Creating Add-Ons

Klara Cihlářová <cihlarov@suse.cz>
Jakub Friedl <jfriedl@suse.cz>
Jiří Srain <jsrain@suse.cz>
Stanislav Višňovský
Rebecca Walter

Creating Add-Ons

Version 1.0 RC2, 4.7.2006

Disclaimer

Copyright and Trademarks

Contents

1 Creating an Add-On Product 3
2 Add-On Structure 5
3 The `content` File 13
4 Package Descriptions 17
5 Selections and Patterns 26
6 Signatures and Other Security Issues 36
7 Autorun 38

This document describes add-on media preparation for SUSE Linux 10.1 and SUSE Linux Enterprise 10 products. The add-on support was developed to support our customers and partners and simplify third-party software distribution for all SUSE products.

1 Creating an Add-On Product

If you want to distribute your software or product for SUSE Linux 10.1, SUSE Linux Enterprise Server 10, SUSE Linux Enterprise Desktop 10, and higher with support from installation programs like YaST and rug, you can create add-on media.

Supported Versions

Including add-ons is supported for SUSE Linux Enterprise Server 10, SUSE Linux Enterprise Desktop 10, and higher. Add-ons cannot be used in older products. SUSE Linux 10.1 does not supported creation of an add-on workflow.

Software distribution as add-on products has the following advantages:

- Your software can be added to the list in YaST Software Management and to the list of installation sources. Users can easily install and reinstall your software.
- You can create an installation workflow and configuration for the add-on product.

- It is possible to influence the second stage of the installation if users add the product during initial installation. This means that you can add new configuration dialogs for your software directly to the installation procedure. You can omit this functionality if you think it is not useful for your software.

To create add-on media, do following:

1. Outline the add-on media structure and create the `content` file with a basic media description for YaST. See Section 2, "Add-On Structure" (page 5) for an add-on media structure example and Section 3, "The `content` File" (page 13) for information about the structure of the file.

2. Prepare signed RPM packages with your software and create package descriptions. For details about creating package descriptions, see Section 4, "Package Descriptions" (page 17).

3. Create the add-on license file and optional special information file. Find information about placing and naming these files in Section "Special Files in the `media.1` Directory" (page 10).

4. Create selections (for SUSE Linux 10.1) or patterns (SUSE Linux Enterprise). This step is optional. For more information, see Section 5, "Selections and Patterns" (page 26).

5. Create the installation workflow for the add-on product. This step is optional.

6. To give autorun functionality, create autorun files for your add-on as described in Section 7, "Autorun" (page 38).

7. Create md5 checksums for all files in the directory and save them to the file `MD5SUMS`. You cannot create md5 checksums for directories with subdirectories only. For more information, see Section 6.2, "MD5" (page 37).

8. Sign the add-on product as described in Section 6, "Signatures and Other Security Issues" (page 36).

9. Create a `directory.yast` file in directories of the add-on that do not have packages. For more information, see Section "The `directory.yast` Files" (page 10).

10 Burn the add-on media as normal data CDs or DVDs. You can use any burning software.

Testing the Add-On Media

YaST and zmd use a cache for the add-on media. The add-on media are cached on disk and used even if the original source is removed. To prevent problems with adding and removing the test media, change the content file for each add-on preversion. It is not possible to add two add-on media with same product name and vendor name in the `content` file.

2 Add-On Structure

YaST can work with two formats of installation media: YaST and YUM. Only the YaST format is described in this document, because it is recommended for local installation sources, such as add-on product CDs or DVDs. For information about the YUM format, read the articles:

- `http://en.opensuse.org/Installation_Sources`
- `http://en.opensuse.org/Secure_Installation_Sources`
- `http://en.opensuse.org/Libzypp`

Before you start creating the add-on media, create a dedicated directory, for example, `/tmp/addon`, to have better overview of included files and directories.

To have functional add-on media, first create a minimal skeleton of the media in your dedicated directory. This minimal skeleton contains all parts to install a product with YaST, but does not include any selections or patterns. The product will not be installed automatically after the CD or DVD is inserted or display special notes.

You can create add-on with multiple media. The minimal skeleton for all media is same. The media skeletons differ only in the number for the `media-n` directory.

Using Minimal Media

If you want to test the minimal add-on functionality, you should:

- Sign metadata with your key
- Create an `MD5SUMS` file with the MD5 checksums of included RPM files in all data subdirectories
- Add the SHA1 digest of all `MD5SUMS` files, files in the `descr` directory, and all used keys to the `content` file
- Provide all used keys

You should also create `directory.yast` as described in Section 1, “Creating an Add-On Product” (page 3). For more information about security issues and metadata signatures, see Section 6, “Signatures and Other Security Issues” (page 36).

The skeleton of an add-on medium has three parts:

The `content` File
: Information about the add-on product.

The `media.n` Directory
: Files with information about the media. The directory is called `media.1` for the first add-on medium. If you prepare an add-on with multiple media, the second media should instead have a `media.2` directory, the third media a `media.3` directory, and so on.

The Data Directory
: Directory with RPM packages.

This example is the first add-on medium, uses the data directory `suse`, and has packages only for the i586 architecture:

```
-content
-gpg-pubkey-*.asc (* is a number of the key)
-media.1/   --media
            --products

-suse/      --i586/  ---*.rpm (* is a name of the package)
```

```
            --setup/ ---descr/   ----packages
                                 ----packages.en
```

A minimal add-on structure with signatures and all integrity check components:

```
-content
-content.asc
-content.key
-directory.yast
-gpg-pubkey-*.asc (* is a number of the key)
-media.1/   --directory.yast
            --media
            --products
            --products.asc
            --products.key
 -suse/ --i586/    ---MD5SUMS
                   ---*.rpm (* is a name of the package)


        --setup/    ---descr/   ----directory.yast
                                ----MD5SUMS
                                ----packages
                                ----packages.en
```

2.1 The Data Directory

The data directory is defined with `DATADIR` in the `content` file, which is described in Section 3, “The `content` File” (page 13). The name of the data directory may be selected freely. Store all RPM packages of your add-on product in this directory sorted into subdirectories by the architecture for which they were built. Name these subdirectories by architecture:

i386
: For the 32-bit i386 Intel architecture (i486 with coprocessor and higher). Since SuSE Linux 8.0 and SLES8, 32-bit SUSE Linux is optimized for the i586 architecture and this directory is obsolete.

i586
: For the 32-bit i586 Intel architecture (Pentium 1 and higher). Since SuSE Linux 8.0 and SLES8, 32-bit SUSE Linux is optimized for this architecture.

i686
: For the 32-bit i686 Intel architecture.

x86_64
: For the 64-bit AMD x86-64 and Intel EMT architectures.

ia64
: For the 64-bit Intel Itanium architecture.

ppc
: For the Power PC architecture.

ppc64
: For the 64-bit Power PC architecture.

noarch
: For architecture-independent packages.

src
: For source packages.

nosrc
: For proprietary software development packages without source packages. This directory does not contain the binary for installation, but only files needed for RPM creation.

Some products are built only for some architectures. It is not necessary to prepare the architecture-specific directories for unsupported architectures.

SUSE Linux Optimization

From SUSE Linux 8.0 and SUSE Linux Enterprise 8, all packages for 32-bit Intel processors are optimized for the i586 architecture. The last versions with i386 optimization are SuSE Linux 7.3 and SUSE Linux Enterprise 7.

For SUSE Linux 10.1, x86_64, i686, i586, ppc, noarch, and src are supported. For SUSE Linux Enterprise Desktop 10, x86_64, i686, i586, noarch, src, and nosrc are supported. For SUSE Linux Enterprise Server 10, x86_64, ia64, i686, i586, ppc, ppc64, noarch, src, and nosrc are supported.

2.2 The `media.n` Directory

The `media.n` directory includes basic information about the add-on media set. Replace the *n* character with the number of the media. For example, if you create the first medium of an add-on with multiple media, use `media.1`. For the second medium, use `media.2`. The directory on all media should contain the files:

`media`
: The mandatory `media` file contains a media description for identification purposes. It is not shown to the user but should contain human-readable data for debugging purposes.

`products`
: This file contains the directory names for each product contained on the media. If this file is not present, a single product on the medium's root directory is assumed.

The `media` File

The `media` file contains a media description for identification purposes. It is not shown to the user but should contain human-readable data for debugging purposes.

The file for the first medium of the add-on should contain three lines:

- Name of the vendor
- Media ID, which can be arbitrary unique number, such as the date of creation in YYYYMMDDHHMMSS format
- Number of media in the product

Example for the first medium of an add-on with five media:

```
SUSE Linux Products GmbH
20060505000500
5
```

All other media can contain only lines with vendor and creation date.

The `products` File

This file contains the directory name for each product contained on the media. If this file is not present, a single product on the root directory is assumed.

`product` is an ASCII file with one line per add-on product. Each line starts with the directory name (relative to the root directory of the medium) followed by whitespace (space or tab) and the product name and version. A leading slash in the directory name is only needed to specify the root directory of the media.

In the following example, `SUSE Linux Add-On Example 10.1` is in the root directory. The other products (Add-On 2 and Add-On 3) have their own subdirectories:

```
/        SUSE Linux Add-On Example 10.1
addon2   Add-On 2
addon3   Add-On 3
```

The `directory.yast` Files

YaST2 uses `directory.yast` files to obtain information about the media directory structure. This file should be in every directory of the media except directories with RPM packages. In the directories with packages, you need only an `MD5SUMS` file, because information about packages are provided by package description files, described in Section 4, “Package Descriptions” (page 17).

To create it, enter the following command in each directory:

```
ls -A1 -p > directory.yast
```

where `-A1` is A and number one.

Special Files in the `media.1` Directory

`media.1` can optionally host the files `license.zip` and `info.txt`. These files are included on the first medium only.

If you want to display information about the add-on product license as a window with *Agree* and *Disagree* buttons before installation starts, include `license.zip` in `media.1`. `license.zip` can include the license in different languages, one file per language. The filename should be in the format `license.`*LANG*`.txt`, where *LANG* is the ISO code of the language. To differentiate variants of one language, you can use the

language-*code_country*-*code* combination. The files should use UTF-8 encoding.

Example `license.zip` content:

```
license.de.txt
license.es.txt
license.fr.txt
license.it.txt
license-ja_JP.txt
license.pt_BR.txt
license.txt
license.zh_CN.txt
license.zh_TW.txt
```

The optional `info.txt` file gives information about the add-on that should be displayed as a pop-up window with an *OK* button. The `info.txt` file is a simple text file in UTF-8 encoding.

2.3 Optional Files

Optional files and directories are not needed for simple package installation, but can extend the add-on functionality or provide useful information. You can, for example, create files that make your add-on product visible as a pattern in YaST or include a simple text file with instructions for how to install your add-on product.

The optional files `autorun.sh` and `autorun.inf` include information for automatic start of the add-on media. For more information about creation, see Section 7, “Autorun” (page 38).

The optional files `INDEX.gz`, `ARCHIVES.gz`, and `ls-lR.gz` include information about files in the packages and descriptions of the packages. The optional files `COPYING`, `COPYRIGHT`, and `LICENSE.TXT` include information about legal issues. You can include localized version of these three files. The example below has German versions in files with the `.de` extension.

The optional files `README` and `README.DOS` include information about how to install the add-on product. `README` is for UNIX and UNIX-like systems and `README.DOS` for Windows systems. You can include localized versions of the files. The example below has a German version in the files `LIESMICH` and `LIESMICH.DOS`.

The optional files `y2update.tgz`, `servicepack.tar.gz`, and `installation.xml` contain the configuration of the add-on installation workflow. It is normally only on the first medium.

The optional `/suse/setup/selection` file and files with the `.sel` extension define a SUSE Linux selection visible in YaST. The configuration file `/suse/setup/patterns` and files with a `.pat` extension are the equivalent for patterns. See Section 5, “Selections and Patterns” (page 26) for more information about patterns and selections.

The optional file `SuSEgo.ico` is the icon for your add-on product.

The following is an example of the add-on media structure with optional files:

```
–ARCHIVES.gz
  -autorun.inf
  -autorun.sh
  -content
  –content.asc
  -content.key
  -COPYING
  -COPYING.de
  -COPYRIGHT
  -COPYRIGHT.de
  -directory.yast
  -gpg-pubkey-*.asc (* is the number of the key)
  -ChangeLog
  –INDEX.gz
  –LICENSE.TXT
  -LIESMICH
  -LIESMICH.DOS
  -ls-lR.gz
  -media.1/   --directory.yast
              --info.txt
              --license.zip
              --media
              --products
              –-products.asc
              --products.key
  -pubring.gpg
  -README
  -README.DOS
  –servicepack.tar.gz
  -y2update.tgz
  -installation.xml
  -suse/ --i586/    ---MD5SUMS
                    ---*.rpm (* is the name of the package)

         --noarch/   ---MD5SUMS
```

```
                ---*.rpm (* is the name of the package)

        --setup/    ---descr/   ----directory.yast
                                ----EXTRA_PROV
                                ----MD5SUMS
                                ----packages
                                ----packages.lan (lang=de,en,hu...)
                                ----selection (only for sl)
                                ----*.sel (only for sl)
                                ----patterns (only for sle)
                                ----*.pat (only for sle)
                    ---LIESMICH
                    ---MD5SUMS
                    ---README

         --x86_64/   ---MD5SUMS
                     ---*.rpm (* is the name of the package)
-SuSEgo.ico
```

2.4 Add-On Products and AutoYaST

For AutoYaST, it is not necessary to create separate entries for all add-on products. You can create only one for all add-ons and add the list of all the add-ons to the special file `addon_products` in the root of the new add-on media.

The `addon_products` file is a simple text file in ASCII encoding. Installation sources are defined with their installation repository URL, one URL per line.

For example, to add SUSE Linux 10.1 add-on from `www.opensuse.org`, enter the installation repository for it. The repository URL is `http://download.opensuse.org/distribution/SL-10.1/inst-source/`. The entry in `addon_products` should be:

```
http://download.opensuse.org/distribution/SL-10.1/inst-source/
```

3 The `content` File

This file is stored in the product directory as specified by the `products` file on the media. Without a `products` file, the product directory defaults to the root directory of the media. The `content` file contains all product-specific data to describe and identify the contents of the product.

Validity of the `content` File

Check your `content` file before you use it. Errors in keywords and keywords without values cause rejection of the add-on media by target systems.

If your `content` file is not valid, find the following error messages in `/var/log/YaST2/y2log`:

```
Downloading metadata failed (is a susetags source?)
  or user did not accept remote source. Aborting refresh.
```

```
[zypp] SourceFactory.cc(createFrom):183 Not SUSE tags source,
trying next type
```

The `content` file is encoded in UTF-8 and consists of keyword and value pairs separated by spaces.

Table 1 *Supported content Keywords*

Keyword	Required	Description
PRODUCT	Yes	Product name.
VERSION	Yes	Product version and release as in RPM *`major.minor-release`*.
DISTPRODUCT	Yes	Distribution ID (vendor specific). The value of the keyword must not contain spaces. Only letters, numbers, and the characters: .~_- are allowed.
DISTVERSION	Yes	Distribution version (vendor specific).
VENDOR	Yes	Vendor name (free form).
ARCH.*`base`*	Yes	Space-separated list of allowed architectures for *`base`*.
DEFAULTBASE	Yes	Minimal architecture base supported by this product. The default is the *`base`* architecture if no matching ARCH.*`base`* is found.

Keyword	Required	Description
REQUIRES	Yes	Resolvables that must be installed on the system to meet the needs of this product. This is a space-separated list of names or kind:name pairs optionally followed by version constraints. Just a name denotes a dependency to a package, such as `sles-release` or `sles-release-10`. The kind can be `package`, `pattern`, or `product`, such as `pattern:basesystem`.
PREREQUIRES	No	Resolvables that must be installed on the system before installation of this product. The syntax is the same as for REQUIRES.
PROVIDES	No	Capabilities this resolvable provides. They can be used to match REQUIRES from others. Every resolvable has a provide by default— its own name and edition. For example, package bar-1.42-1 provides the capability `bar = 1.42-1`.
CONFLICTS	No	This resolvable cannot be installed if the specified resolvable or one that provides the capability is installed.
OBSOLETES	No	When this resolvable is installed, it uninstalls any other resolvable with a name matching this keyword.
RECOMMENDS	No	A weak version of REQUIRES. An attempt is made to fulfill RECOMMENDS, but they are silently dropped if no match is possible.
SUGGESTS	No	These are just hints for an application and not handled during dependency resolution.
SUPPLEMENTS	No	A reverse RECOMMENDS. This resolvable is installed if the specified capability is provided by an installed resolvable. The dependency resolver installs it. Uninstalling it is silently accepted.

Keyword	Required	Description
ENHANCES	No	A reverse SUGGESTS. This resolvable can be installed if this capability is provided by an installed resolvable. It is just a hint for an application. For example, SuSEplugger can suggest packages for installation if specific hardware is found.
LINGUAS	No	ISO language code or language code_country code.
LABEL	No	UTF-8 encoded label. Default label if LINGUAS is omitted or no default language can be determined.
LABEL.*lang*	No	UTF-8-encoded LABEL.*lang* has the same syntax as the LINGUAS values. For each language in LINGUAS, a matching LABEL.*lang* is expected.
DESCRDIR	Yes	Package description directory (relative to product directory).
DATADIR	Yes	Package data directory (relative to product directory).
LANGUAGE	No	Default language code.
RELNOTESURL	No	URL from which to fetch release notes.
FLAGS	No	Product-specific capabilities.
KEY	Yes	SHA1 of the keys used in the media.
META	Yes	Metadata.
UPDATEURLS	No	URL of the update source.

Example of the `content` file:

```
PRODUCT SUSE Linux Add-on
VERSION 10.1
DISTPRODUCT SUSE-Linux-10.1-Add-on
DISTVERSION 10.1-0
```

```
VENDOR SUSE LINUX Products GmbH, Nuernberg, Germany
RELNOTESURL http://www.suse.com/relnotes/i386/SUSE-Linux/10.1/release-notes.rpm
ARCH.x86_64 x86_64 i686 i586 i486 i386 noarch
ARCH.i686 i686 i586 i486 i386 noarch
ARCH.i586 i586 i486 i386 noarch
ARCH.i486 i486 i386 noarch
ARCH.i386 i386 noarch
DEFAULTBASE i586
REQUIRES distribution-release
LINGUAS de en
SHORTLABEL SL 10.1
LABEL SUSE Linux Add-on 10.1
LABEL.de SUSE Linux Add-on 10.1
DESCRDIR suse/setup/descr
DATADIR suse
FLAGS update
LANGUAGE en_US
TIMEZONE America/Los_Angeles
META SHA1 04ef39995b65f02d81d6e1cc22fffda5c3a2a40e  EXTRA_PROV
META SHA1 b8b7146ce7b1e957be54227aa74f79ac95ca4e85  MD5SUMS
META SHA1 12ee25db081bf57beaef32b798262a18f9242216  packages
META SHA1 05279413404e8de127b30082a1ce70049718b849  packages.DU
META SHA1 4061102da14be0cb22ff7cf3ab5c77a27cd0f7df  packages.cs
META SHA1 acf5157177504747bbfa3638596d0afbd29c2762  packages.de
META SHA1 e2e479c179f94cca95b4f2a22facd0bf8cd0bd3a  packages.en
META SHA1 6090dd6ae0343f2470ceab5a1e8e92703d57db4e  packages.es
META SHA1 e2e479c179f94cca95b4f2a22facd0bf8cd0bd3a  packages.fr
META SHA1 e7829946b48a8cc96b0ab5a187e05c58279b063c  packages.hu
META SHA1 4061102da14be0cb22ff7cf3ab5c77a27cd0f7df  packages.sk
KEY SHA1 a108c6aab19fe604fa98ef299cdce6e6ba275f09
gpg-pubkey-0dfb3188-41ed929b.asc
KEY SHA1 af6ee559b573628d89a11239f113f9ece0839673
gpg-pubkey-1d061a62-427a396f.asc
KEY SHA1 b6a95b4cb3f3d0426ed25c0df350006915a803d3
gpg-pubkey-307e3d54-44201d5d.asc
KEY SHA1 0a4cffdc19c5544bc48d22474dd42586be5ac59e
gpg-pubkey-3d25d3d9-36e12d04.asc
KEY SHA1 30726e9a2959dbe9cace5765edd038e0538878ad
gpg-pubkey-9c800aca-40d8063e.asc
```

4 Package Descriptions

The package description files contain the dependencies, size, MD5 checksums, and package descriptions of all packages in the installation source. The encoding of the files is UTF-8. Files should be placed in `setup/descr/` of the data directory defined in the `content` file.

Special keywords are used in all description files. If a keyword starts with =, as in `=Ver`, the system reads only one line. To access multiple line, use a pair of keywords. The opening keyword starts with + and the closing keyword with –, as in `+Ver` and `–Ver`. In the keyword tables, the keywords are provided in their most common form. If the keyword normally has only a one line definition, it is shown with =. If the keyword is normally used in the pair version, see the start and end keywords.

To generate basic description files, in the data directory run the `create_package_descr -C` script, which is contained in the `autoyast2-utils` package. The `-C` option creates package descriptions with MD5 checksums of packages. Refer to `create_package_descr --help` for directions.

The script creates the following files in `setup/descr`:

`packages`
: A cache file for package data needed for package selection and dependency resolution. It contains pure package data and no user readable strings, such as translations.

`packages.`*`lang`*
: *`lang`* is replaced with ISO code of the description language (or language_country-code). For each language defined in `LINGUAS` in `content`, a `packages.`*`lang`* file should exist in the package description directory. This file contains translated strings (package summary, description, etc.) to be viewed by users. These files can be omitted.

`package.DU`
: This file contains disk usage information for each package and for directories used by the packages. It it used to approximate file system requirements, especially when multiple partitions (such as /usr, /var, and /opt) are used. Exact usage information cannot be computed by YaST because it depends heavily on hard or symbolic links (either in the package or in the file system) and the file system in use (such as ext2 or reiserfs).

4.1 The `packages` File

The package description directory must contain the `packages` file. This is basically a cache file for package data needed for package selection and dependency resolution. It contains pure package data and no user-readable strings. The file created by

`create_package_descr` is suitable for most add-on products, making it unnecessary to change it.

The keywords for this file are shown in Table 2, "List of Supported `packages` Keywords" (page 19). Where keywords correspond to dependencies, these are explained in terms of the keywords from Table 1, "Supported `content` Keywords" (page 14). However, these dependencies only relate to packages.

Table 2 *List of Supported packages Keywords*

Keyword	Value	Comment
=Ver	2.0	Version of the file format. For SUSE Linux 10.1, SUSE Linux Enterprise Server 10, and SUSE Linux Enterprise Desktop 10, use version 2.0.
=Pkg	*`name version release architecture`*	These four values identify a package unambiguously and are used as a key.
+Req -Req	*`library or executable`*	REQUIRES.
+Prq -Prq	*`library or executable`*	PREREQUIRES.
+Prv -Prv	*`library or executable`*	PROVIDES.
+Con -Con	*`library or executable`*	CONFLICTS.
+Obs -Obs	*`library or executable`*	OBSOLETES.

Keyword	Value	Comment
+Rec -Rec	*library or executable*	RECOMMENDS.
+Sug -Sug	*library or executable*	SUGGESTS.
+Fre -Fre	*library or executable*	This package is only considered if the resolvable specified here is already installed.
+Sup -Sup	*library or executable*	SUPPLEMENTS.
+Enh -Enh	*library or executable*	ENHANCES.
=Loc	*media_nr filename*	The path to the package is optional and defaults to `DATADIR`/*architecture*/*filename* (see Section 3, “The `content` File” (page 13) for `DATADIR`).
=Siz	*package-size installed-size*	Size in bytes.
=Tim	*buildtime*	Build time in time_t format (seconds since 00:00:00 UTC on January 1, 1970).
=Src	*name version release architecture*	Information about the source package. The architecture must be src or nosrc. If the architecture in the Pkg line already is src or nosrc, the Src value is discarded.
=Grp	*rpmgroup*	The RPM group from the package.

Keyword	Value	Comment
=Lic	*license*	License information.
=Cks	*type checksum*	The type can be SHA1 or MD5 followed by the checksum.
+Aut	*authors*	List of authors.
-Aut		
=Shr	*name version release architecture*	Identity of a package from which to retrieve all values not explicitly set in this package entry. Useful, for example, for optimized versions of the same package (for example, if i686 and i486 versions exists) or for source versions of a package.
+Key	*keywords*	Keywords from your package database, if relevant.
-Key		

4.2 The `packages.`*`lang`* File

For each language defined in `LINGUAS` in the `content` file, a `packages.`*`lang`* file should exist in the package description directory. This file contains translated strings (package summary, description, etc.) to be viewed by the user. These files can be omitted.

The file is encoded in UTF-8 and is line based. Lines starting with `#` are ignored. The `packages.lang` file starts with the `=Ver` tag followed by package entry keywords.

It contains the following information:

Package
: The package identifier (name version revision architecture)

Summary
: The package summary (label), a one line description of the package

Description
: A description of the package, possibly multiline

Installation Notify
: An informal message shown to the user if the package is selected, such as a test version warning or a commercial license

EULA Notify
: A EULA of the package that the user must accept to install the package

Deletion Notify
: An informal message shown to the user if the package is selected for deletion, such as a warning that the system is unusable without the package

Example of `packages.en`:

```
=Ver: 2.0
##----------------------------------------
=Pkg: yast2-pkg-bindings 2.13.79 2 i586
=Sum: YaST2 Package Manager Access
+Des:
This package contains a namespace for accessing the package manager
library in YaST2.



Authors:
--------
Arvin Schnell <arvin@suse.de>
Klaus Kaempf <kkaempf@suse.de>
Mathias Kettner <kettner@suse.de>
Stefan Hundhammer <sh@suse.de>
Stanislav Visnovsky <visnov@suse.cz>
-Des:
##----------------------------------------
=Pkg: yast2-theme-NLD 0.4.5 3 noarch
=Sum: YaST2 NLD Theme
+Des:
This package contains the YaST2 NLD theme.



Authors:
--------
Ken Wimer <wimer@suse.de>
Tuomas Kuosmanen <tigert@ximian.com>
Jakub Steiner <jimmac@ximian.com>
-Des:
```

Table 3 *List of Supported packages.lang Entry Keywords*

Keyword	Value	Comment
=Ver	2.0	Version of the file format. For SUSE Linux 10.1, SUSE Linux Enterprise Server 10, and SUSE Linux Enterprise Desktop 10, use version 2.0.
=Pkg	`name version release architecture`	These four values identify a package unambiguously and are used as a key.
=Sum	`summary`	One line summary.
+Des -Des	`description`	Multiple line package description.
+Ins -Ins	`text`	Text for user notification when the package is selected for installation.
+Del -Del	`text`	Text for user notification when the package is selected for deletion.
=Shr	`name version release architecture`	Identity of the package from which to retrieve all values not explicitly set in the current package entry. Useful, for example, for optimized versions of the same package (for example, if i686 and i486 versions exists) or for source packages.
+Eul: -Eul:	`text`	Text of the EULA. This text is displayed before package installation. If the user does not accept the EULA, package is not installed

4.3 The `packages.DU` File

This file contains disk usage information for each package and for directories used by the package. It is used to approximate file system requirements especially when multiple partitions (such as for `/usr`, `/var`, or `/opt`) are used. Exact usage information cannot be computed by YaST, because it depends heavily on hard or symbolic links (either in the package or in the file system) and the file system in use (such as ext2 or reiserfs). The file created by `create_package_descr` is suitable for most add-on products, making it unnecessary to change it.

The `packages.DU` file is optional, but no size estimations can be given if it is omitted. This also means that insufficient space warnings cannot be given until space runs out during product installation.

The cache file starts with a header defining the version. The file should be encoded in ASCII and is line based. Lines starting with # are ignored.

Example of the `packages.DU`:

```
=Ver: 2.0
##----------------------------------------
=Pkg: MozillaFirefox-translations 1.5.0.4 1.7 i586
+Dir:
/ 0 18939 0 62
usr/ 0 18939 0 62
usr/lib/ 0 18939 0 62
usr/lib/firefox/ 0 18939 0 62
usr/lib/firefox/chrome/ 18939 0 62 0
-Dir:
##----------------------------------------
=Pkg: zlib 1.2.3 15.2 src
+Dir:
/ 0 428 0 6
usr/ 0 428 0 6
usr/src/ 0 428 0 6
usr/src/packages/ 428 0 6 0
-Dir:
##----------------------------------------
=Pkg: zmd 7.1.1.0 39.40 src
+Dir:
/ 0 1049 0 13
usr/ 0 1049 0 13
usr/src/ 0 1049 0 13
usr/src/packages/ 1049 0 13 0
-Dir:
```

Table 4 *List of Supported packages.DU Entry Keywords*

Keyword	Value	Comment
=Ver	2.0	Version of the file format. For SUSE Linux 10.1, SUSE Linux Enterprise Server 10, and SUSE Linux Enterprise Desktop 10, use version 2.0.
=Pkg	*name version release architecture*	These four values identify a package unambiguously and are used as a key.
+Dir -Dir	*directory dir_size size_subdirs files_in_dir files_in_subdir*	The *directory* should be / for the root directory or a relative path to the root directory for others. For *dir_size*, enter the size of data stored only in the main directory in Kb. For *size_subdirs*, enter the size of data stored in subdirectories in Kb. *files_in_dir* is the number of files stored in the main directory. *files_in_subdir* is the number of files stored in subdirectories.

4.4 Pop-Up License Windows

For packages with a proprietary license, it is a good idea to display information about the license. If the user agrees with the license, YaST installs the package. If the users disagrees, YaST does not install the package.

The package license note is a part of package description but it is not created by the `create_package_descr` script. A license note must be added manually.

To add package license information, open the description file `packages.lang`, find the package description, and add the text of the license between the tags `+Eul:` and `-Eul:` at the end of the entry for that package, for example:

```
=Pkg: test-package 7.0.63.0 6 i586
=Sum: Test Package
```

```
+Des:
This is test package.
-Des:
+Eul:
Test Package End User License Agreement
    ....
    ....
-Eul:
```

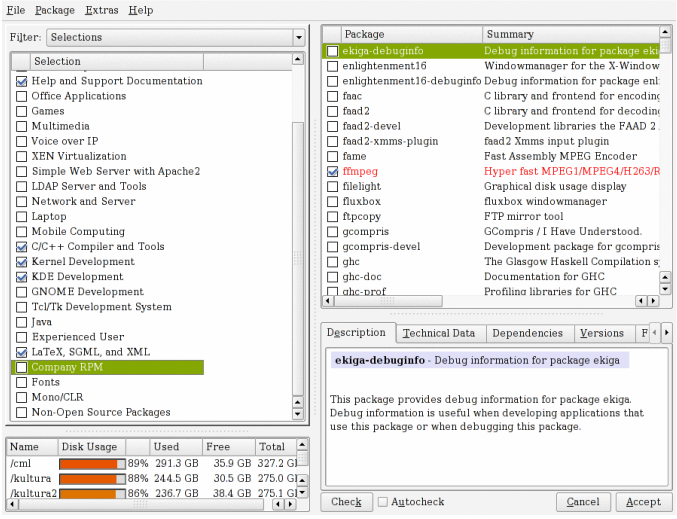
Note

To display a license for the entire add-on product before its installation, use `license.zip`. See Section “Special Files in the `media.1` Directory” (page 10).

5 Selections and Patterns

Both selections and patterns are designed to provide a group of packages for installation in the YaST pattern or selection filter with one click. They have similar syntax, but they are used for different products. Selection are used in SUSE Linux and patterns are used in the SUSE Linux Enterprise products.

Figure 1 *The Selection of the Add-On in the Package Manager*



Patterns versus Selections

If you create an add-on product for SUSE Linux 10.1, create selections. If you create add-on product for SUSE Linux Enterprise products, create patterns. SUSE Linux 10.1 might not work correctly with patterns and SLE products might not work correctly work with selections.

5.1 Selections

To have a functional selection visible in the YaST selection filter, prepare two files:

`selections`
: File with the list of all selection files in the add-on media, each selection on separate line.

`selection-name-version.arch.sel`
: File with a selection definition. The name should resemble `Multimedia-10.1-3.i686.sel`.

These files must be in the `setup/descr` directory in the add-on data directory. For multiple selections, prepare more `*.sel` and include them in `selections`. The keywords for descriptions are shown in Table 5, "List of Supported Selection Entry Keywords" (page 28). Dependency-related keywords are defined in terms of corresponding keywords from Table 2, "List of Supported `packages` Keywords" (page 19), but only relate to selections.

Note

The packages in the list must be written without version numbers and without any suffixes.

Table 5 *List of Supported Selection Entry Keywords*

Keyword	Value	Comment
=Ver:	Syntax version	Minimum parser version needed to parse this file. Set to 4.0 for SUSE Linux 10.1.
=Sel:	*name version release architecture*	All four values are mandatory. Supported architectures are i586, i586, ppc, x86_64, and noarch.
=Sum:	*summary*	One line label in the default language.
=Sum.*lang*:	*summary*	One line language-specific label.
+Des: -Des:	*description*	Multiple line description in the default language.
+Des.*lang*: -Des.*lang*:	*description*	Multiple line description, language specific.
=Cat:	add-on	Use the type `add-on` for selections for add-on products.
=Vis:	true or false	If set to `true`, the selection is shown to the user. Selections set to `false` are hidden from the user.
=Ord:	*ordering*	This three-digit integer value defines the order of the selection when listing multiple selections in the user interface.
+Req: -Req:	*selection*	REQUIRES.
+Prv:	*selection*	PROVIDES.

Keyword	Value	Comment
-Prv:		
+Con:	*selection*	CONFLICTS.
-Con:		
+Obs:	*selection*	OBSOLETES.
-Obs:		
+Rec:	*selection*	RECOMMENDS.
-Rec:		
+Sug:	*selection*	SUGGESTS.
-Sug:		
+Ins:	*package list*	List of packages to install.
-Ins:		
+Ins.*lang*:	*package list*	List of language-specific packages to install if *lang* is used in the system.
-Ins.*lang*:		
+Del:	*package list*	List of packages to delete.
-Del:		
+Del.*lang*:	*package list*	List of packages to delete if .*lang* is used in the system.
-Del.*lang*:		

Example of a selection file:

```
# SuSE-Linux-Package-Selection 10.1-73.x86_64 -- (c) 2004 SuSE Linux AG
# Needs parser version 4.0 or greater

=Ver: 4.0
```

```
=Sel:  Mobile 10.1 73 x86_64

=Sum: Mobile Computing
=Sum.bg: Мобилен компютър
=Sum.cs: Mobilní komunikace
=Sum.da: For bærbare computere
=Sum.de: Mobile Computernutzung
=Sum.el:
=Sum.en: Mobile Computing

+Des:
Support for mobile devices like Palms, mobile phones.
-Des:
+Des.bg:
Support for mobile devices like Palms, mobile phones.
-Des.bg:
+Des.cs:
Podpora mobilních zařízení jako PDA a mobilních telefonů.
-Des.cs:
+Des.da:
Support for mobile devices like Palms, mobile phones.
-Des.da:
+Des.de:
Unterstützung für mobile Ger te wie Palms und Mobiltelefone.
-Des.de:
+Des.el:
Support for mobile devices like Palms, mobile phones.
-Des.el:
+Des.en:
Support for mobile devices like Palms, mobile phones.
-Des.en:


+Req:
X11
Laptop
-Req:

=Cat: addon

=Vis: true

=Ord: 50

+Ins:
bluez-cups
bluez-firmware
bluez-hcidump
bluez-libs
bluez-utils
gnokii
gnome-bluetooth
```

```
ial
initial
kdenetwork3-wireless
kdepim3-mobile
kdepim3-sync
kdeutils3-laptop
lineak_defaultplugin
lineak_kde
lineak_xosdplugin
lineakd
multisync
multisync-backup
multisync-evolution
multisync-irmc
multisync-irmc-bluetooth
multisync-kdepim
multisync-ldap
multisync-opie
multisync-palm
multisync-syncml
ndiswrapper
netapplet
obexftp
openobex
pmtools
scpm
unison
usbview
viki
-Ins:
```

5.2 Patterns

To have a functional pattern visible in the YaST pattern filter, prepare two files:

`patterns`
: File with the list of all pattern files in the add-on media, each pattern on a separate line.

`pattern-name-version.arch.pat`
: File with a pattern definition. The name should resemble `Multimedia-10.1-3.i686.pat`.

These files must be in the `setup/descr` directory in the add-on data directory. For multiple patterns, prepare more `*.pat` files and include them in `patterns`. The keywords for descriptions are shown in Table 6, “List of Supported Pattern Entry

Keywords" (page 32). Dependency-related keywords are defined in terms of corresponding keywords from Table 2, "List of Supported `packages` Keywords" (page 19), but only relate to patterns.

Note

The packages in the list must be written without version numbers and without all suffixes.

Table 6 *List of Supported Pattern Entry Keywords*

Keyword	Value	Comment
=Ver:	Syntax version	Minimum parser version needed to parse this file. Should be set to 5.0 for SUSE Linux Enterprise 10 products.
=Pat:	*`name version release architecture`*	All four values are mandatory. Supported architectures are i586, i586, ppc, ppc64, x86_64, ia64, and noarch.
=Sum:	*`summary`*	One line label in the default language.
=Sum.*`lang`*:	*`summary`*	One line language-specific label.
+Des: -Des:	description	Multiple line description in the default language.
+Des.*`lang`*: -Des.*`lang`*:	*`description`*	Multiple line description, language specific.
=Cat:	*`category`*	One line category in the default language to group patterns. Categories are intended for the user and can be specified freely.
=Cat.*`lang`*:	*`summary`*	Language-specific version of the category.

Keyword	Value	Comment
=Ico:	*filename*	Icon filename. If unspecified, the pattern name is used instead (with blanks in the name replaced by underscores). If the filename does not include a `.png` or `.jpg` extension, `.png` is appended. If no path is specified, icons are searched for in the theme icon path (first `/usr/share/YaST2/theme/current/icons/32x32/apps/` then `/usr/share/YaST2/theme/current/icons/48x48/apps/`). Absolute and relative paths (to the theme path `/usr/share/YaST2/theme/current/`) are allowed.
=Vis:	true or false	Set whether the pattern should be visible in the user interface.
=Ord:	*ordering*	This three-digit integer value defines the order of the pattern when listing multiple patterns in the user interface.
+Req:	*pattern*	REQUIRES.
-Req:		
+Prv:	*pattern*	PROVIDES.
-Prv:		
+Con:	*pattern*	CONFLICTS.
-Con:		
+Obs:	*pattern*	OBSOLETES.
-Obs:		
+Rec:	*pattern*	RECOMMENDS.

Keyword	Value	Comment
-Rec:		
+Sup:	pattern	SUPPLEMENTS.
-Sup:		
+Sug:	*pattern*	SUGGESTS.
-Sug:		
+Prq:	*package list*	List of packages to install.
-Prq:		
+Prc:	*package list*	Recommended packages. These packages are installed by default but can be removed without complaint.
-Prc:		
+Fre:	*pattern*	The current pattern is only considered for installation is the pattern specified here is installed.
-Fre:		

Example of a pattern file:

```
# Pattern for install Company files

=Ver: 1.0

=Pat: Company 1.0 1 noarch

=Cat: Add-on

=Sum: Test add-on

+Des:
One package pattern
-Des:

=Vis: true

=Ord: 8020

+Prq:
```

```
Novell-ricoh-fonts
-Prq:
```

Preselecting the Pattern

If you want to preselect your pattern, add the pattern name to `REQUIRES` in the `content` file. For example, to add a pattern with the name *`example-pattern`*, add the following to the `content` file:

```
REQUIRES pattern:example-pattern
```

For more information about `content` creation, see Section 3, “The `content` File” (page 13).

Creating Dependencies between Patterns

To create dependencies between patterns, use the pattern definition file and the keywords `+/-Req:`, `+/-Con:`, `+/-Obs:`, `+/-Fre:`, and `+/-Sup:`.

Note

The patterns in the list must be written without version numbers and without all suffixes.

The most common situation is a pattern that requires other patterns. To define patterns that must be installed for your new pattern, use the keyword `+/-Req:`. For example, if the pattern `base` must be installed for your pattern, use:

```
+Req:
base
-Req:
```

It sometimes happens that two patterns include the same or incompatible packages. In this situation, only one of these patterns should be installed. If both patterns are installed, the system can be unstable. To ensure this does not happen, use the keyword `+/-Con:`. For example, if the pattern *`minus`* should not be installed with with pattern *`plus`*, add the following to the definition file of the *`minus`* pattern:

```
+Con:
plus
-Con:
```

Include the following in the definition file of the *`plus`* pattern:

```
+Con:
minus
-Con:
```

If your new pattern replaces an older pattern that is already installed, the new pattern can uninstall the old one. To do this, use the keyword `+/-Obs:`. You can also provide all capabilities of the obsolete pattern in your new pattern with `+/-Prv:`.

To have your pattern only be considered for installation if something else is installed, use the keyword `+/-Fre:`. If you use the keyword `+/-Sup:`, your pattern is installed if this capability is provided by an installed pattern. The dependency resolver installs it. Uninstalling it is silently accepted.

You can have multiple patterns with same capability. To install only one, define a pattern group with the keyword `+/-Prv:` in all pattern definition files that provide the capability. If a pattern has the capability in `+/-Req:` and the system finds more than one matching pattern, it asks which one the user wants to install.

For examples of the various keywords in patterns, refer to the SUSE Linux Enterprise Desktop 10 patterns. For more information about `content` creation, see Section 3, “The `content` File” (page 13).

6 Signatures and Other Security Issues

To provide more security and data evidence of integrity, SUSE Linux 10.1, SLES 10, and SLED 10 come with signature support. This support has two levels: the package level and the metadata level.

This means that if you want to provide an add-on media, you should sign your RPM packages first then also the add-on media metadata. If you do not sign packages or metadata, a warning about untrusted media appears during add-on installation. Part of the security is MD5 checking of the files on the add-on media and the SHA1 message digest of the `package.lang` files and keys used.

In this section, find information about the metadata level. If you need information about signing RPM packages, read `http://www.rpm.org/max-rpm/s1-rpm-pgp-signing-packages.html`.

6.1 Signing the Add-On Product

The add-on media metadata is stored in `content` and `media.1/products`. To sign them, do the following:

1. If you do not have a key, create one. To do so, use:

   ```
   gpg -q --gen-key
   ```

2. Sign the files `media.1/products` and `content` and export the key used to the directories of the files. To do so, you can use:

   ```
   gpg --detach-sign -u YOUR_KEY -a content
   gpg --export -a -u YOUR_KEY > content.key
   gpg --detach-sign -u YOUR_KEY -a media.1/products
   gpg --export -a -u YOUR_KEY > media.1/products.key
   ```

3. Export your key. To do so, use:

   ```
   gpg --export -a YOUR_KEY > /add-on-media/gpg-pubkey-YOUR_KEY.asc
   ```

Replace *YOUR_KEY* with your key ID. To find it, use the command:

```
gpg --list-secret-keys|grep "^sec"|sed -e 's/.*\///;s/ .*//g;'|tail -n 1
```

Using Multiple Keys

You can sign RPM packages and add-on media with different keys. In such a situation, you should include all keys and add `pubring.gpg` to the media. This file is a public keyring and contains public keys used for verifying the signature of the add-on media.

6.2 MD5

If a directory includes files, you should create an `MD5SUMS` file in the directory and add MD5 checksums for all included files to it. To do so, use `md5sum`, for example:

```
md5sum FILENAME >> MD5SUMS
```

Replace *FILENAME* with the names of your files from the add-on media.

You do not need to create md5 checksums for directories with subdirectories only and for the add-on media root directory. For information about `md5sum`, use `man md5sum` or `md5sum -h`.

6.3 SHA1 and the `content` File

The SHA1 message digest of the `package.lang` files and keys should be included in the `content` file with the keywords `META` and `KEY`. For normal digests, use `META` with `SHA1`. For keys use `KEY` with `SHA1`. To create the digest, use `sha1sum`.

Example of `META` and `KEY` in the `content` file:

```
META SHA1 04ef39995b65f02d81d6e1cc22fffda5c3a2a40e  EXTRA_PROV
META SHA1 b8b7146ce7b1e957be54227aa74f79ac95ca4e85  MD5SUMS
META SHA1 12ee25db081bf57beaef32b798262a18f9242216  packages
META SHA1 05279413404e8de127b30082a1ce70049718b849  packages.DU
META SHA1 4061102da14be0cb22ff7cf3ab5c77a27cd0f7df  packages.cs
META SHA1 acf5157177504747bbfa3638596d0afbd29c2762  packages.de
META SHA1 e2e479c179f94cca95b4f2a22facd0bf8cd0bd3a  packages.en
META SHA1 6090dd6ae0343f2470ceab5a1e8e92703d57db4e  packages.es
META SHA1 e2e479c179f94cca95b4f2a22facd0bf8cd0bd3a  packages.fr
META SHA1 e7829946b48a8cc96b0ab5a187e05c58279b063c  packages.hu
META SHA1 4061102da14be0cb22ff7cf3ab5c77a27cd0f7df  packages.sk
KEY SHA1 a108c6aab19fe604fa98ef299cdce6e6ba275f09
gpg-pubkey-0dfb3188-41ed929b.asc
KEY SHA1 af6ee559b573628d89a11239f113f9ece0839673
gpg-pubkey-1d061a62-427a396f.asc
KEY SHA1 b6a95b4cb3f3d0426ed25c0df350006915a803d3
gpg-pubkey-307e3d54-44201d5d.asc
KEY SHA1 0a4cffdc19c5544bc48d22474dd42586be5ac59e
gpg-pubkey-3d25d3d9-36e12d04.asc
KEY SHA1 30726e9a2959dbe9cace5765edd038e0538878ad
gpg-pubkey-9c800aca-40d8063e.asc
```

7 Autorun

To maximize the comfort of the installation of the add-on, you can use the autorun functionality. With this functionality, the system calls the YaST add-on module automatically after an add-on medium is inserted into the CD or DVD drive.

For autorun functionality, create the following two files in the add-on's root directory:

- `autorun.inf` for Windows systems

- `autorun.sh` or `setup.sh` for SUSE Linux systems

For `autorun.inf`, prepare files with an icon first, such as `add-on.ico`. Then create the file `autorun.inf` with the content:

```
[autorun]
     icon = add-on.ico
```

`autorun.sh` and `setup.sh` have different functionality. Use `autorun.sh` if you want to run the add-on product as a normal user. If you need privileges, for example, for automatic installation with YaST or for starting the YaST2 add-on module, use `setup.sh`. With `setup.sh` in the root directory of the add-on medium, the system first asks for the `root` password then uses `gnomesu` or `kdesu` to run your script.

Add the following lines to the `autorun.sh` or `setup.sh` file:

```
#!/bin/sh
     /sbin/yast2 add-on cd:///
```

For an add-on DVD, replace `cd:///` with `dvd:///`.