

Creating a Custom Oracle® Solaris 11 Installation Image

Copyright © 2008, 2011, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle America, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Ce logiciel et la documentation qui l'accompagne sont protégés par les lois sur la propriété intellectuelle. Ils sont concédés sous licence et soumis à des restrictions d'utilisation et de divulgation. Sauf disposition de votre contrat de licence ou de la loi, vous ne pouvez pas copier, reproduire, traduire, diffuser, modifier, breveter, transmettre, distribuer, exposer, exécuter, publier ou afficher le logiciel, même partiellement, sous quelque forme et par quelque procédé que ce soit. Par ailleurs, il est interdit de procéder à toute ingénierie inverse du logiciel, de le désassembler ou de le décompiler, excepté à des fins d'interopérabilité avec des logiciels tiers ou tel que prescrit par la loi.

Les informations fournies dans ce document sont susceptibles de modification sans préavis. Par ailleurs, Oracle Corporation ne garantit pas qu'elles soient exemptes d'erreurs et vous invite, le cas échéant, à lui en faire part par écrit.

Si ce logiciel, ou la documentation qui l'accompagne, est concédé sous licence au Gouvernement des Etats-Unis, ou à toute entité qui délivre la licence de ce logiciel ou l'utilise pour le compte du Gouvernement des Etats-Unis, la notice suivante s'applique :

U.S. GOVERNMENT RIGHTS. Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle America, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

Ce logiciel ou matériel a été développé pour un usage général dans le cadre d'applications de gestion des informations. Ce logiciel ou matériel n'est destiné à être utilisé dans des applications à risque, notamment dans des applications pouvant causer des dommages corporels. Si vous utilisez ce logiciel ou matériel dans le cadre d'applications dangereuses, il est de votre responsabilité de prendre toutes les mesures de secours, de sauvegarde, de redondance et autres mesures nécessaires à son utilisation dans des conditions optimales de sécurité. Oracle Corporation et ses affiliés déclinent toute responsabilité quant aux dommages causés par l'utilisation de ce logiciel ou matériel pour ce type d'applications.

Oracle et Java sont des marques déposées d'Oracle Corporation et/ou de ses affiliés. Tout autre nom mentionné peut correspondre à des marques appartenant à d'autres propriétaires qu'Oracle.

AMD, Opteron, le logo AMD et le logo AMD Opteron sont des marques ou des marques déposées d'Advanced Micro Devices. Intel et Intel Xeon sont des marques ou des marques déposées d'Intel Corporation. Toutes les marques SPARC sont utilisées sous licence et sont des marques ou des marques déposées de SPARC International, Inc. UNIX est une marque déposée concédée sous licence par X/Open Company, Ltd.

Contents

Preface	5
1 Introduction to Creating a Custom Installation Image	9
About the Distribution Constructor	9
Oracle Solaris Image Types	10
Image Creation Process	11
SPARC and x86 Archive Differences	11
2 Design a Custom Installation Image	13
System Requirements for Building Images	13
Customizing Images	14
Sample Manifests	14
▼ How to Create and Build a Custom Image	15
Modifying the Manifest Content	15
Creating and Using Custom Scripts	22
3 Building an Image	25
distro_const Command	25
▼ How to Build an Image in One Step	26
▼ How to Build an Image in Stages	26

Preface

Creating a Custom Oracle Solaris 11 Installation Image provides instructions for using the Oracle Solaris Distribution Constructor (DC) tool to build custom Oracle Solaris installation images.

Related Information

Installing Oracle Solaris 11 Systems provides instructions for installing and configuring the Oracle Solaris operating system (OS) using any of the following methods:

- A LiveCD image
- An interactive text installer
- The Oracle Solaris Automated Installer (AI) feature
- The Oracle Solaris SCI Tool interactive system configuration tool
- The `sysconfig(1M)` command line system configuration tool

Creating and Administering Oracle Solaris 11 Boot Environments describes how to manage multiple boot environments on your Oracle Solaris system, including non-global zones.

Chapter 5, “Managing Services (Overview),” in *Oracle Solaris Administration: Common Tasks* describes the Oracle Solaris Service Management Facility (SMF) feature. You can use SMF profiles to configure your system.

The `pkg(5)` man page describes the Oracle Solaris Image Packaging System (IPS) feature, which enables you to store and retrieve software packages for installation. The `pkg(1)` man page explains how to install IPS packages.

See the Oracle Solaris 11 System Administration documentation for more information about how to administer Oracle Solaris 11 systems.

Transitioning From Oracle Solaris 10 JumpStart to Oracle Solaris 11 Automated Installer provides information to help you migrate from JumpStart to AI, both of which are automated installation features of Oracle Solaris.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Typographic Conventions

The following table describes the typographic conventions that are used in this book.

TABLE P-1 Typographic Conventions

Typeface	Meaning	Example
AaBbCc123	The names of commands, files, and directories, and onscreen computer output	Edit your <code>.login</code> file. Use <code>ls -a</code> to list all files. <code>machine_name% you have mail.</code>
AaBbCc123	What you type, contrasted with onscreen computer output	<code>machine_name% su</code> Password:
<i>aabbcc123</i>	Placeholder: replace with a real name or value	The command to remove a file is <i>rm filename</i> .
<i>AaBbCc123</i>	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the <i>User's Guide</i> . <i>A cache</i> is a copy that is stored locally. Do <i>not</i> save the file. Note: Some emphasized items appear bold online.

Shell Prompts in Command Examples

The following table shows the default UNIX system prompt and superuser prompt for shells that are included in the Oracle Solaris OS. Note that the default system prompt that is displayed in command examples varies, depending on the Oracle Solaris release.

TABLE P-2 Shell Prompts

Shell	Prompt
Bash shell, Korn shell, and Bourne shell	\$

TABLE P-2 Shell Prompts (Continued)

Shell	Prompt
Bash shell, Korn shell, and Bourne shell for superuser	#
C shell	machine_name%
C shell for superuser	machine_name#

Introduction to Creating a Custom Installation Image

System administrators and application developers can use the distribution constructor tool to build custom Oracle Solaris installation images.

- If you have not created custom installation images before, read [“About the Distribution Constructor” on page 9](#).
- If you are ready to build custom images, go to [“System Requirements for Building Images” on page 13](#).

About the Distribution Constructor

The distribution constructor is a command-line tool for building preconfigured Oracle Solaris images. The tool takes an XML manifest file as input and builds an image that is based on the parameters specified in the manifest file.

The distribution constructor can build an ISO image, which is an archive file, also known as a disc image, of an optical disc in a format defined by the International Organization for Standardization (ISO). You can also create a USB image based on a generated ISO image. Unlike ISO images, however, a USB image can only be created and used on x86 systems.

Note the following:

- Depending on the image configuration, ISO or USB images can be bootable.
- Both ISO images and USB images can be installed on a system or run in a live media environment.
- An ISO image can be burned to a CD or DVD.
- A USB image can be copied to a flash drive.
- Both ISO images and USB images can be posted on the Internet.

The distribution constructor creates a USB image that could work in various types of flash memory devices, if those devices have driver support provided by the Oracle Solaris release. The

usbcopy utility must be used to copy the USB image into a USB flash drive. This usbcopy utility is available in the `distribution-creator` package.

Oracle Solaris Image Types

You can use the distribution constructor to create the following types of Oracle Solaris images.

- **Oracle Solaris x86 LiveCD** – You can create an x86 ISO image that is comparable to the LiveCD image that's distributed as an Oracle Solaris release. You can also modify the content of this ISO image by adding or removing packages. You can revise the default settings for the resulting booted environment to create a custom ISO image or USB image.

Note – Depending on the size of the packages included in the LiveCD image, the LiveCD image might instead be a LiveDVD image.

For more information about LiveCD installations, see [Chapter 3, “Using the LiveCD,” in *Installing Oracle Solaris 11 Systems*](#).

- **Oracle Solaris x86 or SPARC Text Installation Image** – You can create a SPARC or x86 ISO image that can be used to perform a text installation of the Oracle Solaris operating system. The text installer can be used on systems that do not need graphics cards.

Note – A text installation does *not* install all of the software packages that are included when installing from the LiveCD. For example, the text installer does not install a desktop. After a text installation, you can add additional packages, such as the `solaris-desktop` package.

For more information about text installations, see [Chapter 4, “Using the Text Installer,” in *Installing Oracle Solaris 11 Systems*](#).

- **x86 or SPARC ISO Image for Automated Installations** – The Oracle Solaris release includes the automated installer tool. The automated installer (AI) is used to automate the installation of the Oracle Solaris OS on one or more SPARC and x86 systems over a network. The installations can differ in architecture, packages installed, disk capacity, and other parameters. The automated installer uses a SPARC or x86 AI ISO image to install the Oracle Solaris OS to client systems. You can use the distribution constructor to create a SPARC AI ISO image that can be used to install the Oracle Solaris OS on SPARC clients, or to create an x86 AI ISO image that can be used to install the Oracle Solaris OS on x86 clients.

For information about using the automated installer, see [Part III, “Installing Using an Install Server,” in *Installing Oracle Solaris 11 Systems*](#).

Image Creation Process

The distribution constructor creates images based on settings specified in XML files, called *manifest files*. The manifest files contain specifications for the contents and parameters of the ISO images that you create using the distribution constructor. The distribution constructor contains sample manifests that can be used to create a custom LiveCD, an x86 or SPARC AI ISO image, or an x86 or SPARC text installation image.

All the fields in each manifest file provide preset, default values that will create the type of image you need. You can edit fields in the manifest file to further customize the resulting image. For example, you can edit the target element in the manifest to specify a different location for the build area where the image can be constructed. In addition, you can check the publisher that's specified and ensure that the system you are using can contact that publisher to download the packages needed to build the image. If necessary, you can edit the software name element to specify a different publisher and repository location. For instructions, see [“Customizing Images” on page 14](#).

You can also create *custom scripts* to modify your installation image. Then, you can add checkpoints to the manifest file to run these custom scripts. For further information, see [“Creating and Using Custom Scripts” on page 22](#).

The distribution constructor package also includes a command-line utility, the `distro_const` command that interprets the manifest specifications and builds the image. After you have finished editing the image blueprint in a manifest file to suit your requirements, you run the `distro_const` command to build your image. For further information, see [Chapter 3, “Building an Image.”](#)

You can use the options provided in the `distro_const` command to stop and restart the build process at various stages in the image-generation process, in order to check and debug the image that is being built. This process of stopping and restarting during the build process is called *checkpointing*. Checkpointing is optional. Default checkpoints are specified in each manifest file. For instructions, see [“How to Build an Image in Stages” on page 26](#), or see the `distro_const(1M)` man page.

SPARC and x86 Archive Differences

The root archive for x86 images differs from the root archive for SPARC images. The whole root archive, or `boot_archive`, for x86 images is a UFS file system, compressed by using Lzma. The SPARC platform does not support the compression of the whole root archive in this way. Instead, SPARC root archives use DCFS, which compresses each file individually. These individually compressed files might require specific handling in the manifest. For instructions, see the `<boot_archive_contents>` field in the `dc_manifest(4)` man page.

Design a Custom Installation Image

Review the system requirements and design a custom installation image as described in this chapter.

System Requirements for Building Images

In order to use the distribution constructor, you must have the following set up on your system.

TABLE 2-1 System Requirements

Requirement	Description
Disk space	The recommended minimum size for your distribution constructor work space is 8 Gbytes. Confirm that you have sufficient space on your system to use the distribution constructor.
Oracle Solaris release	<p>You must have the Oracle Solaris operating system (OS) installed on your system. Note the following considerations.</p> <ul style="list-style-type: none"> ■ Your installed system must have network access. The distribution constructor accesses Image Packaging System (IPS) repositories that are available on the network to retrieve packages for the ISO image. You must have network access to the repositories that you specify in the manifest file. ■ When using the distribution constructor, you can only create SPARC images on a SPARC system and x86 images on an x86 system. ■ The Oracle Solaris release version on your system must be the same as the release version of the images that you use with the distribution constructor. <p>Note – You must become the root role to run the distribution constructor.</p>
Required packages	The <code>distribution-creator</code> package, which contains the distribution constructor application.

Customizing Images

The distribution constructor creates images based on settings specified in XML files, called *manifest files*. The manifest files contain specifications for the contents and parameters for the ISO images that you create using the distribution constructor. The `distribution-creator` package provides sample manifests that can be used to create a custom LiveCD, an x86 or SPARC AI ISO image, or an x86 or SPARC text installation image.

The elements in each manifest file provide preset, default values that will create the type of ISO image you need. You can manually edit these preset elements in a manifest file to customize the resulting image. In addition, you can create custom scripts to further modify your image. Then, reference the new scripts in the manifest file.

Sample Manifests

The `distribution-creator` package provides the following sample manifest files.

TABLE 2-2 Sample Manifests

Manifest Type	Manifest Location	Description
x86 LiveCD ISO image	<code>/usr/share/distro_const/ dc_livecd.xml</code>	Used to create an ISO image comparable to the Oracle Solaris LiveCD
x86 text installation image	<code>/usr/share/distro_const/ dc_text_x86.xml</code>	Used to create an ISO image that can be used to perform a text installation of the x86 Oracle Solaris operating system
SPARC text installation image	<code>/usr/share/distro_const/ dc_text_sparc.xml</code>	Used to create an ISO image that can be used to perform a text installation of the SPARC Oracle Solaris operating system
x86 AI ISO image	<code>/usr/share/distro_const/ dc_ai_x86.xml</code>	Used to create an x86 AI ISO image for automated installations of the Oracle Solaris OS to x86 clients
SPARC AI ISO image	<code>/usr/share/distro_const/ dc_ai_sparc.xml</code>	Used to create a SPARC AI ISO image for automated installations of the Oracle Solaris OS to SPARC clients

▼ How to Create and Build a Custom Image

- 1 **Download the `distribution-creator` package, which contains the distribution creator application and the sample manifests.**

You can use the Package Manager tool to install the required package. The Package Manager is available on the menu bar on the desktop of the Oracle Solaris operating system. On the menu bar, go to System>Administration>Package Manager.

Alternately, use IPS commands such as the following to install this package:

```
# pkg install distribution-creator
```

- 2 **Copy one of the sample manifests and create a custom manifest file with a new file name.**

You will reference the manifest file by name when you use the `distro_const` command to create an image.

Note – Always back up the original manifest file and the default scripts before copying them.

- 3 **Edit the manifest elements to suit your needs.**

For example, you can edit the target element in the manifest to specify a different location of the build area where the image can be constructed. And, you can check the publisher to ensure your system can contact that publisher to download the packages needed to build the image. If necessary, you can edit the software name element to specify a different publisher and repository location.

For information, see “[Modifying the Manifest Content](#)” on page 15 and the `dc_manifest(4)` man page.

- 4 **(Optional) Create custom scripts to further modify the image.**

If you do create new scripts, update the script references in the execution section of the manifest file.

For instructions, see “[Creating and Using Custom Scripts](#)” on page 22.

- 5 **Run the `distro_const` utility to create an image.**

For instructions, see [Chapter 3, “Building an Image.”](#)

Modifying the Manifest Content

All the fields in each manifest file provide preset, default values that will create the type of ISO image you need. You can manually edit these preset fields in a manifest file to further customize the resulting image.

Depending on which sample manifest you select, the primary elements are as follows.

TABLE 2-3 Manifest Elements

Element	Description
<code><distro name="Oracle_Solaris_Text_X86" add_timestamp="false"></code>	Specifies image name with optional timestamp
<code><boot_mods></code>	Specifies GRUB menu modifications for image
<code><target></code>	Defines ZFS build dataset where image is built
<code><software name="transfer-ips-install" type="IPS"></code>	Specifies source for software packages to be installed
<code><software_data action="install"></code>	Lists packages to be installed
<code><software_data action="uninstall"></code>	Lists packages to be uninstalled
<code><software name="set-ips-attributes"></code>	Sets different attributes for IPS after the installation has finished.
<code><software name="ba-init"></code>	Specifies boot archive contents Caution – Modify with care. If the boot archive is incorrect, the installed system will fail to boot.
<code><execution stop_on_error="true"></code> <code><checkpoint name="transfer-ips-install"/></code>	Lists build checkpoints
<code><configuration name="pre-pkg-img-mod" type="sysconf" source="/etc/svc/profile/generic_limited_net.xml"></code>	Specifies SMF services to be applied to media during build Caution – Rarely modify.

Provide Image Title

Use the following element to provide a custom or default name for the image you are going to build.

```
<distro name="Oracle_Solaris_Text_X86" add_timestamp="false">
```

If you intend to perform a series of builds of an image and retain the incremental images, you can change the timestamp variable to “true”, and a timestamp will be automatically appended to the name for each image.

If you need to specify an HTTP proxy, uncomment the `distro` name element that includes the proxy variable, and enter the proxy location.

Modify Boot Menu

This boot menu element specifies boot menu modifications to be applied to the image.

In the following example, a specialized boot menu with the title, “boot1”, will be applied to the image. The timeout attribute specifies time before the default boot entry is automatically activated.

```
<boot_mods title="boot1" timeout="5">
```

Within the boot menu element, you can add individual boot menu entries by adding a new `boot_entry` element for each new entry. Entries are added sequentially to the boot menu in the order based on the `insert_at` attribute value of “start” or “end” for each boot entry.

Note – Add new entries before the existing “with magnifier” entry.

See the following example of an individual `boot_entry` element.

```
<boot_entry>
  <title_suffix>with screen reader</title_suffix>
  <kernel_args>-B assistive_tech=reader</kernel_args>
</boot_entry>
```

For detailed information, see the `dc_manifest(4)` man page.

Specify Build Area

You can customize the `target` element. This element defines the ZFS build dataset to be used for the build. This dataset is the area where the image will be created. You must enter a valid dataset location. You should check the default build area to ensure the build will not destroy content you need to keep on your system. Modify the build area if necessary.

Note – The filesystem name should not include the name of the zpool.

See the following example.

```
<target>
  <logical>
    <zpool action="use_existing" name="rpool">
      <dataset>
        <filesystem name="dc/sample-dataset-location"
          action="preserve"/>
      </dataset>
    </zpool>
  </logical>
</target>
```

Specify Publisher

The following element specifies a publisher where the distribution constructor can get packages to download and use to build the image.

```
<software name="transfer-ips-install">
```

In the source element in this section, edit the publisher name and origin name elements to specify which publisher to use and where the package repository is located. Multiple publishers can be listed. When the distribution constructor attempts to locate packages to install, publishers are searched in the order they are listed here.

If mirrors for a publisher need to be specified, uncomment and edit the mirror name element.

See the following example.

```
<source>
  <publisher name="publisher1">
    <origin name="http://example.oracle.com/primary-pub"/>
    <mirror name="mirror.example.com"/>
  </publisher>
  <publisher name="publisher2">
    <origin name="http://example2.com/dev/solaris"></origin>
  </publisher>
  <publisher name="publisher3.org">
    <origin name="http://example3.com/dev"></origin>
  </publisher>
</source>
```

For further information about using publishers, see [Adding and Updating Oracle Solaris 11 Software Packages](#).

List Packages to Install

The `software_data` element with the `install` attribute lists the set of packages to be installed in order to build a particular type of image, depending on which manifest you are using. For example, the `dc_livecd.xml` manifest lists the packages needed to build a LiveCD image. Each name tag lists one package name or the name of a group package that contains many packages.

```
<software_data action="install">
  <name>pkg:/group/system/solaris-desktop</name>
  <name>pkg:/system/install/gui-install</name>
  <name>pkg:/system/install/media/internal</name>
</software_data>
```

If you have packages that you want to add to the image, append the package names by adding a name tag for each package.

By default, the most current package version available in the specified repository is installed. If another version is required, append the version number to the package reference using the following format:

```
<name>pkg:/group/system/solaris-desktop@0.5.11-0.build#</name>
```

Note – Packages with a particular version specified might not be installed if there are other packages with a conflicting version being installed. For further information, see the `pkg(5)` man page.

EXAMPLE 2-1 Adding Packages and Additional Publishers

In this example, a second publisher, `mypublisher`, is specified. And, additional packages, `mypackage1` and `mypackage2`, are specified.

During the build process, the publishers are checked in the order they are listed. If packages are not found at the first publisher, the next publisher is searched for the specified packages.

```
<software name="transfer-ips-install" type="IPS">
  <destination>
    <xi:include xmlns:xi="http://www.w3.org/2003/XInclude"
      href="/usr/share/distro_const/lang_facets.xml"/>
  </destination>
  <source>
    <publisher name="solaris">
      <origin name="http://pkg.oracle.com/solaris/release"/>
    </publisher>
    <publisher name="mypublisher">
      <origin name="http://mypublisher.company.com"/>
    </publisher>
  </source>
  <software_data action="install">
    <name>pkg:/group/system/solaris-large-server</name>
    <name>pkg:/system/install/text-install</name>
    <name>pkg:/system/install/media/internal</name>
    <name>pkg:/mypackage1</name>
    <name>pkg:/mypackage2</name>
  </software_data>
</software>
```

List Packages to Uninstall

The `software_data` element with the `uninstall` attribute can be used to uninstall an individual package or to uninstall a group package definition.

In the following example, `solaris-desktop` is the name of a group package that contains numerous individual packages.

```
<software_data action="uninstall">
  <name>pkg:/group/system/solaris-desktop</name>
</software_data>
```

You could uninstall a group package. Uninstalling a group package means that only the group definition is actually uninstalled. The individual packages that were previously installed as part of that group are not uninstalled. However, you can uninstall those individual packages without uninstalling the group package. Retaining the group package can be useful for ongoing reference. You can also use the `name` tag to uninstall an individual package. Append additional packages to be uninstalled at the end of the `uninstall` section.

Specify Publisher for Installed System

The following element affects a system after that system has been installed with the image created using the distribution constructor.

```
<software name="set-ips-attributes">
```

Provide the publisher name and optional mirror name tags to specify where the installed system can access additional packages to download and install.

You can also set IPS attributes in this element. See the pkg(1) man page IPS property information.

Setup Build Checkpoints

The execution element in the manifest lists a series of checkpoints that are executed during the image construction process. Checkpoints are executed in the order they are listed in this section. The default checkpoints needed to build the default installation image are included in each manifest.

Each checkpoint name tag includes the mod-path attribute which specifies where the checkpoint script is located.

Some of the default checkpoint tags include arguments with default values provided. The following checkpoint example from the dc_ai_sparc.xml sample manifest creates the boot archive for the image build and points to a script that will accomplish that task. The example checkpoint, also, includes argument fields with specific values provided for each argument.

```
<checkpoint name="ba-arch"
  desc="Boot Archive Archival"
  mod_path="solaris_install/distro_const/checkpoints/
  boot_archive_archive"
  checkpoint_class="BootArchiveArchive">
  <kwargs>
    <arg name="size_pad">0</arg>
    <arg name="bytes_per_inode">0</arg>
    <arglist name="uncompressed_files">
      <argitem>etc/svc/repository.db</argitem>
      <argitem>etc/name_to_major</argitem>
      <argitem>etc/minor_perm</argitem>
      <argitem>etc/driver_aliases</argitem>
      <argitem>etc/driver_classes</argitem>
      <argitem>etc/path_to_inst</argitem>
      <argitem>etc/default/init</argitem>
      <argitem>etc/nsswitch.conf</argitem>
      <argitem>etc/passwd</argitem>
      <argitem>etc/shadow</argitem>
      <argitem>etc/inet/hosts</argitem>
    </arglist>
  </kwargs>
</checkpoint>
```

As shown in this example, the `<kwargs>` element contains key word arguments that need to be passed into the checkpoint during the build. Within the `<kwargs>` element are `<arg name>` elements that can be used to specify individual key words to be passed into the checkpoint. And, the `<arg list>` element contains a list of multiple `<arg item>` values to be passed into the checkpoint. This example includes a list of uncompressed files in the `<arg list>` element.

Each `<kwargs>` list item is enclosed in double-quotes. When no double-quotes are used, or if one set of double-quotes encloses the entire string, the entire string including spaces and new lines is interpreted as one argument. Do not use commas between arguments.

If you create a custom script to be used during the building of an image, you must add a checkpoint element pointing to the script location. The checkpoint for a custom script needs only an `<args>` element that points to the custom script location. For further information and examples, see [“Creating and Using Custom Scripts” on page 22](#).

Use the `distro_const` command options to control pausing and restarting the build process at particular checkpoints. See [“How to Build an Image in Stages” on page 26](#).

EXAMPLE 2-2 Adding SVR4 Packages

In this example, a new checkpoint is added to the manifest. This new checkpoint lists SVR4 packages to be added to the image and their location. Then, this new checkpoint is referenced in the execution section.

First, the new checkpoint is created by adding a new `software` element. This checkpoint specifies SVR4 as the software type, where to find the packages, and where to install the packages.

In addition, the specific SVR4 packages to be installed are listed in the `software_data` element.

```
<software name=transfer-svr4-install type="SVR4">
  <destination>
    <dir path={PKG_IMAGE_PATH}/>
  </destination>
  <source>
    <dir path="/path/to/packages"/>
  </source>
  <software_data action="install">
    <name>SUNWpackage1</name>
    <name>SUNWpackage2</name>
  </software_data>
</software>
```

If included in the checkpoint, the values of `{PKG_IMAGE_PATH}` and `{BOOT_ARCHIVE}` are replaced by the `distro_const` utility with `<ZFS Dataset>/build_data/pkg_image` and `<ZFS Dataset>/build_data/boot_archive`, respectively. In this example, the SVR4 packages will be installed into `<ZFS Dataset>/build_data/pkg_image`.

Finally, the new checkpoint is referenced in the execution section.

EXAMPLE 2-2 Adding SVR4 Packages (Continued)

```
<execution stop_on_error="true">
  <checkpoint name="transfer-ips-install"
    desc="Transfer pkg contents from IPS"
    mod_path="solaris_install/transfer/ips"
    checkpoint_class="TransferIPS"/>
  <checkpoint name="set-ips-attributes"
    desc="Set post-install IPS attributes"
    mod_path="solaris_install/transfer/ips"
    checkpoint_class="TransferIPS"/>
  <checkpoint name="transfer-svr4-install"
    desc="Transfer pkg contents from SVR4 packages"
    mod_path="solaris_install/transfer/svr4"
    checkpoint_class="TransferSVR4"/>
```

Note that the software name must match the checkpoint name. In this example, both are “transfer-svr4-install.”

Creating and Using Custom Scripts

The distribution constructor enables you to specify additional scripts that can be used to make customizations based on the type of image you are building. The manifest files point to the scripts, and the scripts transform the generic image into a media-specific distribution. These scripts are referenced in the execution section of the manifest files. Any number of custom-script checkpoints may be specified.

Note – Support for scripts is limited to any unmodified, default scripts that are supplied with the application packages. If you choose to customize these scripts, back up the original scripts first.

▼ How to Create and Use a Custom Script

Before You Begin When you create your own custom scripts, note the following:

- Scripts can be Python programs, shell scripts, or binaries.
- Scripts are executed in the order that they are listed in the execution section of the manifest file.
- Standard output (stdout) and error output (stderr) of commands executed within the scripts (both shell and python modules) are captured in log files that report on the completed or attempted build.

- 1 **Create your new script.**
- 2 **Add your new scripts to your home directory or elsewhere on the system or network.** Make sure that a user assuming the root role can execute these scripts.

3 Reference the new script by adding a checkpoint in the execution section of the appropriate manifest file.

Be sure to specify the full path to your scripts. Checkpoints are executed in the order they are listed in the execution section of the manifest.

When you add a reference for a new script in the execution section of a manifest file, you must specify a checkpoint name that can be used to pause the image build before or after this script performs its task. Optionally, you can include a custom message associated with the checkpoint name. If this message is omitted, the path of the script is used as the default checkpoint message. The checkpoint message displays when the checkpoint is run during the build process.

Note – Use meaningful names for checkpoint names instead of using numbers. If new scripts are added, the new checkpoints for those new scripts will disrupt a numbered checkpoint order.

The following example checkpoint references a custom script named “my-script.”

```
<checkpoint name="my-script"
  desc="my new script"
  mod_path="solaris_install/distro_const/checkpoints/custom_script"
  checkpoint_class="CustomScript">
  <args>/tmp/myscript.sh</args>
</checkpoint>
```

4 (Optional) Specify a build parameter as part of the checkpoint as follows.

Here {PKG_IMAGE_PATH} is specified as the build parameter in the arguments section.

```
<checkpoint name="my-script"
  desc="my new script"
  mod_path="solaris_install/distro_const/checkpoints/my_script"
  checkpoint_class="CustomScript">
  <args>/tmp/myscript.sh {PKG_IMAGE_PATH}</args>
</checkpoint>
```

If included in the checkpoint, the values of {PKG_IMAGE_PATH} and {BOOT_ARCHIVE} are replaced by the distro_const utility with <ZFS Dataset>/build_data/pkg_image and <ZFS Dataset>/build_data/boot_archive, respectively.

5 Build the image.

You can build the image in one step. Or, to check the status of the build, you can stop and restart the build at various checkpoints.

For instructions, see [Chapter 3, “Building an Image.”](#)

6 (Optional) After the build is complete, you can view a log file reporting on the build process.

The build output displays the location of log files.

Building an Image

After you have set up the manifest file that you plan to use and, if desired, customized the finalizer scripts, you are ready to build an image by running the `distro_const` command.

You can use the `distro_const` command to build an image in either of the following:

- In one step
- Pausing and restarting the build as needed to examine the content of the image and debug the scripts during the build process

distro_const Command

The full syntax for the `distro_const` command is as follows:

Syntax: `distro_const build [-v] [-r checkpoint_name] [-p checkpoint_name] [-l] manifest`

Review the following command options.

TABLE 3-1 `distro_const` Command Options

Command Options	Description
<code>distro_const build <i>manifest</i></code>	Builds an image in one step using specified manifest file
<code>distro_const build -v</code>	Verbose mode
<code>distro_const build -l <i>manifest</i></code>	Lists all valid checkpoints at which you can pause and resume building an image
<code>distro_const build -p <i>checkpoint_name</i> <i>manifest</i></code>	Pauses building an image at a specified checkpoint
<code>distro_const build -r <i>checkpoint_name</i> <i>manifest</i></code>	Resumes building an image from a specified checkpoint

TABLE 3-1 distro_const Command Options (Continued)

Command Options	Description
<code>distro_const build -h</code>	Displays help for the command

Note – You must assume the root role to use the `distro_const` command.

▼ How to Build an Image in One Step

Before You Begin Download the `distribution-creator` package and select manifest for your image. If needed, customize the manifest and add custom scripts.

- 1 **Become the root role.**
- 2 **To run a complete build of an image without pausing, use the basic `distro_const` command without options as follows:**

```
# distro_const build manifest
```

Note – The `build` subcommand is required.

Replace *manifest* with the name of the manifest file to be used as the blueprint for your image.

For example, type the following command:

```
# distro_const build /usr/share/distro_const/dc_livecd.xml
```

- 3 **The distribution constructor pulls the needed packages for the image.**
- 4 **The distribution constructor builds the image to the specifications that you set up in the manifest file.**
- 5 **(Optional) After the build is complete, you can view a log file reporting on the build process.**
The build output displays the location of log files.

▼ How to Build an Image in Stages

You can use the options provided in the `distro_const` command to stop and restart the build process at various stages in the image-generation process, in order to check and debug your selection of files, packages, and scripts for the image that is being built. This process uses the checkpointing options that are available in the `distro_const` command as described in the following basic instructions.

- 1 **Become the root role.**
- 2 **Before you build the image, check the valid checkpoints at which you can choose to pause or resume the build.**

```
# distro_const build -l manifest.xml
```

Note – The `build` subcommand is required.

This command displays the valid checkpoints at which you can pause or resume building an image. Use the checkpoint names provided by this command as valid values for the other checkpointing command options.

For example, the following command confirms which checkpoints are available, given a manifest file named `dc_livecd.xml`.

```
# distro_const build -l /usr/share/distro_const/dc_livecd.xml
```

After the command is run, the valid checkpoints are displayed. For example, checkpoints may include the following.

Checkpoint	Resumable	Description
transfer-ips-install	X	Transfer package contents from IPS
set-ips-attributes	X	Set post-installation IPS attributes
pre-pkg-img-mod	X	Pre-package image modification
ba-init		Boot archive initialization
ba-config		Boot archive configuration
ba-arch		Boot archive archiving
grub-setup		Set up the GRUB menu
pkg-img-mod		Package image area modifications
create-iso		ISO image creation

Note – In this sample command output, an “X” in the resumable field indicates that you can restart the build from this checkpoint.

- 3 **Build the image and pause building the image at the specified checkpoint.**

```
# distro_const build -p checkpoint_name manifest
```

Note – The `build` subcommand and the `checkpoint_name`, and `manifest` fields are required.

For example, the following command starts building an image and pauses the build before `ba-arch` modifies the image area:

```
# distro_const build -p ba-arch /usr/share/distro_const/dc_livecd.xml
```

4 Resume building the image from a specified checkpoint.

```
# distro_const build -r checkpoint_name manifest
```

Note – The specified checkpoint must be either the checkpoint at which the previous build stopped executing, or an earlier checkpoint. A later checkpoint is not valid. The *checkpoint_name* and *manifest* fields and the `build` subcommand are required.

For example, the following command resumes building the image at the `ba-arch` stage.

```
# distro_const build -r ba-arch /usr/share/distro_const/dc_livecd.xml
```

Note – You can combine the pause and resume options in a `build` command.

5 (Optional) After the build is complete, you can view a log file reporting on the build process.

The build output displays the location of log files.