



Automated Installation, Configuration, and Update for SLES and OES

Micro Focus Consulting

September 2023

Legal Notice

Copyright 2023 Open Text

The only warranties for products and services of Open Text and its affiliates and licensors ("Open Text") are as may be set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Open Text shall not be liable for technical or editorial errors or omissions contained herein. The information contained herein is subject to change without notice.

Contents

About This Guide	5
1 Solution Overview	7
What's New in CIF for SLES15 SP4 and OES2023	9
Part I Using AutoYaST to Install SLES or OES	11
2 AutoYaST Work Flow Overview	13
Boot Process	14
Installation Process	14
Configuration Process	14
3 Requirements for Unattended Installations via AutoYaST	15
Installation Repositories	15
Local Installation Repositories	15
Remote Installation Repositories	16
Network Repository Server	16
AutoYaST Control File	16
Control File Structure	18
Class Files	19
Retrieving a Control File	20
Installation Boot Medium	21
Custom Boot ISO	21
Network Boot Environment	30
4 Installing and Configuring an AutoYaST Server	39
Hardware Requirements	39
Disk Partition Layout and Directory Structure	40
Software Requirements	40
Installation Repositories	41
Control File and Class File Repository	41
Apache Web Server Configuration	41
5 AutoYaST Extended – The Configuration File Approach	45
Design	45
Implementation	46
Overview	46
Directory Structure of the Consulting Installation Framework	48
Configuration Files	50
XML Snippets	55
Info Files	59

Libraries	60
The Default File Again	60
The Script pre-fetch.sh	61
Post Installation Scripts	63
Server Upgrade Using AutoYaST	65
6 Miscellaneous	69
Advanced Installation	69
Troubleshooting and Monitoring	71
Part II Using ZENworks to Manage SLES or OES	73
7 ZENworks Configuration Management Introduction	75
8 Managing the ZENworks Management Zone	77
Guiding Principles	77
zman	79
Top Level Folder Structure in the ZENworks Management Zone	79
Servers and Server Groups	80
Folder Structure for Servers and Server Groups	80
Server Groups	84
Device Registration	87
Linux Bundles and Bundle Groups	90
Folder Structure for Bundles and Bundle Groups	91
Bundle Groups	93
Subscriptions	94
Managing Pool Bundles, Update Bundles and Update Bundle Groups	102
Managing Configuration Bundles and Configuration Bundle Groups	111
Bundles in the Folder SLES-SERVICES	119
SLESALL_NOV-BOOTSPLASH-CONFIG	121
Agent Commands	125
General Commands	125
Bundle Commands	125
Package Management Commands	126
Registration Commands	126
Diagnostics and Debugging	126
ZCM Server	126
ZCM Agent	127
Part III How To Build Your Own Installation Framework	129
9 Building Your Own AutoYaST Server	131
10 Configuring Your Own ZENworks Management Zone	137
11 Building Your Own Preboot Execution Environment	141

About This Guide

The deployment and maintenance of a large number of server is hardly possible without a certain level of automation. Based on the experience we gained from numerous successful projects Micro Focus Consulting Germany in cooperation with SUSE Consulting Germany has created this Best Practices Guide to share the solution we use to install, configure and maintain systems based on SUSE Enterprise Linux Server (SLES) or MicroFocus Open Enterprise Server (OES).

NOTE: MicroFocus is now OpenText.

This guide is not intended to replace any part of the official documentation or any training material. We assume that you have a working knowledge of SLES, OES, and ZENworks Configuration Management (ZCM). If in doubt, we highly recommend that you consult the related product documentation:

For more information about	See
Open Enterprise Server 2023 Documentation	OES 2023
SUSE Linux Enterprise Server 12 Documentation	SLES12
SUSE Linux Enterprise Server 15 Documentation	SLES15
ZCM2020 Documentation	ZCM 2020

We also assume that you have a working ZCM environment available and will only describe the modifications that you need to make to add the solution to your exiting ZENworks Management Zone. In case you should not have a working ZCM environment we strongly suggest that you consider setting up a single server ZCM zone using the [ZCM 2020 Appliance](#).

Audience

This guide is primarily intended for skilled OES administrators with a good knowledge of Linux, OES and ZCM who aim to install, configure and maintain their systems in a standardized fashion.

Experienced Linux administrators who are interested in best practices for the deployment, configuration, and update/upgrade of SLES and OES systems should also benefit from reading this guide.

1 Solution Overview

Any repetitive task carried out by humans has the potential of more or less pronounced deviations. If you have attempted to install and configure a number of systems with identical configuration you will know that there always are some differences, no matter how hard you try to avoid them. Each of these discrepancies in itself might be considered minor however, in combination they can result in issues that may be difficult to analyze and to resolve.

To avoid this type of problems Micro Focus Consulting Germany has started to work on a framework to deliver a standardized installation and configuration in the OES2 time frame. The design criteria that had been defined back then can be summarized as follows:

- ♦ Automated installation with as little human intervention as possible including any updates that are available at installation time.
- ♦ Automated configuration of the installed system based on the services that the system needs to provide.
- ♦ Regular automated updates with a defined set of patches.

The first criterion is fairly easy to meet. In fact there are many solutions available that can automate the installation of a Linux system. For SLES based systems [SUSE Linux Enterprise Server AutoYaST](#) is the utility of choice.

Integrating updates into the installation process also is an easy undertaking. A SLES based systems can consume update repositories from the customer centers (SUSE Customer Center (SCC) for SLES, Novell Customer Center (NCC) for OES) if the system being installed can access the internet.

If internet access is not available a local system needs to replicate the required channels and provide them to the systems being installed. On SLES12 such functionality can be provided by the Subscription Management Toolkit (SMT). On SLES 15 this can be achieved using the Repository Mirroring Tool (RMT). Both solutions can only access the SCC.

[Micro Focus Subscription Management Tool 2.0](#) (MF-SMT) can be installed on SLES15 SP4 or OES2023 and can access the NCC.

Automated configuration can be implemented if the ZENworks Adaptive Agent (ZAA) is deployed and the system to be managed is registered to a ZENworks Management Zone. SUSE Manager (SUMA) provides similar functionality and has also been used in some projects in particular for customers that already use SUMA to manage a large SLES server population and want to use the same environment to manage their OES systems. There are other solutions that could also be considered to implement the configuration task.

In addition ZCM and SUMA both can be used to maintain defined patch levels as long as they are needed, whereas SMT and RMT have limited capabilities in this area.

Historically our solution has been based on AutoYaST and ZENworks Linux Management (ZLM). ZLM has been replaced by ZCM once it became available. Therefore this guide describes the implementation of the solution based on AutoYaST and ZCM.

However, you will find that very little modification is required if for instance you should want to use SMT to provide updates to your systems. Using SUMA instead of ZCM to perform the configuration tasks is also possible but will be more involved than replacing ZCM as provider of the update repositories.

By enhancing SUSE's automatic installation method through AutoYaST and combining it with ZCM, we have developed the Consulting Installation Framework (CIF) to provide standardized installation, configuration, and maintenance of SUSE Linux Enterprise Server (SLES) as well as MicroFocus Open Enterprise Server (OES) that is easy to implement and use.

CIF supports OES and SLES from version 11. It may be possible to deploy older versions but this has not been tested with the current code. CIF also supports the deployment of Domain Services for Windows (DSfW) domain controllers for the forest root domain. Please note that DSfW child domains are not supported by the current CIF release.

Most customers maintain test systems where they first apply new updates or evaluate new configurations before they are deployed into production. We refer to these different environments as staging areas and explain how to support three such environments – development, testing, and production. However, our solution can very easily be extended to support additional environments if the need should arise.

In principal building your own installation framework requires the configuration of services such as an installation repository, remote access to this installation repository, a customized boot medium or PXE environment and the AutoYaST control file that defines the properties of the target devices.

[“Using AutoYaST to Install SLES or OES” on page 11](#) explains how AutoYaST works in general and what is required to set up an installation server. Most of this section focuses on the customization of AutoYaST that has been developed for and is used by CIF.

[“Using ZENworks to Manage SLES or OES” on page 73](#) presents the methodology to use ZENworks Configuration Management to provide management of configuration settings across many servers as well as the management of updates for SLES and OES servers. This includes well-defined (frozen) patch levels for different staging areas such as development, testing and production. The management of field test files (FTF) and the upgrade to the next support pack are also covered in this section.

Finally, [“How To Build Your Own Installation Framework” on page 129](#) explains what is available on the [CIF download site](#) and how to use it to build your own AutoYaST/ZCM environment.

What's New in CIF for SLES15 SP4 and OES2023

Since the last release of this guide in December 2020 the Consulting Installation Framework has been enhanced to support SLES15 SP4 and OES2023.

However, as the principals on which our solution is based do still apply the decision has been made to not update the text of the guide and the figures as a whole. Instead this section briefly explains what has changed and refers the reader to sections that have been added to this guide.

The updated version of the CIF has been extensively tested with AutoYaST on SLES15 SP4 and ZENworks Configuration Management 2020.3. Earlier releases of SLES have not been tested but should work.

IMPORTANT: Due to changes in the way dependencies between packages can be defined you must have Official FTF 959 for ZENworks 2020 Update 3 applied to your ZCM Primary servers before creating any YUM repositories for SLES5 SP4 or OES2023. For more information, see [Boolean Dependencies](#).

Using YUM repositories provided by ZCM as add-on repositories during the installation of SLES15 SP4 or OES 2023 will also only succeed if you have FTF 959 deployed before creating the YUM repositories.

WARNING: YUM repositories for SLES15 SP4 or OES2023 created without FTF 959 or on earlier releases of ZCM will not be operational and can break a system beyond repair!

As the ZENworks Adaptive Agent does not understand Boolean Dependencies, `zac bin` or `zac up` must not be used to apply updates to either SLES15 SP4 or OES2023. For this reason we no longer recommend to assign the ZCM Update Bundle Groups to device groups. See, [Assigning a Frozen Patch Level](#).

Use the `zypper patch` command instead to apply the updates available from the YUM repositories of the ZCM Update Bundle Groups. With `btrfs` using `zypper patch` has the additional advantage that you automatically create file system snapshots of your system before and after the patches are applied.

You can still use `zac bin` or `zac up` to update releases of SLES or OES that do not use Boolean Dependencies but we strongly recommend to switch to `zypper patch` and to remove the assignment between ZCM Update Bundle Groups to device groups for all releases.

Micro Focus Subscription Management Tool 2.0 that became available in November 2022 is supported on SLES15 SP4 or on OES2023 and can be used to provide updates to all OES versions.

For production systems using `btrfs` we now recommend a minimum size of 70 GB for the system disk while 30 GB are now recommended for test systems.

For new installations we recommend to configure your systems with UEFI firmware. This requires some changes to the disk partitioning to accommodate the EFI system partition. The partitioning XML snippets `*UEFI*.xml` have the required modifications.

SLES15 SP4 and therefore OES2023 use `chronyd` instead of `ntpd` to receive time. By default `chronyd` uses a set of public NTP servers. As many customers do not allow their servers to access any system on the internet the function `remove_chrony_include` that comments the `include` command at the end of `/etc/chrony.conf` has been added to `post-inst.sh`. to prevent the inclusion of public time sources on any SLES15 SP4 or OES203 system installed with the CIF.

The definitions for the different OES2023 Service Types have been added to `Your_CUSTOMER.txt` and the software selection files for these OES2023 Service Types are also provided.

OES2023 adds the Unified Management Console (UMC) intended to replace iManager for the management of OES services. The configuration information for this new component is contained in the XML class file `umc.xml`. Anticipating that customers will deploy UMC in the same way as the currently deploy iManager we have added the new service type `OES2023_EDIR_UMC` to `Your_CUSTOMER.txt`. The corresponding software selection is defined in `soft-oes2023_edir-umc.xml`.

Parameters to accommodate the UMC configuration information have been added to `Your_CUSTOMER.txt`, `Your_Tree.txt` and to `Your_Location.txt`.

Finally the new release contains numerous corrections and clarifications based on the feedback received from our customers. Remember, the comments in the configuration files are the only documentation that we maintain.

There are also a number of changes with respect to the ZCM configuration bundles we provide. The following bundles are new:

- ♦ `SLES-SERVICES > SLESALL_MFC-DHCP-BOND0`
- ♦ `SLES-SERVICES > SLESALL_MFC-DISABLE-ZISLNX-SERVICE`

The following bundles have been updated:

- ♦ `iMan-BASECONFIG > IMAN_NOV-change-config-xml`
- ♦ `OESLES-SERVICES > OESALL_NOV-BASHRC`
- ♦ `SLES-SERVICES > SLESALL_NOV-NTP-SYSCONFIGS-SERVICES`

Finally, ZCM configuration bundle `OES2018_MFC-DISABLE-CIS-AGENTS` in folder `OES-SERVICES` needs to be replaced with bundle `OES2018_MFC-DISABLE-OES-AGENTS`.

Upon popular demand we have included the `net-tools-deprecated` package that provides legacy commands such as `route`. Therefore you need to replicate the `OES2023-SLE-Module-Legacy15-SP4-Pool` channel as dependency bundle and create a YUM repository for it.

In each chapter of [Part III, "How To Build Your Own Installation Framework,"](#) on page 129 we have added a section with detailed instructions on how to update an existing build environment to the latest CIF release.

Using AutoYaST to Install SLES or OES

Micro Focus Consulting Germany in cooperation with SUSE Consulting Germany has developed a methodology to quickly and reproducibly install SUSE Linux Enterprise Servers (SLES) and Novell Open Enterprise (OES) using AutoYaST.

This section describes the principals of AutoYaST and how to setup your own AutoYaST server.

In [“AutoYaST Work Flow Overview” on page 13](#) we briefly describe the different stages of an AutoYaST installation.

[“Requirements for Unattended Installations via AutoYaST” on page 15](#) discusses what is needed to build an installation framework, the repositories and the server providing them, the control file, that governs the installation; and how to use a Custom Boot ISO or PXE to initiate the installation.

[“Installing and Configuring an AutoYaST Server” on page 39](#) gives our recommendations on how to build an AutoYaST server.

[“AutoYaST Extended – The Configuration File Approach” on page 45](#) explains how our solution separates the classical AutoYaST profile into service specific XML snippets and configuration files and how they are dynamically combined into an AutoYaST profile at installation time.

All scripts and the configuration files are available at the [CIF download site](#). Refer to [“Building Your Own AutoYaST Server” on page 131](#) to learn how to use these downloads to build your own AutoYaST server.

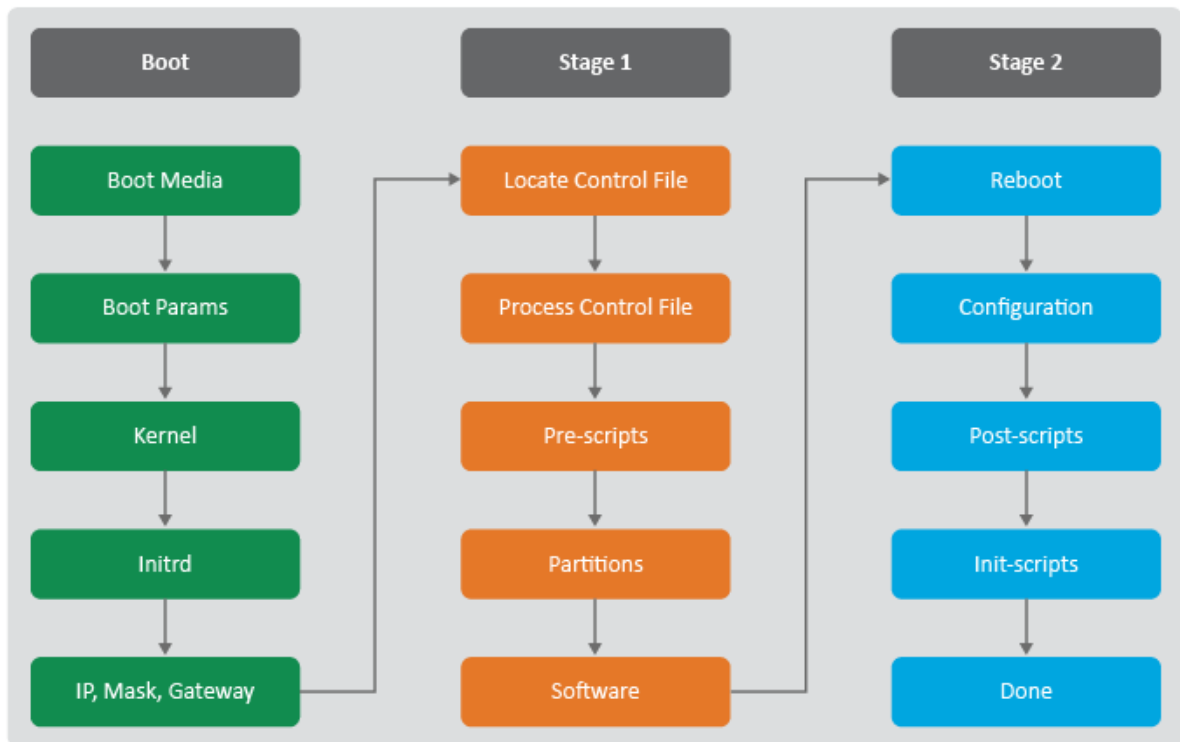
However, before you proceed, please read this chapter carefully to understand how AutoYaST is used as a part of our solution.

2 AutoYaST Work Flow Overview

The automated installation of any SLES based system consists of three distinct steps:

- ♦ The installation system is booted and network connectivity is established.
- ♦ In Stage 1 the AutoYaST control file is located and retrieved. Pre-scripts, if any, are executed, the system is installed and rebooted.
- ♦ In Stage 2 the system is configured and post-scripts and init-scripts, if any, are executed.

Figure 2-1 AutoYaST Process Overview



This is illustrated in the figure above. Note that even so a manual installation of SLES12 and later is performed in one stage an AutoYaST installation of these releases is still executed in two stages.

Boot Process

When a computer is powered on and has performed its power-on self test (POST), firmware routines are executed that recognize and initialize hardware components such as hard disk controllers, hard disks, network adapters, etc.

Afterwards control of the boot process is passed to the boot loader, which is located on a bootable device (for example, CD-ROM, hard disk, network (PXE)). The boot loader starts a kernel and optionally an initial RAM disk (initrd). Up to SLES 11 the default boot loader has been `grub` which has been replaced with `grub2` in SLES 12 and later.

Most operating systems accept custom boot options to pass instructions to the kernel or to the `initrd`. These instructions can be used to influence hardware initialization or which functions to execute. One application of custom boot options is to initiate an automated installation.

Installation Process

The boot process of a system being installed must execute other routines than the boot process of a system already installed. Some of these routines are executed in the background while other routines may require human intervention.

The routines and modules that are necessary to install a new SLES based system are contained in a special `initrd` provided by the boot medium. This typically large `initrd` differs dramatically from the `initrd` used during a normal operating system boot.

SLES based systems use `linuxrc` to setup the installation environment. `linuxrc` accepts custom boot options to influence how the system will be installed. Some options ensure a proper network setup, if necessary. Other parameters determine which installation repositories shall be used. An unattended installation via AutoYaST is initiated by specifying the `autoyast=` boot option:

```
autoyast=<URL to AutoYaST control file>
```

This parameter is recognized by the `initrd` and the installation process is taking another direction. Instead of prompting the administrator for various system settings an AutoYaST control file is retrieved from the URL specified. YaST modules then parse the control file and perform an unattended installation according to the information provided in the AutoYaST control file.

Once all actions specified in the control file have been executed the system being installed is automatically rebooted.

Configuration Process

After the initial reboot YaST processes all configuration information for the patterns and packages that have been installed in the previous step. Post-scripts and init-scripts specified in the control file will also be executed before the automated installation is completed.

3 Requirements for Unattended Installations via AutoYaST

The single hard requirement for an AutoYaST installation is a valid AutoYaST control file. However, we recommend that you set up some additional components to build a standardized and easy-to-maintain installation framework.

Installation Repositories

Any installation requires one or multiple repositories that contain the patterns and packages for the operating system and Add-On products to be installed.

Up to and including OES2015 the Add-On product OES could be installed together with the underlying SLES operating system or in a separate step, after SLES had been installed.

With OES2018 OES has become a product in its own right and can only be installed from the OES Install Media, which is a single integrated install media that includes both SLES and OES. Installation of OES on top of an existing SLES server is no longer supported.

Optionally updates for the operating system and any Add-On product can be deployed as part of the installation process. The updates can be provided by different types of repositories:

- Local repositories on CD-ROMs, DVDs, HDs or USB devices, etc.
- Remote repositories accessible over network protocols such as HTTP, FTP, NFS, or SMB. These repositories can be advertised via SLP.

Supported repository types are `yast2` and `rpm-md`. `yast2` repositories are corresponding to SUSE's installation media format and are the only repository type that can be used for operating system installations.

`rpm-md` (Repository Metadata) is the original repository type of YUM (Yellowdog Updater, Modified) and only supported for the installation of Add-On products or updates.

Local Installation Repositories

Local installation repositories have advantages over remote repositories in rare cases where network bandwidth and/or network latency are limiting factors.

One example would be the installation of a branch office server behind a slow WAN link. We have made very good experiences with the use of internal USB sticks, memory cards, or ISO images stored on a virtualization host in the location to provide all repositories required to install servers in remote locations, including the repositories providing the updates.

Remote Installation Repositories

We strongly recommend that you use remote installation repositories for your AutoYaST framework wherever possible, because they provide the following advantages:

- ♦ A central repository for all servers prevents the necessity to distribute large physical installation media to all systems being installed.
- ♦ Only one small boot medium or PXE is required to start the installation process. Everything else is retrieved over the network.
- ♦ In most environments network based installations achieve higher throughput than installations using physical media.
- ♦ Multiple products can be installed together without the need to change the installation media (i. e. SLES11 SP4 and OES2015 SP1).
- ♦ Installation sources can be made available permanently allowing for easy installation of additional software packages at a later stage.

A remote installation repository must be specified at the boot prompt as follows:

```
install=<protocol>://<IP Address|DNS-Name>/<path to media content>
```

For example:

```
install=http://10.10.10.101/oes2018sp2
```

Network Repository Server

As pointed out in [“Installation Repositories” on page 15](#) various network protocols can be used to access network repositories which requires to set up an appropriate service (web server, FTP server, SMB server, etc.)

Based on experiences gained in numerous projects we recommend that you use the HTTP protocol and the Apache2 web server to implement your remote repositories.

Compared to other servers and protocols the Apache2 web server has the following advantages:

- ♦ Most administrators are familiar with the configuration of Apache2.
- ♦ Apache2 is part of all server distributions delivered by MicroFocus.
- ♦ Extensive logging mechanisms.
- ♦ Support for Symbolic links.

AutoYaST Control File

An AutoYaST control file is a configuration description in XML format that can contain just a few instructions or it may be comprised of hundreds of instructions that describe the details of the system to be installed.

A control file does not need to provide every configuration detail. If configuration items are omitted, AutoYaST will configure the system by using meaningful default values for the configuration items of the lacking sections. This works for most situations but can lead to unexpected results in some

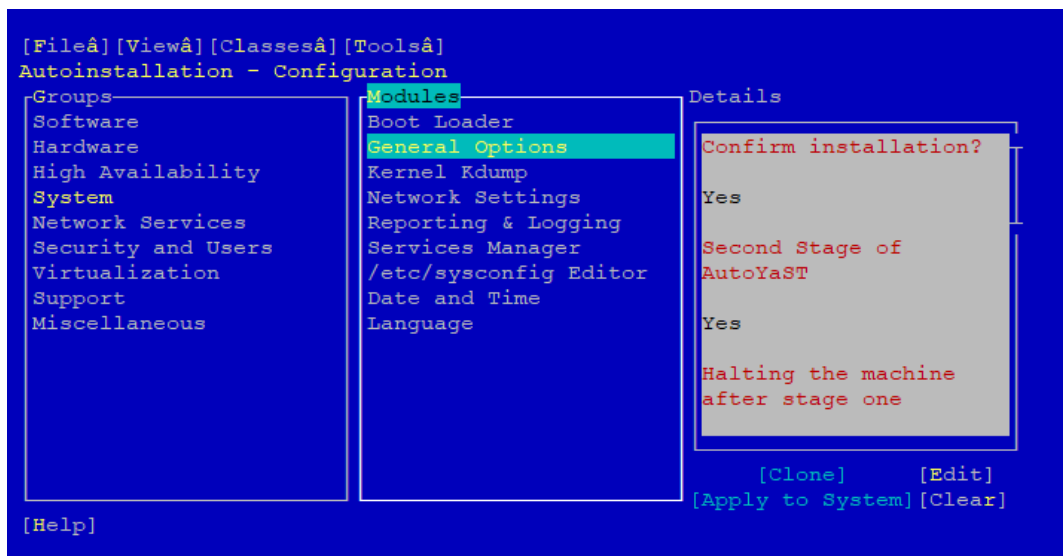
circumstances. Therefore we strongly recommend to include the configuration tags for every setting where the desired value differs from the default or must be protected against a change in the default.

In theory a control file can be created completely manually using your favorite editor. However, because this approach would be very error-prone it is not recommended.

The easiest way to obtain a valid control file that can be used as a template for automated installations is to use the `/root/autoinst.xml` file of an existing system, provided the option to create this file had been selected during installation. If this should not have been the case the file can be created easily at any time by issuing the `yast clone_system` command.

Another way to create a control file is to use `yast autoyast`. This allows you to create a control file for the complete system or for individual components. It provides a save method to create a syntactically correct XML file. You can even clone the existing system and only modify the values that need to be different for the system that you want to install. The resulting XML files are stored in the directory `/var/lib/autoinstall/repository`.

Figure 3-1 YaST: Autoinstallation - Configuration Items



The packages `autoyast2` and `autoyast2-installation` must be installed for this functionality to be available in YaST.

However, the control file of an existing system should only be used as template for a new installation with great care. You need to ensure that any value specific for the machine where the file has been generated is adjusted before you use the control file to install a new system. Whenever you modify the contents of a XML file we strongly recommend to validate the control file using a XML parser such as `xmllint`.

The AutoYaST control file can be located on any local or remote installation repository from where it can be retrieved at the beginning of an unattended installation.

Control File Structure

An AutoYaST control file must be written in valid XML syntax. It is organized in sections that define different aspects of an installation. The following sections could be part of an AutoYaST control file that drives an OES 2018 installation:

- ◆ add-on
- ◆ boot loader
- ◆ ca_mgm (CA management)
- ◆ eDirectory
- ◆ general
- ◆ groups
- ◆ host
- ◆ ...
- ◆ networking
- ◆ partitioning
- ◆ users

The following example shows a XML snippet to configure some aspects of the YaST Certificate Authority for SLES 12:

```
<?xml version="1.0"?>
<!DOCTYPE profile>
<profile xmlns="http://www.suse.com/1.0/yast2ns"
xmlns:config="http://www.suse.com/1.0/configs">
<ca_mgm>
  <CAName>YaST_Default_CA</CAName>
  <ca_commonName>my-company.us</ca_commonName>
  <country>US</country>
  <importCertificate config:type="boolean">>false</importCertificate>
  <locality>Provo</locality>
  <organisation>IT</organisation>
  <organisationUnit>Servermanagement</organisationUnit>
  <password>secret</password>
  <server_email>jdoe@novell.com</server_email>
  <state>UTAH</state>
  <takeLocalServerName config:type="boolean">>true</takeLocalServerName>
</ca_mgm>
</profile>
```

Class Files

Very few instructions within a control file are specific for a particular server. Most of them can be re-used. To support this AutoYaST offers the possibility to separate sections of the control file out into external class files. This approach is also known as class-based AutoYaST installation. Classes have a few advantages over monolithic AutoYaST control files:

- ♦ Code is easier to maintain since class files contain only parts of the whole.
- ♦ Classes can be re-used for different installation scenarios, i. e. the NTP class can be used for the installation of different versions of SLES and OES.

When using this technique it is possible to start the installation with a minimal control file only containing class directives (see control file `default` in section [“Retrieving a Control File” on page 20](#) below). The real configuration information is then provided by the class files.

Just like a control file a class file must be in valid XML syntax and it can contain a single or multiple sections of the complete AutoYaST control file. It must be specified in a control file by a `class` directive. This directive has attributes that define the location of the external class file.

When a `class` configuration tag is encountered during control file processing the corresponding external class file is loaded from the specified location and merged into the main control file that is temporarily stored in `/tmp/profile/autoinst.xml` on the system being installed. Once it is complete this control file is parsed in the same manner as a monolithic control file when using the non-class based approach.

For using classes some preparations have to be made where the AutoYaST control file is stored:

- ♦ All class files must be located in a directory named `classes` immediately subordinate to the directory specified in the AutoYaST URL. For example, if the AutoYaST URL has been specified as `autoyast=http://10.10.10.101/xml/default` the class files must be located in the `http://10.10.10.101/xml/classes` directory.
- ♦ A `class` directive always contains a `class_name` tag and a `configuration` tag:

```
<class>
  <class_name>general</class_name>
  <configuration>bootloader.xml</configuration>
</class>
```

- ♦ A `class_name` tag represents a directory underneath the `classes` directory, e. g. if `general` is used in the `class_name` tag the following directory must exist:

```
http://10.10.10.101/xml/classes/general
```

- ♦ A `configuration` tag specifies the name of the XML file in the directory defined by the `class_name` tag. Using the `class_name` tag from the above example the class file `bootloader.xml` would be accessed through the following URL:

```
http://10.10.10.101/xml/classes/general/bootloader.xml
```

Retrieving a Control File

The URL for a control file must be specified by one of the following syntaxes:

```
autoyast=<protocol>://<IP Address|DNS-Name>/<path to control file or directory/>,
```

e.g.:

```
autoyast=http://10.10.10.101/xml/mycontrolfile
```

or

```
autoyast=http://10.10.10.101/xml/
```

The first example points directly to a control file at the repository server. This will locate, retrieve and process the AutoYaST control file `mycontrolfile`.

The second example points to a directory (note the trailing “/”) resulting in the following retrieval process:

1. A file whose name is equivalent to the hexadecimal value of the IP address of the system being installed is searched in the specified directory. For example:

```
192.168.2.91 -> http://10.10.10.101/xml/C0A8025B
```

2. If this file does not exist, the last character of the file name is removed and the system searches for the resulting file name. For example:

```
http://10.10.10.101/xml/C0A8025
```

3. If the file does not exist the process is repeated until the file name has been truncated to one character. For example:

```
http://10.10.10.101/xml/C0A802
http://10.10.10.101/xml/C0A80
http://10.10.10.101/xml/C0A8
http://10.10.10.101/xml/C0A
http://10.10.10.101/xml/C0
http://10.10.10.101/xml/C
```

4. If a file with a name matching the name derived by the above process can be located the unattended installation starts. If no file can be found the process continues.

5. The system tries to retrieve a file with a name matching the MAC address of the network adapter from which the connection to the web server has been initiated.

Two attempts are made: First, it looks for a file name with characters in all uppercase. If it is not found, a file name with characters in all lowercase will be searched for.

For example:

```
http://10.10.10.101/xml/0080A8F6484C
http://10.10.10.101/xml/0080a8f6484c
```

6. Finally, if this still does not deliver a control file, the process looks for a file named `default` in the directory which has been specified with the `autoyast=` parameter. For example:

```
http://10.10.10.101/xml/default
```

7. If the default file is found, it is evaluated and its instructions are executed. The resulting control file that is used to perform the installation initially is stored as `/tmp/profile/autoinst.xml` and moved to `/var/adm/autoinstall/cache/installedSystem.xml` on the installed system as the installation proceeds.

If the installation engine has not located a control file at this point, the installation process terminates and informs the administrator that the URL for the AutoYaST control file is invalid.

Installation Boot Medium

A boot medium is required to invoke the installation process by loading a kernel and an `initrd` that contains the installation logic.

As with repositories a wide range of media is available. Servers can be booted from Floppy, CD, DVD, USB devices, over the network (PXE) or from a local hard drive. In our experience smaller environments typically use images or physical media for installations via remote management connections such as ILO boards whereas PXE tends to be preferred in larger environments.

Both methods are explained in detail in this section.

Custom Boot ISO

The standard installation medium is an ISO image of the OES or SLES DVD provided by MicroFocus or SUSE.

However, we favour a customized boot image with nested boot menus for the following reasons:

- ♦ Multiple versions of OES and SLES can be installed from one single boot image.
- ♦ The image requires only a small number of kernels, `initrds` and some boot loader files. This allows to reduce the image size to just a couple hundred MB depending on the number of OES and SLES releases and service packs that need to be supported.
- ♦ Customized menus can be created to initiate unattended installations using pre-defined boot options reflecting different installation requirements.
- ♦ Background images can be included to reflect corporate identity

The boot medium for our solution supports BIOS and UEFI systems and can be built from a working directory (`/data/boot_cd_build`) representing the root of the boot image.

In this working directory the stage files for the grub boot loader used to boot systems with BIOS and the file `menu.lst` that defines the first level of the nested menu structure for grub need to reside in the directory `.../boot/grub`. The files required by grub2 used to boot systems with UEFI must be located in `.../boot/x86_64`.

The file `message` along with the grub configuration files that define the other levels of the nested menu structure for BIOS systems need to be located in the directory `.../grub`. The grub2 configuration files defining the menus for UEFI systems must be placed in `.../EFI`.

Finally, for each target platform the kernel (`linux`) and the corresponding `initrd` are required on the boot medium. Typically they have to be copied from the `/boot/<architecture>/loader` directory on the original boot media shipped by MicroFocus or SUSE to the directory `.../kernel/<product><version><sp>`, for example to `.../kernel/oes2018sp2`.

When you copy these two files make sure to preserve the timestamps. Note that for OES2018 the kernel and the initrd from the OES2018 Install media must be used. An OES2018 installation using a pure SLES12 kernel and/or initrd will not succeed.

[Table 3-1, “Directory Structure of AutoYaST Boot Medium,” on page 22](#) summarizes the directory structure of the working directory from which the Installation Boot Medium is created.

Table 3-1 Directory Structure of AutoYaST Boot Medium

Directory	Content
/boot/EFI	Configuration files required by grub2. <code>grub.cfg</code> corresponds to <code>main.lst</code> used for grub and defines the first level of the grub2 menu structure; see Figure 3-5 on page 27 .
/boot/EFI/locale	Locale files copied from <code>/EFI/BOOT/locale/</code> of a current SLES15 ISO.
/boot/grub	<code>stage1</code> , <code>iso9660_stage1_5</code> , <code>stage2</code> These files need to be copied from a system using a current version of grub. <code>menu.lst</code> Grub configuration file defining the first level of the grub menu structure; see Figure 3-2 on page 24 .
/boot/x86_64/grub2-efi	Fonts, images and <code>theme.txt</code> required by grub2 copied from <code>/boot/x86_64/grub2-efi/</code> of a SLES15 ISO.
/grub	<code>message</code> (optional). This file can be used to customize the language and the background image. <code>main.lst</code> Grub configuration file defining the second level of the grub menu structure; see Figure 3-3 on page 25 . <code><product><version><sp>.lst</code> For example: <code>oes2018sp2.lst</code> Grub configuration file defining the third level of the grub menu structure; see Figure 3-4 on page 26 . These files contain the predefined boot options to invoke an AutoYaST installation or upgrade for the specified version and support pack of a product.
/kernel/<product> <version><sp>	<code>initrd</code> and <code>linux</code> from directory <code>/boot/<architecture>/</code> loader on the original <code><product><version><sp></code> installation medium for the specified version and support pack of a product (copy both files with <code>-p!</code>).

Table 3-2, “linuxrc Boot Options,” on page 23 lists the most important `linuxrc` boot options that are used by our solution to control the installation process. For a complete list of the available `linuxrc` boot options, please refer to [linuxrc Parameter Reference](#).

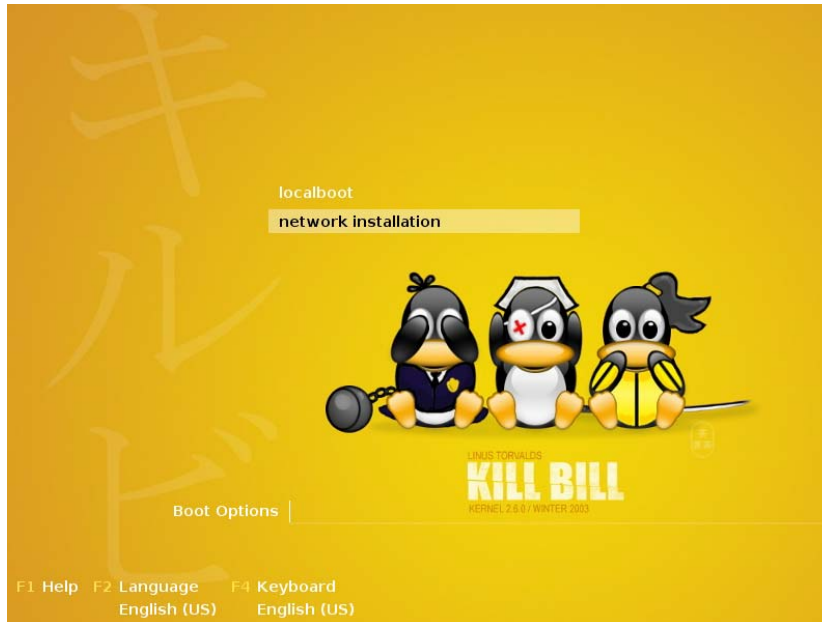
Table 3-2 *linuxrc Boot Options*

Parameter	Value and Meaning
<code>autoupgrade</code>	This parameter can be used to initiate an automatic upgrade using AutoYaST. Its value must be 1 and <code>autoyast</code> must point to a valid control file for the upgrade.
<code>autoyast</code> (mandatory)	URL to AutoYaST control file or directory. For example: <code>http://<IP-Address>/xml/default</code>
<code>domain</code> (optional)	Suffix search list for DNS.
<code>info</code> (optional)	Specifies a file to read more options from as URL. This is useful if not all options fit on the command line (some <code>linuxrc</code> versions have a limitation of 255 chars).
<code>install</code> (mandatory)	URL to installation repository. For example: <code>http://<IP-Address>/oes2018sp2</code>
<code>kernel</code> (mandatory)	Path to the kernel image of the target installation system on the boot medium such as <code>kernel /kernel/oes2018sp1/linux</code>
<code>nameserver</code> (optional)	Space or comma separated list of DNS servers; required if you want to use DNS names instead of IP addresses.
<code>netsetup</code> (optional, mandatory for our solution)	Used to setup a static network configuration in the pre-installation system to gain network access to repositories and/or special metadata files. Otherwise DHCP is used. A value of 1 initiates a dialogue to specify IP address, net mask, gateway, DNS name server and DNS search path.
<code>netmask</code> , <code>gateway</code> , <code>nameserver</code>	Values for these parameters can be pre-defined at the boot prompt in full or only partly. These values will be displayed in the dialog windows where they can be modified.
<code>netwait</code>	Delay after activating the network interface in seconds.
<code>self_update</code>	URL to a repository providing installer updates. Available from SLES12 SP3 onwards. Do not use this option with OES!

There are many ways how to organize the menu structure of a customized boot image. Here we illustrate one example that supports the installation of and the upgrade to OES2018 SP2, SLES12 SP5, and SLES15 SP1 of systems with traditional BIOS.

The first level of the menu hierarchy for BIOS systems only provides the selections **localboot** and **network installation**. To prevent loops in the installation process **localboot** is the default selection so that the system being installed will boot from the hard disk when it reaches stage2.

Figure 3-2 Customized Boot Image for BIOS Systems With Nested Menus - Level 1



This is implemented in the grub configuration file `.../boot/grub/menu.lst`:

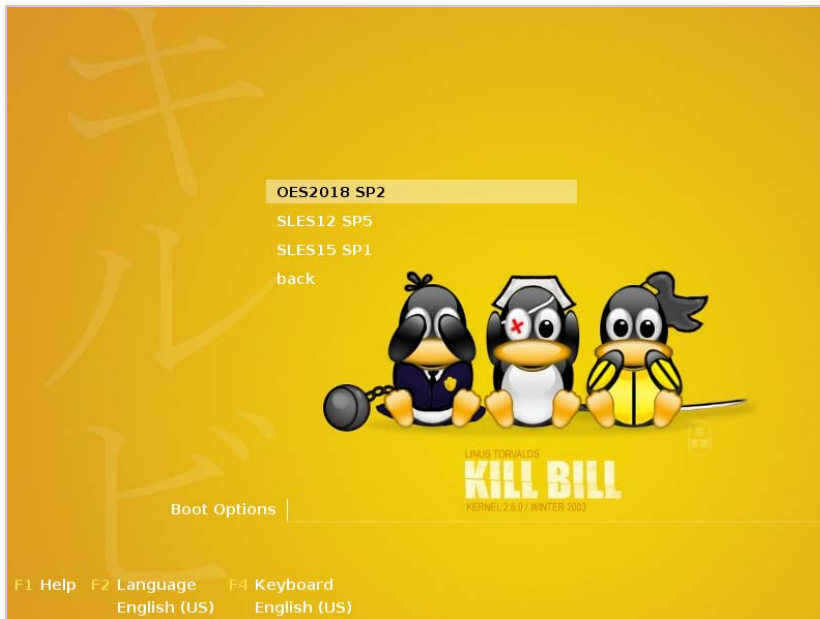
```
# autoinst level 1
color white/blue black/light-gray
default 0
timeout 60
root (cd)
gfxmenu /grub/message

### localboot
title localboot
    rootnoverify (hd0)
    chainloader +1

### network
title network installation
    configfile /grub/main.lst
```


The second level of the menu structure allows the selection of the release and support pack of the product to be installed.

Figure 3-3 Customized Boot Image for BIOS Systems With Nested Menus - Level 2



The grub menu file `.../grub/main.lst` to implement this level of the boot selection menu looks as follows:

```
# autoinst level 2
color white/blue black/light-gray
default 0
timeout 60
root (cd)
gfxmenu /grub/message

### OES2018 SP2
title OES2018 SP2
    configfile /grub/oes2018sp2.lst

### SLES12 SP5
title SLES12 SP5
    configfile /grub/sles12sp5.lst

### SLES15 SP1
title SLES15 SP1
    configfile /grub/sles15sp1.lst

### back
title back
    configfile /boot/grub/menu.lst
```

The third and final level of the boot menu structure determines how exactly the installation is going to happen. This example provides options for a new installation and for an upgrade of an existing system to OES2018 SP2:

Figure 3-4 Customized Boot Image for BIOS Systems With Nested Menus - Level 3



For OES2018 SP2 the final level of the boot selection menu is implemented through the grub menu file `.../grub/oes2018sp2.lst`:

```
# autoinst level 3
color white/blue black/light-gray
default 0
timeout 30
root (cd)
gfxmenu /grub/message

### oes2018sp2
title OES2018 SP2
    kernel /kernel/oes2018sp2/linux autoyast=http://<IP address of your
AutoYaST server>/xml/default-<IP address of your AutoYaST server>
install=http://<IP address of your AutoYaST server>/oes2018sp2
netsetup=hostip,gateway netmask=255.255.255.0 gateway=10. netwait=10
    initrd=/kernel/oes2018sp2/initrd

### upgrade to oes2018sp2
title UPGRADE to OES2018 SP2
    kernel /kernel/oes2018sp2/linux autoupgrade=1 autoyast=http://<IP
address of your AutoYaST server>/upgrade/autoupgrade-oes2018sp2_ZCM.xml
install=http://<IP address of your AutoYaST server>/oes2018sp2
netsetup=hostip,gateway netmask=255.255.255.0 gateway=10. netwait=10
    initrd=/kernel/oes2018sp2/initrd

### back
title back
    configfile /grub/main.lst
```

The menu system to install UEFI based systems only has two levels that look slightly different than the screens for BIOS systems. The first level provides the selections **localboot** and the releases and support packs of the products that can be installed. Again **localboot** is the default selection to prevent loops when the system being installed boots into stage2.

Figure 3-5 Customized Boot Image for UEFI Systems With Nested Menus - Level 1



This level of the boot menu structure for UEFI systems is provided by the grub2 configuration file `.../EFI/grub.cfg`. Only the part defining the menu structure is shown here:

```

:
menuentry "localboot" {
    set my_os=localboot
    echo "booting from local disk..."
    insmod part_gpt
    insmod part_msdos
    insmod btrfs
    if search --no-floppy --file /efi/boot/fallback.efi --set; then
        for os in opensuse sles caasp ; do
            if [ -f /efi/$os/grub.efi ] ; then
                chainloader /efi/$os/grub.efi
            fi
        done
    fi
}
submenu "OES2018 SP2" {
    configfile /EFI/CIF_oes2018sp2.cfg
}
submenu "SLES12 SP5" {
    configfile /EFI/CIF_sles12sp5.cfg
}
submenu "SLES15 SP1" {
    configfile /EFI/CIF_sles15sp1.cfg
}
:

```

The second level of the boot menu in this example also provides the options for a new installation and for an upgrade of an existing system:

Figure 3-6 Customized Boot Image for UEFI Systems With Nested Menus - Level 2



The above menu is implemented in the file `.../EFI/CIF_oes2018sp2.cfg`. It uses variables for the IP address of the AutoYaST server, the repository server and the gateway defined in lines 2-4. This example assumes that the AutoYaST server is also your repository server (`ISO_SERVER`).

```
# autoinst level 2
set AY_SERVER="<IP of your AutoYaST server>"
set ISO_SERVER="${AY_SERVER}"
set GATEWAY="<IP of your gateway>"

set my_os="oes2018sp2"
set display_os="OES2018 SP2"

    menuentry "$display_os" {
        echo "CIF - Consulting Installation Framework - Boot ISO"
        echo ""
        echo "AutoYaST Server:    ${AY_SERVER}"
        echo "ISO Server:         ${ISO_SERVER}"
        echo "Gateway:           ${GATEWAY}"
        echo ""
        echo "Installing ${display_os} . . ."
        echo ""
        echo "loading $display_os kernel ..."
        linuxefi /kernel/$my_os/linux autoyast=http://"${AY_SERVER}"/xml/
default-"${AY_SERVER}" install=http://"${ISO_SERVER}"/"${my_os}"
netsetup=hostip,gateway netmask=255.255.255.0 gateway=${GATEWAY}netwait=10
        echo "loading $display_os initrd ..."
        initrdefi /kernel/$my_os/initrd
    }

    menuentry "UPGRADE to $display_os" {
        echo "CIF - Consulting Installation Framework - Boot ISO"
        echo ""
```

```

echo "AutoYaST Server:      ${AY_SERVER}"
echo "ISO Server:          ${ISO_SERVER}"
echo "Gateway:             ${GATEWAY}"
echo ""
echo "Upgrading to ${display_os} . . ."
echo ""
echo "loading $display_os kernel ..."
linuxefi /kernel/$my_os/linux autoupgrade=1
autoyast=http://"${AY_SERVER}"/upgrade/"${my_os}"-"${AY_SERVER}".xml
install=http://"${ISO_SERVER}"/"${my_os}" netsetup=hostip,gateway
netmask=255.255.255.0 gateway=${GATEWAY} netwait=10
echo "loading $display_os initrd ..."
initrdefi /kernel/$my_os/initrd
}
menuentry 'Back' {
    configfile /EFI/grub.cfg
}

```

Independent of whether the system being installed is booted from BIOS or UEFI the actual installation is triggered by the boot options of the kernel command line for the selection you chose.

We only have demonstrated the basics of how to design the menu structure of an Installation Boot Medium here. This can be expanded in many ways. For instance, the system is not limited to three levels of menus and customers have been very creative in how they use the menu structure.

One possibility that could be implemented is the choice of the repository that will be used for the installation. For instance you could provide one option to use a remote repository when installing systems in your main location or in branch offices with high bandwidth WAN connections to the AutoYaST server. For locations with a low bandwidth connection a selection to install using a local repository might be more appropriate.

Another option could be to set the location specific boot options in different grub configuration files and to provide separate selections for each location. The design possibilities for the boot menus are endless.

Finally, when the required kernels and initrds, the boot binaries, and the configuration files for grub and grub2 are prepared, the boot image must be created using `xorrisio`.

To simplify the creation of the Installation Boot Medium we provide the script `create-ay-iso.sh` available at the [CIF download site](#). This script allows to specify the following command line parameters:

Table 3-3 Command Line Parameters for `create-ay-iso.sh`

Parameter	Value and Meaning
<code>--customer</code>	Customer name; will be part of the name of the ISO file that is created Default: CIF
<code>--name</code>	Name of the iso file that is created. Default: <code>autoyast-cif.iso</code>
	NOTE: the default name of the iso file is derived from a variable <code>CUSTOMER</code> that is set in line 77 of the script.

Parameter	Value and Meaning
<code>--isodir</code>	Output directory where the ISO file is created. Default: <code>/data/isos</code>
<code>--tempdir</code>	Directory for temporary files required for EFI support Default: <code>/tmp</code>
<code>--workdir</code>	Directory where the directories <code>boot</code> , <code>grub</code> , and <code>kernel</code> are located. Default: <code>/data/boot_cd_build</code>
<code>--help</code>	Displays Help screen.
<code>--debug</code>	Optional: increases verbosity.

The size of the resulting image file `autoyast-<Customer>.iso`, that will be stored in the path provided to the parameter `--isodir` depends on the number of products, versions and support packs that the boot image needs to support. The kernels and corresponding initrds consume the most space, but the size of such a custom Installation Boot Medium is always far smaller than the size of the original boot media.

Network Boot Environment

A network boot environment consists of a DHCP server and a TFTP server. Depending on the solution a proxyDHCP server might also be required.

We first will describe how to use your ZCM Primary Server to provide a network boot environment for BIOS and UEFI systems assuming that DHCP is provided by other servers and that you have configured your routers to forward DHCP and TFTP requests to your DHCP and TFTP servers. We will also explain how to use your AutoYaST server as TFTP server.

Sample PXE configuration files for all configurations are available at the [CIF download site](#).

PXE Boot for BIOS Systems Using a ZCM Primary Server

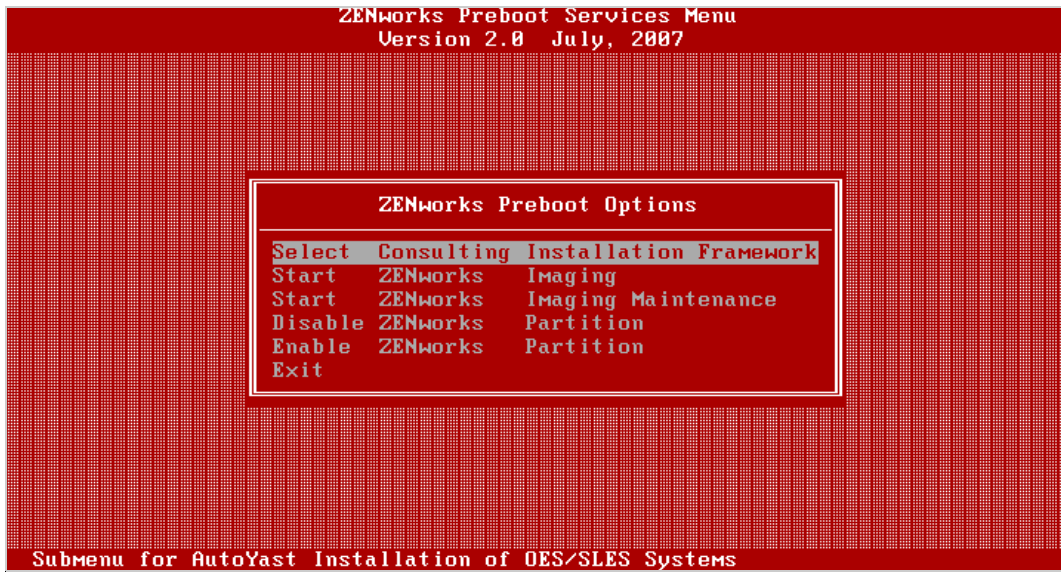
Every ZCM Primary Server has the pattern `novell-tftp` installed and the Novell TFTP server is active by default. `novell-tftp` uses the base directory `/srv/tftp`.

The DHCP options required for network boot need to be supplied by the Novell Proxy DHCP server (`novell-proxydhcp`) that needs to be activated and started on one server (please refer to [Preparing a Preboot Services Imaging Server](#) in the ZENworks Preboot Services and Imaging Reference for further details).

The kernel and initrd of the installation systems that CIF shall support need to be copied to the directory `/srv/tftp/kernel`.

The first step in extending the Novell TFTP server configuration to support the installation of OES servers and SLES servers using the Consulting Installation Framework is to add a corresponding entry to the main menu, that is defined in `/srv/tftp/pxemenu.txt`.

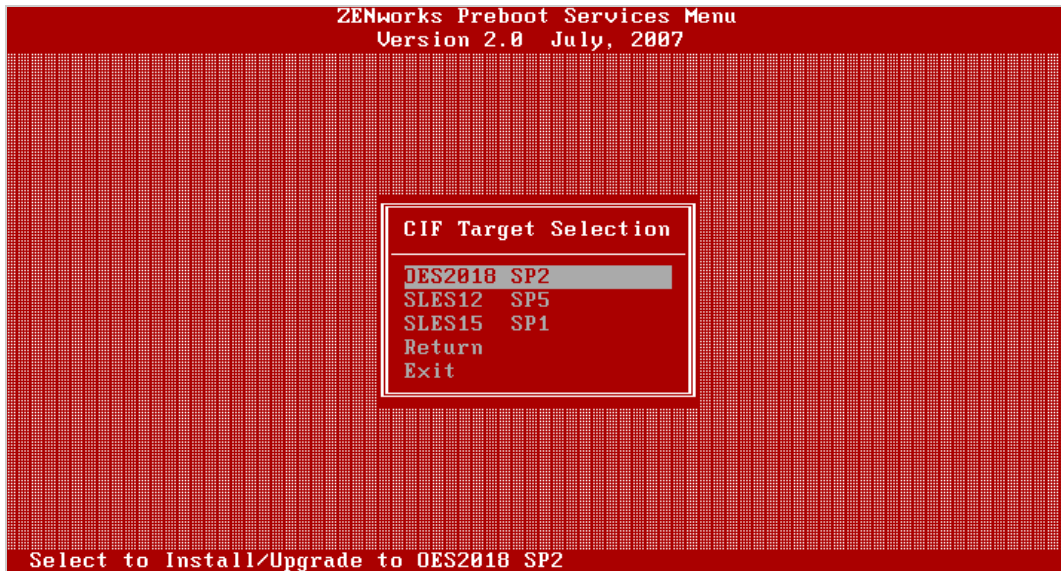
Figure 3-7 PXE Main Menu for BIOS Systems on ZCM Primary Server



```
:  
[Main]  
MenuTitle = ZENworks Preboot Options  
option = submenu ; "Select Consulting Installation Framework" ; "Submenu  
for AutoYast Installation of OES/SLES Systems" ; SUBMenu_CIF  
option = execute ; "Start ZENworks Imaging" ; "ZENworks Imaging in  
Automated Mode";  
pxelinux.0 ; z_auto.cfg  
:
```

The section [SUBMenu_CIF] in this file defines a sub-menu with one entry for each of the supported releases.

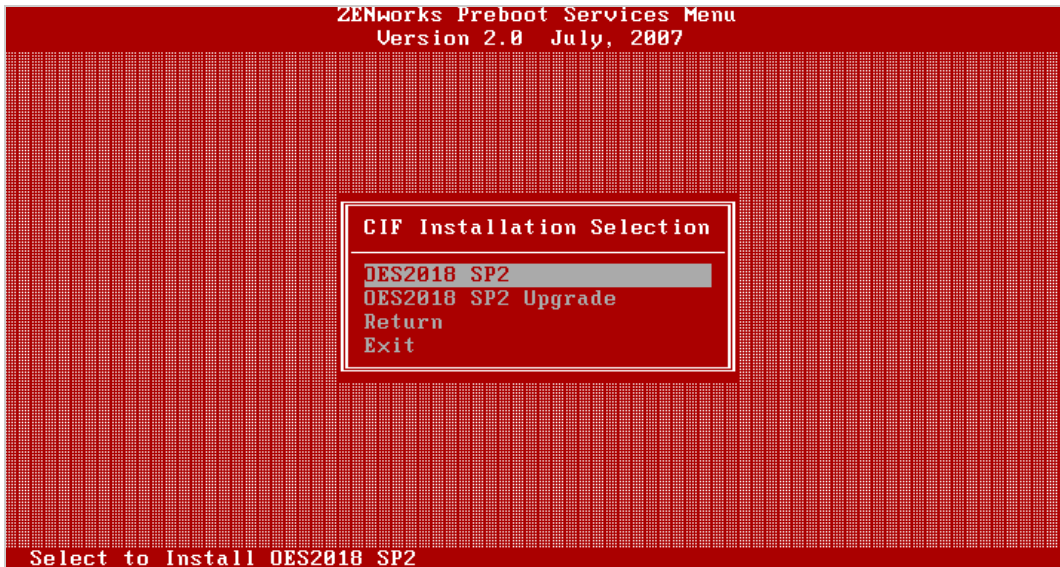
Figure 3-8 CIF Sub Menu for BIOS Systems on ZCM Primary Server



```
[SUBMenu_CIF]
MenuTitle = CIF Target Selection
option = submenu ; "OES2018 SP2"; "Select to Install/Upgrade to OES2018
SP2" ;
SUBMenu_CIF_OES2018SP2
option = submenu ; "SLES12 SP5"; "Select to Install/Upgrade to SLES12
SP5" ;
SUBMenu_CIF_SLES12SP5
option = submenu ; "SLES15 SP1"; "Select to Install/Upgrade to SLES15
SP1" ;
SUBMenu_CIF_SLES15SP1
option = return ; "Return" ; "Return to main menu"
option = exit ; "Exit" ; "Boot to local hard drive"
```


Each selection in this sub-menu allows to either install the selected release or to upgrade an existing system to the selected release.

Figure 3-9 OES2018 SP2 Options for BIOS Systems on ZCM Primary Server



```
[SUBMenu_CIF_OES2018SP2]
MenuTitle = CIF Installation Selection
option = execute ; "OES2018 SP2" ; "Select to Install OES2018 SP2" ;
        pxelinux.0 ; CIF_oes2018sp2.cfg
option = execute ; "OES2018 SP2 Upgrade" ; "Select to Upgrade to OES2018
SP2" ;
        pxelinux.0 ; CIF_oes2018sp_upgrade.cfg
option = return ; "Return" ; "Return to main menu"
option = exit ; "Exit" ; "Boot to local hard drive"
```

These options execute configuration files that load the kernel and the initrd of the installation system of the selected release and pass the defined boot options to it. For instance, if you select the installation of OES2018 SP2 /srv/tftp/CIF_oes2018sp2.cfg is processed.

```
##### WARNING! #####
# Consulting Installation Framework PXE Configuration
#
# DO NOT EDIT THIS FILE!
#
# Modifying this file is unsupported and can have
# unpredictable results on ZENworks Preboot Services
#####
ZENWORKSAPPEND 1
DEFAULT OES2018SP2
LABEL OES2018SP2
        kernel kernel/oes2018sp2/linux
        append autoyast=http://<IP of your AutoYaST server>/xml/default-<IP
of your AutoYaST server> install=http://<IP of your ISO server>/oes2018sp2
netsetup=hostip,gateway netmask=255.255.255.0 gateway=<IP of your gateway>
netwait=10 initrd=kernel/oes2018sp2/initrd
```

These files are similar to the grub configuration files for BIOS systems on the Custom Boot ISO discussed in the previous section with two distinct differences: the boot options are defined on the append line and the initrd is part of this line as well.

The ownership of `pxemenu.txt`, of all configuration files and of the kernels and initrds underneath `/srv/tftp/kernel` need to be changed to `zenworks:zenworks`.

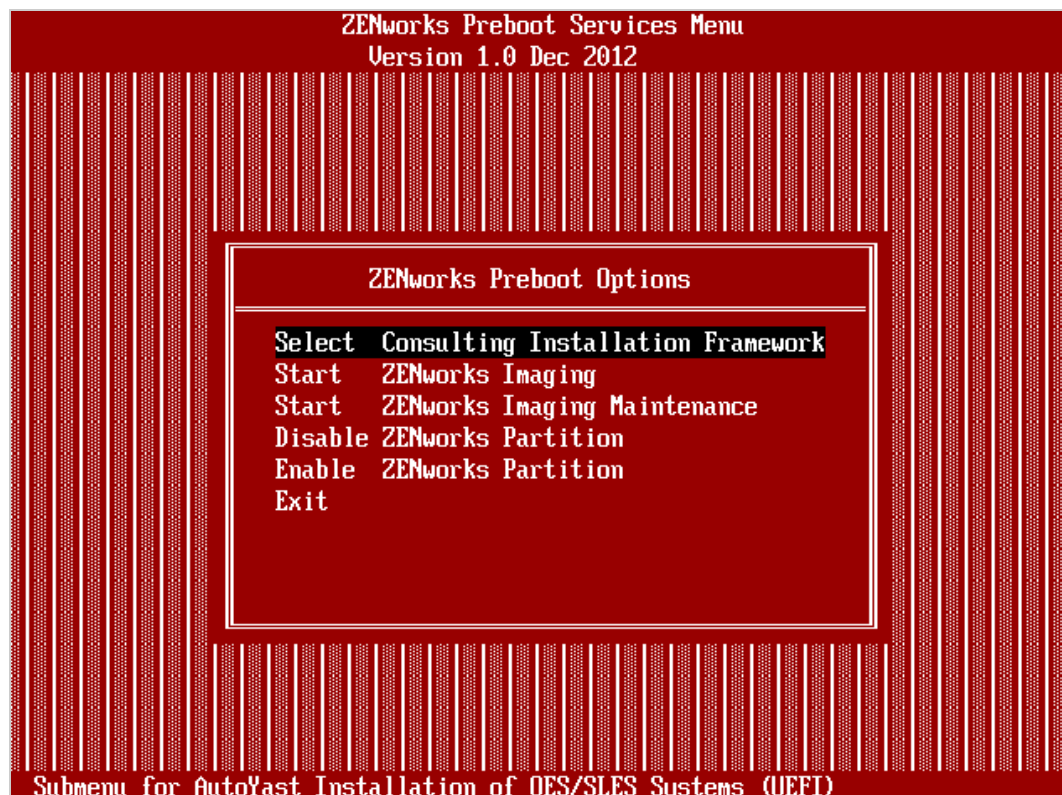
When an installation is initiated over PXE once it has processed the PXE configuration file the installation will proceed just as if the system being installed would have been booted from the Custom Boot ISO.

PXE Boot for UEFI Systems Using a ZCM Primary Server

For a ZCM Primary Server to support PXE boot for UEFI systems `/boot/x86_64/grub2-efi/` from a current SLES15 ISO needs to be copied to `/srv/tftp/boot/x86_64/grub2-efi` and `/EFI/BOOT/locale/` needs to be copied to `/srv/tftp/efi/x86_64/locale`. The ownership of the copied files needs to be set to `zenworks:zenworks` and a relative symbolic link `kernel` pointing to `/srv/tftp/kernel` needs to be created in `/srv/tftp/efi/x86_64`.

The option to select the Consulting Installation Framework needs to be added to PXE main menu for UEFI systems:

Figure 3-10 PXE Main Menu for UEFI Systems on ZCM Primary Server



This is defined in `/srv/tftp/efi/x86_64/pxemenu.txt`.

```
:
[Main]
MenuTitle = ZENworks Preboot Options
option = "execute;Select Consulting Installation Framework;Submenu for
AutoYast Installation of OES/SLES Systems (UEFI);efi/x86_64/
bootx64.efi;efi/grub.cfg"
option = "execute;Start ZENworks Imaging;ZENworks Imaging in Automated
Mode;efi/x86_64/bootx64.efi;efi/x86_64/z_auto.conf"
option = "execute;Start ZENworks Imaging Maintenance;ZENworks Imaging
Linux Session in Interactive Mode;efi/x86_64/bootx64.efi;efi/x86_64/
z_maint.conf"
option = "execute;Disable ZENworks Partition;Disable Existing ZENworks
partition; efi/x86_64/bootx64.efi;efi/x86_64/z_zpdis.conf"
option = "execute;Enable ZENworks Partition;Re-enable Existing ZENworks
partition; efi/x86_64/bootx64.efi;efi/x86_64/z_zpen.conf"
option = "exit;Exit;Boot to local hard drive"
```

In general PXE uses the same grub2 configuration files that are used by the Custom Boot ISO. The main configuration file `grub.cfg` and the configuration files that implement the sub-menus need to be copied to `/srv/tftp/efi`. To be able to use the configuration files from your AutoYaST server without modification you need to create a symbolic link `/srv/tftp/EFI` pointing to the directory `efi`. The ownership of all grub2 configuration files needs also to be set to `zenworks:zenworks`.

PXE Boot for BIOS Systems Using the AutoYaST Server

The first step to use your AutoYaST server as PXE server is to install the `tftp` package. `tftp` uses the base directory `/srv/tftpboot`.

Next you need to install the package `tftpboot-installation-SLE-OS_VERSION-ARCHITECTURE`. Replace `OS_VERSION` with the version and `ARCHITECTURE` with the architecture of your SUSE Linux Enterprise Server installation, for example `SLE-15-SP1-x86_64`.

This package copies all files required to boot a `SLE-OS_VERSION-ARCHITECTURE` server to `/usr/share/tftpboot-installation/SLE-OS_VERSION-ARCHITECTURE`. You then need to copy the boot image `pxelinux.0` and its configuration directory `pxelinux.cfg` from `/usr/share/tftpboot-installation/SLE-OS_VERSION-ARCHITECTURE/net` to `/srv/tftpboot/net`.

Next you need to copy the kernels and initrds for all releases that the CIF shall support to `/srv/tftpboot/kernel`.

NOTE: As the TFTP server runs chrooted in `/srv/tftpboot` you cannot use a symbolic link to point to `/data/boot_cd_build/kernel` here!

Linking `/data/boot_cd_build/kernel` to `/srv/tftpboot/kernel` however will work.

Finally the files `menu.c32` and `chain.c32` need to be copied from `/usr/share/syslinux` to `/srv/tftpboot/net`. They are required to use a menu system with the PXE server and to have the ability to boot a local installation.

A SLES based TFTP server is configured through /srv/tftpboot/pxelinux.cfg/default.

```
# activate menu system
UI menu.c32
PROMPT 0

# menu color settings
MENU COLOR border      37;41
MENU COLOR disabled    37;41
MENU COLOR hotkey      1;37;41
MENU COLOR hotssel     1;37;47
MENU COLOR sel         31;47
MENU COLOR title       1;37;41
MENU COLOR unsel       37;41

# do not allow user to specify command line options
ALLOWOPTIONS 0

# menu system
MENU TITLE Consulting Installation Framework

# OES2018 SP2 options
MENU BEGIN
    MENU TITLE OES2018 SP2
    LABEL OES2018 SP2
    MENU LABEL ^OES2018 SP2
    MENU INCLUDE pxelinux.cfg/CIF_oes2018sp2.cfg
    MENU CLEAR

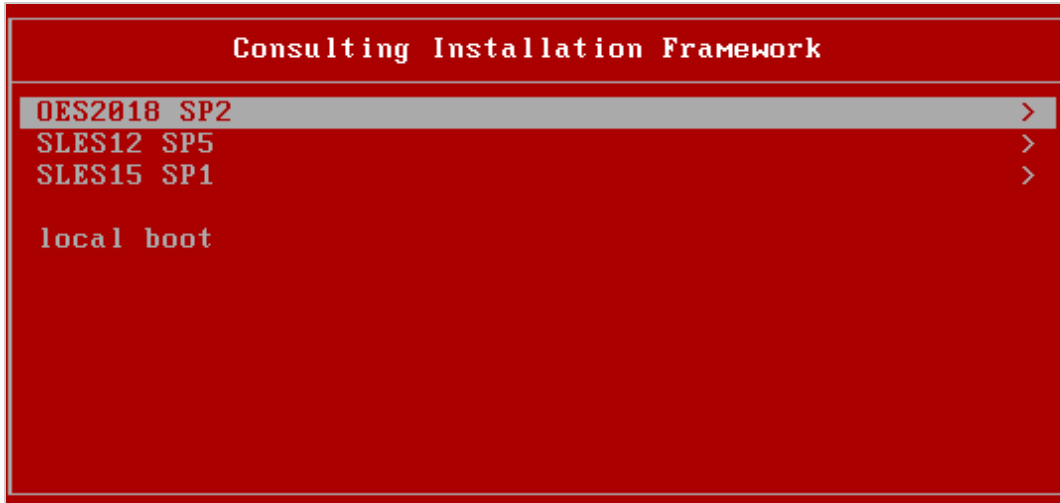
    LABEL OES2018 SP2 Upgrade
    MENU LABEL OES2018 SP2 ^Upgrade
    MENU INCLUDE pxelinux.cfg/CIF_oes2018sp2_upgrade.cfg
    MENU CLEAR
MENU END

:
# Boot to local hard drive
MENU SEPARATOR
LABEL local boot
KERNEL chain.c32
APPEND hd0 0
MENU CLEAR
```

The first part of this configuration file activates the simple menu system, defines the menu colors, and prevents users from entering command line options. The menu system comes next before the last six lines implement the option to boot from local disk.

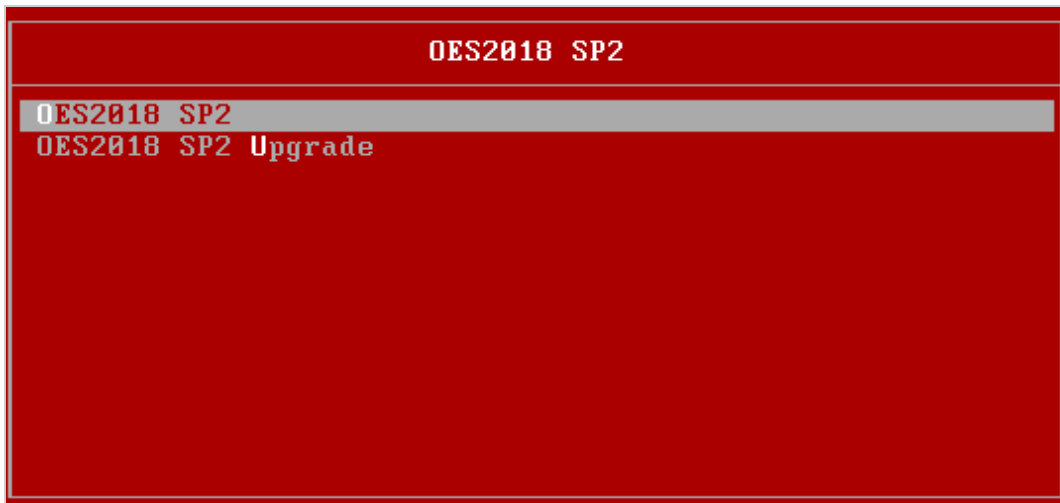
The main menu allows the selection of the supported products and releases or to boot from the local disk.

Figure 3-11 PXE Main Menu for BIOS Systems on AutoYaST Server



The menu system consists of a sub-menu (enclosed in `MENU BEGIN ... MENU END`) for each product. The options for the installation/upgrade of each product use the `MENU LABEL` directive to implement a selection hotkey (preceded by `^`) and to include a configuration file with a `KERNEL` and `APPEND` statement.

Figure 3-12 OES2018 SP2 Options for BIOS Systems on AutoYaST Server



Basically this are the same configuration files used for PXE on a ZCM Primary Server however, the lines with the directives ZENWORKSAPPEND, DEFAULT, and LABEL need to be removed.

```
##### WARNING! #####  
# Consulting Installation Framework PXE Configuration  
#  
# DO NOT EDIT THIS FILE!  
#####  
  
kernel kernel/oes2018sp2/linux  
append autoyast=http://<IP of your AutoYaST server>/xml/default-<IP  
of your AutoYaST server> install=http://<IP of your ISO server>/oes2018sp2  
netsetup=hostip,gateway netmask=255.255.255.0 gateway=<IP of your gateway>  
netwait=10 initrd=kernel/oes2018sp2/initrd
```

PXE Boot for UEFI Systems Using the AutoYaST Server

PXE boot for UEFI systems on the AutoYaST server requires almost the same preparation as for the ZCM Primary Server. `/boot/x86_64/grub2-efi/` from a current SLES15 ISO needs to be copied to `/srv/tftpboot/boot/x86_64/grub2-efi` and `/EFI/BOOT/locale/` needs to be copied to `/srv/tftpboot/EFI/locale`. In addition `/usr/share/tftpboot-installation/SLE-15-SP1-x86_64/EFI/BOOT/grub.efi` needs to be copied to `srv/tftpboot/EFI`.

A relative symbolic link `kernel` pointing to `/srv/tftpboot/kernel` needs to be created in `/srv/tftpboot/net`.

PXE boot for UEFI systems uses the same grub2 configuration files as the boot ISO. The main configuration file `grub.cfg` and the configuration files that implement the sub-menus need to be placed in `/srv/tftpboot/EFI`.

4 Installing and Configuring an AutoYaST Server

This chapter details the setup and configuration of the AutoYaST server for the Consulting Installation Framework (CIF). We also use this server as the repository server for the solution. The goal is to provide a standardized installation environment that is easy to maintain and to adjust.

Hardware Requirements

The AutoYaST server does not require high-performance hardware. In many environments, a virtual instance on VMware, XEN, KVM, or another type of virtualization with a single processor and 2 GByte of RAM has been used successfully.

As the speed of the network connection has the biggest influence on the duration of an installation a 1 Gbits⁻¹ network interface is recommended.

Every AutoYaST server should be equipped with two hard disks. A 30 GB disk is sufficient to install the operating system and the additional packages needed.

The second disk will accommodate the AutoYaST system and in particular it will store the repositories that will consume most of the disk space. Therefore the capacity of this disk greatly depends on the number of releases that the system needs to support. If you only plan to support the latest releases of current OES and SLES versions 50 GB will be a good starting point. If you plan to support more releases or multiple support packs per release 100 GB might be a better choice. As we use Logical Volume Management (LVM) for this disk its capacity can easily be increased if the need should arise.

Disk Partition Layout and Directory Structure

We strongly recommend to use GPT partitioned disk devices. For grub2 to be able to boot from such a device on BIOS systems a small BIOS Boot partition (type: 0x107, 263) is required to hold the second sector of the bootloader. 8 MB is sufficient for this purpose.

On UEFI systems the first partition must be the EFI system partition (type 0xEF, 239) with a FAT file system. On SLES15 this partition has a capacity of 512 MByte.

We recommend that you create a 2 GB swap partition next and format the remainder of the first disk with btrfs with default subvolume handling and mounted at /. On the second disk we recommend to create a single LVM volume group `data` with a single logical volume `data` formatted with xfs and mounted at `/data`. The recommended disk partitioning is summarized in the following table.

Table 4-1 Partition Layout for an UEFI AutoYaST Server

Disk	Partition	Type	Volume Group	Volume	Mount Point	File System	Size [GB]
1	1	bios_grub	n/a	n/a	n/a	fat	0.5
1	2	Swap	n/a	n/a	swap	swap	2
1	3	Linux	n/a	n/a	/	btrfs	27.5
2	1	LVM	data	data	/data	xfs	50

Underneath `/data` the following directories need to be created to accommodate the CIF:

- ♦ `autoyast` scripts, configuration files, control file and class files used by the solution
- ♦ `boot_cd_build` working directory for Custom Boot ISO
- ♦ `install` directory for the mount-points for the installation sources
- ♦ `isos` directory for the installation sources and the AutoYaST boot ISO

Software Requirements

The AutoYaST server can be based on SLES12 (pattern “Base System”) or on SLES15 (pattern “Enhanced Base System”, modules Basesystem and Server-Applications) with the latest support pack. If you follow our recommendation to use HTTP to access the repositories the server also needs to have the `apache2` package installed.

The creation of the custom AutoYaST boot medium requires `xorriso`. On SLES12 this is provided as part of the `libburnia-tools` package while SLES 15 comes with a `xorriso` package that you need to install. On SLES15 you also need to install the `dosfstools` package. On SLES15 SP2 and later the `mtools` package needs to be installed as well.

`create-ay-iso.sh` is used to create the custom AutoYaST boot medium and verifies that the required packages are installed.

Installation Repositories

An installation repository must contain the same directory structure as the corresponding installation media. The simplest way would be to copy the installation media into a repository directory and to provide remote access to the resulting directory structure over HTTP.

However, to protect against unintended alterations of installation sources we very strongly recommend to place the ISO files of the different installation media into a directory `/data/isos` and to loop mount them read-only to `/data/install/<product><version>/<sp>/cd1` through `/etc/fstab`. In case of an initial release of a product `fcs`, `ga`, or `sp0` can be used instead of `<sp>`.

With SLES15 the installer media is mounted at `/data/install/sles15/<sp>/cd1`. The associated packages ISO is mounted at `/data/install/sles15/<sp>/Packages` instead. This is illustrated in the following figure:

Figure 4-1 Sample `fstab` Entries for an AutoYaST Server

```
# AutoYaST server
/dev/data/data                                /data                                xfs          defaults    1 2

# Loop mounts for AutoYaST server
/data/isos/OES2018-SP2-DVD-x86_64-DVD1.iso    /data/install/oes2018/sp2/cd1/      iso9660      loop,ro     0 0
/data/isos/SLE-12-SP5-Server-DVD-x86_64-GM-DVD1.iso /data/install/sles12/sp5/cd1/      iso9660      loop,ro     0 0
/data/isos/SLE-15-SP1-Installer-DVD-x86_64-GM-DVD1.iso /data/install/sles15/sp1/cd1/      iso9660      loop,ro     0 0
/data/isos/SLE-15-SP1-Packages-x86_64-GM-DVD1.iso /data/install/sles15/sp1/Packages  iso9660      loop,ro     0 0
```

Control File and Class File Repository

The main AutoYaST control file (`default`) is stored in `/data/autoyast/xml`. All class files must be located in `/data/autoyast/xml/classes` as this directory name is hard coded in the AutoYaST plugins (see [“Class Files” on page 19](#)).

We do not use any class files for the installation of SLES systems. For OES systems our solution uses class files to query the passwords for the OES installation user, for the AD Administrator in case of NSS-AD, and for the DSfW Administrator.

Apache Web Server Configuration

Apache2 listens for requests on all interfaces by default as defined by the following directive in `/etc/apache2/listen.conf`:

```
Listen 80
```

If only a particular interface is to be used, its IP address must be added to this directive. For example:

```
Listen 10.10.10.101:80
```

Apache2 should be configured for automatic service start:

```
systemctl enable apache2
```

In the next step the directories that will be accessed over HTTP and their URLs need to be configured. Apache2 has a default directive which ensures that all files located in the directory `/etc/apache2/conf.d` and suffixed with `.conf` will be read and processed every time Apache2 is loaded.

We are using the file `inst_server.conf` to provide all configuration directives for the Consulting Installation Framework.

```
# httpd configuration for the Consulting Installation Framework AutoYaST
# server included by httpd.conf
```

```
Alias /oes2018sp2           /data/install/oes2018/sp2/cd1
Alias /sles12sp5           /data/install/sles12/sp5/cd1
Alias /sles15sp1          /data/install/sles15/sp1/cd1
Alias /sles15sp1_Packages /data/install/sles15/sp1/Packages

Alias /install             /data/install
Alias /isos                /data/isos

Alias /autoyast            /data/autoyast
Alias /configs             /data/autoyast/configs
Alias /files               /data/autoyast/files
Alias /info                /data/autoyast/info
Alias /lib                 /data/autoyast/lib
Alias /scripts             /data/autoyast/scripts
Alias /upgrade             /data/autoyast/upgrade
Alias /xml                 /data/autoyast/xml

<Directory "/data/autoyast">
# +FollowSymLinks does work in Directory but it does not work
# in DirectoryMatch!!!
# (https://httpd.apache.org/docs/2.4/mod/core.html#options)

Options +Indexes +FollowSymLinks
IndexOptions +NameWidth=*
Require all granted
</Directory>

<DirectoryMatch "^/data/(install|isos).*">
Options +Indexes
IndexOptions +NameWidth=*
Require all granted
</DirectoryMatch>
```

The various `Alias` directives are used to map descriptive URLs to lengthy file system paths allowing for easier access to the different parts of the installation framework.

There are two different methods to grant access to these directories. The `Directory` directive grants access to the specified directory and its sub-directories with support for symbolic links. This approach has been used to grant access to the `autoyast` branch of the file system structure used by the CIF. Access to another part of the file system with symbolic link support would require a separate `Directory` directive with the same options.

The `DirectoryMatch` directive uses a regular expression to match multiple directories but does not support symbolic links. It has been used to grant access to the directory where the ISO files are stored and to the directory where they are loop-mounted.

AutoYaST Extended – The Configuration File Approach

AutoYaST control files created by YaST modules always contain values from the system where the file has been created. As a consequence, every such control file can only be used without modification to re-install the same server again.

The installation of a different server requires that a new XML file with the information specific for the new server needs to be provided. Theoretically this could be achieved by copying the control file of an existing server and making the required adjustments. However, this would be a very involved process requiring a highly skilled administrator and considerable effort before a new server could be installed. As a result, the advantage of an automated installation would be greatly reduced by such a cumbersome approach.

To overcome this limitation we have developed the Consulting Installation Framework (CIF), which almost completely automates the creation of the AutoYaST control file and allows for fast, standardized, unattended installations with minimal administrator intervention.

This approach uses standard processes explained in [“AutoYaST Control File” on page 16](#), as well as some additional AutoYaST features described in this section.

Design

After evaluating numerous automated installations of OES servers and SLES servers we have found that these installations have some commonalities:

- ♦ AutoYaST control files contain information that is identical for all customer environments, such as meta XML information, system user settings, and services configurations.
- ♦ There are always properties which differ between customer environments and that might even differ between locations or sites of a particular customer but are the same for all servers in that particular environment. Time sources, name resolution (DNS, SLP, hosts), LDAP server, and replica server are examples for this kind of configuration information.
- ♦ Control files typically also contain some server specific information such as the host name, IP addresses, network masks, MAC addresses, and PCI IDs.

The Consulting Installation Framework aims at reducing the complexity of managing control files. Its design is based on the following principles:

- ♦ The framework must fit into every customer environment without code modification.
- ♦ The installation is based on XML class files.
- ♦ The configuration of SLES and OES services is split into individual XML files (snippets) that can be combined into service types as needed.
- ♦ All dynamic information is removed from the XML files and replaced by placeholder variables. The names of these variables are in all uppercase and enclosed in %%. These modified XML files containing placeholders are referred to as **template files**.

- ◆ Information relevant for multiple systems is stored in a set of configuration text files supporting the same set of variables to allow overwriting general configuration information with more specific information for a subset of systems.
- ◆ Server-specific information is stored in a semicolon-separated server configuration file.
- ◆ The system being installed retrieves the required template files to its `/tmp/profile` directory and merges them into a complete template control file.
- ◆ The system being installed also retrieves the required configuration text files to its `/tmp/profile` directory and replaces all placeholders with the actual values, resulting in a server-specific `autoinst.xml` file created on the fly.
- ◆ The set of variables along with their assigned values used during the installation will be stored in the file `/root/install/variables.txt` for documentation purposes and future use.
- ◆ The framework is implemented in a main library and a set of shell scripts that are developed and maintained by Micro Focus Consulting Germany in cooperation with SUSE Consulting Germany.
- ◆ Customer-specific requirements can be implemented in a custom library that is already integrated in the framework. It can use every function of the main library.

This design approach has two fundamental requirements: the ability to execute scripts on the system being installed and the ability to modify the default `/tmp/profile/autoinst.xml` control file created by AutoYaST, before the actual installation begins.

The first requirement is met by AutoYaST's **pre-script stage** at the beginning of an installation within the initial RAM disk. In this stage, shell and Perl scripts specified in the control file can be retrieved from local or remote repositories and can be executed.

After the pre-script stage, AutoYaST checks for a file named `/tmp/profile/modified.xml`. If this file exists, `autoinst.xml` is renamed to `pre-autoinst.xml` and `modified.xml` is renamed to `autoinst.xml`. The installation then starts and processes the information in this new control file. This meets the second requirement.

If `modified.xml` should not exist, the original control file is used instead.

Implementation

This section first gives a brief overview of how the Consulting Installation Framework is used to implement the AutoYaST workflow depicted in [Figure 2-1, “AutoYaST Process Overview,” on page 13](#) before it provides a detailed description of each element.

Overview

By default the `autoyast=` parameter at the boot prompt of the system being installed points to the URL `http://<AutoYaST-Server>/xml/default`.

This control file is retrieved and processed. Contrary to what one might expect the control file for our solution does not contain a `class` section. Instead it contains a `scripts` section to define the pre-script `pre-fetch.sh` and a `software` section that is only required for the installation of SLES15.

5

After processing the control file AutoYaST enters the pre-script stage and retrieves the `pre-fetch.sh` script via HTTP. `pre-fetch.sh` is the command center of the solution that manages all aspects of building the customized control file for the server to be installed.

The script first retrieves and sources the system configuration file `AY_MAIN.txt` and the main library `ay_lib.sh` to define the environment for the CIF. From this point forward, `pre-fetch.sh` exclusively uses functions from the main library unless additional functions are defined in the custom library that is retrieved and sourced next to further modify the CIF environment, if required.

`CUSTOMER.txt` is the first configuration text file that is retrieved and sourced resulting in a first set of values that are assigned to the variables that describe the system that will be installed.

In the next step the server configuration file `server.txt` is searched for the IP address that must be provided at the beginning of the installation. This address is used to identify the entry that provides the server type, the intended partitioning, the software to be installed and some other information.

If the entry for the new server in `server.txt` should specify an eDirectory tree name and/or a server location, the corresponding configuration text files are retrieved and sourced as well. This will overwrite any values that have been assigned to variables from `CUSTOMER.txt` earlier. `server.txt` is processed a second time to preserve more specific values from the server configuration file over values from the configuration text files. This will assign the final value to every variable.

The variables along with their value are compiled in the file `variables.txt` that will be stored in the `/root/install` directory of the installed system. This file is used to make the variable definitions available to other scripts.

In the next step, the original control file, `/tmp/profile/autoinst.xml`, is copied to `/tmp/profile/modified.xml` and all subsequent control file modifications happen on this copy.

Based on the specifications obtained from the configuration text files additional template files are retrieved and are merged into the control file. Among other things, these files determine the disk partitioning of the new server and which patterns and packages will be installed.

Finally, the service type specified in Field 14 of each line in the server configuration file `server.txt` is evaluated and the template files required to build the specified service type are retrieved and merged into the control file.

These XML snippets contain some SLES configuration information, such as the boot loader configuration, the memory reservation for the `kdump` kernel, or the scripts to be executed in the post-scripts and init-scripts phase of the installation.

For OES servers additional files contain information about the configuration of the various services such as eDirectory, NSS, and CIFS that are also merged into the control file.

As the last execution step, `pre-fetch.sh` replaces all placeholder variables that originated from the different template files and XML snippets that are now part of the `/tmp/profile/modified.xml` control file with the values that have been derived from the various configuration text files and the server configuration file. This is the same configuration information that you would enter into YaST during a manual installation of SLES or OES.

If any error should occur during this process, `pre-fetch.sh` outputs an error message on the screen of the system being installed indicating where the error has occurred. The administrator can inspect the error message and decide whether to abort or to continue the installation. For the great majority of errors that can occur the safe option is to abort the installation!

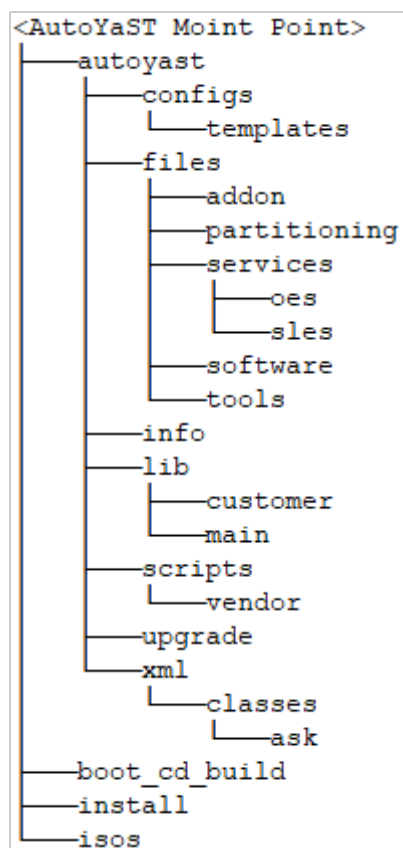
When `pre-fetch.sh` terminates without error, AutoYaST picks up the `/tmp/profile/modified.xml` control file and processes it as explained in [“Design” on page 45](#).

The remainder of this section explains this process in more detail.

Directory Structure of the Consulting Installation Framework

[Figure 5-1](#) illustrates the directory structure for the Consulting Installation Framework underneath the top level directory `/data/autoyast`.

Figure 5-1 Directory Structure



`/data/autoyast/xml` is the driver file repository directory introduced and explained in section [“Control File and Class File Repository” on page 41](#).

The directory `/data/boot_cd_build` is used to build the Installation Boot Medium as explained in [“Installation Boot Medium” on page 21](#).

The `/data/isos` sub-directories is used to store the installation sources and the Installation Boot Medium. `/data/install` is used to mount the installation sources as discussed in [“Installation Repositories” on page 41](#).

The purpose of the directories underneath `/data/autoyast` is explained in the following table.

Table 5-1 Directory Layout for the Consulting Installation Framework

Directory	Description
<code>.../configs</code>	All configuration files are stored in this directory: <ul style="list-style-type: none">◆ The system configuration file <code>AY_MAIN.txt</code>.◆ The customer configuration file <code>CUSTOMER.txt</code>.◆ Your tree name configuration files <code><TreeName>.txt</code>.◆ Your location configuration files (optional).◆ The server configuration file <code>server.txt</code>.
<code>.../configs/templates</code>	Holds template files to create your customer configuration file, your tree name configuration files, and your location configuration files.
<code>.../files/addon</code>	Holds the XML class files defining the Add-On products for the various Server Types used in Field 04 of <code>server.txt</code> .
<code>.../files/partitioning</code>	Holds the XML class files for the partitioning of the various disk types used in Field 07 of <code>server.txt</code> .
<code>.../files/services/oes</code>	Holds the XML class files configuring the OES services that are part of the service types defined in <code>CUSTOMER.txt</code> .
<code>.../files/services/sles</code>	Holds the XML class files configuring the SLES services that are part of the service types defined in <code>CUSTOMER.txt</code> .
<code>.../files/software</code>	Holds the XML class files for the different software selections used in Field 08 of <code>server.txt</code> .
<code>.../files/tools</code>	Stores the <code>check_errors.xml</code> ask file required for the error handling.
<code>.../info</code>	Holds files providing additional boot options.
<code>.../lib/customer</code>	Stores the custom library <code>customer_lib.sh</code> that can be used to implement customer-specific functions to further customize the installation process.
<code>.../lib/main</code>	Stores the main library <code>ay_lib.sh</code> provided by Micro Focus Consulting and SUSE Consulting. Most of the functionality of the Consulting Installation Framework is implemented in this library. It must not be changed or modified in any way. Use the customer library instead.
<code>.../scripts</code>	Every script that is executed in the pre-, post- or init-phase of the installation process is stored in this directory.
<code>.../scripts/vendor</code>	Stores scripts to perform additional configuration tasks. Any script in this directory named <code>*.sh</code> will be executed by the script <code>post-inst.sh</code> .
<code>.../upgrade</code>	The control file and any other files that need to be retrieved to perform an upgrade via AutoYaST need to be stored in this directory.

Configuration Files

The definition of the environment in which the Consulting Installation Framework (CIF) operates and the configuration information for the servers being installed using the CIF is defined in a set of configuration files that by default are stored in `/data/autoyast/configs`.

Configuration file entries consist of three pieces of information:

- ♦ The type of the variable.
 - This can be a string, an IP address or DNS name, an URL, a file or directory name, a list of values, or any custom definition.
- ♦ A short description of the information stored in the variable,
- ♦ The actual definition in the format `VARIABLE="value"`.

IMPORTANT: The variables are only documented in the files. If you want to learn more about the meaning of a certain variable or the syntax to use with a variable please refer to the comments in the configuration files.

The following example defines the name of the OES Installation user that is used to authenticate against the eDirectory tree when installing a new server into an eDirectory tree:

```
## Type String
## Description: fully distinguished name of the OES Installation user in
LDAP syntax
OES_INSTALL_USER="cn=Admin,o=MF"
```

Although strictly technical it is not always required, we strongly recommend that you enclose all values in double quotes.

There is one notable exception: because the encrypted password for the root user typically contains “\$”, it must be enclosed in single quotes instead of double quotes. As a best practice, any setting that is not used in a particular configuration file should be commented.

System Configuration File `AY_MAIN.txt`

This configuration file defines the directory structure to store the various components of the CIF, the name of the files used by the CIF, and the URLs to retrieve this information.

IMPORTANT: As explained in [“The Script `pre-fetch.sh`” on page 61](#), the name and path of this file must be hard-coded in `pre-fetch.sh` and any other shell script used by the CIF that needs access to the configuration files.

These scripts use the variable `PREFIX`, to identify the CIF base directory (default: `autoyast`). The variable `AY_CONFIG_DIR` contains the path to the configuration files relative to the CIF base directory (default: `configs`).

The framework has been tested only with the default names used in this document and defined in our version of the system configuration file. If you should want to use different names, you can do so at your own risk, making the appropriate changes in `AY_MAIN.txt` and the scripts. However, we strongly recommend that you use the default name wherever possible.

Environment Configuration Files

The purpose of the environment configuration files is to separate information that applies to multiple servers from server-specific information provided by the server configuration file (default: `server.txt`). DNS name servers, time servers, the distinguished name of the UNIX configuration object, or the list of LDAP servers are examples for the type of information that is provided by these files.

To allow for sufficient flexibility to accommodate a wide variety of environments, the framework supports up to three environment configuration files defining the same set of variables. The variables are organized in the sections infrastructure, OES with the sub sections DHCP, DNS, DSfW, LDAP, NSS, and NSS-AD, and SLP and SLES. The files are discussed below in a global-to-local order.

Customer Configuration File `CUSTOMER.txt`

This file is the most global configuration file. It is mandatory for every systems that is installed using the CIF. Information that applies to all servers in all environments should be defined in this file.

There are three sections that by default are only found in this configuration file:

- ◆ Customer name (informational only).
- ◆ Definition of the service types.
These definitions combine all the XML class files that are required to configure all services that have been installed. The service type for an OES system always needs to include the service type for the corresponding SLES base.
- ◆ Information about the systems used for software updates.

Typically this information applies to all servers and environments. However, if required any of these three sections can be copied to any of the other environment configuration files. For instance if you plan to use a different system to provide updates to servers in a specific environment just copy the last section of `CUSTOMER.txt` to the configuration file for this environment and specify the appropriate values for the copied variables.

Tree Name Configuration File `<TREE_NAME>.txt`

Most customers have more than one eDirectory tree in their environment. Each of these eDirectory trees requires its own configuration file and only values that apply to all servers of a tree should be specified in these files.

Even so the `CUSTOMER.txt` configuration file may appear to be sufficient to provide all required settings in small environments with a single tree and only a few servers, the tree name configuration file must always exist to install OES servers (even if not a single values should be set in the file) as it determines in which eDirectory tree the server will be installed.

The name of this file consists of the eDirectory tree name specified in Field 10 of the `server.txt` server configuration file and the suffix ".txt". For example, if an eDirectory tree is named MYCOMPANY-TREE, then `pre-fetch.sh` searches for a file `.../autoyast/configs/MYCOMPANY-TREE.txt` (case-sensitive).

Location Configuration File

Configuration information defined in the customer configuration file or in the tree name configuration file can be overwritten with information from an optional location configuration file. You can choose any file name for this type of configuration file. If it is used, it must be specified in Field 13 of the server configuration file. Each server can only use a single location configuration file.

“Location” can represent a lot of things. It could provide configuration information specific for a country, a city, a site, a building, a data center or a server room, or even just a LAN segment - basically anything that defines a group of servers that need different configuration information than other systems.

You also can use this configuration file to apply specific settings to a logical group of servers. For instance if you would want to set a particular root password on a group of servers this could be achieved by using a location configuration file that only defines that password.

If the only configuration information that is specific for groups of servers is their default gateway it is not required to create a location configuration file for every group. Instead, the default gateway of each server can be provided in Field 3 of the server configuration file from where it is retrieved when this file is processed the second time.

Configuration File Templates

Templates for the three environment configuration files are available in `/data/autoyast/configs/templates`. When you build your CIF environment you need to copy these template files to `/data/autoyast/configs` and supply the values for the different variables. Tree name configuration file(s) and location configuration file(s) need to be named to match your environment.

Server Configuration File `server.txt`

The server configuration file defines the server-specific information for the Consulting Installation Framework.

Each server configuration is represented by a single line, with its fields separated by semicolons. The current server configuration file contains 14 fields explained in the following table:

Table 5-2 *Field Descriptions for the `server.txt` Server Configuration File*

No.	Field Name	Meaning	Example	Multivalue Required
1	HOST_NAME	Short name of the server being installed	machine01	No Mandatory
2	IP_ADDRESS/CIDR	IP address and net mask of the server being installed; used as system identifier. Net mask in CIDR notation. For example, 255.255.240.0 = 20	10.10.10.10/24	No Mandatory
3	GATEWAY	IP address of the default gateway	10.10.10.1	No Optional

No.	Field Name	Meaning	Example	Multivalue Required
4	SERVER_TYPE	<p>Pure SLES or OES systems, including version and support pack level.</p> <p>ovl at the end of the identifier denotes that the system will be installed from the combined SLES/OES installation medium.</p> <p>Server types for different environments can be created by appending identifiers such as -DEV or -TST.</p> <p>IMPORTANT: There must always be a corresponding <code>addon_products-\$SERVER_TYPE.xml</code> file in the <code>.../files/addon</code> directory.</p>	<p>sles12sp5-DEV</p> <p>sles15sp1-TST</p> <p>oes2018sp2-PRD</p> <p>oes2015sp1-ovl-DEV</p>	No Mandatory
5	DEVICE_NAME0	Name of the first system disk device.	/dev/sda, /dev/xhda	No Mandatory
6	DEVICE_NAME1	Name of the second system disk device.	/dev/sda, /dev/xhda	No Optional
7	PART_FILE	Partitioning class file. Must be located in <code>.../files/partitioning</code> .	The file name should identify type and size of the device(s) for which the partitioning has been defined (“Partitioning” on page 57).	No Mandatory
8	SOFT_FILE	Software class file. Must be located in <code>.../files/software</code> .	<p>The file name must identify the SERVER_TYPE and the purpose of the server for which the software selection is valid.</p> <p>For example: <code>soft-oes2018_eDir.xml</code> <code>soft-sles15_ZCM.xml</code></p>	No Mandatory
9	ZCM_KEY_LIST	<p>Keys for the registration of the new device with the ZENworks Management Zone. Multiple keys, separated by ':' are possible.</p> <p>The first key is used for the registration with the ZCM zone. All other keys are used for subscribing to server groups for managing configuration and software updates.</p>	<p>Examples for valid ZCM keys for a production Environment are:</p> <p>Location: PRD_eDir Configuration: PRD_eDir_GRP Update: PRD_OES2018SP2</p>	Yes Optional

No.	Field Name	Meaning	Example	Multivalue Required
10	TREE_NAME	eDirectory tree name. There must be a configuration file <TREE_NAME>.txt in .../configs. The file name is case-sensitive.	My_Tree	No Mandatory for servers with eDir.
11	TREE_TYPE	Determines whether a new tree will be created or the server will join an existing tree.	existing new	No Mandatory for servers with eDir.
12	SERVER_CONTEXT	Server context in LDAP syntax.	ou=servers, ou=services, o=MF	No Mandatory for servers with eDir.
13	SERVER_LOCATION	Configuration file determining all aspects of the physical server location such as the following: <ul style="list-style-type: none"> ◆ Default gateway ◆ LDAP server list ◆ NTP server list There must be a configuration file <SERVER_LOCATION> in .../configs. The file name is case-sensitive. In smaller environments, all of this information can be provided in the tree name configuration file or in the customer configuration file.	Utah.txt Provo.loc DC1.info	No Optional
14	SERVICE_TYPE	XML profile as defined in the customer configuration file. The file names are case-sensitive. The XML files referenced by the profile must exist in .../files/services/oes and .../files/services/sles.	SLES15_BASE, OES2018_EDIR_IMAN	No Mandatory

XML Snippets

The files stored in the directory structure underneath `.../files` have been created by retrieving the relevant section from an `autoinst.xml` file saved after a manual installation of an OES or SLES system with all patterns and by replacing dynamic information with placeholders.

In the following example the DNS information has been replaced with place holders enclosed in `%%`.

```
<dns>
  <dhcp_hostname config:type="boolean">false</dhcp_hostname>
  <dhcp_resolv config:type="boolean">>true</dhcp_resolv>
  <hostname>%%HOST_NAME%%</hostname>
  <domain>%%DOMAIN%%</domain>
  <searchlist config:type="list">
    <search>%%SEARCH_LIST%%</search>
  </searchlist><nameservers config:type="list">
  <nameservers config:type="list">
    <nameserver>%%NAMESERVER%%</nameserver>
  </nameservers>
</dns>
```

If you build your own AutoYaST server you should carefully inspect each of these files to ensure that the files provided by our solution meet your specific requirements.

Add-On Products

Each server installed using the Consulting Installation Framework requires one add-on XML file. This file must be named `addon_products- $\$$ SERVER_TYPE.xml`, where `$\$$ SERVER_ TYPE` is the value of Field 04 in the server configuration file `server.txt`.

For current OES and SLES systems, these files determine which updates for OES and SLES (YUM repositories) are deployed as part of the installation process including updates for SLES Modules if they should be installed.

Typically the YUM repositories for the various updates will be provided by a ZCM Primary server configured as described in [“YUM Repositories Derived From Frozen Patch Level Bundles and Pool Bundles”](#) on page 107.

In the case of OES2018 SP2 each of the two update repositories needs to be specified as a separate list entry in the add-on XML file:

```
<listentry>
  <!-- SLES12 SP5 for OES2018 SP2 Updates -->
  <media_url>%%YUM_SERVER%%/zenworks-yumrepo/OES2018-SP2-SLES12-SP5-
Updates-PRD</media_url>
  <product>OES2018-SP2-SLES12-SP5-UPDATES</product>
  <product_dir>/</product_dir>
  <name>OES2018 SP2 SLES12 SP5 Updates (PRD)</name>
  <alias>OES2018-SP2-SLES12-SP5-UPDATE-PRD</alias>
</listentry>
<listentry>
  <!-- OES2018 SP2 Updates -->
  <media_url>%%YUM_SERVER%%/zenworks-yumrepo/OES2018-SP2-Updates-PRD</
media_url>
  <product>OES2018-SP2-UPDATES</product>
  <product_dir>/</product_dir>
  <name>OES2018 SP2 Updates (PRD)</name>
  <alias>OES2018-SP2-UPDATE-PRD</alias>
</listentry>
```

SMT creates YUM repositories when it synchronizes updates from the customer centers. The list entries for the OES2028 SP2 updates provided by an SMT server only differ in the media URL:

```
<listentry>
  <!-- SLES12 SP5 for OES2018 SP2 Updates -->
  <media_url>%%YUM_SERVER%%/repo/$RCE/OES2018-SP2-SLES12-SP5-Updates-/
sle-12-x86_64</media_url>
  <product>OES2018-SP2-SLES12-SP5-UPDATES</product>
  <product_dir>/</product_dir>
  <name>OES2018 SP2 SLES12 SP5 Updates (PRD)</name>
  <alias>OES2018-SP2-SLES12-SP5-UPDATE-PRD</alias>
</listentry>
<listentry>
  <!-- OES2018 SP2 Updates -->
  <media_url>%%YUM_SERVER%%/repo/$RCE/OES2018-SP2-Updates/sle-12-
x86_64</media_url>
  <product>OES2018-SP2-UPDATES</product>
  <product_dir>/</product_dir>
  <name>OES2018 SP2 Updates (PRD)</name>
  <alias>OES2018-SP2-UPDATE-PRD</alias>
</listentry>
```

If an URL should be misspelled or a repo is not reachable AutoYast will report “Failed to add add-on product” early in the installation process.

For OES systems, up to and including OES2015 these files can also define the installation source for OES if the combined OES Install Media should not be used. In this case the URL for the OES add-on product uses the aliases defined in the `inst_server.conf` Apache configuration file described in [“Apache Web Server Configuration” on page 41](#).

Partitioning

Files in this sub-directory contain partitioning information for up to two disk devices. As a general rule, from SLES12 onwards we strongly recommend that you boot from GPT partitioned devices. Class files for this kind of partitioning are identified by `GPT` in the file name.

We also recommend to use UEFI instead of BIOS as your system firmware which requires an EFI system partition on your system disk. Class files for this kind of partitioning are identified by `UEFI` in the file name.

Finally, we recommend that you use `btrfs` as the file system for the system disk of any system based on SLES12 or later. Due to the snapshots created and maintained by `btrfs` these disks need to be larger than the system disks you have been using with earlier releases of SLES. We consider a capacity of 30 GB and a 2 GB swap partition sufficient for test and development systems. Production systems should use a minimum disk capacity of 70 GB and a swap partition of 4GB.

However, `eDirectory` does not support and cannot be installed on `btrfs`. Therefore we create an additional primary partition formatted with `xfs` and mounted at `/var/opt/novell/eDirectory`. The size of this partition obviously depends on the size of your `eDirectory` tree. As a guideline determine the disk space used by `eDirectory` on a server that holds your complete `eDirectory` tree and use three times this space as the absolute minimum for the `xfs` partition.

Most customers prefer to put this `xfs` partition into LVM for easier expansion. For this purpose we name the LVM Volume Group and the LVM Logical Volume `eDir`. Class files that implement this kind of partitioning are identified by `btrfs-lvm` in the file name.

WARNING: These partitioning files use the device names specified in Field 05 and Field 06 of `server.txt`, typically `/dev/sda` and optionally `/dev/sdb`.

When you (re-)install a system with access to shared devices these kernel device names could be assigned to a path to a shared device. This would result in data loss because the installation would partition and format the shared device!

Therefore, when you (re-)install such a system, make sure that access to shared devices has been temporarily disabled by either removing host access on the storage system, by revoking access to the SAN through a zoning change or by simply disabling ALL host bus adapters for shared devices!

Only re-enable access to shared devices once the installation and configuration has been completed successfully.

For ZENworks Configuration Management servers we recommend to also use LVM and to place the ZCM data on a separate disk mounted at `/var/opt/novell/zenworks`. The size of this disk obviously depends on the content that will be replicated to the Satellite server.

Following the above recommendation OES servers that also are a ZCM Satellite Server use two LVM VGs and LVs named `eDir` and `ZCM` respectively (`part-vmware-UEFI-btrfs-lvm-70G_2nd-HDD_ZCMSat.xml`).

For a ZCM Primary Server a system disk of 50 GB is not sufficient and we recommend a minimum of 100 GB. `part-vmware-UEFI-btrfs-lvm_2nd-HDD_ZCM.xml` creates the BIOS Boot partition and a 4 GB swap partition on the first disk and assigns the remaining capacity to the `btrfs` partition mounted at `/`. The second disk is used for the `xfs` formatted LVM volume `ZCM`.

These are only a few examples to illustrate how XML class files can be used to define the desired disk partitioning.

Software

The files in this sub-directory are referenced in Field 08 of `server.txt` and determine which patterns and packages will be installed on the target system.

The guideline for software selection should always be to only install what is required. For instance, you might want to define one software selection for dedicated eDirectory/Login/iManager servers and another one for cluster nodes providing print and file services accessed through AFP, CIFS, or NCP. Branch office servers providing additional services such as DHCP and DNS might require yet another software selection.

A SLES system accommodating a ZCM Primary Server might require other packages than a SLES system hosting some monitoring system or GroupWise.

Services

This area is divided into two directories where the class files defining the configuration of the SLES and OES components that have been installed are stored.

In general these files apply to all versions of SLES or OES supported by the framework. File names including a version number such as `ca_mgm_sles12.xml` apply to versions up to and including the version specified as part of the file name.

Services – OES

Files in this sub-directory apply to OES systems only. The `oes_base.xml` file contains all of the configuration settings that are required by every OES server independent of specific patterns, such as the LDAP servers for OES, the eDirectory, SLP, and NTP configuration information, or the configuration of LUM or the NCP Server.

Note that Domain Services for Windows (DSfW) domain controllers for the Forrest Root Domain need to use their own version of this file (DSfW child domains are currently not supported by CIF). One of the `oes_base*.xml` files needs to be part of every service type for OES systems defined in `CUSTOMER.txt`.

The other files in this directory configure individual OES services such as AFP, NSS, or iPrint, to only name a few. You cannot configure Novell Cluster Services as part of the initial installation, because NCS in most cases needs access to shared storage which can only be configured once the initial installation is complete. Therefore the configuration of NCS must always be done manually after you have completed your storage configuration and verified proper storage access. Also consider the warning on shared devices on the previous page.

Services – SLES

Files in this sub-directory such as `bootloader.xml`, `general.xml`, or `net.xml` apply to SLES systems as well as to OES systems. `net-forward.xml` configures the network in the same way as `net.xml` but additionally enables IP forwarding. This is required when using containers.

There is a different version of the class file `system*.xml` configuring the system services for every SLES release. The file to configure the SLES certificate authority does not apply to SLES15.

`scripts-zcm.xml` defines the scripts that are executed in the post-phase of the installation. This class file will execute the script `zcm-install.sh` and if a ZCM key is defined for the system being installed the ZENworks Adaptive Agent will be installed and the system will be registered with the ZCM zone.

Tools

This directory currently only holds the `check_errors.xml` ask file that is part of the error checking mechanism of the `pre-fetch.sh` pre-script.

Info Files

These files provide an overflow mechanism to accommodate boot options that do not fit on the limited kernel command line of `linuxrc`. We provide templates for SLES15 SP1 to either initiate an automated installation, a manual installation or to boot into the rescue system.

The following example shows an info file with all non-network related boot options to perform an AutoYaST installation of SLES15 SP1 using the installer updates that are provided by a YUM repository on a ZENworks Configuration Management Primary Server.

```
autoyast=http://<IP address of your AutoYaST server>/xml/default-<IP
address of your AutoYaST server>
install=http://<IP address of your ISO server>/sles15sp1/
## Installer self-updates - only enable ONE option!!!
## from SMT
# self_update=http://<IP address of your YUM server>/repo/SUSE/Updates/
SLE-INSTALLER/15-SP1/x86_64/update/
## from YUM
self_update=http://<IP address of your YUM server>/zenworks-yumrepo/SLE15-
SP1-Installer-Updates
```

IMPORTANT: To use these files you need to copy them and to manually replace `<IP address of your . . .>` with the IP address or your AutoYaST, ISO and YUM server respectively.

As these files are evaluated very early in the installation process AutoYaST cannot do this replacement for you.

This info file is integrated into the menu entry of a grub or grub2 configuration file by means of the `info=` boot option.

```
### sles15sp1
title SLES15 SP1
    kernel /kernel/sles15sp1/linux info=http://10.10.10.101/info/info-
sles15sp1-ay.txt netsetup=hostip,gateway netmask=255.255.255.0 gateway=10.
netwait=10
    initrd=/kernel/sles15sp1/initrd
```

Libraries

The framework uses two libraries. The main library `ay_lib.sh` (a set of shell functions that can be reused to reduce code fragments) developed and maintained by Micro Focus Consulting Germany in collaboration with SUSE Consulting Germany contains nearly all code used by `pre-fetch.sh` and other scripts of the framework.

This library must not be changed or modified in any way as any potential update to the Consulting Installation Framework almost certainly will include a newer version of this library that will replace the current version removing any customer modification.

To accommodate customer-specific functionality, the empty `/data/autoyast/lib/customer/customer_lib.sh` library file is available. This file can either accommodate completely new functions or it can amend functions from the main library. It can use any function that is already implemented in the main library.

New functions intended to be used in the pre-script phase most likely will also require modifications of `pre-fetch.sh` (that may or may not get lost with a potential update to the framework). To extend a function from the main library the function needs to be copied to the customer library and modified there. The modifications may only add functionality but not modify or remove existing functionality as the function in the customer library will replace the function with the same name from the main library.

For example, you could copy the function `do_replace()` to the customer library and enhance it to replace additional placeholders defined by you, but you must not alter the existing replacements in any way. However, if your customer library should replace functions from the main library you need to be prepared to adjust your functions with any modification that might be contained in the main library of a potential newer version of the framework.

`customer_lib.sh` is sourced and executed by `pre-fetch.sh` before the configuration files are retrieved and processed. No update to the framework will ever touch this particular library.

The Default File Again

The full default file used by the Consulting Installation Framework is displayed below:

```
<?xml version="1.0"?>
<!DOCTYPE profile>
<profile xmlns="http://www.suse.com/1.0/yast2ns" xmlns:config="http://
www.suse.com/1.0/configs">
  <scripts>
    <!--
      This section is mandatory and must not be removed.
      Amongst other things it is used to determine the
      IP address of the AutoYaST server in the post-inst
      phase
    -->
    <pre-scripts config:type="list">
      <script>
        <interpreter>shell</interpreter>
        <filename>pre-fetch.sh</filename>
        <location>http://<IP address of your AutoYaST server>/autoyast/
scripts/pre-fetch.sh</location>
```

```

        <notification>Please wait while pre-fetch.sh is running...</
notification>
        <feedback config:type="boolean">false</feedback>
        <debug config:type="boolean">false</debug>
        </script>
    </pre-scripts>
</scripts>
<software>
<!-- Required for SLES 15 -->
    <products config:type="list">
        <product>SLES</product>
    </products>
</software>
</profile>

```

The scripts section of this file defines the `pre-fetch.sh` pre-script file located in the `scripts` directory on the AutoYaST server. This information causes the AutoYaST engine to retrieve and execute this script.

This section is one of the few places where customer-specific values must be hard-coded. The IP address of the AutoYaST server is unique to every customer environment and cannot be derived from environment parameters. Therefore it must be explicitly specified in the URL to the `pre-fetch.sh` script in the location directive of the pre-script definition in the default file, making this file customer specific.

The software section is only required for the installation of SLES15 servers. Earlier SLES versions and OES ignore this section.

The Script `pre-fetch.sh`

The first action of this script is to retrieve the system configuration file `AY_MAIN.txt`. This is implemented in the function `get_main_config_files()` that first derives the IP address of the AutoYaST server being used from the command line.

The script needs three other pieces of information to be able to retrieve the required files from the AutoYaST server:

- ♦ The name of the top-level directory of the Consulting Installation Framework directory structure (`PREFIX="autoyast"`)
- ♦ The name of the directory where the configuration files are stored relative to the top-level directory (`AY_CONFIG_DIR="configs"`)
- ♦ The name of the system configuration file (`MAIN_CONFIG_FILE="AY_MAIN.txt"`)

None of this information can be retrieved from the system environment and therefore is hard-coded in the function `get_main_config_files()`. If you want to use a different name for the system configuration file or for any of the directories making up the path to it, ensure that you set the correct values here and that you adjust the aliases in the Apache configuration file `inst_server.conf` accordingly.

All other file names, directory names, and URLs required to retrieve files from the AutoYaST server are read from the system configuration file.

The same function also retrieves and sources the main library `ay_lib.sh` and the customer configuration file `CUSTOMER.txt` (or whatever name is defined for this file in your system configuration file).

In some cases, a customer might have requirements that cannot be accomplished by the code of the main library. For these cases, the `customer_lib.sh` file is provided. Code contained in this file is evaluated after code from the main library but before the replacement of the template variables takes place.

Configuration information such as the default gateway, a list of DNS name servers, or the DNS suffix search list, and also OES specific information such as the name of the installation user or the IP address of an existing replica server can be specified in three different files:

- ♦ The customer configuration file (default name `CUSTOMER.txt`; mandatory)
- ♦ The eDirectory tree configuration file (must be named `<TreeName>.txt`; mandatory for OES servers)
- ♦ A location configuration file (can use any name; optional)

After processing the customer configuration file, the script parses each line of the server configuration file `server.txt` and compares the IP address that has been entered at the beginning of the installation to the IP address part of Field 02 using the library function `parse_line`. The first match completes the parse operation and the resulting line is used to configure the server.

The tree name configuration file and the optional location configuration file for a particular server are defined in Field 10 and Field 13 of the server configuration file `server.txt`.

Overall `pre-fetch.sh` sources the three possible environment configuration files in the following order:

```
customer configuration file > eDirectory tree configuration file > location
configuration file
```

This process overwrites the value assigned to a variable from a global configuration file with information from more specific configuration files. For example, a list of NTP time servers specified in the customer configuration file as a company-wide default can be overwritten by specifying a different set of NTP servers in a configuration file representing a particular eDirectory tree. This NTP server list can be further modified by specifying yet another list of NTP servers in a location configuration file representing a particular site.

One of the central functions performed by `pre-fetch.sh` is to merge XML snippets into the final control file. In particular, the partitioning, software, and services XML files are merged completely into `/tmp/profile/modified.xml`. This functionality is based on XSLT techniques utilizing `/usr/share/autoinstall/xslt/merge.xslt`, which is part of any installation RAM disk provided by SUSE.

Another important function of the pre-script is the replacement of the placeholders with their real values. After parsing the server-specific line in `server.txt` and processing the specified configuration files, every variable will have its final value assigned. The replacement process now searches for every placeholder (enclosed in `%%`) within `/tmp/profile/modified.xml` and

replaces the whole string with the value of the corresponding variable. For example, any occurrence of `%%HOST_NAME%%` anywhere in the control file is replaced with the value held by the variable `HOST_NAME`.

The last action performed by `pre-fetch.sh` is a basic error check. First, it checks to see if the control file `/tmp/profile/modified.xml` still contains `%%`-signs (which implies that “`%%`” must exclusively be used to identify placeholders. Any other usage, even in comments, will result in an error during this check).

Then it looks for the file `/tmp/profile/errors.txt`. Various functions from the main library will create this file when an error condition is encountered, for instance, if the IP address that has been entered could not be located in the server configuration file, or in case the retrieval of a file fails. If this file exists, a pop-up is generated that displays the contents of the file along with the list of variables and their assigned values. The administrator can then decide to abort the installation process or to continue it.

Post Installation Scripts

Post-installation scripts perform actions that require an installed system. They must be specified within the `<scripts>` section of the AutoYaST control file.

The Consulting Installation Framework uses a separate class file (`/data/autoyast/files/services/sles/scripts-zcm.xml`) that specifies which scripts are to be executed at the end of the installation:

```
<?xml version="1.0"?>
<!DOCTYPE profile>
<profile xmlns="http://www.suse.com/1.0/yast2ns"
  xmlns:config="http://www.suse.com/1.0/configns">
  <scripts>

    <!-- chroot-scripts (after the package installation, before the first
boot) -->
    <chroot-scripts config:type="list">
      <listentry>
        <filename>create-variables.sh</filename>
        <interpreter>shell</interpreter>
        <source>

          <![CDATA[

# save variables in /root/install/variables.txt
if [ -f /tmp/profile/variables.txt ]; then
  mkdir -p /mnt/root/install
  cat /tmp/profile/variables.txt >>/mnt/root/install/variables.txt
fi
]]>
        </source>

        <notification>Please wait while variables list is
stored in /root/install/variables.txt . . .</notification>
      </listentry>
    </chroot-scripts>

    <!-- init-scripts (during the first boot of the installed system, all
services up and running) -->
```

```

<init-scripts config:type="list">
  <listentry>
    <!-- Miscellaneous changes -->
    <filename>post-inst.sh</filename>
    <interpreter>shell</interpreter>
    <location>%%AY_SERVER%%/%%PREFIX%%/scripts/post-inst.sh</location>
  </listentry>
  <listentry>
    <!-- Install ZCM agent -->
    <filename>zcm-install.sh</filename>
    <interpreter>shell</interpreter>
    <location>%%AY_SERVER%%/%%PREFIX%%/scripts/zcm-install.sh</
location>
  </listentry>
</init-scripts>
</scripts>
</profile>

```

In the chroot phase of the installation, just before the first boot the script `create-variables.sh`, defined inside the class file is executed and stores the file `variables.txt` that holds the list of variables and their assigned values into the `/root/install` directory.

When the system has performed its first boot and all services are running the installation reaches the init stage and executes the scripts specified inside the `<init-scripts>` tag. The Consulting Installation Framework uses two such scripts.

The Post-Installation Script `post-inst.sh`

This script sources `/root/install/variables.txt` created during the first stage of the unattended installation giving it access to the whole set of variables and their assigned values that have been used during that installation stage.

It is used to make configuration adjustments that are not possible in AutoYaST. Currently, it performs the following operations:

- ◆ Add comments to `/etc/ntp` in SLES12 and earlier.

If NTP is configured through AutoYaST `/etc/ntp.conf` only contains a few directives to configure the daemon. However, many customers would like to have a complete `ntp.conf` including the comments with explanations. This is not possible with AutoYaST and therefore the function `complete_ntp()` has been introduced.
- ◆ Disable IPv6.
- ◆ Remove “localhost” from the IPv6 loopback entry in `/etc/hosts`.
- ◆ Enable X forwarding for SSH connections over IPv4.
- ◆ Comment the include instruction at the end of `/etc/chrony.conf` on SLES15 SP4.

The script also executes any scripts named `*.sh` from `/data/autoyast/scripts/vendor`. This is intended as a simple way to add any customization you may need to the build environment.

The Post-Installation Script `zcm-install.sh`

If the service type of the system being installed defined in Field 09 in `server.txt` does not contain “ZCM” and at least one ZCM registration key is specified in Field 09 in `server.txt` this script will retrieve the ZENworks Adaptive Agent from the source specified by the variable `ZCM_AGENT_URL`.

The agent is installed and the system is registered with the ZENworks Management Zone (ZCM zone) using the first registration key. Additional ZCM registration keys specified in Field 09 are processed executing the `zac add-reg-key` command for each of them.

This process ensures the creation of the server object in the ZCM zone in the desired folder (using the first registration key specified). The server is then made a member of device groups (as defined by the additional registration keys) that are assigned to bundle groups holding configuration bundles and update bundles.

As a result configuration bundles are deployed and immediately installed on the new server. Updates have already been installed as part of the installation process however, their status in ZCM needs to be updated (see “[Assigning Pool Bundles and Update Bundle Groups](#)” on page 105).

After a reboot the new server is ready for production.

Server Upgrade Using AutoYaST

AutoYaST can not only be used to install new servers it can also be used to upgrade existing servers to the next support pack.

For OES2018 SP2 this is described in [Using AutoYaST for an OES 2018 SP2 Upgrade](#) in the OES 2018 SP2 Installation Guide. The AutoYaST control file required to perform an upgrade is the file `autoupgrade.xml` provided in the root directory of the installation medium for the new support pack.

However, this control file can only be used directly for servers installed in English using an US English keyboard. If your servers should use a different keyboard or are installed in a different language or with support for multiple languages you will need to adjust the corresponding sections of the control file.

```
:
<keyboard>
  <keymap>Your_Keyboard</keymap>
</keyboard>
<language>
  <language>Your_Language</language>
  <languages>Your_LanguageList</languages>
</language>
:
```

If you are upgrading to the next support pack after updates have been released for it you can integrate these updates into the upgrade by adding the appropriate add-on section to your control file.

To do so you can simply copy the section from the Add-On XML snippet that you would use to install a server with the new support pack (see [“Add-On Products” on page 55](#)). However, as the upgrade does not use the process that replaces placeholders with actual values you need to modify the media URLs and need to replace `%%YUM_SERVER%%` with the access protocol and the IP address or DNS name used to access your YUM repository server.

For example, if you want to include updates for OES2018 SP2 provided by a ZCM Primary Server for servers in your production environment this section would look as follows:

```

:
<add-on>
  <add_on_products config:type="list">
    <listentry>
      <!-- SLES12 SP5 for OES2018 SP2 Updates -->
      <media_url>http://<IP|DNS of your YUM server>/zenworks-yumrepo/
OES2018-SP2-SLES12-SP5-Updates-PRD</media_url>
      <product>OES2018-SP2-SLES12-SP5-UPDATES</product>
      <product_dir></product_dir>
      <name>OES2018 SP2 SLES12 SP5 Updates (PRD)</name>
      <alias>OES2018-SP2-SLES12-SP5-UPDATES-PRD</alias>
    </listentry>
    <listentry>
      <!-- OES2018 SP2 Updates -->
      <media_url>http://<IP|DNS of your YUM server>/zenworks-yumrepo/
OES2018-SP2-Updates-PRD</media_url>
      <product>OES2018-SP2-UPDATES</product>
      <product_dir></product_dir>
      <name>OES2018 SP2 Updates (PRD)</name>
      <alias>OES2018-SP2-UPDATES-PRD</alias>
    </listentry>
  </add_on_products>
</add-on>
:

```

Your modified control file should be stored in `/data/autoyast/upgrade` with a name that clearly identifies the version to which it will upgrade a system and where it will access the YUM repositories with the updates, for instance `autoupgrade-oes2018sp2_ZCM.xml`. The `autoyast=boot` option in your grub configuration file needs to point to this file along with the `autoupgrade=1` option ([“Installation Boot Medium” on page 21](#)).

Just like an installation an upgrade consists of two phases. When the system has performed its reboot the YaST console will open and request the password for the OES installation user. If the system being upgraded is a DSfW server, YaST will also request the password for the DSfW Administrator.

When this happens you can simply enter the password(s) and the upgrade will complete. However, having to do so in the middle of the upgrade somewhat contradicts the purpose of an automated process. This can be avoided by [Creating an Answer File to Provide the eDirectory and DSfW Passwords](#) in the OES2018 SP2 Installation Guide.

To create the answer file for an OES2018 server you simply issue the following command:

```
yast /usr/share/YaST2/clients/create-answer-file.rb <eDirectory password>
```

WARNING: As there is absolutely no validation of the passwords entered you need to make sure that the passwords are correct. It may be a good idea to create two answer files and to verify they are identical.

If the password(s) you entered should not be correct, the upgrade will fail and the server that you are upgrading may become unrecoverable.

Note that the password you enter will appear in `y2log`! This can be avoided by setting the passwords in the variables `OES_EDIR_DATA` and `OES_DSFV_DATA` before executing the YaST command to generate the answer file however, this will be logged in your bash history. Either way you will have to do some cleaning up to protect your password(s)!

The answer file is created in your current working directory and needs to be copied to the `/opt/novell/oes-install` directory of every server that will be upgraded. When this file is found at the beginning of stage 2 the required passwords will be retrieved from it and the upgrade will complete without administrative intervention. At the end of the upgrade the file is deleted automatically.

When the upgrade is completed successfully, the Add-On repositories for the earlier support pack have been replaced with those for the current support pack.

Figure 5-2 Zypper and ZCM Repositories After Upgrade to OES 2018 SP2

```
Repository priorities are without effect. All enabled repositories share the same priority.
# | Alias | Name | Enabled | GPG Check | Refresh
-----+-----+-----+-----+-----+-----
1 | OES2018-SP2-2018.2-0 | OES2018-SP2-2018.2-0 | Yes | ( r ) Yes | Yes
2 | OES2018-SP2-SLES12-SP5-UPDATE-PRD | OES2018 SP2 SLES12 SP5 Updates (PRD) | Yes | ( p ) Yes | Yes
3 | OES2018-SP2-UPDATE-PRD | OES2018 SP2 Updates (PRD) | Yes | ( p ) Yes | Yes

Processing Command: bl

Display Name | Version | Bundle Type | Status | Assigned
-----+-----+-----+-----+-----
_OES2018-SP1-SLES12-SP3-Updates-20200420 | 0 | Linux Bundle | Available | Device
_OES2018-SP1-Updates-20200420 | 0 | Linux Bundle | Available | Device
-----+-----+-----+-----+-----
Total Bundles Assigned: 2
```

However, ZCM still has the updates from the previous support pack assigned to the system. This needs to be corrected as described in [“Assigning Update Bundles After a Server Upgrade”](#) on page 110.

Miscellaneous

This section provides additional information that might be useful in specific situations.

Advanced Installation

Driver Updates

In rare cases, it might be necessary to provide driver update files (DUD) to the installation environment. These files contain binaries or configuration files that are needed at the beginning of the installation but are not part of the current `initrd`. They are mostly used to support very new hardware components or to provide a missing feature via a YaST module. DUDs are provided by the [SUSE SolidDriver Program](#).

A driver update can be specified as follows:

```
dud=<Protocol://path_to_dud_or_rpm>
```

A driver update can also be included in the info file described in [“Info Files” on page 59](#).

Boot Parameter `y2confirm`

Another way to troubleshoot installation issues is to inspect the installation proposal created by YaST before the installation is invoked by specifying the `y2confirm` option at the boot prompt.

When this option is set, YaST stops when the installation proposal is complete even with `<confirm>no</confirm>` in the control file, and will only continue when the proposal has been accepted by the administrator, just as you need to do in a manual installation.

This is particularly useful when you want to test a new server type or you have defined a new disk partitioning scheme that you want to validate.

Secure Download of AutoYaST Control Files via HTTPS

AutoYaST does support certificate-based authentication to protect the control file repository against unauthorized access via HTTP. For this purpose, a public key and private key generated by the CA that created the certificate used by the web server must be inserted into `/etc/ssl/clientcerts` on the `initrd` of the boot medium. The keys must be named `client-cert.pem` for the public key and `client-key.pem` for the private key.

For more information how to modify the `initrd` of the installation system please refer to [“How to modify or customize the installation `initrd`”](#).

SSH Based Installation

Any installation may get stuck or the initialization of a hardware component may fail due to hardware-related issues or a misconfiguration. In such cases, the ability to inspect the affected system can be very useful.

This can easily be achieved by adding the boot option `ssh=1`, which invokes the SSH daemon on the system being installed. If you do not specify a temporary root password by using the additional boot option `sshpassword=<TemporaryRootPassword>` the system will prompt for it.

Figure 6-1 Instructions for SSH-Based Installation

```
Loading Installation System (1/6) -      100%
Loading Installation System (2/6) -      100%
Loading Installation System (3/6) -      100%
Loading Installation System (4/6) -      100%
Loading Installation System (5/6) -      100%
Loading Installation System (6/6) - missing (optional)
starting rsyslogd (logging to /dev/tty4)... ok
starting klogd... ok
starting nscd... ok
setting temporary root password to 'C1F@1tsb3st'
Checking for missing server keys in /etc/ssh
ssh-keygen: generating new host keys: RSA1 RSA DSA ECDSA ED25519
Starting SSH daemon... ok
IP addresses:
  172.17.2.222

*** login using 'ssh -X root@172.17.2.222' ***
*** run 'yast.ssh' to start the installation ***

Active interfaces:
eth0: 00:0c:29:18:7c:c3
     172.17.2.222/24
     fe80::20c:29ff:fe18:7cc3/64
```

When the system is fully initialized, a message with instructions on how to log in to the server and how to invoke the installation appears on the screen.

To start the installation you need to establish a SSH connection with X-redirection enabled and issue the `yast.ssh` command.

Figure 6-2 Starting the Installation Over SSH

```
Open Enterprise 2018 SP1 Installation

Run yast.ssh to start the installation.

/usr/bin/xauth:  file /root/.Xauthority does not exist
0:install:~ #
```

The installation will occur in the graphical user interface and you can use additional SSH connections to the server to determine the root cause of your installation problems.

6 Troubleshooting and Monitoring

Installation Server

Access to installation repositories and control files via HTTP or HTTPS can be monitored in the Apache 2 log files. By default, these files are:

- ♦ `/var/log/apache2/access.log`
- ♦ `/var/log/apache2/error.log`

These log files can help to troubleshoot issues caused by files that do not exist or cannot be retrieved from the repository server.

Server Being Installed

The files you need to inspect to troubleshoot installation issues from the perspective of the system being installed are stored in different locations depending on where in the installation process the issue is encountered.

Installation Stage

The inspection of the server being installed can be done via a remote SSH session as described in [“SSH Based Installation” on page 70](#) above.

Configuration files and temporary files involved in the AutoYaST process are located in two different places:

- ♦ `/tmp/profile`

The installation process copies control files, configuration files, and the libraries of the Consulting Installation Framework to this directory. The file `variables.txt` is also created here.

- ♦ `/tmp/YaST-nnnnn-xxxxxx`

In the `/tmp` directory of the server being installed, there are at least three directories with a name starting with YaST followed by five random numbers and six random alphanumeric characters, such as `YaST-03200-AxfV0b`.

One of them contains a sub-directory `pre-script` that contains the script used in the pre-script stage of the installation. The same directory contains a log sub-directory that holds a log file for every `pre-script`.

These directories exist only during the installation stage. When the system is rebooted at the end of Stage 1 this information is lost.

All processes invoked by YaST are logged to `/var/log/YaST/y2log`. This file can be inspected during the installation stage as well as after the installation has finished.

If the option `loghost=<[IP-Address|DNS name] of syslog server>` is given at the boot prompt of the device being installed, all log files are redirected to the syslog server specified.

The log level can also be specified by using the `loglevel=[0-10]` boot parameter. Of course, the syslog server needs to be configured for remote logging.

Installed System

Information regarding aspects of the AutoYaST process can be found on a running system in the `/var/adm/autoinstall` directory. This directory contains the following sub-directories:

- ◆ `logs`

Contains a log file for every script that has been specified in the `<scripts>` section of the control file and that has been executed during the installation.
- ◆ `scripts`

Contains every script specified in the `<scripts>` section of the control file.
- ◆ `cache`

Contains the initial control file `pre-autoinst.xml` and the resulting control file `installedSystem.xml`. The later file contains the same information as `/root/autoinst.xml` on a system that has been installed manually.
- ◆ `files`

Contains any file specified in the `<files>` section of the control file.



Using ZENworks to Manage SLES or OES

Micro Focus Consulting Germany in cooperation with SUSE Consulting Germany has developed a methodology to configure and manage SUSE Linux Enterprise Servers (SLES) as well as Micro Focus Open Enterprise Servers (OES) using ZENworks Configuration Management (ZCM).

This section describes the guiding principles for the ZCM part of our solution. It first introduces the folder structures that we recommend you use in your ZCM zone to accommodate your server objects and your server group objects.

We then explain the registration keys used to put the server objects into the desired folder during device registration and how to use additional registration keys to establish server group memberships.

Next we cover the folder structures for bundle objects and bundle group objects and the naming scheme for Configuration Bundle Groups.

This is followed by a discussion of the Subscriptions used to replicate Pool channels and Update channels from the Customer Centers.

The creation of frozen patch levels, how to assign the resulting update bundles to your devices, and the use of YUM repositories as add-on repositories for updates during the installation are covered next.

We also explain how to use ZCM to deploy Field Test Files (FTF) and we discuss how the assignment of update bundles needs to be modified once a system has been upgraded to the next support pack.

The remainder of this section discusses some examples of configuration bundles used in Consulting projects and how they are assigned to devices.

The most frequently used agent commands and some debugging hints complete this section.

A scripts to create the recommended folder structure, the configuration bundles discussed in [“Managing Configuration Bundles and Configuration Bundle Groups” on page 111](#) as well as a script to make theses bundles members of the bundle groups used to assign them are available at the [CIF download site](#). Refer to [“Configuring Your Own ZENworks Management Zone” on page 137](#) to learn how to use these downloads to build your own ZCM zone. However, before you do so please carefully read this part of the guide to understand how ZCM is used as part of our solution.

7 ZENworks Configuration Management Introduction

ZENworks Configuration Management (ZCM) is the part of the ZENworks product line that provides centralized configuration management together with imaging, reporting, and remote management of servers and workstations running Linux or Windows. It can also be used to deploy updates to managed devices (not to be confused with ZENworks Patch Management, that is only available for SUSE Linux Enterprise Servers (SLES) but not for Micro Focus Open Enterprise (OES)).

As most customers are using ZCM to manage their Windows desktops Micro Focus Consulting has decided to also use ZCM to configure and manage SLES systems as well as OES systems. While developing the solution special attention has been given to the following questions that typically arise in every project:

- ◆ How to manage different patch levels (frozen patch level) across different staging areas such as test, development, and production?
- ◆ How to incorporate Field Test Files into deployment and management?
- ◆ How to efficiently manage a wide range of configuration settings across a large number of managed systems?

A ZENworks Management Zone (ZCM zone) is used to manage a set of devices and consists of one or more Primary Servers, optional Satellites, and managed devices.

Typically a single ZENworks Primary Server is sufficient to manage the SLES and OES systems in most customer environments and therefore the solution will be illustrated using a ZCM zone with a single ZCM 2020 server on SLES15 SP1. In case you should be using a ZENworks Management Zone with multiple Primary Servers we will discuss what needs to be considered in such a setup where applicable.

Whether or not ZENworks Satellite Servers are required depends on the WAN infrastructure and on what other systems the customer is managing with ZCM. These aspects are beyond the scope of this guide. Typically our solution integrates nicely in any well designed ZCM infrastructure.

When using ZCM to manage Linux devices some thought must be given to the question which structures to use to organize devices and bundles. [“Linux Bundles and Bundle Groups” on page 90](#) explains this in more detail and gives recommendations how to organize your ZCM zone.

Updates that are publicly available must be obtained from external sources such as the Micro Focus Customer Center or the SUSE Customer Center by means of subscriptions that result in update bundles being created in the ZENworks Management Zone. Field test files (FTF) provided by engineering to address a specific issue can also be incorporated in the solution.

Ultimately any bundle needs to be assigned to managed devices one way or another to deploy it to the devices.

Updates can either be distributed by a push process (distribution of updates predetermined by the ZCM server) or by a pull process (installation of updates initiated through the ZENworks Adaptive Agent (ZAA) from the system being updated). This is covered in more detail in [“Managing Pool Bundles, Update Bundles and Update Bundle Groups” on page 102](#).

Configuration bundles can be used to easily deploy configuration files to managed devices such as the `bindings` file that is used to assign user-friendly names to shared disk devices in a multi-path environment.

More sophisticated configuration bundles can be used to modify existing configuration files using the “Edit Text File” actions available in ZCM. One example would be to add the configuration settings to the `slp.conf` file that are required to turn a server into a SLP Directory Agent.

“Run Script” actions can be used to implement even the most advanced configuration operations in ZCM. [“Managing Configuration Bundles and Configuration Bundle Groups” on page 111](#) gives an overview of the most important configuration bundles that are used in literally every Consulting project and how they are deployed.

The bundles discussed in this section are available at the [CIF download site](#). This site also provides some input files for `zman`. Refer to [“How To Build Your Own Installation Framework” on page 129](#) to learn how to use these downloads to build and configure your ZCM zone.

8

Managing the ZENworks Management Zone

We assume that you either start with a new ZCM zone or that your ZCM zone has not been used to manage Linux devices. If this should not be the case you will need to adjust our suggestions to the structure of your ZCM zone.

When you start to manage your Linux devices in ZCM you will find that there are many different ways to perform an operation using the ZENworks Control Center (ZCC). For instance, to make a bundle a member of a bundle group you can browse to the bundle and modify its relationship to bundle groups. Another way to achieve the same outcome is to browse to the bundle group and modify its members. If you only want to assign a single bundle to a single bundle group you obviously can do it either way. However, if you have to assign a bundle to multiple bundle groups you will find it to be far more efficient to add the bundle to multiple bundle groups rather than modifying each bundle group individually.

Careful planning of your ZENworks Management Zone (ZCM zone) is crucial to build an environment that does provide easy administration of your Linux devices. In this section, we describe the approach that has been used successfully in many of our projects. This includes the recommended folder structures as well as a naming standard for the objects that you need to create to manage your Linux devices.

Building the ZCM portion of the installation framework in principal consists of the following steps:

- ♦ Build the folder structure for server and server group objects
- ♦ Build the folder structure for bundle and bundle group objects
- ♦ Create bundle groups and server groups
- ♦ Assign bundle groups to server groups
- ♦ Assign servers to server groups
- ♦ Assign bundles to bundle groups

Guiding Principles

When the ZENworks Adaptive Agent (ZAA) is installed on a server, the device is registered with the ZENworks Management Zone (ZCM zone) and a server object representing the new managed device is created in the **> Devices > Servers** folder by default. This server object is a member of the dynamic server group for the installed operating system.

The registration process can be modified by use of registration keys or registration rules. Our solution is based on registration keys. As one of the last steps of an AutoYaST deployment the ZAA is installed and the device is registered with the ZCM zone using the following registration keys (see [“The Post-Installation Script zcm-install.sh” on page 65](#)):

- ♦ **Location key:** The first key is always used to determine where in the ZCM zone folder structure to place the server object.
- ♦ **Configuration Group key:** This key determines the membership in **exactly one** server group that is assigned to exactly one bundle group that only holds **Configuration Bundles**.
- ♦ **Update Group key:** This key determines the membership in **exactly one** server group that is assigned to exactly one bundle group that only holds **Update Bundles**.

For configuration tasks we suggest to take the following approach:

- ♦ Every configuration task is implemented as a separate configuration bundle.
- ♦ Configuration bundles are members of as many Configuration Bundle Groups as needed.
- ♦ Configuration Bundle Groups are assigned to Configuration Server Groups. For the ease of management it is very important to strictly maintain this 1:1 relationship between Configuration Bundle Groups and Configuration Server Groups.

Updates are managed in the following way:

- ♦ Every update channel needed for a specific release and service pack (e. g. OES2018 SP2) is replicated from a customer center to the ZCM zone through a subscription (see [“Subscriptions” on page 94](#)).

This results in a Linux bundle being created in the folder defined in the subscription for that channel.

By default this bundle is named `<Release>-<SP>-Updates-bundle`, for example `OES2018-SP2-Updates-bundle` and we recommend to keep this name.

- ♦ This Linux bundle is copied to a new Linux bundle freezing the content of the channel at the time the copy is taken (see [“Creating a Frozen Patch Level” on page 102](#)).

The bundle with the Frozen Patch Level (FPL) is named `_<Release>-<SP>-Updates-
yyyymmdd`, where `yyyy` denotes the year, `mm` indicates the month and `dd` is the day at which the copy of the Linux bundle has been created.

The bundle name is prefixed with “_” to get the update bundles sorted to the bottom of the list of all bundles that are assigned to a device.

The resulting bundle will be named something like `_OES2018-SP2-Updates-20200324`.

Note that copying a bundle will not consume any additional disk space and FPLs therefore can easily be kept as long as they are needed.

- ♦ When the subscription for the same channel is executed again only the initial Linux bundle `<Release>-<SP>-Updates-bundle` will be modified. FPL bundles will remain untouched.
- ♦ Update bundles providing a FPL are made a member of Update Bundle Groups that in turn are assigned to Update Server Groups.

If the installation media is not permanently available to the installed systems the pool channel(s) for the installed software can be used for dependency resolution instead.

- ◆ Unlike update channels the pool channels only need to be replicated once as they provide the same patterns and packages as the original installation media or a subsequent support pack. Typically they should not change during the life cycle of the software product or its support pack.

However there are exceptions to the rule and whenever a new version of an installation media is released you must update the corresponding pool channel.

- ◆ As there is no FPL for a pool channel, these channels do not need to be frozen and are directly assigned to the Update Server Groups.

Through the assignments between bundle groups and server groups all required configuration bundles and updates bundles are automatically assigned to every new device.

Other methods of assignment, in particular assignments to server objects or folders or assignments of individual bundles instead of bundle groups are not used (with the exception of Pool Bundles).

zman

Most administrative tasks for a ZCM zone can be performed from the ZCC. However, you might want to perform certain operations like the creation of the required folder structure from the command line using zman.

When you execute a zman command, you will be asked for an username and the corresponding password. This can be very cumbersome when performing many actions with zman. For convenience, you can store the credentials in the file system by issuing the following command:

```
zman asc <ZCCUserID>
```

ZCCUserID is the user issuing the zman commands. This will create the encrypted credentials file (.zman/.zmaninfo) in the home directory of the logged-in user.

Password-free execution of zman instructions can be tested by executing the following command:

```
zman ql -s F
```

This command displays all failed actions from the main queue of the ZENworks Configuration Management server.

Top Level Folder Structure in the ZENworks Management Zone

When designing the folder structure for a ZCM zone there are many ways to organize servers, server groups, bundles, bundle groups and registration keys. However, three aspects proved to be of particular importance:

- ◆ A well defined folder structure that keeps the number of mouse clicks required to perform management tasks such as to make assignments between objects at a minimum.

- ◆ A naming standard that clearly indicates the purpose of every object.
- ◆ A concept to manage which administrators have which rights in which part of your ZCM zone. For details please refer to the [ZENworks Administrator Accounts and Rights Reference](#).

Assuming that you are not yet managing Linux devices with ZCM we very strongly recommend to separate Linux devices and bundles from devices and bundles for systems using other operating systems by creating the following folders as the first level of the folder structure for the management of Linux devices:

- ◆ > **Devices > Server > Linux**
- ◆ > **Bundles > Linux**

These folders are primarily introduced to allow the assignment of access rights that ensure that only qualified administrators will have the rights to modify Linux devices and bundles.

Servers and Server Groups

By default ZCM places all server objects in the Devices area, that is subdivided into the folders "Servers" and "Workstations".

This may be acceptable for small lab environments. However, for an efficient management of devices in a production management with a large number of servers it is vital to organize the server objects into folders.

Even so it is possible to make all bundle assignments directly to server objects it is much more efficient to use server groups. For ease of administration we recommend to place them in a separate folder structure.

Folder Structure for Servers and Server Groups

We recommend to organize the folder structure to accommodate server objects and server group objects representing Linux servers in three folders:

Figure 8-1 Device Folder Structure

Devices					
Status	Name	Type	Operating System	Last Contact	Retired
<input type="checkbox"/>	GROUPS (Details)	Server Folder			
<input type="checkbox"/>	NOKEY (Details)	Server Folder			
<input type="checkbox"/>	SERVERS (Details)	Server Folder			

1 - 3 of 3 items 1 / 1 show 25 items

The folders shown in this figure have the following purposes:

- ♦ **GROUPS** – Server group objects that are created to simplify the assignment of configuration bundles and updates bundles to devices are located in this branch of the folder structure.
- ♦ **NOKEY** – The purpose of the folder > **Devices** > **Servers** > **NOKEY** is to collect the server objects for devices that accidentally have been registered with a registration key intended for group assignment only.

If all systems have been registered using the correct location key this folder will always be empty!

- ♦ **SERVERS** – Any server object that is created as part of the registration process will be placed into a folder subordinate to this folder.

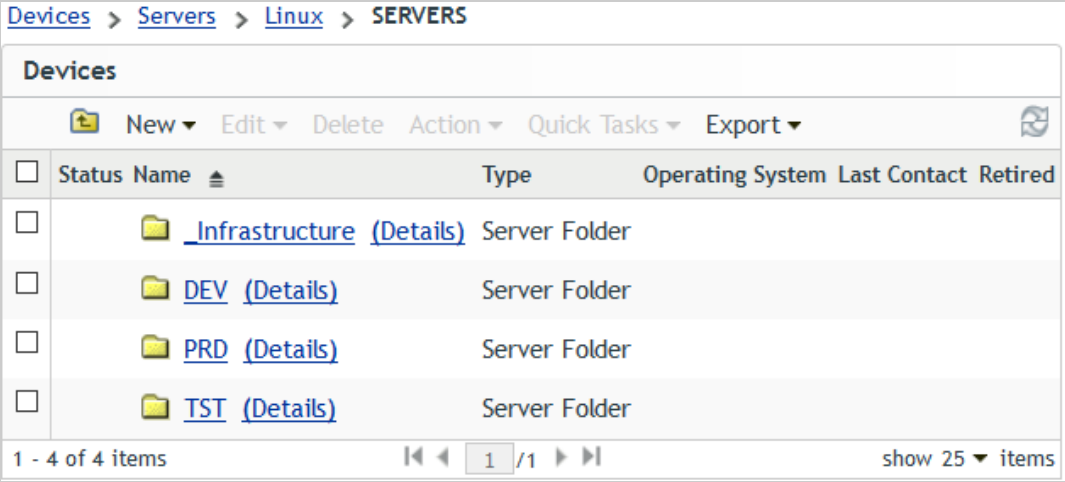
The Folder SERVERS

One important aspect of the folder structure for servers objects are the staging areas. Most customers besides their production environment are using a test environment. Some customers even maintain additional development systems.

Very typically these environments use different sets of updates and the first level of the folder structure for the server objects is designed to accommodate this requirement.

In this example the server objects for all systems that are part of the build environment are placed in the folder `_Infrastructure`. Typically this will be your AutoYaST server, any repository server and possibly your ZENworks Primary Servers.

Figure 8-2 The *SERVERS* Folder



The screenshot shows a web-based interface for managing devices. The breadcrumb path is `Devices > Servers > Linux > SERVERS`. Below the breadcrumb is a toolbar with options: `New`, `Edit`, `Delete`, `Action`, `Quick Tasks`, and `Export`. A table lists the contents of the `SERVERS` folder:

<input type="checkbox"/>	Status	Name	Type	Operating System	Last Contact	Retired
<input type="checkbox"/>		_Infrastructure (Details)	Server Folder			
<input type="checkbox"/>		DEV (Details)	Server Folder			
<input type="checkbox"/>		PRD (Details)	Server Folder			
<input type="checkbox"/>		TST (Details)	Server Folder			

At the bottom of the table, it shows `1 - 4 of 4 items`, navigation arrows, `1 / 1`, and `show 25 items`.

The other three folders are intended to accommodate the server objects representing systems from the development environment, from the test environment and from the production environment respectively.

Obviously you can adjust this level of the folder structure to meet your needs. Sometimes customers that do not (yet) have a test environment tend to skip this level of the folder structure. However, if they should later on discover that they need this level, re-organizing the folder structure of their

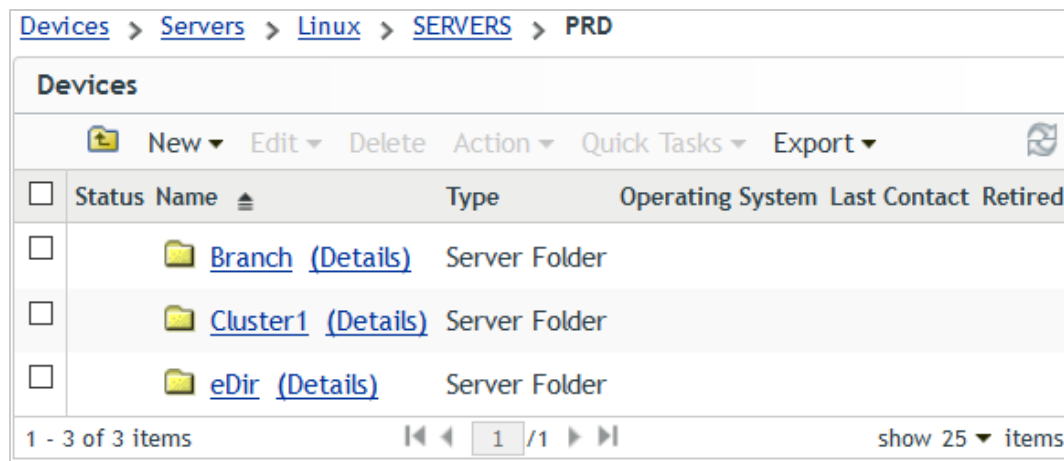
ZCM zone can be a cumbersome and error prone process. Therefore we highly recommend to implement this level of the folder hierarchy even if it (initially) should only consist of a single folder representing the production environment.

There are many ways to organize the next level of the folder structure. In smaller environments it is not unusual that this folder is not sub-divided any further and all server objects are simply put into the folder representing a staging area.

Some customers put the server objects into folders that represent the physical location of the systems.

Most commonly server objects are put into folders based on the services provided by a system.

Figure 8-3 Server Folders for the Production Environment



In the above example servers are organized in three folders based on the function of the servers:

- ◆ **Branch** – The server objects for all Branch Office servers are collected in this folder. This approach is typically used by customers that have a main office and a number of mostly small remote offices where a single server provides all the services needed by the users in the location.
- ◆ **Cluster 1** – Even so there is no strict technical requirement to do so many customers tend to place the server objects representing the nodes of a given NCS cluster into a separate folder.
- ◆ **eDir** – Another group of servers that often is put into a separate folder are all kinds of dedicated eDirectory servers. The first server typically found in this folder is the eDirectory Master / Certificate Authority host server of an eDirectory tree. Other servers placed in this folder are those that are configured to handle all user logins or to service LDAP request. iManager servers are also common in this folder.

Customers that only have a few but larger locations might create a folder for each location and then create the above folder structure for each location.

In very large environments that use different build environments for their staging areas it might even make sense to have a folder `_Infrastructure` for each staging area instead of the global folder depicted here.

The possibilities are endless and you need to decide which folder structure does make the most sense in your situation.

The Folder NOKEY

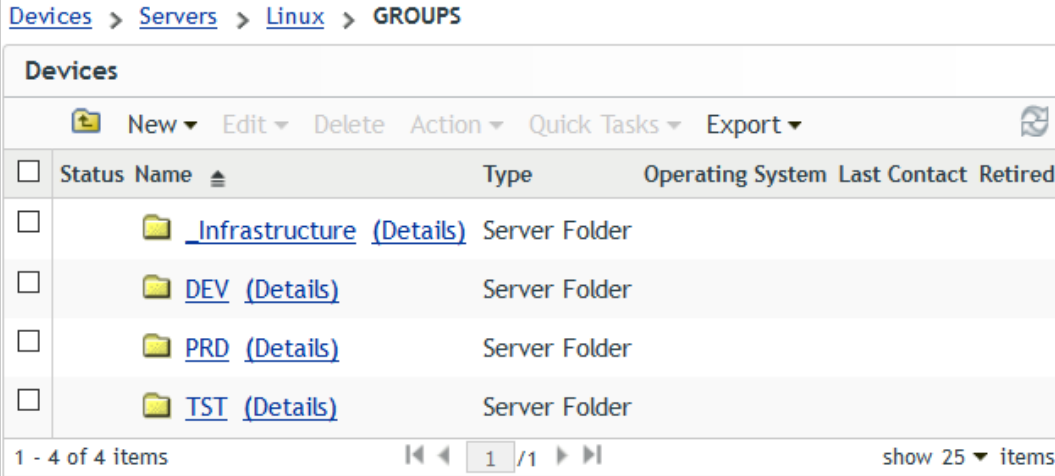
The purpose of this folder is to collect server objects for devices that have been registered with the zone using a registration key that is not a location key. In other words – if everything goes well you should never see a server object in this folder. If you do see a server object in this folder you know that something did not go as planned and you may want to take a closer look.

The Folder GROUPS

It is possible to assign bundles and bundle groups to folders so that the bundles become effective on every device who's server object ends up in the folder that has the bundles or bundle groups assigned.

However, as discussed in “The Folder SERVERS” on page 81 above there are many ways how customers may organize the server objects of their systems into folders. This variability is one of the reasons why we have decided to assign bundle groups to server groups and not to folders.

Figure 8-4 The GROUPS Folder



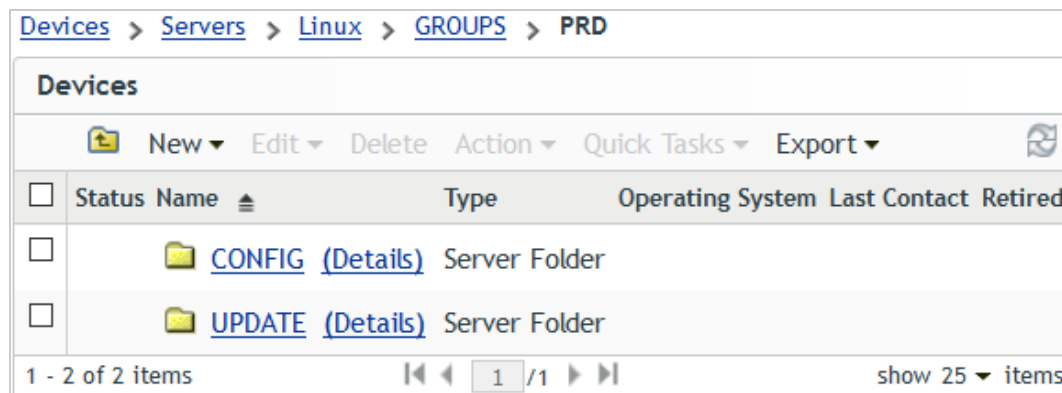
The screenshot shows a web-based interface for managing devices. The breadcrumb navigation at the top reads: [Devices](#) > [Servers](#) > [Linux](#) > **GROUPS**. Below the breadcrumb is a toolbar with buttons for New, Edit, Delete, Action, Quick Tasks, and Export. The main content area is a table with the following columns: Status, Name, Type, Operating System, Last Contact, and Retired. The table contains four rows, each representing a folder:

<input type="checkbox"/>	Status	Name	Type	Operating System	Last Contact	Retired
<input type="checkbox"/>		_Infrastructure (Details)	Server Folder			
<input type="checkbox"/>		DEV (Details)	Server Folder			
<input type="checkbox"/>		PRD (Details)	Server Folder			
<input type="checkbox"/>		TST (Details)	Server Folder			

At the bottom of the table, there is a pagination control showing "1 - 4 of 4 items" and a "show 25 items" dropdown.

Just like the folder SERVERS the folder GROUPS is sub-divided in one folder for each staging area and a folder for your infrastructure servers. Again we highly recommend that this layer of the folder structure be implemented even if there only should be a production environment, at the time when the initial folder structure is created.

Figure 8-5 Server Group Folders for the Production Environment



Each of the folders for server groups is divided into two folders – CONFIG and UPDATE. These folders contain the server group objects that will be used to assign configuration bundles and update bundles to the devices.

Server Groups

Server groups are objects which represent multiple server objects. By using this object type associations of tasks and software to devices can be handled more easily.

For instance, if a specific bundle has to be deployed to many servers this could be achieved by assigning the bundle to every server object individually. If the assignment would need to be deleted at a later time this again would require to touch every server object individually.

It is much more efficient to make the appropriate server objects a member of a server group and to assign the desired bundle to this group. This will deliver the bundle to all members of the server group and in turn if a bundle association is removed from the group it is removed from all members of the group automatically.

For the assignment of multiple bundles we recommend to make the bundles a member of a bundle group and assign this bundle group to the server group (see “[Bundle Groups](#)” on page 93 below).

We distinguish between the following two types of server groups:

- ◆ **Configuration Server Group:** Devices with identical configuration requirements are combined into **Configuration** Server Groups.

Application packages such as antivirus software or backup software as well as configuration bundles are assigned to these server groups through a corresponding bundle group.

- ◆ **Update Server Group:** Servers that are on the same product, version and support pack, such as all OES2018 SP2 servers, are combined into **Update** Server Groups.

The following bundles are assigned to these groups:

- ◆ Pool Bundles for dependency resolution
- ◆ Update Bundle Groups providing the frozen patch level.

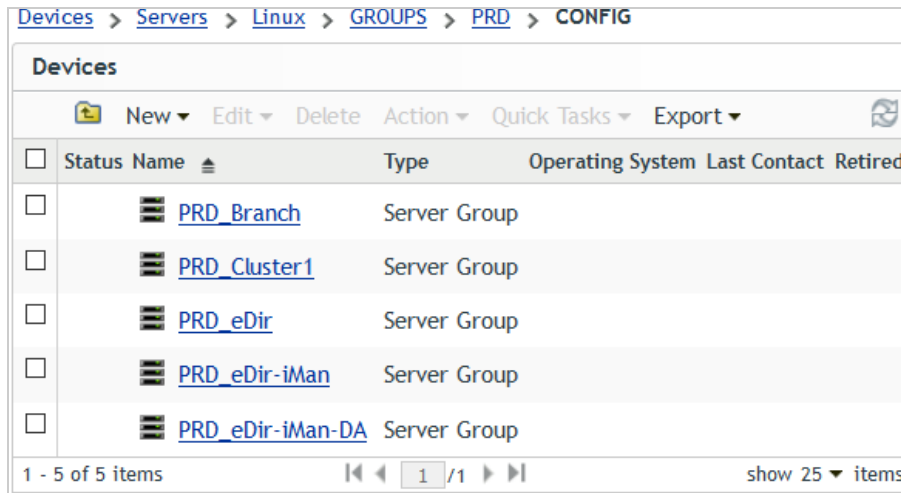
The following table provides an example for a naming scheme for Configuration Server Groups:

Table 8-1 Configuration Server Group naming scheme: %ENVIRONMENT%_%CONFIG_GROUP%

Element	Description
ENVIRONMENT	DEV TST PRD This part of a group name identifies to which environment the members of this group belong.
CONFIG_GROUP	Branch Cluster1 eDir eDir-iMan eDir-iMan-DA ... This part of the group name identifies the function of the members of the Configuration Server Group: Branch: Branch Office Servers (typically providing all services) Cluster1: Nodes of NCS cluster Cluster1 eDir: dedicated eDirectory/LDAP servers eDir-iMan: dedicated eDirectory/LDAP servers with iManager eDir-iMan-DA: dedicated eDirectory/LDAP servers with iManager and configured as SLP Directory Agent

The server groups in the CONFIG folder depicted below consist of server objects for systems from the production environment with the exact same configuration requirements.

Figure 8-6 Server Groups for the Assignment of Configuration Bundles



In this example we see server groups for Branch Office Servers, for the nodes of NCS cluster Cluster1, for dedicated eDirectory servers, for eDirectory servers that have iManager installed, and for eDirectory servers that have iManager installed and are configured as SLP Directory Agent. Bundle groups holding the configuration bundles required by these systems are assigned to the server groups in this folder.

As certain server types may not exist in all environments the corresponding CONFIG folders do not necessarily contain the same set of server groups. For instance a test environment may only require the server groups TST_Branch and TST_eDir-iMan-DA.

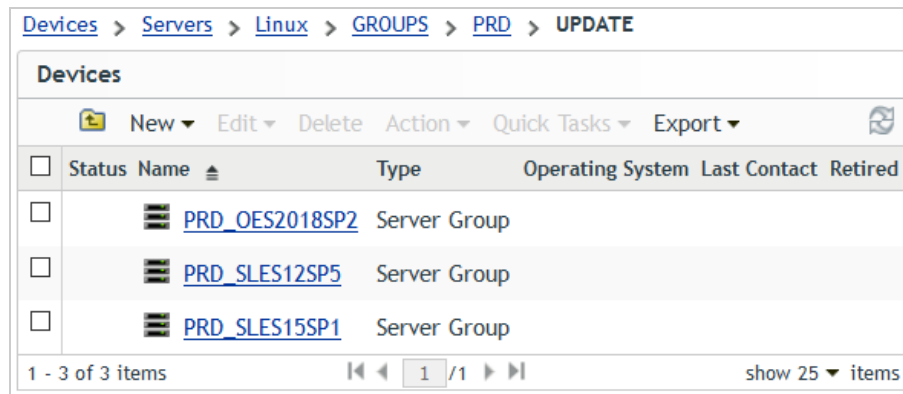
The following table defines the naming scheme used for update server groups.

Table 8-2 Update Server Group naming scheme: %ENVIRONMENT%_%PRODUCT%%VERSION%%SERVICE PACK%

Element	Description
ENVIRONMENT	DEV TST PRD This part of a server group name identifies to which environment the members of this group belong.
PRODUCT	SLES OES This part of the naming standard identifies which product is installed on its members.
VERSION	11 12 15 2015 2018 ... This part of a group name identifies which version of the product installed on its members
SERVICE PACK	[FCS GA SP0] SP1 SP2 SP3 ... The final part of a group name identifies the service pack level of the product installed on its members. To maintain the folder structure of the ZCM zone we recommend to use a generic term for the initial release of a product such as FCS (first customer shipment), GA (General Availability) or simply SP0.

The server groups in this UPDATE folder encompass server objects for systems from the production environment running the same release and support pack of SLES or OES.

Figure 8-7 Server Groups for the Assignment of Update Bundles



Bundle groups holding update bundles for these releases and support packs are assigned to the server groups in this folder.

The `UPDATE` folders for different environments only contain the server groups for the releases and support packs that are used in those environment.

Device Registration

The first step to integrate a device into a ZENworks Management Zone is its registration. By default the device object will be named with its hostname and the resulting server object will be placed in the folder > **Devices** > **Server**.

This process can be modified by use of registration keys. The first and mandatory purpose of a registration key is to create a device object during its first registration and to place it into a certain device folder.

Although server objects can be manually moved within the ZENworks Configuration Management folder structure, we recommend that you use registration keys to determine the final location during initial registration.

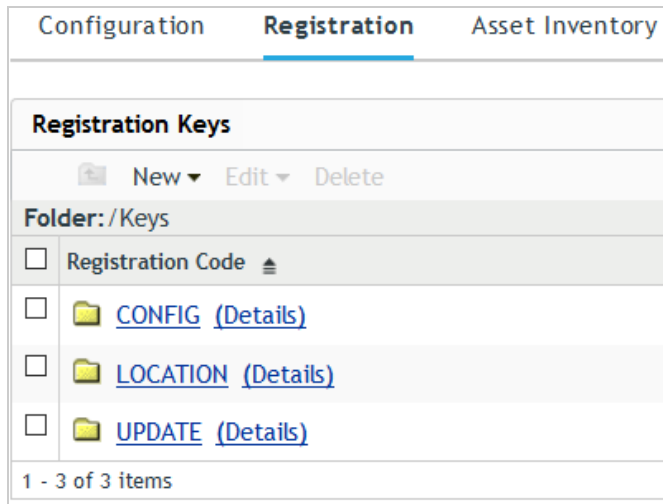
The second and optional purpose of a registration key is to add the server object to one or more server groups. To simplify administration we recommend that you use separate registration keys to determine the target folder of a device object and to add the server object to server groups. You only should add a server object to one server group with each registration key. This simplifies the administration.

When the ZENworks Adaptive Agent (ZAA) is installed as one of the last steps of the automated deployment the device is registered with the ZCM zone using the registration keys (see [“The Post-Installation Script `zcm-install.sh`” on page 65](#)):

- ♦ **Location Registration Key:** the first key is always used to determine in which folder of the ZCM zone folder structure to place the server object.
- ♦ **Group Registration Keys:** all subsequent registration keys are used to make the server objects a member of exactly one server group that either is assigned to a Configuration Bundle Group or to an Update Bundle Group.

These registration keys are created as described in [Creating a Registration Key](#) in the “ZENworks Discovery, Deployment, and Retirement Reference” and placed in three folders in the Registration Keys area of the ZCC.

Figure 8-8 Registration Key Folders



To avoid any confusion, you must be able to clearly identify the purpose for a registration key. Using the Micro Focus Consulting naming scheme will help you to accomplish this. The registration keys should be named and defined as follows:

Location Registration Keys: We recommend that you restrict the location keys to the initial registration of devices and do not use them to assign any server group memberships. Obviously, you will need one registration key for each folder where you want to place server objects.

Table 8-3 Location Registration Keys Definition

Location registration key naming scheme:	%ENVIRONMENT%_%SERVER_FOLDER%
Location registration key folder:	/Keys/LOCATION
Folder for server object:	/Devices/Servers/Linux/ %ENVIRONMENT%_%SERVER_FOLDER%
Group membership:	none

Element	Description
ENVIRONMENT	DEV TST PRD This part of the registration key name identifies to which environment the new device will belong.
SERVER_FOLDER	Branch Cluster1 eDir This part of the registration key name identifies in which folder to place the server object for the new device.

Server Group Registration Keys: We suggest that you use the same naming standards for update and configuration group registration keys as defined for the corresponding server groups in Configuration Server Group and Update Server Group.

This will result in an easy-to-understand relationship between registration keys and the server groups to which they add the server objects.

Table 8-4 Configuration Registration Keys Definition

Configuration registration key naming scheme:	%ENVIRONMENT%_%CONFIG_GROUP%_GRP
Configuration registration key folder:	/Keys/CONFIG
Folder for server object:	/Devices/Servers/Linux/NOKEY
Group membership:	/Devices/Servers/Linux/GROUPS/ %ENVIRONMENT%/CONFIG/ %ENVIRONMENT%_%CONFIG_GROUP%_GRP

Element	Description
ENVIRONMENT	DEV TST PRD This part of the registration key name identifies to which environment the new device will belong.
CONFIG_GROUP	Branch_GRP Cluster1_GRP eDir_GRP eDir-iMan_GRP eDir-iMan-DA_GRP ... This part of the registration key name identifies the Configuration Server Group to which the new device will be added.

NOTE: As registration keys need to be unique in a ZCM zone the Configuration registration keys are suffixed with “_GRP” to distinguish them from the Location registration key for the same devices.

Table 8-5 Update Registration Keys Definition

Update registration key naming scheme:	%ENVIRONMENT%_%PRODUCT%%VERSION% %SERVICE_PACK%
Update registration key folder:	/Keys/UPDATE
Folder for server object:	/Devices/Servers/Linux/NOKEY
Group membership:	/Devices/Servers/Linux/GROUPS/ %ENVIRONMENT%/UPDATE/ %ENVIRONMENT%_%PRODUCT%%VERSION% %SERVICE_PACK%

Element	Description
ENVIRONMENT	DEV TST PRD This part of the registration key name identifies to which environment the new device will belong.
PRODUCT	SLES OES This part of the naming standard identifies which product will be installed on the new device.
VERSION	11 12 15 2015 2018 ... This part of the naming standard identifies which version of the product will be installed on the new device.
SERVICE PACK	[FCS GA SP0] SP1 SP2 SP3 ... The final part of a registration key name identifies the service pack level of the product that will be installed on the new device. To maintain the folder structure of the ZCM zone we recommend to use a generic term for the initial release of a product such as FCS (first customer shipment), GA (General Availability) or SP0.

Linux Bundles and Bundle Groups

In this section we describe how we recommend to manage bundles and bundle groups to assign configuration bundles and update bundles to devices.

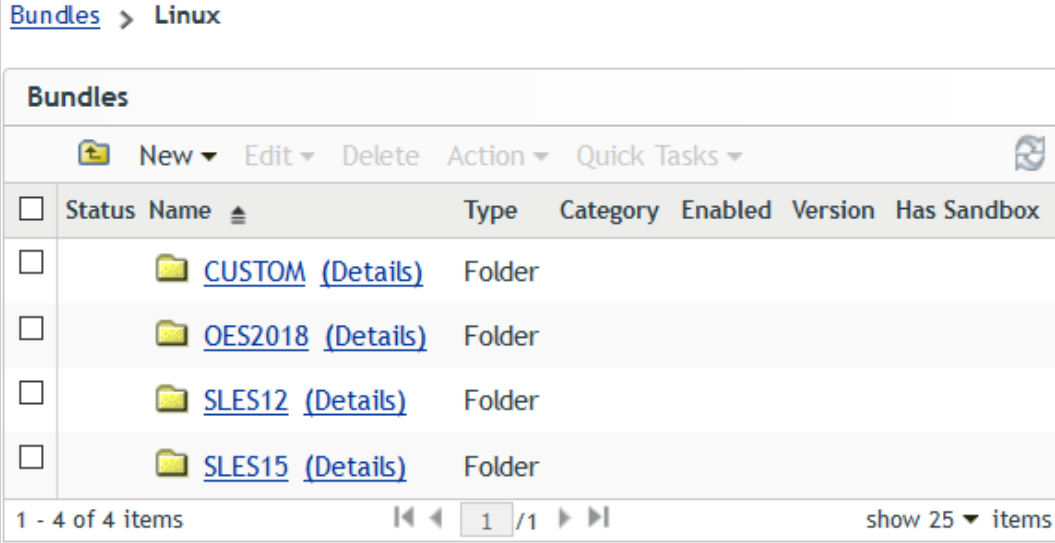
This includes folder structures to store the bundles and bundle groups as well as a standard for the names for these two object types. We will also explain the principles that have been used when developing configuration bundles.

For a complete description of the Linux package management in ZCM please refer to the [ZENworks Linux Package Management Reference](#).





Folder Structure for Bundles and Bundle Groups

At the first level of the folder structure configuration bundles and Configuration Bundle Groups are placed in the folder CUSTOM.

Figure 8-9 Folder Structure for Bundles and Bundle Groups



The screenshot shows a web interface for managing bundles. At the top, there is a breadcrumb path: [Bundles](#) > **Linux**. Below this is a section titled "Bundles" with a toolbar containing "New", "Edit", "Delete", "Action", and "Quick Tasks" buttons. A table lists the bundles, each with a checkbox, a folder icon, a name with a "(Details)" link, a type, and other attributes. The table has columns for Status, Name, Type, Category, Enabled, Version, and Has Sandbox. The listed items are CUSTOM, OES2018, SLES12, and SLES15, all of which are folders. At the bottom of the table, there is a pagination control showing "1 - 4 of 4 items" and a "show 25 items" dropdown.

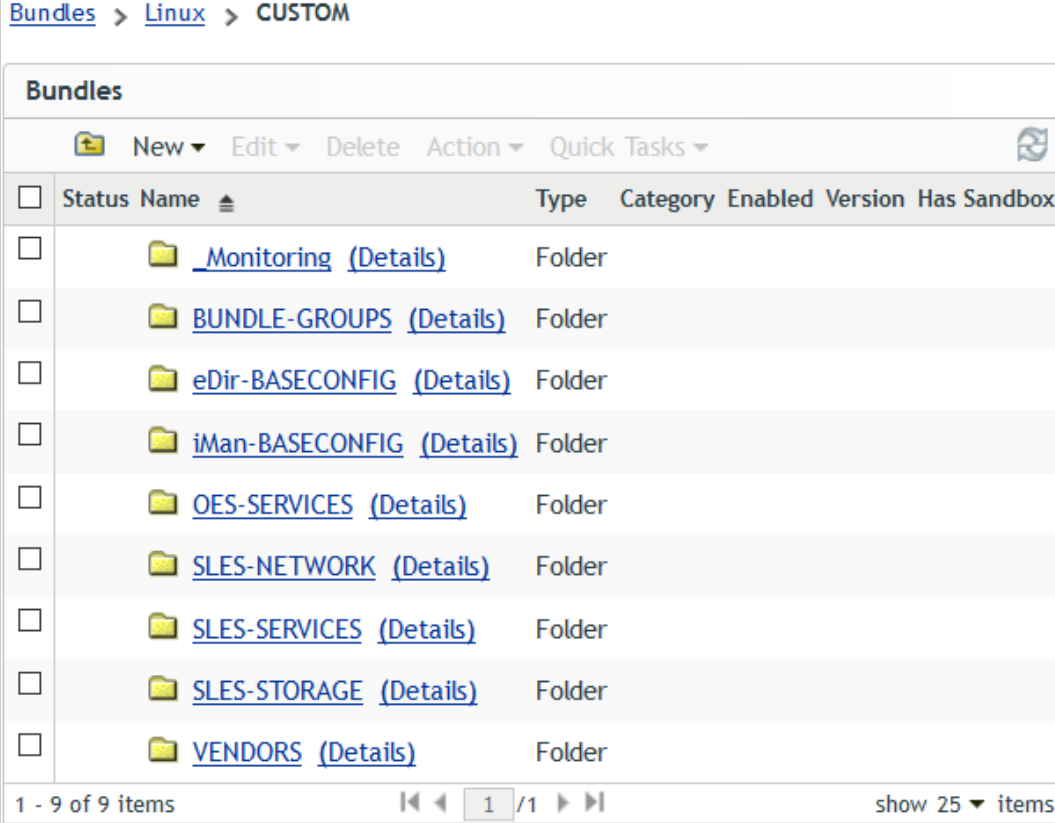
<input type="checkbox"/>	Status	Name	Type	Category	Enabled	Version	Has Sandbox
<input type="checkbox"/>		 CUSTOM (Details)	Folder				
<input type="checkbox"/>		 OES2018 (Details)	Folder				
<input type="checkbox"/>		 SLES12 (Details)	Folder				
<input type="checkbox"/>		 SLES15 (Details)	Folder				

For update bundles and update bundle groups we recommend to use one folder for each software product and version for example OES2018, SLES12, or SLES15.

Folder Structure for Configuration Bundles and Configuration Bundle Groups

The folder CUSTOM holds all configuration bundles and the bundle groups used to assign them to devices. As the number of bundles and bundle groups can become quite substantial additional folders are used to group these objects.

Figure 8-10 The Folder CUSTOM



The screenshot shows a web-based interface for managing configuration bundles. The breadcrumb path is Bundles > Linux > CUSTOM. The main area is titled 'Bundles' and contains a table with the following columns: Status, Name, Type, Category, Enabled, Version, and Has Sandbox. There are 9 items listed, all of which are folders. The folders are: _Monitoring (Details), BUNDLE-GROUPS (Details), eDir-BASECONFIG (Details), iMan-BASECONFIG (Details), OES-SERVICES (Details), SLES-NETWORK (Details), SLES-SERVICES (Details), SLES-STORAGE (Details), and VENDORS (Details). The interface includes a toolbar with 'New', 'Edit', 'Delete', 'Action', and 'Quick Tasks' options, and a pagination bar at the bottom showing '1 - 9 of 9 items' and 'show 25 items'.

Status	Name	Type	Category	Enabled	Version	Has Sandbox
<input type="checkbox"/>	_Monitoring (Details)	Folder				
<input type="checkbox"/>	BUNDLE-GROUPS (Details)	Folder				
<input type="checkbox"/>	eDir-BASECONFIG (Details)	Folder				
<input type="checkbox"/>	iMan-BASECONFIG (Details)	Folder				
<input type="checkbox"/>	OES-SERVICES (Details)	Folder				
<input type="checkbox"/>	SLES-NETWORK (Details)	Folder				
<input type="checkbox"/>	SLES-SERVICES (Details)	Folder				
<input type="checkbox"/>	SLES-STORAGE (Details)	Folder				
<input type="checkbox"/>	VENDORS (Details)	Folder				

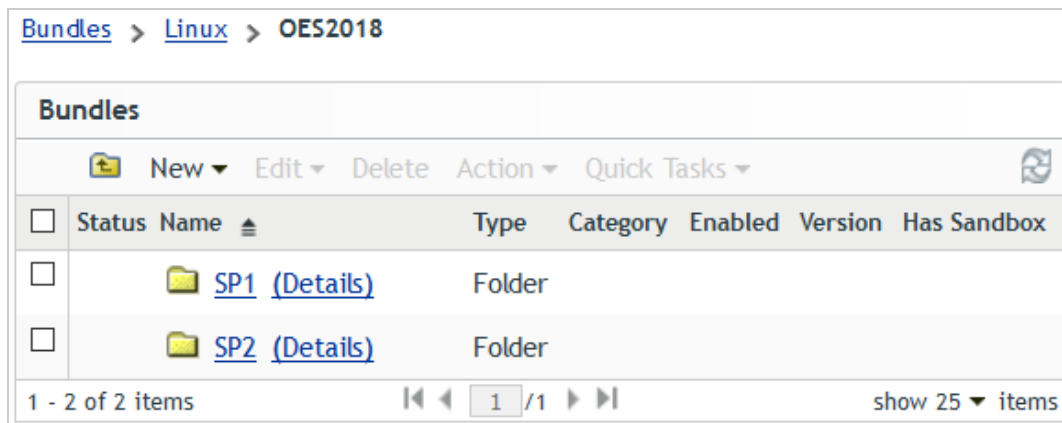
- ♦ **_Monitoring:** Any configuration bundle that is used to configure how your OES servers and your SLES servers interact with your monitoring solution should be placed into this folder. This includes bundles to configure Ganglia that is installed on every OES server by default.
- ♦ **BUNDLE-GROUPS:** Any bundle group used to assign configuration bundles to devices should be placed in this folder.
- ♦ **eDir-BASECONFIG:** This folder is intended to store bundles that configure eDirectory on OES or SLES.
- ♦ **iMan-BASECONFIG:** This folder is intended to store bundles that configure iManager on OES or SLES.
- ♦ **OES-SERVICES:** This folder accommodates bundles that apply configuration settings to OES servers such as a bundle to configure aliases for OES specific commands or bundles that configure a specific OES service like CIFS or FTP.
- ♦ **SLES-NETWORK:** Any bundle that configures any aspect of the network connectivity such as the configuration of a bond device or increasing the MTU for an iSCSI interface is stored in this folder.

- ♦ **SLES-SERVICES:** This folder accommodates bundles that apply general configuration settings on SLES servers or bundles that configure a specific SLES service such as SLP or SSH.
- ♦ **SLES STORAGE:** Any bundle that configures an aspect of the storage connectivity such as `multipath.conf` or the `bindings` files used to assign user friendly names to the shared device used by the nodes of a cluster come to mind..
- ♦ **VENDORS:** Bundles to distribute or configure third party solutions such as anti-virus software or backup software are typically placed in this folder.

Folder Structure for Update Bundles and Update Bundle Groups

The folders that accommodate update bundles and update bundle groups for a software product are split into separate folders for the initial release (FCS | GA | SP0) of the product and for each of its support packs.

Figure 8-11 Folder Structure for OES2018 Updates



These are the folders that are selected when defining the subscriptions for update channels or for pool channels (Figure 8-5). All the other folders that are needed to manage update bundles, pool bundles and Update Bundle Groups are automatically created when you execute the subscriptions.

Bundle Groups

Bundle groups are used to assign multiple bundles with a single operation. If a number of bundles has to be deployed to devices it is more efficient to make these bundles member of a bundle group and assign the bundle group instead of assigning each bundle individually.

We distinguish between the following two types of bundle groups:

- ♦ **Configuration Bundle Group:** These bundle groups combine bundles that manage configuration settings or add additional software such as anti-virus software or backup software to the devices.
- ♦ **Update Bundle Group:** These bundle groups are used to assign updates to all devices that are on the same product, version and support pack, such as all OES2018 SP2 servers.

Configuration Bundle Groups use the same naming standard as Configuration Server Groups.

Table 8-6 Configuration Bundle Groups Naming scheme: %ENVIRONMENT%_%CONFIG_GROUP%

Element	Description
ENVIRONMENT	DEV TST PRD This part of a group name identifies to which environment the members of this group belong.
CONFIG_GROUP	Branch Cluster1 eDir eDir-iMan eDir-iMan-DA ... This part of the group name identifies the function of the members of the Configuration Server Group: Branch: Branch Office Servers (typically providing all services) Cluster1: Nodes of NCS cluster Cluster1 eDir: dedicated eDirectory/LDAP servers eDir-iMan: dedicated eDirectory/LDAP servers with iManager eDir-iMan-DA: dedicated eDirectory/LDAP servers with iManager and configured as SLP Directory Agent

The name of the Update Bundle groups is derived from the updates they assign to the devices of a particular environment. This is discussed in detail in [“Assigning a Frozen Patch Level” on page 104](#).

Subscriptions

Subscriptions in ZENworks Configuration Management define how content is replicated from external repositories to create update bundles in the ZCM zone. From the many types of subscriptions available in ZCM only Novell Subscriptions and SUSE Subscriptions are discussed here.

Subscription are characterized by the following properties:

- ◆ The descriptive name of the subscription
- ◆ The folder where to place the subscription
- ◆ The folder where to place the replicated content
- ◆ An optional description
- ◆ The credentials to access the external repositories
- ◆ The catalog and targets to replicate
- ◆ The replication schedule
- ◆ The ZCM servers that executes the subscription

Once a subscriptions has been created some additional properties can be configured

- ◆ The local catalog name (not used by Micro Focus Consulting)
- ◆ The local name of the resulting bundles (only used for Pool bundles)

- ♦ Common options (Dry Run / Force Replication / Create Sandbox / Rollback on Failure)
- ♦ Kernel package install type options (Install / Upgrade)
- ♦ The bundle type (Static Replication / Linux Bundle / Linux Dependency Bundle)

The following sections describe the steps to create subscriptions in more detail.

Proxy Settings for Subscriptions

If you need to use a proxy server to access the external repositories, this cannot be configured in the ZCC. Instead the relevant information needs to be provided in the file `/etc/opt/novell/zenworks/lpm-server.properties`.

```
#####
## LPM Server Settings
#####
#
# Debug logging for Subscription Replication Module
Debug=false
# Time to Live(days) for Subscription WebCache metadata
TTL=24
#
#Subscription Proxy Settings
subscription-proxyaddress=<IP|DNS of proxy server>
subscription-proxyport=<port of proxy server>
subscription-proxyuser=
# Obfuscated password. Do not specify the raw password here directly.
# In stead use "zman srpp" command to specify the password in obfuscated
format
subscription-proxypassword=
subscription-useNTLM=false
```

Be aware that in a ZCM zone with multiple ZCM Primary servers this file needs to be configured on each server with internet access.

If you have very strict security requirements you may even be required to configured this file to use different proxies depending on which sites a particular ZCM Primary server is allowed to access.

Credential Vault

Novell update repositories or repositories from other vendors like SUSE or Red Hat are not freely accessible. A credential tied to some sort of maintenance contract is typically needed to gain access to such repositories. In the Micro Focus Customer Center you can find your mirror credentials under “Software”, the SUSE Customer Center lists them under “Proxies”.

Instead of entering the credentials every time a subscription is executed they can be stored in the ZCM zone as objects in the credential vault that then can be consumed by the subscriptions. We recommend to create a separate credential for each external repository source. The credential vault can be accessed in the ZCC by selecting > **Configuration** and then scrolling to the bottom of the page.

Figure 8-12 Credential Setup

Add Credential ? X

Credential Name: NCC *

Login Name: 123456 *

Password: ●●●●●●●

Reenter Password: ●●●●●●●

Description: Credentials to mirror channels from the Novell Customer Center. Contract expires December 31, 2027.

Fields marked with an asterisk are required.

OK Cancel

A credential has the following properties:

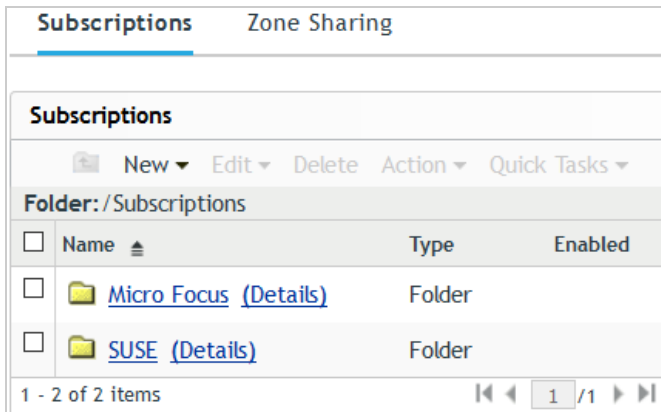
Credential Name:	A descriptive name
Login Name:	The credential login name
Password:	The credential password
Description:	A meaningful description to tell others what this credential is for. Should the credential have an expiration date it might be helpful to add this information to the description.

Folder Structure for Subscriptions

As you will integrate more and more operating system releases and their support packs into your build environment you will find that the number of subscriptions will increase just as quickly.

To make it more easy to keep an overview of your subscriptions we recommend that you create a separate folder for each repository provider.

Figure 8-13 Subscription Folders

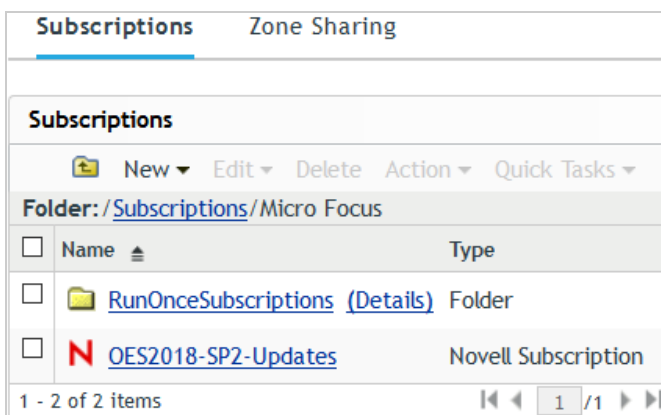


Subscriptions		Zone Sharing	
Subscriptions			
New Edit Delete Action Quick Tasks			
Folder: /Subscriptions			
<input type="checkbox"/>	Name	Type	Enabled
<input type="checkbox"/>	Micro Focus (Details)	Folder	
<input type="checkbox"/>	SUSE (Details)	Folder	
1 - 2 of 2 items		1 / 1	

Another aspect of the design is the fact that subscriptions for pool channels are typically only executed once while subscriptions for update channels will be executed regularly. Therefore we recommend that you create a sub-folder to accommodate subscriptions for pool channels for each repository provider.

This is illustrated for Micro Focus subscriptions in the following figure where the subscriptions for pool channels have been placed in the folder `RunOnceSubscriptions`.

Figure 8-14 Micro Focus Subscription



Subscriptions		Zone Sharing	
Subscriptions			
New Edit Delete Action Quick Tasks			
Folder: /Subscriptions/Micro Focus			
<input type="checkbox"/>	Name	Type	Enabled
<input type="checkbox"/>	RunOnceSubscriptions (Details)	Folder	
<input type="checkbox"/>	OES2018-SP2-Updates	Novell Subscription	
1 - 2 of 2 items		1 / 1	

Creating Subscriptions

Maintaining a software product typically will require you to replicate multiple channels from the software repository of the manufacturer to your ZCM zone. At the very minimum you will have to sync the update channel for your product and support pack level.

If you cannot or do not want to have the original installation sources permanently attached to the installed systems you may also need to replicate the corresponding pool channel.

However, if you have installed additional components there might be other channels that you need to include into your subscriptions. For instance, if you should be using any of the modules that are available with SLES12 and later, for instance the Scripting Module or the Containers Module you also need to include their catalogs into your subscription.

SUSE has introduced a separate Installer Updates channel with SLES12 SP3. If you want to use these updates with pure SLES systems they need to be part of the corresponding SLES updates subscriptions. Note that there is no pool channel for these Installer Updates and that **you may not use this channel with OES2018!!**

With SLE15 SUSE has introduced the Unified Installer concept. This is why your SLE15 subscriptions need to include the Product Update channel in addition to the Module Updates channels and the Installer Updates channel.

Earlier OES versions have been Add-On products and the updates for the underlying SLES operation system have been replicated from the SCC while the OES specific updates were retrieved from the NCC.

OES2018 is a product in its own right and all its approved channels are provided by the NCC. In addition to the OES2018 updates you also need to include the NCC channel with the updates for the SLES 12 operating system on which OES2018 is based. **You must not apply any packages from the SCC to any OES2018 system!**


There are also update channels for the various SLES 12 Modules that are available for OES2018 system. For instance, if you plan to use any docker based services such as the Cloud Integrated Storage (CIS) you will need to install and maintain the Containers module. Depending on the services you plan to use you may need to include additional modules. Note that not all modules that are available for SLES 12 are also available for OES2018.

In any case you create one subscription for all the pool channels and another subscription for all the update channels for each product and support pack level that you use in your environment.


When you create a new subscription you first have to select what type of subscriptions (Novell / SUSE) you want to create. In the next step you give the subscription a descriptive name and select the folder in which the bundles replicated by the subscription will be placed (see [“Folder Structure for Update Bundles and Update Bundle Groups” on page 93](#)).


Figure 8-15 Subscription for OES2018 SP2 Updates (1)

Create New Subscription

 **Step 2: Define Details**
Enter the details for the Subscription.

Subscription Name: *

Folder: *
 

Download To Folder: *
 

Description:

In the following example the download filter for the `sles-12-x86_64` target includes the updates for OES2018 SP2, the underlying SLES12 SP5 operating system and the corresponding Container Module required for dockerized containers.

Figure 8-16 Subscriptions for OES2018 SP2 Updates (2)

N OES2018-SP2-Updates

Summary
Download Filter
Settings
Audit

Catalogs ^

Get from the repository

Add Remove

	Catalog Name	Targets
<input type="checkbox"/>	OES2018-SP2-Updates	sles-12-x86_64
<input type="checkbox"/>	OES2018-SP2-SLES12-SP5-Updates	sles-12-x86_64
<input type="checkbox"/>	OES2018-SP2-SLE-Module-Containers12-Updates	sles-12-x86_64

1 - 3 of 3 items 1 / 1 show 10 items

Finally the subscription is configured to create a Monolithic Bundle containing all packages and the `Create Sandbox` flag is removed.

Figure 8-17 Subscription for OES2018 SP2 Updates (3)

Options

Common options

- Dry Run
- Force Replication
- Create Sandbox
- RollBack Installation of Bundles on Failure

Kernel Package Install type options

- Install
- Upgrade

Bundle options

- Static Replication
 - Monolithic Bundle
 - Source RPMs
 - Patches
- Create Linux Bundle
 - Monolithic Bundle
 - Source RPMs
 - Patches
 - Create Category based Bundle Groups
- Create Linux Dependency Bundle
 - Publish Packages

We strongly recommend that you keep a 1:1 relationship between the subscription for the update channels and the corresponding subscription for the pool channels. Subscriptions to replicate the pool channels are created similar to the subscription for the update channels with three differences:

- ◆ The subscription will replicate the catalogs for the pool channels.
- ◆ The subscription will create a Linux Dependency Bundle instead of a Linux Bundle.
- ◆ The local name of each bundle will be set to `_%PRODUCT%%VERSION%-SP%SUPPORTPACK%-Pool`, in other words the bundle name is prefixed with “_” and “_bundle” is removed from the original bundle name, i. e. `_OES2018-SP2-Pool`.

Figure 8-18 Modified Local Bundle Name

N OES2018-SP2-Pool			
Summary	Download Filter	Settings	Audit
General			
Catalog Name:	OES2018-SP2-Pool		
LocalCatalog Name:	(Edit)	OES2018-SP2-Pool	
Folder:	(Edit)		
LocalBundle Name:	(Edit)	_OES2018-SP2-Pool	

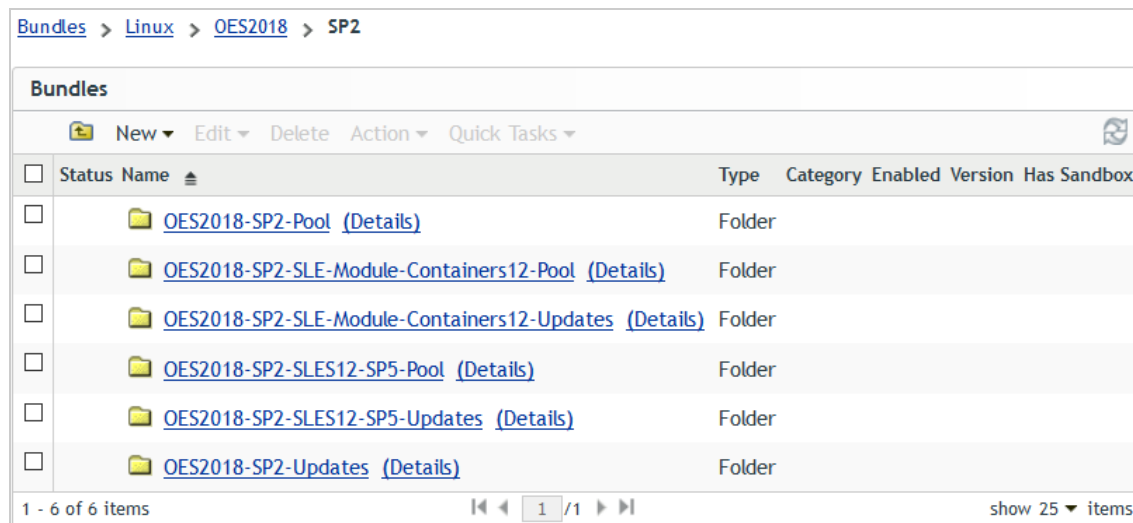
While a pool subscription typically only is executed once, subscriptions for update channels need to be repeated regularly.

As OES updates typically are only released once every three month it is sufficient to execute these subscriptions monthly to catch any interim patches and security updates.

For SUSE updates it may make sense to execute the subscriptions in shorter intervals as SUSE releases these updates more frequently.

When subscriptions are executed for the first time a folder is automatically created for each catalog being replicated.

Figure 8-19 Catalog Folders for OES2018 SP2

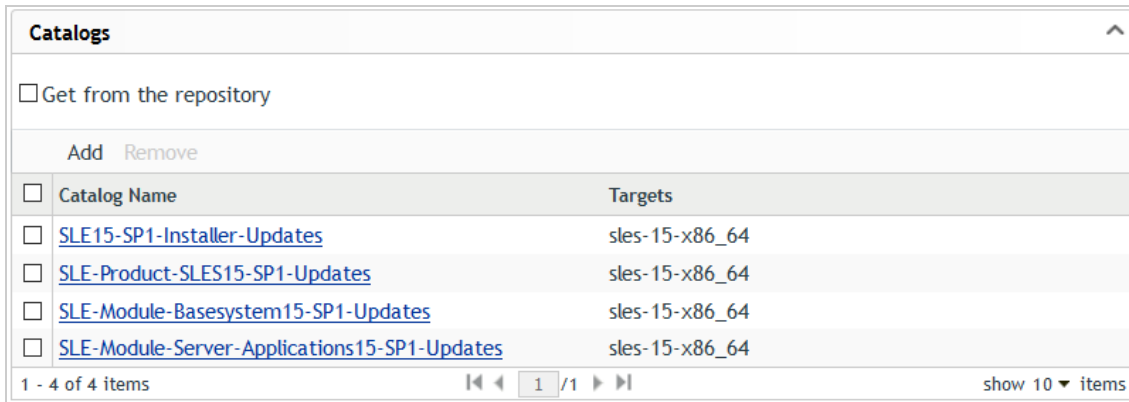


The above example depicts the folders that are created by the two subscriptions for the update channels and the pool channels for OES2018 SP2 defined in the preceding steps.

Setting up subscriptions for SLES15 is a little bit more involved as the subscriptions will need to include the pools and updates for the product as well as pools and the updates for all modules that you are using on your SLES15 SP1 servers.

In this example the subscription only contains the modules Basesystem and Server-Applications that are both selected for installation by default. Depending on the modules that you are using you may need to add more update catalogs to your subscription.

Figure 8-20 Subscription for SLES15 SP1 Updates - Catalogs



The screenshot shows a web interface titled "Catalogs" with a search bar and a list of catalogs. The list has columns for "Catalog Name" and "Targets". There are four entries, each with a checkbox on the left. The bottom of the interface shows pagination: "1 - 4 of 4 items", "1 / 1", and "show 10 items".

<input type="checkbox"/>	Catalog Name	Targets
<input type="checkbox"/>	SLE15-SP1-Installer-Updates	sles-15-x86_64
<input type="checkbox"/>	SLE-Product-SLES15-SP1-Updates	sles-15-x86_64
<input type="checkbox"/>	SLE-Module-Basesystem15-SP1-Updates	sles-15-x86_64
<input type="checkbox"/>	SLE-Module-Server-Applications15-SP1-Updates	sles-15-x86_64

Initially only the subscription server will receive the content from the external provider. If your ZCM zone should have multiple ZCM Primary Servers the other servers will only receive the new content when the next Primary Server recurring content replication cycle is executed.

For more details on subscriptions please refer to [Subscriptions](#) in the “ZENworks Linux Package Management Reference”.

Managing Pool Bundles, Update Bundles and Update Bundle Groups

When subscriptions are executed they populate the bundle folder structure. This section describes how to use the bundles created by subscriptions.

Pool Bundles

When installing a package on a Linux device there is always the possibility that the package you try to install is dependent on other packages that also need to be installed. This process is known as dependency resolution.

To resolve a dependency the Linux device needs access to the packages it needs to install to resolve this dependency. If your systems have been installed from physical media or from an ISO image this source might no longer be available when you later attempt to install additional packages.

One way to avoid this problem is to assign the pool bundles for your software to the devices.

Creating a Frozen Patch Level

A subscription ensures that local repositories and remote repositories are kept in sync. Every time a subscription is executed it updates the local bundle with any changes it finds on the remote repository.

When you have to re-install a system, for whatever reason you typically will want the system to be in the exact same state it was in when it was lost. This includes its configuration as well as the updates. To be able to meet this requirement patch levels must be provided unchanged over long periods of time.

Furthermore applications that depend on specific kernel versions and kernel modules may require a patch level which differs from patch levels needed by systems providing other applications. We term bundles that contain all the updates that have been available in the source repository at a given date Frozen Patch Level (FPL).

In ZCM the way to “freeze” a patch level is to simply copy the bundle created by the subscription. The name of the copied bundle is derived by replacing “bundle” in the name of the bundle maintained by the subscription with the date the copy is created as `yyyymmdd`, where `yyyy` denotes the year, `mm` indicates the month and `dd` is the day.

Figure 8-21 Frozen Patch level for OES2018 SP2 SLES12 SP5 Updates

Bundles						
Status	Name	Type	Category	Enabled	Version	Has Sandbox
<input type="checkbox"/>	_OES2018-SP2-SLES12-SP5-Updates-20200324	Linux Bundle		Yes	0	No
<input type="checkbox"/>	OES2018-SP2-SLES12-SP5-Updates-bundle	Linux Bundle		Yes	0	No

Finally, the name of the bundle with the FPL is prefixed with “_”. This is done to get the update bundles sorted to the bottom of the list of all bundles that are assigned to a device.


Copying a bundle will not consume any additional disk space as it only adds additional pointers in the ZCM database.

Assigning a Frozen Patch Level

The first step in assigning a Frozen Patch Level bundle to a device is to make it a member of an Update Bundle Group. The name of this bundle group is obtained by adding the identifier of the environment to which the target devices belong to the name of the bundle belonging to the subscription.

Figure 8-22 Assignment of a Frozen Patch Level to an Upgrade Bundle Group

[Bundles](#) > [Linux](#) > [OES2018](#) > [SP2](#) > [OES2018-SP2-SLES12-SP5-Updates](#) > OES2018-SP2-SLES12-SP5-Updates-PRD

 OES2018-SP2-SLES12-SP5-Updates-PRD

[Summary](#) [Share](#) [Audit](#)

General ^


Object type: Bundle Group


GUID: fde952078d93b07fd77dcc5664d0c9c0

Description: [\(Edit\)](#)

Members ^

[Add](#) [Remove](#) [Move Up](#) [Move Down](#) [Copy Members](#) [Move Members](#)

<input type="checkbox"/>	Name	In Folder
<input type="checkbox"/>	 _OES2018-SP2-SLES12-SP5-Updates-20200324	/~bundles~/Linux/OES2018/SP2/OES2018-SP2-SLES12-SP5-Updates

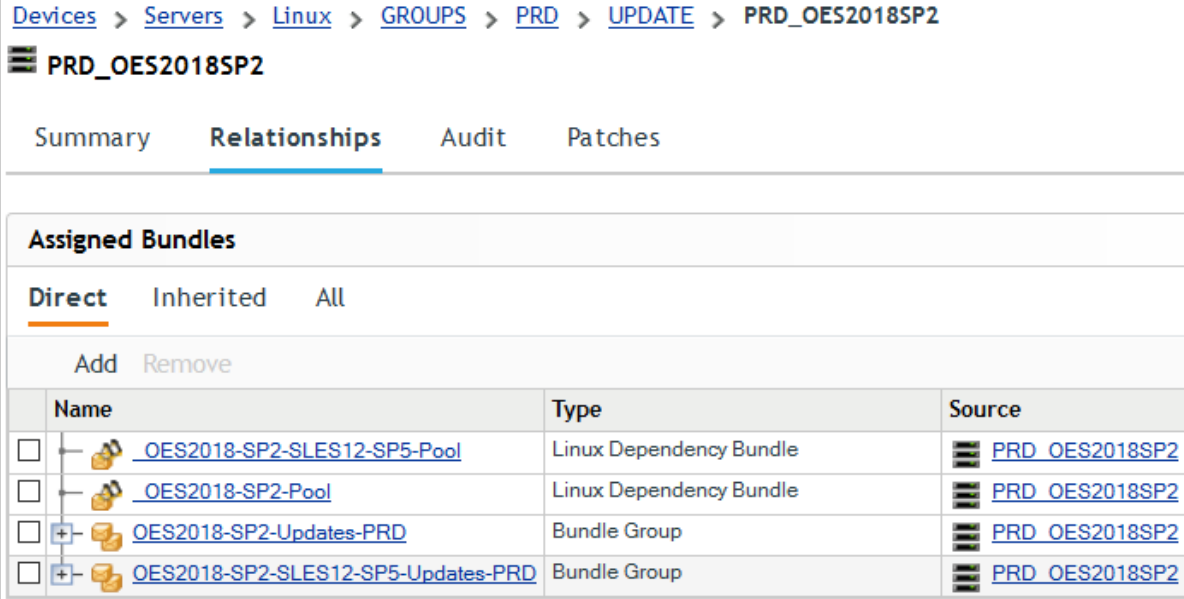
1 - 1 of 1 items  1 / 1 [show 25](#) items

In the above example the SLES12 SP5 updates for OES2018 SP2 have been frozen in the bundle `_OES2018-SP2-SLES12-SP5-Updates-20200324` and have been added to the Update Bundle Group `OES2018-SP2-SLES12-SP5-Updates-PRD`.









Assigning Pool Bundles and Update Bundle Groups

As the second step the Update Bundle groups need to be assigned to the target devices. We recommend that you make all bundle assignments for Linux devices to server groups instead of individual server objects or to folders holding server objects. This will greatly simplify bundle assignments.

Figure 8-23 Assignment of Pool Bundles and Update Bundle Groups



The screenshot shows the management console for the server group **PRD_OES2018SP2**. The breadcrumb navigation is **Devices > Servers > Linux > GROUPS > PRD > UPDATE > PRD_OES2018SP2**. The **Relationships** tab is selected, showing a list of assigned bundles. The table below details these assignments:

	Name	Type	Source
<input type="checkbox"/>	 OES2018-SP2-SLES12-SP5-Pool	Linux Dependency Bundle	 PRD_OES2018SP2
<input type="checkbox"/>	 OES2018-SP2-Pool	Linux Dependency Bundle	 PRD_OES2018SP2
<input type="checkbox"/>	 OES2018-SP2-Updates-PRD	Bundle Group	 PRD_OES2018SP2
<input type="checkbox"/>	 OES2018-SP2-SLES12-SP5-Updates-PRD	Bundle Group	 PRD_OES2018SP2

This example illustrates the assignment of Pool Bundles and Update Bundle groups to the server group for OES2018 SP2 devices in production.

As Pool bundles required for dependency resolution are expected to not change they are directly assigned to the server group **PRD_OES2018SP2**. Update bundles on the other hand will change regularly and are therefore assigned through bundle groups to the same server group.

There are various options how you can configure when the contents of the bundles is distributed to the managed devices. The distribution schedule can be set for specific times or it can be set to occur with the next device refresh. The setting **Install Immediately after Distribution** determines automatic installation of the bundle content. If you enable this option keep in mind that many updates require a reboot to become effective.

These assignments are made when you introduce a new product or a new support pack for a product to your ZCM zone. Once the assignment is made it will not be changed as long as that product or support pack is serviced by ZCM.

After a device refresh executing the command `zac bl` will list the bundles assigned to the device.

Figure 8-24 Bundles Assigned to a Managed Device

Display Name	Version	Bundle Type	Status	Assigned
_OES2018-SP2-Pool	0	Linux Dependency Bundle	Available	Device
_OES2018-SP2-SLES12-SP5-Pool	0	Linux Dependency Bundle	Available	Device
_OES2018-SP2-SLES12-SP5-Updates-20200324	0	Linux Bundle	Downloaded	Device
_OES2018-SP2-Updates-20200324	0	Linux Bundle	Downloaded	Device

Note that the two pool bundles are shown with a status of Available. This is expected, because Linux Dependencies Bundles are never installed, they are only used to resolve dependencies.

The two update bundles on the other hand are shown as Downloaded. This indicates that the content of the assigned bundles has been downloaded to the ZENworks cache but has not yet been installed. This is the expected behavior if the option `Install Immediately` after `Distribution` has not been enabled.

To install update bundles use the `zac bin` command:

Figure 8-25 Installation of Update Bundles with `zac bin`

```
Processing Command: bin
Starting Distribution of _OES2018-SP2-SLES12-SP5-Updates-20200324
Finished Distributing _OES2018-SP2-SLES12-SP5-Updates-20200324
Starting Install of _OES2018-SP2-SLES12-SP5-Updates-20200324
Finished Installing _OES2018-SP2-SLES12-SP5-Updates-20200324
Starting Distribution of _OES2018-SP2-Updates-20200324
Finished Distributing _OES2018-SP2-Updates-20200324
Starting Install of _OES2018-SP2-Updates-20200324
Finished Installing _OES2018-SP2-Updates-20200324
Successfully executed bin command
```

If you do this on a device that has been installed with the same Update Bundles configured as Add-On repositories executing `zac bin` will just update the status of the ZENworks Update Bundles to Available. No packages will be installed because they have already been installed during the installation of the system.

If the update bundles contain newer packages than what is already installed on the managed device, i.e. if you are deploying a new Frozen Patch Level, executing `zac bin` will update all packages installed on the managed device to the latest available version.

Figure 8-26 Bundles Installed on a Managed Device

Display Name	Version	Bundle Type	Status	Assigned
OES2018-SP2-Pool	0	Linux Dependency Bundle	Available	Device
OES2018-SP2-SLES12-SP5-Pool	0	Linux Dependency Bundle	Available	Device
OES2018-SP2-SLES12-SP5-Updates-20200324	0	Linux Bundle	Available	Device
OES2018-SP2-Updates-20200324	0	Linux Bundle	Available	Device

When the ZAA has verified that all available and required packages have been installed the status of the update bundles will change to Available.

IMPORTANT: Due to the changes in the way package dependencies can be defined in RPM 4.13 and later and due to the fact that the ZENworks Adaptive Agent does not understand these [Boolean Dependencies](#) `zac bin` and `zac up` can no longer be used to deploy updates to SLES and OES systems.

We therefore no longer recommend to assign Update Bundle Groups to server groups. Instead use the `zypper patch` command to deploy the updates from the YUM repos of the Update Bundle Groups as described in the next section.

We recommend that you delete any Update Device Group. Also delete the registration keys that make devices a member of an Update Device Group and remove these registration keys from Field 09 of every entry in your `server.txt` file.

YUM Repositories Derived From Frozen Patch Level Bundles and Pool Bundles

Frozen Patch Levels and Pool bundles created as described above need the ZCM Adaptive Agent on the managed device. If the ZAA cannot be installed or if the FPL is to be used as Add-On repository in the initial installation ZCM bundles are not suitable due to the format of the ZCM bundles.

However, ZENworks Configuration Management has the ability to create a YUM (Yellowdog Updater Modified) repository from ZENworks Linux bundles and ZENworks Linux Dependency bundles. These repositories can be consumed by `zypper`, the main package management tool of SLES. This allows you to update devices that do not have the ZAA installed.

IMPORTANT: Due to changes in the way dependencies between packages can be defined (see Boolean Dependencies for details) you must have **Official FTF 959 for ZENworks 2020 Update 3** applied to your ZCM Primary servers before creating any YUM repositories for SLES5 SP4 or OES2023.

In addition, YUM repositories can be consumed during installations to integrate the updates from the FPL. This reduces administrative overhead as devices installed in this way do not need to be updated until the next patching cycle.

As a Pool bundle hardly ever changes there is no need to freeze it and the YUM service for it can be created on the bundle itself. We recommend that you edit the YUM Service Name and remove the leading “_”.

Figure 8-27 YUM Repository for the OES2018 SP2 SLES12 SP5 Pool

Bundle YUM Service

Step 1: Create YUM Service

Service Name:*

Auto update the repository when bundle is published

Primary Servers

Add Remove

<input type="checkbox"/>	Name	In Folder
<input type="checkbox"/>	zcm2020-ref	/Devices/Servers

1 - 1 of 1 items show 5 ▾ items

The YUM service for update bundles is created on the bundle group, not on the bundle itself.

Figure 8-28 YUM Repository for OES2018 SP2 SLES12 SP5 Updates

Bundle YUM Service

Step 1: Create YUM Service

Service Name:*

Auto update the repository when bundle is published

Primary Servers

Add Remove

<input type="checkbox"/>	Name	In Folder
<input type="checkbox"/>	zcm2020-ref	/Devices/Servers

1 - 1 of 1 items show 5 ▾ items

This is done to preserve the URL of the YUM service used by the managed devices. When a new FPL becomes available it is simply added to the Update Bundle Group. The bundle for the old FPL needs to be removed from the bundle group and the YUM service must be refreshed to make the new updates available to the devices where they can be installed using `zypper patch`.

If your ZCM zone should have multiple ZCM Primary Server each of them can be configured as YUM repository server. You can configure multiple YUM servers for fault tolerance and load sharing. However, a YUM repository can only be created or updated on a ZCM Primary Server after it has received the corresponding content from the subscription server.

These repositories can be consumed by package management tools such as `rpm` or `zypper`. While the ZAA always displays the name of the individual bundles (see [Figure 8-26](#)) `zypper` shows the YUM service names and aliases.

Figure 8-29 YUM Repositories for OES2018 SP2 Pools and Updates

#	Alias	Name	Enabled	GPG Check	Refresh
1	OES2018-SP2-2018.2-0	OES2018-SP2-2018.2-0	Yes	(r) Yes	No
2	OES2018-SP2-Pool	OES2018-SP2-Pool	Yes	(p) Yes	Yes
3	OES2018-SP2-SLES12-SP5-Pool	OES2018-SP2-SLES12-SP5-Pool	Yes	(p) Yes	Yes
4	OES2018-SP2-SLES12-SP5-Updates-PRD	OES2018-SP2-SLES12-SP5-Updates-PRD	Yes	(p) Yes	Yes
5	OES2018-SP2-Updates-PRD	OES2018-SP2-Updates-PRD	Yes	(p) Yes	Yes

As the YUM service names for update bundles are based on bundle groups it is not possible to see from the repository listing when the last FPL has been created.

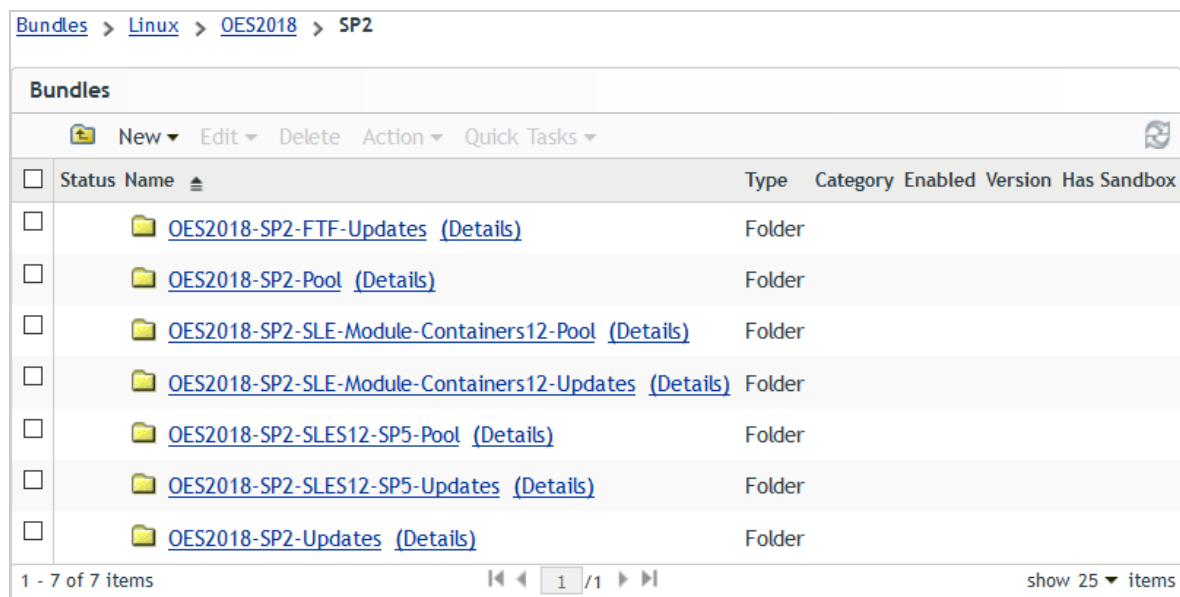
Field Test Files

Field Test Files (FTFs), also known as Patch Test Files (PTFs) are RPMs provided by engineering as fix for a reported issue.

They can be deployed using ZCM much the same way as ordinary updates. The major difference is that there is no repository from which FTFs can be replicated, so we need to build this source repository ourselves using a RPM Application bundle.

The first step is to create a folder for the FTFs parallel to the folders for Pool Bundles and Update Bundles.

Figure 8-30 Folder for OES2018 SP2 FTF Updates



Inside this folder we create the RPM Application Bundle and upload the RPM files for the FTFs into it.

Figure 8-31 Packages in an OES2018 SP2 FTF Updates bundle

Name	Epoch	Version	Release	Architecture	Targets	Freshen	Install Type
ncs-kmp-default	0	2.6.7_k4.12.14_120	0.34.7.1712.0.PTF.1171981	x86_64	All	No	Auto
novell-cluster-services	0	2.6.7	0.100.7.1712.0.PTF.1171981	x86_64	All	No	Auto

From here onwards the process to deploy the FTFs is the same as creating and deploying a FPL for an Update Bundle obtained through a subscription (“Creating a Frozen Patch Level” on page 102).

The bundle is copied and added to a bundle group that in turn is assigned to the Upgrade Server Group that has the target devices as members.

Figure 8-32 OES2018 SP2 FTF Updates Bundle and FTF Updates Bundle Group

Status	Name	Type	Category	Enabled	Version	Has Sandbox
<input type="checkbox"/>	OES2018-SP2-FTF-Updates-PRD	Bundle Group				
<input type="checkbox"/>	OES2018-SP2-FTF-Updates-20200615	Linux Bundle	RPM Application	Yes	0	No
<input type="checkbox"/>	OES2018-SP2-FTF-Updates-bundle	Linux Bundle	RPM Application	Yes	0	No

Finally the YUM service for this FTF Updates Bundle Group is configured.

Every time a FTF s added or removed from this bundle a new FPL needs to be created for it.

Assigning Update Bundles After a Server Upgrade

IMPORTANT: Due to the changes in the way package dependencies can be defined in RPM 4.13 and later and due to the fact that the ZENworks Adaptive Agent does not understand these [Boolean Dependencies](#) `zac bin` and `zac up` can no longer be used to deploy updates to SLES and OES systems.

We therefore no longer recommend to assign Update Bundle Groups to server groups. Instead use the `zypper patch` command to deploy the updates from the YUM repos of the Update Bundle Groups.

After a server has been upgraded to the next support pack using AutoYaST package management tools such as `zypper` automatically have access to the Add-On repository for the new support pack (see [Figure 5-2 Zypper and ZCM Repositories After Upgrade to OES2018 SP2](#)).

To supply the ZENworks Adaptive Agent with the same updates the server must be removed from the server group representing systems with the old support pack and must be made a member of the server group representing the systems that already have been upgraded to the new support pack. For example, if you have upgraded a server from OES2018 SP1 to OES2018 SP2 in your production environment you need to move its server object from the server group `PRD_OES2018SP1` to the server group `PRD_OES2018SP2`.

After a `zac ref bypasscache` the ZAA will be using the same updates as `zypper`. The `zac bin` command needs to be executed to update the bundle status from Downloaded to Available ([Figure 8-25, "Installation of Update Bundles with zac bin," on page 106](#)).

Figure 8-33 Zypper and ZCM Repositories on OES2018 SP2

```
Repository priorities are without effect. All enabled repositories share the same priority.
```

#	Alias	Name	Enabled	GPG Check	Refresh
1	OES2018-SP2-2018.2-0	OES2018-SP2-2018.2-0	Yes	(r) Yes	Yes
2	OES2018-SP2-SLES12-SP5-UPDATE-PRD	OES2018 SP2 SLES12 SP5 Updates (PRD)	Yes	(p) Yes	Yes
3	OES2018-SP2-UPDATE-PRD	OES2018 SP2 Updates (PRD)	Yes	(p) Yes	Yes

Processing Command: bl

Display Name	Version	Bundle Type	Status	Assigned
OES2018-SP2-SLES12-SP5-Updates-20200324	0	Linux Bundle	Downloaded	Device
OES2018-SP2-Updates-20200324	0	Linux Bundle	Downloaded	Device

Total Bundles Assigned: 2

Use the commands `zypper lr` and `zac bl` to verify that `zypper` and the ZAA on the system are using the desired repositories.

Managing Configuration Bundles and Configuration Bundle Groups

The creation of a configuration bundle is always an administrative task. As a general rule all our configuration bundles have been created as empty Linux bundle but you can use any category of Linux bundle available in ZCM to create a configuration bundle.

Bundle Design Guidelines

In theory configuration bundles are only assigned to devices where these bundles make sense. However, there is always a chance of error and therefore, with the exception of bundles that apply to all devices we use requirements to ensure that a bundle will only become effective on the intended target devices.

Requirements are available on the bundle level as well as on the individual action. For ease of administration requirements on the bundle are preferred. For instance, if you want to create a bundle to configure some NSS setting (bundle “OESALL_NOV-NSSSTART-CFG” on page 118) this bundle should have a requirement to only apply to devices where the file `/etc/opt/novell/nss/nssstart.cfg` does exist.

Action requirements are used as an exception. For instance there are differences in the default NTP configuration on SLES11 and SLES12. Therefore modifications to `/etc/sysconfig/ntp` (bundle “SLESALL_NOV-NTP-SYSCONFIG” on page 121) depend on the SLES release and actions that only apply to a specific release must be defined with a requirement for that particular release.

Finally all Linux bundles must execute their actions as root otherwise they will fail to execute.

We propose that for configuration bundles you use the following naming scheme.

Table 8-7 Configuration Bundle Naming Scheme:

%PRODUCT%[%VERSION%[SP%SP-LEVEL%]]_%INITIATOR%-[%PURPOSE%[-%TARGET%]]

Name	Description
PRODUCT	EDIR IMAN SLES OES This part of the bundle name identifies the component or product to which a bundle applies. Bundles for SLES also apply to OES systems. Bundles for eDirectory or iManager apply to both, SLES systems and OES systems.
VERSION	[11 12 15 2015 2018 .. ALL] This part of the bundle name indicates the minimum version of a product to which a bundle applies. If a bundle does not have a version limitation, ALL is used instead. This part is not used for eDirectory or iManager specific bundles. For instance, as NLVM was only introduced in OES11 the name of any NLVM specific bundle needs to begin with OES11... to indicated that this bundle must only be assigned to devices running OES11 and later versions.
SP-LEVEL	[[1 2 3 ..] [only]] ALL This part of the bundle name indicates the minimum support pack level of a product to which a bundle applies. If a bundle does not have a support pack level limitation, ALL is used instead. If a bundle should only apply to a specific support pack only will be added. This part is not used for eDirectory or iManager specific bundles. For instance, as Ganglia was only introduced in OES11 SP2 the name of any Ganglia related bundle needs to being with OES11SP2... to indicate that this bundle must only be assigned to devices running OES11 SP2 or later.

INITIATOR	<p>MFC NOV <CUST></p> <p>This part of the bundle name identifies who created the initial version of a bundle. Bundles that are not specific for a customer and have been created by Micro Focus Consulting use MFC. Bundles that have been created before the merger between Novell and Micro Focus still use NOV.</p> <p>Customer specific as well as customer created bundles use a customer specific identifier <CUST> instead (max. 4 characters).</p>
PURPOSE	<p>This part of the bundle name is fairly flexible and is intended to give a hint on which part of a system is affected by the bundle.</p> <p>This can be the name of a configuration file that is modified by the bundle such as NTP-CONF (note that we use “-” instead of “.”).</p> <p>Where a bundle modifies multiple configuration files or modifies different files in different releases it may be appropriate to indicate the function of the system that is being configured, i. e. BOOTSPASH-CONFIG.</p>
TARGET	<p><Cluster1> PRV ...</p> <p>This is an optional part of the naming standard that only will be used if a bundle is specific for certain systems such as bundle distributing the cluster specific bindings files to the nodes of a given NCS cluster.</p> <p>Another use case would be a bundle that only applies to systems in a specific location.</p>

The suggested names for configuration bundles are case-sensitive and the separators “_” and “-” are a relevant part of their name.

Assigning Configuration Bundles

Just like update bundles are made a member of Update Bundle Groups, configuration bundles are made a member of Configuration Bundle Groups. The naming scheme for these bundle groups has already been defined in [Table 8-6 Configuration Bundle Groups Naming Scheme](#).

These Configuration Bundle Groups are assigned to the various Server Groups with a distribution schedule of On Device Refresh and immediate installation after distribution.

Selected Configuration Bundles

Over the years many configuration bundles have been developed for different customers and projects. This section describes a set of ZCM bundles used in literally every consulting project.

We describe how these bundles are build and designed and we also give some more background on the use case applicable to these bundles.

Archives with these bundles can be downloaded from the [CIF download site](#) and can be directly imported into your ZCM zone.

Please be aware that these bundles are only intended as examples. You should carefully inspect each bundle and verify that it meets your specific needs before you apply it to your environment.

Process the file `zman_bundle_assign.in.cif` through the `zman bex` command to make the imported bundles a member of your Configuration Bundle Groups (see [“Configuring Your Own ZENworks Management Zone” on page 137](#)).

Bundles in the Folder EDIR-BASECONFIG

Bundles in this folder are used to modify the configuration of eDirectory or its services.

EDIR_MFC-disable-FIPS

All eDirectory 9.0 servers run in FIPS mode for OpenSSL by default however, FIPS is not supported with OES ([eDirectory Features Not Supported in OES 2018 SP2](#)).

Operating OES servers with FIPS enabled can have detrimental side effects. Therefore FIPS mode needs to be disabled after OES2018 has been installed or a server has been upgraded to OES2018. If you should have non-OES eDirectory servers in the same tree as your OES servers it is recommended to apply this bundle to these servers as well.

The bundle uses an Edit Text File action that searches a line beginning with `"n4u.server.fips_tls=1"` in `/etc/opt/novell/eDirectory/conf/nds.conf`. If such a line is found, the whole line is replaced with `"n4u.server.fips_tls=0"`.

The unavoidable backup files that are created in the process are removed through a File Removal action.

Summary

- ◆ Environments: all OES2018 servers
- ◆ Components: eDirectory, FIPS
- ◆ Configuration File(s): `/etc/opt/novell/eDirectory/conf/nds.conf`
- ◆ Documentation: [Configuring eDirectory in FIPS Mode for OpenSSL](#)

EDIR_NOV-NMAS-CONF

By default NMAS refreshes its login policies with every login attempt. In particular on servers that do not hold a replica of the Security container this is not desirable. In addition the NMAS login policies are not likely to change frequently so it is sufficient to refresh them at scheduled intervals.

The bundle uses an Edit Text File action to create the NMAS configuration file with a contents of `"nmas RefreshRate 30"`.

Summary

- ◆ Environments: all eDirectory servers
- ◆ Components: NMAS
- ◆ Configuration File(s): `/var/opt/novell/eDirectory/data/nmas.config`
- ◆ Documentation: [Using the Policy Refresh Rate Command](#)

Bundles in the Folder IMAN-BASECONFIG

The bundles in this folder modify the configuration of iManager and manage which users are allowed to make configuration changes.

IMAN_NOV-change-config-xml

In the real world some default settings of iManager such as the timeout of 30 minutes are not very practicable. This bundle is used to modify some of these settings. It first deploys the python script `change_config_xml.py` to `/usr/local/zac`. Subsequent actions execute the python script to set the following parameters:

- ◆ Enable “Remember login credentials”
- ◆ Disable “Hide specific reason for login failure”
- ◆ Timeout 180 minutes

Finally an embedded bash script is used to set the tree name and to add the server name to the login dialog.

The bundle can easily be extended with additional actions to set other settings to meet your needs.

Summary

- ◆ Environments: all iManager servers
- ◆ Configuration File: `/var/opt/novell/iManager/nps/WEB-INF/config.xml`

IMAN_NOV-users

By default only the user that installs a server has the rights to manage iManager on this server. In many environments it is desirable to have multiple users or groups configured in this role.

An embedded bash script is used to set the users in the iManager configuration file. You must modify this script and add the users and/or groups that you want to grant iManager manager capabilities.

Summary

- ◆ Environments: all iManager Servers
- ◆ Configuration File:
`/var/opt/novell/iManager/nps/WEB-INF/configiman.properties`

Bundles in the Folder OES-SERVICES

Bundles in this folder configure various aspects of different OES services.

OES11ALL_NOV-CHECK-MOVE

By default OES does not provide a means to easily monitor the progress of a pool move. This bundle deploys the script `check_move.sh` to `/usr/local/ncif`. When this script is executed with a pool name as the only parameter it documents the status of the corresponding pool move in the file `/usr/local/ncif/move_<PoolName>`. Executing this script in regular intervals through cron will document the progress of a pool move.

Summary

- ◆ Environments: all OES servers with NSS
- ◆ Components: NSS
- ◆ Tools: NLVM

OES11ALL_NOV-NLVM-CMD-COMPLETIONS

This bundle implements command completion for `nlvm` commands by deploying the completion file `nlvm_completion.sh` to the `/etc/profile.d` directory. With this file in place possible completions for a `nlvm` command are displayed when hitting the `<TAB>` key.

Summary

- ◆ Environments: all OES servers with NSS
- ◆ Components: NLVM
- ◆ Configuration Files: `/etc/profile.d/nlvm_completion.sh`

OES11ALL_NOV-NLVM-CONF

As a general rule it is not advisable that `nlvm` has access to the system device(s). This bundle only applies to systems installed through the Consulting Installation Framework. An embedded bash script determines devices from `/root/install/variables` and excludes the first system device from `nlvm.conf`. If the second device should be used by a LVM VG named ZCM it will also be excluded.

Summary

- ◆ Environments: all OES servers with NSS
- ◆ Configuration File: `/etc/opt/novell/nss/nlvm.conf`

OES2018SP3_MFC-DISABLE-OES-AGENTS

Cloud Integration Services (CIS) agents are active on every OES2018 servers that has NSS installed by default. With OES2018 SP3 the same applies to the OES Dashboard agents. If CIS is not implemented in an environment this is not advisable. The bundle uses an embedded bash script to stop and disable the following agents:

- ◆ `oes-core-agent`
- ◆ `oes-dashboard-agent`

- ♦ oes-cis-agent
- ♦ oes-cis-recall-agent
- ♦ oes-cis-scanner

Summary

- ♦ Environments: all OES2018 SP3 servers and later with NSS not using CIS

OESALL_NOV-BASHRC

This bundle deploys `bash.bashrc.local` to `/etc`. This file primarily contains aliases for cluster commands that are executed via `sudo`. For instance the command `cluster status` can be executed using its alias `cs`.

Summary

- ♦ Environments: all OES cluster nodes
- ♦ Components: cluster services, bash
- ♦ Configuration File: `/etc/bash.bashrc.local`
- ♦ Documentation: [Novell Cool Solutions: Bash - Making use of your .bashrc file](#)

OESALL_NOV-CIFS-CONFIG

The bundle uses an embedded bash script to set the following CIFS parameters:

- ♦ block invalid users for 5 minutes
- ♦ disable offline caching
- ♦ enable DFS support for CIFS
- ♦ enable sub-tree search for CIFS

A second embedded bash script is used to remove the server context from and to add the organization as additional context to the CIFS context file.

Summary

- ♦ Environments: all OES servers with CIFS
- ♦ Components: n/a
- ♦ Tools: n/a
- ♦ Configuration File:
 - `/etc/opt/novell/cifs/cifs.conf`
 - `/etc/opt/novell/cifs/cifsctxs.conf`

OESALL_NOV-NAM-CONF

This bundle uses an embedded bash script to set the following LUM configuration options:

- ♦ `cache-only=yes`

- ♦ `persistent-search=no`
- ♦ `case-sensitive=no`
- ♦ `convert-lowercase=yes`
- ♦ `dont-deny-pamservice=yes`

Summary

- ♦ Environments: all OES servers
- ♦ Configuration File: `/etc/nam.conf`

OESALL_NOV-NCPSEV-CONF

This bundle uses an embedded bash script to disable NCP broadcasts and to configure the handling of duplicate shadow files in DST. It sets the following NCP server options:

- ♦ `DISABLE_BROADCAST = 1`
- ♦ `DUPLICATE_SHADOW_FILE_ACTION = 4`
- ♦ `DUPLICATE_SHADOW_FILE_BROADCAST = 0`

Summary

- ♦ Environments: all OES servers
- ♦ Configuration File: `/etc/opt/novell/ncpserv.conf`
- ♦ Technical Information Document:
 - ♦ [TID 7004848: Usage of the FIRST_WATCHDOG_PACKET NCP parameter on OES Linux](#)
 - ♦ [TID 7004888: NCP performance tuning on Open Enterprise Server Linux](#)

OESALL_NOV-NSSSTART-CFG

This bundle uses an embedded shell script to set the following NSS parameters:

- ♦ `/ListXattrNWmetadata`
- ♦ `/CtimeIsMetadataModTime`
- ♦ `/FastWriteOfMessyBeasts`

Summary

- ♦ Environments: all OES servers with NSS
- ♦ Configuration File: `/etc/opt/novell/nss/nssstart.cfg`

OESALL_NOV-PureFTP-local-bind

The FTP server by default listens on all interfaces of a server. When you want to implement FTP for a cluster resource it is important to reconfigure the FTP server to only listen to the IP address of the cluster node. This is achieved with an embedded bash script that determines the primary IP address of the host and sets it as the only IP address for FTP. The FTP service is restarted.

Summary

- ◆ Environments: all OES cluster nodes with FTP enabled cluster resources
- ◆ Configuration File: `/etc/pure-ftpd/pure-ftpd.conf`

OESALL_NOV-PureFTP-SYSLOG-ClusterName

Pure FTP by default uses the syslog facility ftp and logs to `/var/log/messages`. This can be pretty confusing when using FTP to access cluster resources and most customers prefer if the logging goes to the corresponding cluster resource in such cases.

This bundle uses Edit Text File actions to illustrate how to reconfigured the syslog facility local1 to point to `/media/nss/FTP/var/log/pure-ftpd-FTP.log`. local1 must be set as SyslogFacility in the Pure-FTP configuration file on the cluster resource.

This is obviously only an example assuming that the FTP cluster resource will use an NSS volume named FTP. Also ClusterName in the name of the bundle needs to be replaced with the name of the cluster hosting the FTP service.

The syslog service is reloaded to activate the configuration change.

Summary

- ◆ Environments: all OES cluster nodes with FTP enabled cluster resources
- ◆ Configuration File: `/etc/rsyslog.conf`
- ◆ Documentation: [Cluster Enabling Pure-FTPd in an OES Environment](#)

Bundles in the Folder SLES-SERVICES

SLES12ALL_MFC-LFTP-CONFIG

lftp is the default FTP client in SLES12 and beyond. This bundle uses an embedded bash script to enable SSL support for FTP and to set the path where the CA certificates need to be stored to `/etc/ssl/certs/`.

Summary

- ◆ Environments: all OES2018 servers that need a SFTP client
- ◆ Configuration File: `/etc/lftp.conf`

SLES15ALL_MFC-enable-chrony-server

Chrony by default is configured as NTP client only. To enable the server functionality of chrony the `allow` directive must be added to `/etc/chrony.conf`.

Without any additional parameter this directive will allow any IPv4 and any IPv6 address to retrieve time from the server. `man chrony.conf` has all the gory details on how to further manage which clients may retrieve time from a server and which may not.

Summary

- ◆ Environments: all OES2023/SLES15 servers that need to be time providers.

By default this bundle is a member of all our bundle groups named *eDir*.

This may or may not meet your requirements.

- ◆ Configuration File: `/etc/chrony.conf`.

SLESALL_Customer-SSH-Keys

This bundle implements a centralized management for SSH public keys for root access. It uses an Edit Text File action to create/replace the `authorized_keys` file in `/root/.ssh`.

The file can be used to deploy the public SSH key for users that need to access the target system as root. It also can be used to deploy public SSH keys for other servers that shall be able to execute ssh commands on the target system.

Summary

- ◆ Environments: all OES servers
- ◆ Components: SSH
- ◆ Configuration File: `/root/.ssh/authorized_keys`

SLESALL_MFC-DHCP-BONDO

If a DHCP server is equipped with a bonded network interface the value of `DHCPD_INTERFACE` needs to be adjusted accordingly. The bundle uses an Edit Text File action to modify the DHCP configuration file. A File Removal action removes the undesired backup files created by the first action.

Summary

- ◆ Environments: all OES2018 servers that provide DHCP over bond0
- ◆ Configuration File: `/etc/sysconfig/dhcpd`

SLESALL_MFC-DISABLE-ZISLNX-SERVICE

The imaging service of the ZAA cannot work on systems with GPT partitioned disks because the area where it wants to store its ZIS record is occupied by the first GPT partition table. In addition as the CIF does not use imaging this service should always be stopped and disabled. The bundle uses an embedded bash script to stop and disable `novell-zislrx`.

Summary

- ◆ Environments: all systems installed and managed through the CIF

SLESALL_NOV-BOOTSPLASH-CONFIG

SLES servers by default display a splash screen during boot. This bundle disables this splash screen so that the messages during boot can be viewed on the screen without the need to first have to hit the `ESC` key.

For SLES11 this is implemented through an embedded bash script that modifies the configuration for the grub boot loader and the sysconfig settings that are used when re-creating the initrd.

For SLES12 and beyond an Edit Text File action is used to modify the default settings for the grub2 boot loader and the initrd is recreated.

Requirements ensure that these actions are only executed on servers with matching SLES release.

Summary

- ◆ Environments: all SLES servers
- ◆ Components: `grub` / `grub2`
- ◆ Tools: `SLES12: /usr/sbin/grub2-mkconfig`
- ◆ Configuration File: `SLES11: /boot/grub/menu.lst`
`SLES11: /etc/sysconfig/bootloader`
`SLES12: /etc/default/grub`
`SLES12: /boot/grub2/grub.cfg`

SLESALL_NOV-no-SuSE-Register

By default `cron_suse_register` is executed at 19:41 of the 5th of every month to (re-)register a system with the customer center. In many environments the servers do not have access to the internet and therefor this registration will always fail.

This bundle uses an embedded bash script to delete the corresponding crontab file.

Summary

- ◆ Environments: all SLES servers without internet access
- ◆ Configuration File: `/etc/cron.d/novell.com-suse_register`

SLESALL_NOV-NTP-SYSCONFIG

The NTP daemon terminates silently if the time it receives deviates from the local time by more than 15 minutes. This can easily happen whenever the system time is changed because daylight saving time begins or ends, the hardware clock does not get updated accordingly and the system is restarted.

In an eDirectory environments this is not desirable. Therefor this bundle uses an embedded bash script to set `NTPD_FORCE_SYNC_ON_STARTUP` to `yes`. A second embedded bash script does also set `NTPD_FORCE_SYNC_HWCLOCK_ON_STARTUP` to `yes` on SLES11 servers. As this setting is default on SLES12 a restriction is used to ensure that this action is only executed on SLES11. A third action restarts the NTP daemon.

A bundle requirement for the existence of the configuration file ensures that this bundle does not get executed on SLES15 servers that are using `chrony` instead of `ntp`.

Summary

- ◆ Environments: all SLES11 and SLES12 servers
- ◆ Configuration File: `/etc/sysconfig/ntp`

SLESALL_NOV-SLPDA-CONFIG

This bundle uses an embedded bash script to configure the SLP daemon to act as Directory Agent. It also enables the synchronization of SLP service registrations between different DAs and enables the backup of local and remote service registrations every 15 minutes.

Local IP addresses are removed from the list of DAs. Finally `slpd` is stopped and started again after a delay of 5 seconds.

Summary

- ◆ Environments: ONLY servers that need to be configured as SLP Directory Agent
- ◆ Configuration File: `/etc/slp.conf`

SLESALL_NOV-ZYPP-CONFIG

This bundle uses an embedded bash script to configure `zypper` to download all packages before it starts to install them. This is of particular importance if `zypper` is used to retrieve and install a series of packages over a WAN link that might become unavailable during the process.

The bundle also disables the use of delta rpms because this typically results in a high load on the system being updated.

Summary

- ◆ Environments: all SLES servers
- ◆ Configuration File: `/etc/zypp/zypp.conf`

Bundles in the Folder SLES-STORAGE

The bundles in this folder are used to manage the multipath configuration typically required when servers access storage systems over a Storage Area Network (SAN).

DM-MPIO uses two configuration files. The behavior of the daemon is configured in `multipath.conf`. To avoid any inconsistency the goal is that exactly one configuration file shall be used for all systems in a customer environment. This file may contain settings for multiple storage systems. The daemon will only use those settings that are relevant for the storage systems a particular server is connected to.

By default the disk devices provided by a storage system are uniquely identified by 32-byte World Wide Identifier, the WWID. If a server is attached to many storage devices it can become difficult to distinguish these devices based on their WWIDs. DM-MPIO allows to use a `bindings` file to translate WWIDs into user friendly names. The `bindings` file should be specific for a server, or a group of servers in case of a cluster.

SLESALL_NOV-BINDINGS-ClusterName

This bundle just copies the file `bindings` to `/etc/multipath` with mode 600 into the directory `/etc/multipath` and reconfigures the multipath daemon with this new file using an embedded bash script.

The `bindings` file in this bundle is only a template. Whenever multipath discovers new devices it will add the corresponding entries to the end of the file using the default multipath device names. To assign a user friendly name to a device you simply replace the default multipath device name with the name of your choice. If the device is used by NSS the device name you choose must not match the name of the NSS pool or the NSS volume. To avoid this problem the device names in the sample file have been appended with “-1”.

For better readability it is recommended to separate the two columns with tabs rather than just spaces.

Finally, if you are working in a cluster and compare the `bindings` file of different cluster nodes you might find that the device that has the name `mpatha` assigned on the first node is using a different name on another node. As this can become pretty confusing in a cluster using many devices it is mandatory to ensure that all nodes of a give cluster are using the same `bindings` file.

Therefore, whenever there are any changes to the devices assigned to a cluster you only make the change in the `bindings` file of one cluster node, upload it to your ZCM zone and let ZENworks distribute it to all cluster nodes.

Summary

- ◆ Environments: all servers attached to a Storage Area Network
- ◆ Configuration File: `/etc/multipath/bindings`
- ◆ Documentation: [Configuring User-Friendly Names or Alias Names](#)

SLESALL_NOV-MULTIPATH-CONF

This bundle deploys the `multipath.conf` file to `/etc` and reconfigures the multipath daemon using an embedded bash script.

The configuration file in this bundle sets a number of `defaults` that apply to all storage systems. Device specific settings for a number of commonly used storage systems are contained in the `devices` section.

The `blacklist` section demonstrates various ways to prevent the multipath daemon to manage local devices independent of their WWID. This is very important to avoid systems with `btrfs` and `multipath` to exit to emergency mode.

Summary

- ◆ Environments:
- ◆ Components: n/a
- ◆ Tools: n/a
- ◆ Configuration File: `/etc/multipath.conf`
- ◆ Documentation: [Creating or Modifying the /etc/multipath.conf File](#)
- ◆ Technical Information Document: [TID 19607: System exit to emergency shell at boot with multipath enabled](#)

Bundles in the Folder SLES-NETWORK

The bundles in this folder are used to manage various aspects of network connectivity.

SLES11ALL_MFC-BONDING

Manually setting up a bonded network interface requires many steps that are automated by this bundle.

As a precaution this bundle is only installed if no bond device does exist on the target system. If this is the case it deploys the bash script `config_bond.sh` to `/usr/local/zac`. A second action will create an active-passive bond device with slaves `eth0` and `eth1` (up to four slaves are possible) using one ARP ping per second to the default gateway to monitor the link status. As this process will restart the network this bundle should not be executed on an active cluster node.

The script can also be used to create a bond that cannot reach the default gateway. In this case the IP address of the ARP ping target must be specified using the `--target` command line parameter.

Summary

- ◆ Environments: all servers that require bonded network interfaces.
- ◆ Tools: `/usr/local/zac/config_bond.sh`
- ◆ Configuration File: `/etc/sysconfig/network/ifcfg-bond0`

SLESALL_MFC-MTU-9000-eth2-eth3

When using iSCSI it is very important to increase the maximum packet size to improve throughput.

This bundle deploys the bash script `set_mtu.sh` to `/usr/local/zac`. A second action executes the script and sets the MTU for all interfaces specified as command line parameter to 9000.

Summary

- ◆ Environments: all servers with iSCSI connections
- ◆ Configuration File: `/etc/sysconfig/network/ifcfg-eth*`

Agent Commands

A detailed description of the available agent commands and their usage can be found in the [ZENworks Command Line Utilities Reference](#), in the man page for `zac` or by using `zac --help` to call the agent help.

This section only gives a brief overview of some frequently used `zac` commands. Use `zac <command> --help` for a more specific help on any sub-command.

General Commands

Command	Description
<code>zac agp (agent properties)</code>	Displays a summary about the agent state, operating system details, and component versions.
<code>zac cl (configuration location)</code>	Displays the configuration location.
<code>zac get (get-preferences)</code>	Lists agent settings such as proxy, rollback, and log settings.
<code>zac set</code>	Changes settings listed with <code>zac get</code> .
<code>zac logger</code>	Displays or changes agent log configurations.
<code>zac ref (refresh)</code>	Refreshes all agent information. Typically used with <code>bypasscache</code> to avoid using data from the server cache.
<code>zac zc (zone-config)</code>	Displays information about the ZENworks Server that the device is accessing for configuration information.

Bundle Commands

Command	Description
<code>zac bl (bundle list)</code>	Lists all assigned bundles.
<code>zac bin (bundle install)</code>	Installs a given bundle.
<code>zac bu (bundle uninstall)</code>	Uninstalls a given bundle (use with caution).
<code>zac bv (bundle verify)</code>	Reinstalls a given bundle and executes the Verify Action sets.

Package Management Commands

Command	Description
<code>zac in (install)</code>	Installs a package and its dependencies.
<code>zac lu (list updates)</code>	Lists all updates.
<code>zac rm (remove)</code>	Removes a package and its dependencies.
<code>zac se (search)</code>	Searches for packages with a name matching the specified pattern.
<code>zac up (update)</code>	Updates packages where higher versions are available from any assigned channel.

Registration Commands

Command	Description
<code>zac ark (add-registration-key)</code>	Adds a key for group registration.
<code>zac reg (register)</code>	Registers an agent with the ZENworks Management Zone.
<code>zac retr (reestablish-trust)</code>	Reestablishes trust with the current Management Zone

Diagnostics and Debugging

ZENworks Configuration Management consists of many different components. This section gives brief instructions on how to enable debug logging for these components and where to find the resulting log files. A more detailed description is available in [TID 3418069: How to enable debug logging for ZENworks Configuration Management](#).

ZCM Server

ZENworks Control Center

Configuration:

```
/opt/novell/zenworks/share/tomcat/webapps/zenworks/WEB-INF/config.xml  
Setting ID: debug.tags;
```

uncomment required values under this tag as requested by Novell Technical Support.

Log Location: `/var/opt/novell/log/zenworks/zcc.log`

ZENworks Web Service

Log Location: `/var/opt/novell/log/zenworks/services-messages.log`

CASA

Configuration:

```
/etc/CASA/authtoken/svc/log4j.properties
```

```
set log4j.rootLogger= (change warn to debug and restart the service)
```

Log Location: /srv/www/casaats/logs/*

zman

Configuration:

```
/etc/opt/novell/zenworks/zman/properties/zman-config.properties
```

```
set DEBUG_LEVEL=4
```

Log Location: /var/opt/novell/log/zenworks/zman.log

ZCM Agent

Changing Log Level

```
zac log level debug
```

```
zac set CaptureAxis2Logs true
```

Logfiles

File	Description
/var/opt/novell/zenworks/novell-zenworks-xplatzmd.out	Contains agent start and stop messages.
/opt/novell/zenworks/logs/LocalStore/zmd-messages.log	Most relevant agent information is found here.
/var/log/zmd-satbackend.log	This log deals only with actions taken by the agent to install an RPM. It also shows the communication that happens (essentially API calls) between the agent and SATLIB. SATLIB is an interface to the RPM that actually does the install.
/var/opt/novell/log/zenworks/preboot/novell-zislnx.log	Contains information about the managed ZIS partition of a device. The information is written or read from the ZIS sector.



How To Build Your Own Installation Framework

This solution has been developed and tested by Micro Focus Consulting Germany in cooperation with SUSE Consulting Germany. The scripts, the configuration files, the PXE configuration files, the ZCM bundles and some other supporting material that is intended to simplify the process of building your own installation framework are available at the [CIF download site](#). This material is provided as is and is not officially supported by Micro Focus. Micro Focus [Terms of Use](#) do apply.

This section provides instructions how to use the contents of the archive `cif.tgz` to build your own AutoYaST, to build and configure your ZENworks Management Zone supporting SLES12 SP5, SLES15 SP1 and OES2018 SP2, and to optionally create your PXE environment.

The archive contains the following files:

- ♦ **AutoYaST-cif_yyyymmdd.tgz**

Everything needed to build your own AutoYaST server

- ♦ **ZCM-cif_yyyymmdd.tgz**

Everything needed to build and configure your ZENworks Management Zone

- ♦ `zman_params.cfg`

This is the parameter file that defines the names for the folders for devices, bundles, and registration keys and is used by all bash scripts used to build your ZCM zone. If you plan to use more or other releases and/or support packs you will need to set the parameters appropriately in `zman_params.cfg` to include them into your setup.

- ♦ `/zman_structure_create`

The bash script `zman_structure_create.sh` consumes `zman_params.cfg` and creates the file `zman_structure_create.in` which contains all zman commands required to build the desired structure in your ZCM zone.

If you want to use the names for folders, groups and registration keys that we have used throughout this guide you just can process the file `zman_structure_create.in_cif` provided in this directory.

- ♦ `/zman_bundle_assign`

The bash script `zman_bundle_assign.sh` consumes `zman_params.cfg` and creates the file `zman_bundle_assign.in` which contains all zman commands required to assign the configuration bundles to the various bundle groups..

If you want to use the names for folders, groups and registration keys that we have used throughout this guide you just can process the file `zman_structure_create.in_cif` provided in this directory.

- ◆ `/ZCM_bundles`

This folder contains archives with exported ZCM configuration bundles discussed in [“Managing Configuration Bundles and Configuration Bundle Groups”](#) on page 111, one archive for each bundle folder.

- ◆ `PXE_AutoYaST-cif_yyyymmdd.tgz`

Everything needed to use your AutoYaST server as TFTP server.

- ◆ `PXE_ZCM-cif_yyyymmdd.tgz`

Everything needed to use a ZCM Primary server as TFTP server.

In this archive `/srv/tftp/pxemenu.txt` and `/srv/tftp/efi/x86_64/pxemenu.txt` are renamed to `CIF-pxemenu.txt` to protect any existing PXE configuration.

It will take some time to build your own installation framework. In particular it will take some time to figure which information should be provided in which configuration file.

Most customers find that the best thing to do is to start with a small test environment that uses all OES components to understand the effect of the various settings. Before starting to customize the solution to meet your needs.

9 Building Your Own AutoYaST Server

The first step in creating your own AutoYaST server supporting OES2018 SP2, SLES12 SP5, and SLES15 SP1 is to install a SLES12 SP5 or a SLES15 SP1 server as described in [“Installing and Configuring an AutoYaST Server” on page 39](#).

Once this server is available copy the archive `AutoYaST-cif_yyyymmdd.tgz` (see [CIF Download site](#) above) into the directory `<YourDirectory>` and extract it. This will create the directory structure shown in [Figure 5-1](#) and the directory `<YourDirectory>/data/etc`.

Move the contents of `<YourDirectory>/data` to `/data`.

Next download and copy the following files to the `/data/isos` directory:

- `OES2018-SP2-DVD-x86_64-DVD1.iso`
- `SLE-12-SP5-Server-DVD-x86_64-GM-DVD1.iso`
- `SLE-15-SP1-Installer-DVD-x86_64-GM-DVD1.iso`
- `SLE-15-SP1-Packages-x86_64-GM-DVD1.iso`

Make sure to compute the checksums of the copied files and to compare them to the checksums provided on the download site of the vendor.

Loop mount the ISOs to `/data/install` and make them available through Apache by performing the following actions:

- Add the contents of the file `<YourDirectory>/etc/fstab.cif` to the end of `etc/fstab` and execute the `mount -a` command.

This will loop mount the ISOs copied in the previous step under `/data/install`.

- Move `<YourDirectory>/etc/apache2/conf.d/inst_server.conf` to the `/etc/apache2/conf.d` directory, enable and start Apache. If Apache should be already running reload it instead.

Verify that all your installation sources are accessible via HTTP by entering the following URLs into your browser:

```
http://<IP address of your AutoYaST server>/<product><version><sp>
```

For example:

```
http://10.10.10.101/oes2018sp2
```

The next step is to prepare your environment so that it can create the required Custom Boot ISO. For every release you want to install using CIF you first have to copy the kernel and the corresponding `initrd` into the directories under `/data/boot_cd_build/kernel` as explained [Table 3-1, “Directory Structure of AutoYaST Boot Medium,” on page 22](#).

In all `grub` configuration files under `/data/boot_cd_build/grub` replace `"<IP address of your AutoYaST server>"` with the IP address of your AutoYaST server. Use search and replace to ensure you will replace all occurrences. Also replace `"10."` with the gateway for your environment. You can provide only parts of the gateway address.

If you want your Custom Boot ISO to support systems with UEFI you have to define the IP address of your AutoYaST server, the IP address of your ISO server, and the IP address of your gateway in lines 2-4 of each *.cfg file for the various releases in the /data/boot_cd_build/EFI directory. If UEFI support is not required simply delete this directory.

Optionally, in line 120 of /data/boot_cd_build/create-ay-iso.sh replace “CIF” with your customer name.

Execute /data/boot_cd_build/create-ay-iso.sh. This will create your boot ISO “autoyast-\${CUSTOMER}.iso” supporting OES2018 SP2, SLES12 SP5, and SLES15 SP1 in your /data/isos directory.

Rename /data/autoyast/xml/default-(ip) replacing “(ip)” with the IP address of your AutoYaST server next. For example, rename the file to default-10.10.10.101 if the IP address of your AutoYaST server is 10.10.10.101.

In the renamed file replace “<IP address of your AutoYaST server>” with the IP address of your AutoYaST server.

For SLES15 SP1 you need to copy the info template files and replace <IP address of your ...> with the IP address of your AutoYaST, ISO and YUM server.

You can also use the template files to create info files for other products and releases. This will require you to adjust your grub and grub2 configuration files.

The final step is to populate the configuration files used by CIF with the information that defines your environment.

IMPORTANT: Before you copy any configuration file from the templates directory you should carefully inspect each of these files to ensure that the files provided by our solution cover your specific requirements.

If this should not be the case modify the template files to meet your needs before you copy them.

When you have verified that the template files do meet your needs copy /data/autoyast/configs/templates/Your_CUSTOMER.txt to /data/autoyast/configs/CUSTOMER.txt (see “[Customer Configuration File CUSTOMER.txt](#)” on page 51).

For every eDirectory tree in your environment to which you want to install servers using CIF copy /data/autoyast/configs/templates/Your_Tree.txt to /data/autoyast/configs/<TreeName>.txt (see “[Tree Name Configuration File <TREE_NAME>.txt](#)” on page 51).

For example, if an eDirectory tree is named XYZCORP-TREE this file needs to be named XYZCORP-TREE.txt. Remember this file name is case-sensitive.

If you plan to also use locations (see “[Location Configuration File](#)” on page 52) create a copy of /data/autoyast/configs/templates/Your_Location.txt in /data/autoyast/configs for each location. You may use any name you wish for these files.

Once you have copied all the required configuration files from the template directory carefully plan which information will go into which configuration file. Make sure to provide all required information. All variables in the sections “infrastructure information” and “OES information” must

be defined or your installation will fail. Other sections such as “DHCP information” or “DNS information” are optional and only need to be populated if you install the patterns “novell-dhcp” or “novell-dns”.

Remember that `CUSTOMER.txt` should only store information that applies to **all** servers in **all** environments. Information that is specific for an eDirectory tree, for instance the name of the installation user goes into the corresponding tree configuration file and information specific for a location will go into the location configuration file.

The tree configuration file provides a set of settings that is also available in `CUSTOMER.txt`. If you provide a different value for any of these settings in your tree configuration file it will overwrite the corresponding value from `CUSTOMER.txt`.

Even if you want to use all settings from `CUSTOMER.txt` you still need to create a tree configuration file – otherwise you would get an error during the installation of your OES server.

In every tree file in line 222 replace “Your_O” with the base context for LDAP searches in the corresponding eDirectory tree. These searches are performed by AFP, CIFS, iPrint and NetStorage. The base for the LDAP sub-tree search can be any container of your eDirectory structure. Typically all four components use the same search base however, if this should not be the case you can set a different search base for each component.

If you do not use any of these components you should comment lines 222, 227, 232, 236 and 241.

It may sound obvious but it is often overlooked that the tree configuration files only apply to OES servers. Any information defined in these files is never applied to pure SLES systems!

Finally you can create optional location configuration files to further fine tune your configuration settings. Values provided in this configuration will overwrite the corresponding values from the tree configuration file and from `CUSTOMER.txt`.

Once you have entered all the configuration information into all the configuration files there is some additional information that you need to specify in `CUSTOMER.txt`:

Line Numbers Action to Perform

- Line 38: Replace “CIF” with your customer name.
- Line 54-71: Adjust the service types to match your requirements.
- Line 380: Specify the country for the SLES CA.
- Line 407/408: Take a note of the comment concerning password security.
- Line 421: Specify if your hardware clock will be set to UTC or to localtime.
- Line 425: Specify your keyboard mapping.
- Line 429: Specify the language in which you want to install.
- Line 434-437 Set a simple default root password that will ensure you can login if you should have forgotten to specify the root password in other configuration files.
- Line 441: Specify your default time zone.
- Line 498: Specify the IP address of your ZCM Primary server.
- Line 512: Specify the protocol and IP address of the server providing your installation repositories if this should not be your AutoYaST server.

Line Numbers Action to Perform

Line 517: Specify the protocol and IP address of the server providing your YUM repositories for updates if this should not be your ZCM Primary server.

There are different ways to obtain the encrypted password for the root user. One way is to use the autoinstallation feature in YaST (> **Miscellaneous > Autoinstallation Configuration**).

Edit the root user, set your desired password and save the XML file. Copy and paste the encrypted password from the XML file you created to line 437 of `/data/autoyast/configs/CUSTOMER.txt`.

Another way is to just set the password on a running system and to copy it from `/etc/shadow`. The encrypted password is the string between the first colon immediately following the username and the second colon. It always starts with "\$".

Always ensure that the encrypted password for the root user is enclosed in single quotes!

`CUSTOMER.txt` is also the file where you define your Service Types. Basically a Service Type is collection of XML snippets that provide the configuration information for the patterns defined in the software selection files.

We provide base definitions of Service Types for typical SLES11, SLES12 and SLES15 servers. We also provide Service Type definitions for six different types of OES2023 servers:

- ◆ **OES2023_BRANCH**

This variable defines all the XML snippets required to configure the OES services for a branch office server.

- ◆ **OES2023_DSFW_FRD_ADC**

This variable defines all the XML snippets required to configure an additional domain controller for the DSfW Forest Root Domain.

- ◆ **OES2023_DSFW_FRD_FDC**

This variable defines all the XML snippets required to configure the forest domain controller for the DSfW Forest Root Domain.

- ◆ **OES2023_EDIR_IMAN**

This variable defines all the XML snippets required to configure the OES services for an eDirectory/iManager server.

- ◆ **OES2023_EDIR_UMC**

This variable defines all the XML snippets required to configure the OES services for an eDirectory/UMC server.

- ◆ **OES2023_NCS**

This variable defines all the XML snippets required to configure the OES services for a branch office server.

As OES2023 is based on SLES15 services these Service Type definitions must include the Service Type definition for `SLES15_BASE`.

You can define additional Service Types or modify these Service Types to meet your particular needs. For instance, if you do not want to use AFP remove “:oes/afp.xml” from the service type definitions for branch office servers and cluster nodes. In this case make sure you also remove the pattern “novell-afp” from the corresponding software selection file.

NOTE: `.../files/services/oes/imanager.xml` does not install all iManager PlugIns by default. As a consequence, when you log in to iManager the very first time you will not see any Roles and Tasks. You will have to go to **> Configure > Plug-In Installation > Available Novell Plug-In Modules** and install the PlugIns that you need starting with iManager Base Content, iManager Framework, and iManager Framework Content.

We recommend that the you only install the Plug-Ins for the components you are using. This will result in a much leaner, easier to work with iManager.

If you instead want AutoYaST to install all available PlugIns just modify `imanager.xml` and set `<install_plugins>yes</install_plugins>`.

The last configuration file that you need to modify is `server.txt`. You need to have one line in this file for each server that you want to install using CIF.

We provide sample entries to install an eDirectory server with iManager as the first server into a tree and to add a branch office server as the second server to the same tree.

Both servers require a 50 GB hard disk and a minimum of 1.5 GByte of RAM. Just copy the appropriate entry to a new line and provide the information for the first server you want to install. Carefully read the description of the 14 fields and make sure you understand what information is required in which field.

Replace the IP address, specify the tree file, the server context and possibly the location file. If you want to register the server with your ZCM zone provide at least one ZCM registration key in field 09 of the entry.

Also use the instructions in lines 19 of `server.txt` to ensure that your entry has the correct number of fields. Always double-check that the IP address in Field 02 is unique as it is this address that will tell the installation process which line of `server.txt` to use.

If you want to use the full solution you will have to configure your ZENworks Management Zone before you can perform your first AutoYaST installation.

Once your ZCM zone will be configured you can add the ZENworks registration keys to Field 09 of each entry in your `server.txt` file.

To update an existing AutoYaST server to the latest release of the CIF we recommend that you extract the latest `AutoYaST-cif_yyyymmdd.tgz` archive into a temporary directory and make the same modifications you have made for the earlier release.

It is of particular importance to rebuild your own `CUSTOMER.txt`, every tree configuration file and every location configuration file using the updated templates to ensure your build environment will properly support all the enhancements.

Once you have done all your modifications replace any file in `/data/autoyast` and in `/data/boot_cd_build` with the newer file from your temporary directory. Any file that exists in your work directory that you did not create yourself but does not exist in the temporary directory needs to be removed.

NOTE: Do not mix files from the current release of the CIF with files from earlier releases.

Add the ISOs for the newer releases you want your build environment to support to `/data/isos`, Add the mount points for the added ISOs underneath `/data/install` and add them to `/etc/fstab`. Add the new mountpoints to the configuration of the Apache web server on your AutoYaST server by adding the Alias directives for the new releases to `/etc/apache2/conf.d/inst_server.conf`.

Finally add the kernel and initrd for the releases of SLES12, SLES15 and OES2023 that you want to support to `.../boot_cd_build/kernel/<release>` and create a new custom AutoYaST boot medium using the updated `create-ay-iso.sh` script.

10 Configuring Your Own ZENworks Management Zone

Before you begin to build your ZENworks Management Zone make sure to store the credentials for your ZCM users in the home directory of the logged-in user on a ZCM Primary server using the `zman asc` command (see “[zman](#)” on page 79).

If you did install your own ZCM zone you most likely will log in to your ZCM Primary server as `root` (if you are using an appliance make sure to enable SSH access) and will use `Administrator` to login to the ZCC.

Next copy the archive `ZCM-cif_yyyymmdd.tgz` (see [CIF download site](#)) to any directory on the same ZCM Primary server and extract it. Inspect the file `zman_params.cfg` and carefully read the description of each variable. If you do not want to use the names for folders, groups, and registration keys used throughout this document please modify `zman_params.cfg` to meet your requirements.

Change into the directory `zman_structure_create`. If you have made changes to `zman_params.cfg` execute `zman_structure_create.sh`. This will create the file `zman_structure_create.in` that you can process using `zman bex <input file>` to create the required structures in your ZCM zone.

If you did not make any changes to `zman_params.cfg` simply use `zman bex zman_structure_create.in_cif` to create a default structure in your ZCM zone.

`zman` will create all required folders, groups and registration keys for infrastructure servers and the environments DEV, PRD, and TST. The bundle groups will be assigned to server groups as required.

The next step is to create and execute your subscriptions. Login to the ZCC, select **Configuration** and scroll down to **Credential Vault**. Setup one credential for each customer center you will be working with as explained [Figure 8-12](#).

Change to **Subscribe and Share**, create your subscriptions and place them in the folders that have been created in the preceding step as explained in “[Creating Subscriptions](#)” on page 98 starting with the subscriptions for the update channels. Remember that subscriptions for pool channels are optional while the subscriptions for update channels are mandatory:

When a subscription is complete execute it. By default you can only have two subscriptions active in parallel. This can be increased to a maximum of 5 under **> Configuration > Infrastructure Management > Subscription Settings**.

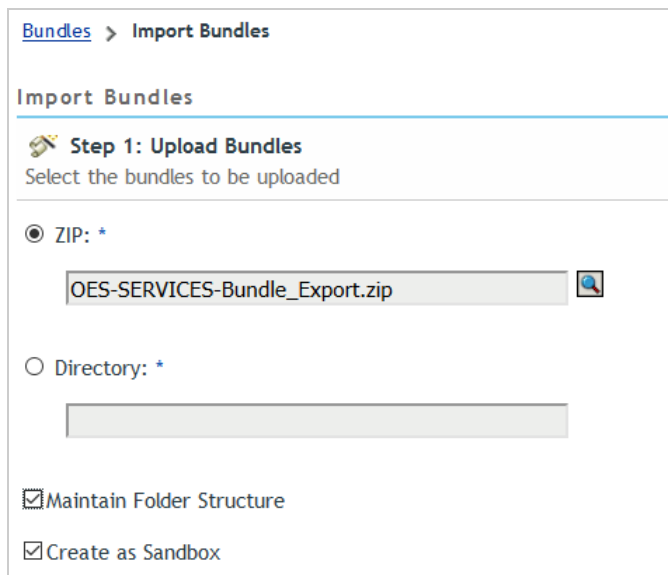
Occasionally, it can happen that a subscription will abort with an error message about a checksum mismatch. If this happens just start the replication again. Depending on your WAN connectivity you may even have to start the subscriptions repeatedly until they are complete.

Once a subscription for an update channel is complete create a Frozen Patch Level for it as explained in “[Managing Pool Bundles, Update Bundles and Update Bundle Groups](#)” on page 102. Add the FPL to the bundle groups you want to use and create the YUM service for the bundles group.

When all required update bundles have been replicated and frozen it is time to take care of the configuration bundles. Copy the Folder `ZCM_Bundles` that has been created when extracting the archive to the workstation from where you are managing your ZCM zone.

In the ZCC go to **Bundles** and select **> New > Import Bundle**. This will require the ZCC helper to be installed on your workstation. The ZCC will discover if this should not be the case. Follow the on screen instructions to install the helper.

Figure 10-1 Importing A Bundle into the ZCM zone




Bundles > **Import Bundles**

Import Bundles

Step 1: Upload Bundles
Select the bundles to be uploaded

ZIP: *



Directory: *

Maintain Folder Structure

Create as Sandbox

Check **Maintain Folder Structure** before you browse for any of the zip files in the copied folder. For example, to import all configuration bundles for OES services select `OES-SERVICES_Bundle_Export.zip`, select **Upload** and once the upload is completed verify that **Maintain Folder Structure** is checked before you select **Next**.

When the bundles contained in the archive are listed select **Next** again and when the import summary is displayed select **Finish**.

If you have more bundles to import select **Import Another Bundle** and repeat this process until all zip files have been imported into your ZCM zone. Again, make sure to check **Maintain Folder Structure** before you select **Next**. Once all the bundles have been imported select **Return to Folder**.

The bundles contained in the zip file should have been imported into your ZCM zone. In the above example you will find them in the folder `/Bundles/Linux/CUSTOM/OES-SERVICES`.

The final step is to assign the bundles you just imported to the appropriate bundle groups. Go back to your ZCM primary server and change into the directory `zman_bundle_assign`.

If you have not made any changes to `zman_params.cfg` you can simply execute the command `zman bezzman_structure_create.in_cif`. This will make all the required membership assignments for the imported configuration bundles.

In case of any changes to the parameter file you first need to execute `zman_bundle_assign.sh` and process the resulting file `zman_bundle_assign.in` with `zman bez`.

Finally you need to publish every bundle that you want to become effective on your managed Linux systems.

To update your ZCM zone you can first use the script `zman_structure_create.sh`. The script has been greatly enhanced so that it not only can be used to configure a ZENworks Management Zone from scratch but to add additional releases to an existing CIF build environment. It now supports the command line switches explained in [Table 10-1 on page 139](#).

Table 10-1 *Command Line Switches for `zman_structure_create.sh`*

Element	Description
<code>--parameter -p <parameter file></code>	file containing the parameters that describe the desired ZCM zone structure default: <code>../zman_params.cfg</code>
<code>--output -o <output file></code>	file that will contain the zman commands that will be used to create the desired ZCM zone structure default: <code>zman_structure_create.in</code>
<code>--assign -a</code>	Assign Update Bundle groups to Update Device groups. This is only required when using <code>*zac*</code> to deploy updates which does not work with rpm 4.13 and later. Not recommended when using <code>btrfs</code> . possible values: <code>[y Y yes Yes YES]</code> default: <ul style="list-style-type: none"> ◆ Update Device groups will not be created ◆ Update Bundle groups will not be assigned to Update Device groups Registration Keys for Update Device groups will not be created
<code>--pool -P</code>	Creation of structures to include pool channels possible values: <code>[y Y yes Yes YES]</code> default: pool channels will NOT be included
<code>--debug -d</code>	optional: enable debug

Along with this script we provide two parameter files and two `zman` command files:

- ◆ `zman_params.cfg_cif`

This parameter file is intended to create the `zman` command file to create all required ZCM structures from scratch using default names.

Pool channels and ZCM Update Device Groups are not included.

→ `zman_structure_create.in_cif`

- ◆ `zman_params_add_SLES15SP4_OES2023.cfg_cif`

This parameter file is intended to add the structures to support SLES15SP4 and OES2023 to an existing CIF ZENworks Management Zone using default names.

→ `zman_structure_create.in_cif.add_SLES15SP4_OES2023`

Carefully inspect the error messages that occur when processing this file with the `zman bex` command. Errors for objects that already exists in your ZENworks Management Zone are expected. All other errors need to be researched and resolved!

Please note that additional variables have been added to these parameter files. If you are using our default names you can just use the parameter files and `zman` command files provided in the archive. If you prefer other names make sure to add your values to all variables in the new command files.

There are also a number of changes with respect to the ZCM configuration bundles we provide. The following bundles are new:

- ◆ `SLES-SERVICES > SLESALL_MFC-DHCP-BOND0`
- ◆ `SLES-SERVICES > SLESALL_MFC-DISABLE-ZISLNX-SERVICE`

The following bundles have been updated:

- ◆ `iMan-BASECONFIG > IMAN_NOV-change-config-xml`
- ◆ `OES-SERVICES > OESALL_NOV-BASHRC`
- ◆ `SLES-SERVICES > SLESALL_NOV-NTP-SYSCONFIG`

Finally, ZCM configuration bundle `OES2018_MFC-DISABLE-CIS-AGENTS` in folder `OES-SERVICES` needs to be replaced with bundle `OES2018_MFC-DISABLE-OES-AGENTS`.

The script `zman_bundle_assign.sh` has been modified to accommodate the changes listed above.

The easiest way to incorporate the new ZCM configuration bundles is to first rename the existing bundles that will be replaced or deleted. Then you need to import the three new bundles and the three updated bundles to the correct folders.

Processing the `zman` command file `zman_bundle_assign.in_cif` (or a version of this file created with your names) with `zman bex` will make the new bundles member of the correct bundle groups.

Obviously this will result in an error message whenever a bundle is already a member of a bundle group. These errors are expected. All other errors need to be researched and resolved. Finally the old bundles need to be deleted from the ZENworks Management Zone.

11 Building Your Own Preboot Execution Environment

The first step in creating your own AutoYaST preboot execution environment is to decide whether you will use a ZCM Primary server or your AutoYaST server as TFTP server.

You then need to configure the chosen server as explained in [““Network Boot Environment” on page 30.](#)

PXE Boot Using a ZCM Primary Server

If you want to use a ZCM Primary server you first need to copy the kernel and initrd of the installation systems that CIF shall support to the directory `/srv/tftp/kernel`. Basically this directory is an exact copy of the directory `/data/boot_cd_build/kernel` on your AutoYaST Server (see [Table 3-1, “Directory Structure of AutoYaST Boot Medium,” on page 22](#)).

If you also want to support PXE for UEFI systems some additional preparation is required (see [“PXE Boot for UEFI Systems Using a ZCM Primary Server” on page 34](#)).

Once all files are copied and linked copy the archive `PXE_ZCM-cif_yyyymmdd.tgz` to `/srv` on the selected ZCM Primary server and extract it.

If you have not been using PXE on this server simply rename `/srv/tftp/CIF-pxemenu.txt` and `/srv/tftp/efi/x86_64/CIF_pxemenu.txt` to `pxemenu.txt`. This will just integrate the CIF PXE configuration into the default configuration.

If you have customized the PXE configuration on the selected server you will need to extract the relevant sections of the two files and integrate them into your PXE configuration (see [“Network Boot Environment” on page 30](#) for details).

NOTE: `pxemenu.txt` has a maximum size of 4.096 bytes. If your file is only a single byte larger you will not see any boot menu.

In all configuration files (`/srv/tftp/CIF*.cfg`, `/srv/tftp/efi/CIF*.cfg`) you need to replace `<IP of your AutoYaST server>`, `<IP of your ISO server>`, and `<IP of your gateway>` with the IP addresses of your systems.

Verify that the ownership of all files that you have copied or modified is set to `zenworks:zenworks`.

PXE Boot for Using the AutoYaST Server

To use your AutoYaST server for PXE you first have to install the required packages and to copy the required files to `/srv/tftp` as described in [“PXE Boot for BIOS Systems Using the AutoYaST Server” on page 35](#).

Supporting PXE for UEFI systems with your AutoYaST server requires some additional preparation (see [“PXE Boot for UEFI Systems Using the AutoYaST Server” on page 38](#)).

Once these preparatory steps are completed you need to copy the archive `PXE_AutoYaST-cif_yyyymmdd.tgz` to `/srv` on the AutoYaST server and extract it.

Finally you need to replace “<IP of your AutoYaST server>”, “<IP of your ISO server>”, and “<IP of your gateway>” with the IP addresses of your systems in all configuration files (`/srv/tftpboot/EFI/CIF*.cfg`, `/srv/tftpboot/net/pxelinux.cfg/CIF*.cfg`).

Updating your PXE environment is a fairly simple process. Just remove the boot loader configuration files, the kernel and *initrd* for the releases that you do no longer want to support.

Then add the new bootloader configuration files from the `PXE_ZCM-cif_yyyymmdd.tgz` or the `PXE_AutoYaST-cif_yyyymmdd.tgz` archive depending on your platform of choice and add the IP address of your AutoYaST server and your gateway as explained above.

Finally add the kernel and *initrd* for the new releases to the kernel directory.