swap</mount> |
| Description | This entry describes the mount point of the partition. For a swap partition the special value "swap" has to be used. |

### fsys

| | |
|---|---|
| Example | <fsys>reiser</fsys> |
| Description | This entry describes the filesystem type created on this partition. Possible Filesystem types are: reiser, ext2, ext3, xfs, vfat, jfs, swap If no filesystem type is given for a partition, reiserfs is used. |

### formatopt

| | |
|---|---|
| Example | <formatopt>reiser<formatopt> |
| Description | This entry describes the options given to the format command. Multiple options have to be separated by blanks. There must not be a blank between option letter and option value. This entry is optional. |

### fstopt

| | |
|---|---|
| Example | <fstopt>acl,user_xattr<fstopt> |
| Description | This entry describes the options written to `/etc/fstab`. Multiple options have to be separated by comma. This entry is optional. |

### label

| | |
|---|---|
| Example | <label>emil<label> |
| Description | If the filesystem can have a label, the value of the label is set to this value. |

### id

| | |
|---|---|
| Example | <id>0x8E<id> |
| Description | This keyword makes it possible to create partitions with partition ide other than 0x83 (for normal filesystem partitions) or 0x82 (for swap partitions). This make it possible to create LVM or MD partitions on a disk. |

### size

| | |
|---|---|
| Example | <size>2G<size> |
| Description | This keyword determines the size that is at least needed for a partition. A size value of zero means that YaST2 should try to make the partition as large as possible after all other demands regarding partition size are fulfilled. The special value of "auto" can be given for the `/boot` and swap partition. If auto is set for a /boot or swap partition YaST2 computes a suitable partition size by itself. |

### percent

| | |
|---|---|
| Example | <percent>30<percent> |
| Description | This keyword determines that a partition should be allocated a certain percentage of the available space for installation on a disk. |

### maxsize

| | |
|---|---|
| Example | <maxsize>4G<maxsize> |
| Description | This keyword limits the maximal amount of space that is allocated to a certain partition. This keyword is only useful in conjunction with a size specification by keyword "percent" or by an entry of "size=0". |

### increasable

| | |
|---|---|
| Example | <increasable config:type="boolean">true<increasable> |
| Default | false |
| Description | After determining the optimal disk usage the partition may be increased if there is unallocated space in the same gap available. If this keyword is set, the partition may grow larger than specified by the maxsize and percent parameter. This keyword is intended to avoid having unallocated space on a disk after partitioning if possible. |

### disk

| | |
|---|---|
| Example | <disk>2<disk> |
| Description | This keyword specifies which partitions should be placed on which disks if multiple disks are present in the system. All partitions with the same disk value will be placed on the same disk. The value after the keyword determines the priority of the partition group. Lower numbers mean higher priority. If there are not enough disks in the system a partition group with lower priority is assigned a separate disks before a partition group with higher priority. A partition without disk keyword is implicitly assigned the highest priority 0. |

## Example 1. Flexible Partitioning

If in the example below the machine has three disks then each of the partition groups gets on a separate disk. So one disk will hold `/var`, another disk will hold /home and another disk will hold `/`, `/usr` and `/opt`. If in the above example the machine has only two disks then `/home` will still be on a separate disk (since it has lower priority than the other partition groups) and `/`, `/usr`, `/opt` and `/var` will share the other disk.

If there is only one disk in the system of course all partitions will be on that disk.

```
  <partitions config:type="list">
    <partition>
        <disk config:type="integer">3</disk>
        <mount>/var</mount>
        <percent config:type="integer">100</percent>
    </partition>
    <partition>
        <disk config:type="integer">2</disk>
        <mount>/</mount>
        <size>1G</size>
    </partition>
    <partition>
        <disk config:type="integer">2</disk>
        <mount>/usr</mount>
        <size>2G</size>
    </partition>
    <partition>
        <disk config:type="integer">2</disk>
        <mount>/opt</mount>
        <size>2G</size>
    </partition>
   <partition>
        <disk config:type="integer">1</disk>
        <mount>/home</mount>
        <percent config:type="integer">100</percent>
    </partition>
</partitions>
```

A more complete example with other options is hown below:

```
      <partitioning>
        <partitions config:type="list">
            <partition>
                <disk config:type="integer">2</disk>
                <mount>swap</mount>
                <size>auto</size>
            </partition>
            <partition>
                <disk config:type="integer">1</disk>
                <fstopt>defaults</fstopt>
                <fsys>reiser</fsys>
                <increasable config:type="boolean">true</increasable>
                <mount>/</mount>
                <size>2gb</size>
            </partition>
            <partition>
                <disk config:type="integer">2</disk>
                <fstopt>defaults,data=writeback,noatime</fstopt>
                <fsys>reiser</fsys>
                <increasable config:type="boolean">true</increasable>
                <mount>/var</mount>
                <pct config:type="integer">100</pct>
                <size>2gb</size>
            </partition>
        </partitions>
        <prefer_remove config:type="boolean">true</prefer_remove>
        <remove_special_partitions config:type="boolean">false</remove_special_partitions>
    </partitioning>
```

## 3.6. Hooks

It is possible to add hooks before and after any workflow step for further customization of the installed system and to to perform non-standard tasks during installation.

Two additional elements define custom script hooks:

- prescript: Executed before the module is called.
- postscript: Executed after the module is called.

Both script types accept two elements, the interpreter used (shell or perl) and the source of the scripts which is embedded in the XML file using CDATA sections to avoid confusion with the XML syntax. The following example shows how scripts can be embedded in the control file:

```
                <module>
                    <name>info</name>
                    <arguments>false,true</arguments>
                    <prescript>
                        <interpreter>shell</interpreter>
                        <source>
<![CDATA[#!/bin/sh
touch /tmp/anas
echo anas > /tmp/anas
]]>
                        </source>
                    </prescript>
                </module>
```