Guide du constructeur de distribution Oracle® Solaris 11 Express

Référence : 821–1711–10
Novembre 2010

ORACLE®

101124@25097

Table des matières

◆ ◆ ◆ CHAPITRE 1

1 Introduction au constructeur de distribution

Vous pouvez utiliser le constructeur de distribution pour créer des images Oracle Solaris personnalisées. Vous pouvez ensuite utiliser ces images pour installer le logiciel Oracle Solaris sur un ou plusieurs systèmes. Vous pouvez également utiliser le constructeur de distribution pour créer des images de machine virtuelle (VM) qui exécutent le système d'exploitation Oracle Solaris.

- Si vous n'avez jamais utilisé le constructeur de distribution, lisez l'introduction suivante.
- Si vous êtes prêt à utiliser le constructeur de distribution, reportez-vous au Chapitre 2, “Conception et création d'images Oracle Solaris”.
- Si vous souhaitez construire une machine virtuelle, reportez-vous aux instructions du Chapitre 3, “x86 : conception et création d'une machine virtuelle”.

Qu'est-ce que le constructeur de distribution ?

Le constructeur de distribution est un outil de ligne de commande permettant de créer des images Oracle Solaris préconfigurées et des images de machine virtuelle. Cet outil récupère le fichier manifest XML en entrée et génère une image ISO ou une image de machine virtuelle en fonction des paramètres spécifiés dans le fichier manifest. Une image ISO, ou image disque, est un fichier archive d'un disque optique dans un format défini par l'Organisation internationale de normalisation (ISO).

Vous pouvez également créer une image USB d'après une image ISO x86 générée. Selon la configuration d'image, ces images ISO ou USB peuvent être initialisables. Elles peuvent être installées sur un système ou exécutées dans un environnement média en direct. Une image ISO peut être gravée sur un CD ou un DVD. Une image USB peut être copiée vers un lecteur Flash.

Remarque – Le constructeur de distribution crée une image USB qui peut fonctionner avec plusieurs types de périphériques de mémoire flash, dans la mesure où la version Oracle Solaris prend en charge le pilote de périphérique. L'utilitaire `usbcopy` est le seul qui vous permet de copier l'image USB sur un disque flash USB. Il est disponible avec la version Oracle Solaris.

Quels types d'images Oracle Solaris puis-je créer ?

Vous pouvez vous servir du constructeur de distribution pour créer les types d'images Oracle Solaris suivants.

- **Image Live CD x86 Oracle Solaris** : en utilisant le constructeur de distribution, vous pouvez créer une image ISO x86, appelée image Live CD ou « Slim CD », comparable à l'image Live CD distribuée avec la version Oracle Solaris. Cette image Live CD s'initialise sur un bureau GNOME sur lequel vous pouvez explorer le système d'exploitation Oracle Solaris. Vous pouvez installer le système d'exploitation à partir de ce bureau. Vous pouvez utiliser le constructeur de distribution pour modifier le contenu de cette image ISO du Live CD en ajoutant ou en supprimant des packages. Vous pouvez également modifier les paramètres par défaut pour l'environnement initialisé de manière à créer une image ISO ou USB personnalisée. Voir Chapitre 2, “Conception et création d'images Oracle Solaris”.

  Pour plus d'informations sur l'utilisation du Live CD x86 Oracle Solaris, reportez-vous à *Démarrage d'Oracle Solaris 11 Express* .
- **Image du programme d'installation en mode texte Oracle Solaris x86 ou SPARC** : en utilisant le constructeur de distribution, vous pouvez créer une image ISO permettant de démarrer une installation en mode texte du système d'exploitation Oracle Solaris sur les ordinateurs x86 ou SPARC. Cette image du programme d'installation en mode texte peut être utilisée pour installer le système d'exploitation Oracle Solaris sur les ordinateurs qui n'ont pas de carte graphique. Le programme d'installation en mode texte n'installe pas de bureau avec le système d'exploitation. En revanche, vous pouvez ajouter des packages au système d'exploitation, y compris des packages pour un bureau GNOME, après l'installation.

  Pour plus d'informations sur l'exécution d'une installation en mode texte, reportez-vous à *Démarrage d'Oracle Solaris 11 Express* .
- **Image ISO x86 ou SPARC pour l'installation automatisée** : la version Oracle Solaris inclut l'outil d'installation automatisée. Le programme d'installation automatisée (IA) permet d'automatiser l'installation du SE Oracle Solaris sur un ou plusieurs systèmes SPARC et x86 sur un réseau. Les installations peuvent différer en termes d'architecture, de packages installés, de capacité du disque et d'autres paramètres. Le programme d'installation automatisée utilise une image ISO d'installation automatisée SPARC ou x86 pour installer le SE Oracle Solaris sur les systèmes clients. Vous pouvez vous servir du constructeur de distribution pour créer une image ISO d'installation automatisée SPARC permettant d'installer le SE Oracle Solaris sur des clients SPARC, ou pour créer une image ISO d'installation automatisée x86 permettant d'installer le SE Oracle Solaris sur des clients x86.

Pour toute information sur l'utilisation du programme d'installation automatisée, reportez-vous au *Guide du programme d'installation automatisée d'Oracle Solaris 11 Express*.

Remarque – Lorsque vous utilisez le constructeur de distribution, vous ne pouvez créer que des images SPARC sur un système SPARC. De même, vous pouvez uniquement créer des images x86 sur un système x86. Par ailleurs, la version Oracle Solaris sur votre système doit être la même que celle des images d'installation automatisée que vous utilisez avec le constructeur de distribution.

- **Machine Virtuelle Oracle Solaris x86** : vous pouvez utiliser le constructeur de distribution pour créer une machine virtuelle préinstallée avec le système d'exploitation Oracle Solaris. Voir Chapitre 3, “x86 : conception et création d'une machine virtuelle”.

Quels sont les composants du constructeur de distribution ?

Le constructeur de distribution utilise des composants de base pour concevoir et créer des images.

- “Fichiers manifest” à la page 7
- “Scripts de finalisation” à la page 8
- “Utilitaire `distro_const`” à la page 8 avec “Options de création de points de contrôle” à la page 8

Reportez-vous aux brèves descriptions de ces composants de base.

Remarque – Pour obtenir des instructions complètes sur l'utilisation du constructeur de distribution, reportez-vous au Chapitre 2, “Conception et création d'images Oracle Solaris”.

Fichiers manifest

Le constructeur de distribution crée des images en fonction des paramètres indiqués dans des fichiers XML, appelés *fichiers manifest*. Les fichiers manifest contiennent les spécifications relatives au contenu et aux paramètres des images ISO créées à l'aide du constructeur de distribution. Le constructeur de distribution contient des fichiers manifest par défaut permettant de créer un Live CD, une image du programme d'installation en mode texte x86 ou SPARC, une image ISO d'installation automatisée x86 ou SPARC, ou une image de machine virtuelle personnalisés.

Tous les champs des fichiers manifest fournissent des valeurs par défaut, prédéfinies, qui créent le type d'image dont vous avez besoin. Vous pouvez également modifier manuellement ces champs prédéfinis dans un fichier manifest afin de personnaliser davantage l'image obtenue.

Pour plus d'informations sur la personnalisation des fichiers manifest, reportez-vous à "Personnalisation de l'image en modifiant les fichiers manifest" à la page 10.

Scripts de finalisation

Le constructeur de distribution permet de spécifier les scripts qui peuvent être utilisés afin de personnaliser l'installation selon le type d'images créées. Ces scripts sont appelés *scripts de finalisation*. Les fichiers manifest désignent les scripts de finalisation, et ceux-ci transforment l'image générique en une distribution spécifique au support. Les scripts par défaut sont fournis lorsque vous installez le package du constructeur de distribution.

Pour plus d'informations sur la création et l'utilisation des scripts de finalisation, reportez-vous à "Personnalisation avancée d'une image à l'aide des scripts de finalisation" à la page 13.

Utilitaire `distro_const`

Le package du constructeur de distribution inclut également un utilitaire de ligne de commande, la commande `distro_const`, qui construit votre image pour vous. Une fois l'empreinte de l'image modifiée selon vos besoins dans un fichier manifest, exécutez la commande `distro_const` pour construire votre image.

Pour plus d'informations sur l'utilisation de la `commande distro_const`, reportez-vous à "Création d'une image" à la page 18.

Options de création de points de contrôle

Vous pouvez utiliser les options offertes par la commande `distro_const` pour arrêter et redémarrer le processus de création à différentes étapes du processus de génération d'image, afin de contrôler et de déboguer l'image en cours de création. Ce processus d'arrêt et de redémarrage au cours de la création est appelé *création de points de contrôle*.

Pour plus d'informations sur l'utilisation des points de contrôle, reportez-vous à "Création d'une image par étapes" à la page 18.

◆ ◆ ◆ CHAPITRE 2

Conception et création d'images Oracle Solaris

Pour concevoir et créer des images Oracle Solaris avec le constructeur de distribution, réalisez les quatre tâches suivantes.

Remarque – La création d'une machine virtuelle nécessite des instructions particulières, qui diffèrent des instructions générales ci-dessous. Pour créer une machine virtuelle, reportez-vous au Chapitre 3, “x86 : conception et création d'une machine virtuelle”.

---

1. Vérifiez la section “Configuration système requise” à la page 9 et installez l'application du constructeur de distribution sur votre système.
2. Facultatif : “Personnalisation de l'image en modifiant les fichiers manifest” à la page 10.
3. Facultatif : “Personnalisation avancée d'une image à l'aide des scripts de finalisation” à la page 13.
4. “Création d'une image” à la page 18.

Pour obtenir des instructions complètes, reportez-vous aux sections suivantes.

Configuration système requise

Pour pouvoir utiliser le constructeur de distribution, configurez les fonctionnalités suivantes sur votre système.

TABLEAU 2–1 Configuration système requise

Exigence	Description
Espace disque	Vérifiez que vous disposez de suffisamment d'espace sur votre système pour utiliser le constructeur de distribution. La taille minimale recommandée pour l'espace de travail du constructeur de distribution est de 8 Go.

TABLEAU 2–1 Configuration système requise *(Suite)*

Exigence	Description
Version d'Oracle Solaris	Un système d'exploitation (SE) Oracle Solaris SPARC ou x86 doit être installé sur votre système. Veuillez prendre en compte les points importants suivants : ■ Votre système Oracle Solaris installé doit avoir accès au réseau. Le constructeur de distribution accède aux référentiels IPS (Image Packaging System, système d'empaquetage d'image) disponibles sur le réseau pour récupérer les packages pour l'image ISO. Vous devez disposer d'un accès réseau aux référentiels spécifiés dans le fichier manifest. ■ Lorsque vous utilisez le constructeur de distribution, vous ne pouvez créer que des images SPARC sur un système SPARC. De même, vous pouvez uniquement créer des images x86 sur un système x86. Par ailleurs, la version Oracle Solaris sur votre système doit être la même que celle des images d'installation automatisée que vous utilisez avec le constructeur de distribution. **Remarque –** Pour exécuter le constructeur de distribution sur votre système, vous devez assumer le rôle root en exécutant la commande `su -`.
Packages requis	Installez le package `distribution-constructor`, qui contient l'application du constructeur de distribution, sur votre système. Vous pouvez utiliser le gestionnaire de packages pour installer le package requis. Le gestionnaire de packages se présente sous la forme d'une icône sur le bureau du système d'exploitation Oracle Solaris et dans la barre de menu sur le bureau. Dans la barre de menu, cliquez sur Système>Administration>Gestionnaire de packages. Vous pouvez également utiliser les commandes IPS (la suivante, par exemple) pour installer ce package : `# pkg install distribution-constructor`

Personnalisation de l'image en modifiant les fichiers manifest

Le constructeur de distribution crée des images en fonction des paramètres indiqués dans des fichiers XML, appelés *fichiers manifest*. Les fichiers manifest contiennent les spécifications relatives au contenu et aux paramètres des images ISO créées à l'aide du constructeur de distribution. Le constructeur de distribution contient des fichiers manifest par défaut permettant de créer un Live CD, une image du programme d'installation en mode texte x86 ou SPARC, une image ISO d'installation automatisée x86 ou SPARC ou une image de machine virtuelle personnalisés.

Spécifiez des paramètres manifest, tels que :

- Nom des packages à inclure dans l'image
- Emplacement réseau du référentiel auquel accéder pour récupérer les packages pour l'image
- Nom et emplacement des scripts utilisés pour terminer la création de la nouvelle image

Astuce – Lorsque vous utilisez le constructeur de distribution pour créer des images ISO, notez les points suivants :

L'archive root pour les images x86 diffère de l'archive root pour les images SPARC. L'archive whole root, ou `boot_archive`, pour les images x86, est un système de fichiers UFS compressé à l'aide de `gzip`. La plate-forme SPARC ne prend pas en charge la compression de l'archive whole root de cette manière. Au lieu de cela, les archives SPARC utilisent DCFS, qui compresse chaque fichier individuellement. Ces fichiers compressés séparément peuvent nécessiter un traitement spécifique dans le fichier manifest. Pour plus d'instructions, reportez-vous au champ `<boot_archive_contents>` de la page de manuel `dc_manifest`(4).

Les fichiers manifest par défaut inclus dans le package `distribution-constructor` sont répertoriés dans le tableau ci-dessous.

TABLEAU 2–2 Fichiers manifest par défaut

Type de fichier manifest	Emplacement du fichier manifest	Description
Image ISO du Live CD x86	`/usr/share/distro_const/slim_cd/all_lang_slim_cd_x86.xml`	Ce fichier manifest est utilisé pour créer une image ISO comparable au Live CD d'Oracle Solaris, également appelé « Slim CD ».
Image ISO du programme d'installation en mode texte x86	`/usr/share/distro_const/text_install/text_mode_x86.xml`	Ce fichier manifest est utilisé pour créer une image ISO que vous pouvez initialiser pour lancer une installation en mode texte du SE Oracle Solaris sur des ordinateurs x86.
Image ISO du programme d'installation en mode texte SPARC	`/usr/share/distro_const/text_install/text_mode_sparc.xml`	Ce fichier manifest est utilisé pour créer une image ISO que vous pouvez initialiser pour lancer une installation en mode texte du SE Oracle Solaris sur des ordinateurs SPARC.
Image ISO d'installation automatisée SPARC	`/usr/share/distro_const/auto_install/ai_sparc_image.xml`	Ce fichier manifest est utilisé pour créer une image ISO d'installation automatisée SPARC pour l'installation automatisée du SE Oracle Solaris sur des clients SPARC.
Image ISO d'installation automatisée x86	`/usr/share/distro_const/auto_install/ai_x86_image.xml`	Ce fichier manifest est utilisé pour créer une image ISO d'installation automatisée x86 pour l'installation automatisée du SE Oracle Solaris sur des clients x86.

TABLEAU 2–2 Fichiers manifest par défaut *(Suite)*

Type de fichier manifest	Emplacement du fichier manifest	Description
Machine virtuelle x86	`/usr/share/distro_const/vmc/vmc_image.xml`	Ce fichier manifest est utilisé pour créer une image de machine virtuelle. Pour plus d'instructions, reportez-vous à la section Chapitre 3, “x86 : conception et création d'une machine virtuelle”.

Modification du contenu des fichiers manifest

Tous les champs des fichiers manifest fournissent des valeurs par défaut, prédéfinies, qui créent le type d'image ISO dont vous avez besoin, ainsi que, pour les systèmes x86, des images USB. Vous pouvez modifier manuellement ces champs prédéfinis dans un fichier manifest afin de personnaliser davantage l'image obtenue.

Si vous souhaitez modifier les informations d'un fichier manifest, effectuez la procédure suivante :

1. Copiez un des fichiers manifest par défaut et créez un fichier manifest personnalisé avec un nouveau nom de fichier.

   Vous ferez référence au fichier manifest par son nom lorsque vous utiliserez la commande `distro_const` pour créer une image.

   Remarque – Sauvegardez toujours le fichier manifest d'origine et les scripts par défaut avant de les copier.

2. Modifiez les champs du fichier manifest selon vos besoins. Reportez-vous à la page de manuel `dc_manifest`(4) pour obtenir des instructions.
3. Si les scripts de finalisation par défaut ne laissent pas `build_area` comme vous le souhaitez, vous pouvez créer vos propres scripts pour procéder à des modifications supplémentaires. Si vous créez de nouveaux scripts, mettez à jour les références de script dans la section de finalisation du fichier manifest.

   Pour les instructions sur la modification de la section relative au script de finalisation dans le fichier manifest, reportez-vous à la page de manuel `dc_manifest`(4).

   Pour obtenir des instructions sur la création de nouveaux scripts, reportez-vous à la section “Personnalisation avancée d'une image à l'aide des scripts de finalisation” à la page 13.
4. Une fois toutes les modifications apportées au fichier manifest, et, éventuellement, les scripts de finalisation personnalisés comme décrit à la section suivante, vous pouvez passer à l'exécution de l'utilitaire `distro_const` pour créer une image. Pour plus d'instructions, reportez-vous à la section “Création d'une image” à la page 18.

Personnalisation avancée d'une image à l'aide des scripts de finalisation

Le constructeur de distribution vous permet de spécifier les scripts qui peuvent être utilisés afin de personnaliser l'installation selon le type d'images créées. Ces scripts sont appelés *scripts de finalisation*. Les fichiers manifest désignent les scripts de finalisation, et ceux-ci transforment l'image générique en une distribution spécifique au support. Un ensemble de scripts par défaut est fourni dans les packages d'application.

L'application du constructeur de distribution inclut les scripts par défaut dans le répertoire `/usr/share/distro_const` et ses sous-répertoires. Ces scripts sont référencés dans la section de finalisation des fichiers manifest.

Par exemple, le tableau suivant répertorie certains scripts de finalisation par défaut utilisés pour créer un Live CD ou « Slim CD » x86.

TABLEAU 2–3 Scripts de finalisation Live CD x86

Script de finalisation	Description
`pre_boot_archive_pkg_image_mod`	Modifie la zone d'image pour tous les types d'images
`slim_cd/slimcd_pre_boot_archive_pkg_image_mod`	Modifie la zone d'image, en particulier pour les images Live CD
`slim_cd/slimcd_gen_cd_content`	Génère une liste de fichiers qui font partie de l'image Live CD
`boot_archive_initialize.py`	Initialise l'archive d'initialisation
`slim_cd/slimcd_boot_archive_configure`	Configure l'archive d'initialisation, en particulier pour l'image Live CD
`boot_archive_configure`	Configure l'archive d'initialisation
`boot_archive_archive.py`	Archive l'archive d'initialisation
`slim_cd/slimcd_post_boot_archive_pkg_image_mod`	Applique des modifications postérieures à la construction à la zone d'image de l'archive d'initialisation, en particulier pour l'image Live CD
`grub_setup`	Initialise le menu Grub
`post_boot_archive_pkg_image_mod`	Applique des modifications postérieures à la construction à la zone d'image de l'archive d'initialisation
`create_iso`	Crée une image ISO
`create_usb`	Crée une image USB

Création de scripts de finalisation personnalisés

Il est recommandé d'utiliser les scripts de finalisation fournis sans les modifier. Vous pouvez toutefois écrire et ajouter vos propres scripts afin d'effectuer d'autres opérations, comme décrit ci-dessous. Si vous créez de nouveaux scripts, modifiez les fichiers manifest pour qu'ils les désignent.

Remarque – La prise en charge des scripts est limitée aux scripts par défaut, non modifiés, fournis avec les packages d'application. Si vous décidez de personnaliser ces scripts, sauvegardez les scripts d'origine au préalable.

---

1. Concevez vos nouveaux scripts. Utilisez les scripts existants comme modèles pour les nouveaux scripts. Reportez-vous à la section "Caractéristiques des scripts de finalisation" à la page 15.

   Consultez également les exemples de scripts personnalisés suivants :

   - "Exemple de script permettant d'ajouter des packages à une image" à la page 37
   - "Exemple de script permettant de tester d'autres packages" à la page 41
   - "Exemple de script permettant d'ajouter un package SVR4 à une image" à la page 43
2. Créez vos nouveaux scripts.
3. Ajoutez les nouveaux scripts au répertoire `/usr/local/`, à votre répertoire personnel ou ailleurs sur le système ou réseau. Assurez-vous que le rôle root puisse exécuter ces scripts.
4. Ajoutez le nom des nouveaux scripts dans la section de finalisation du fichier manifest approprié. Veillez à spécifier le chemin d'accès complet à vos scripts dans le fichier manifest, même s'ils se trouvent dans le répertoire `/usr/share/distro_const`.
5. Lorsque vous ajoutez une référence pour un nouveau script dans la section de finalisation d'un fichier manifest, vous devez spécifier un nom de point de contrôle utilisé pour interrompre la création de l'image avant que ce script ait effectué sa tâche ou après. Vous pouvez également inclure un message personnalisé associé au nom du point de contrôle. Si vous n'ajoutez pas un tel message, le chemin du script est utilisé comme message de point de contrôle par défaut.

   Remarque – Attribuez aux points de contrôle des noms explicites plutôt que des nombres. Si de nouveaux scripts sont ajoutés, les nouvelles étapes correspondant à ces nouveaux scripts perturberont l'ordre des points de contrôle numérotés.

   Par exemple, la référence suivante dans un fichier manifest spécifie le nom de point de contrôle "ba-arch" pour un script d'archivage de l'archive d'initialisation et le message associé est "archivage de l'archive d'initialisation".

   ```
   <script name="/usr/share/distro_const/boot_archive_archive.py">
   <checkpoint name="ba-arch" message="Boot archive archiving"/>
   </script>
   ```

Cet exemple vous permet de spécifier une commande `distro_const` pour créer votre image, ou interrompre ou reprendre la création à partir du point de contrôle "ba-arch". La création d'image s'interrompt juste avant que le script d'archivage de l'archive d'initialisation effectue sa tâche.

Pour en savoir plus sur le référencement des points de contrôle dans la commande `distro_const`, reportez-vous à la section "Création d'une image par étapes" à la page 18.

Caractéristiques des scripts de finalisation

Lorsque vous créez vos propres scripts de finalisation, prenez en compte les points suivants :

- Les scripts peuvent être des programmes Python, des scripts shell ou des fichiers binaires.
- Les scripts sont exécutés dans leur ordre d'apparition dans le fichier manifest.
- La sortie standard (`stdout`) et la sortie d'erreur (`stderr`) des commandes exécutées dans les scripts de finalisation (modules shell et python) sont enregistrées dans les fichiers journaux. La sortie `stderr` est enregistrée dans le journal simple. Les sorties `stdout` et `stderr` sont capturées dans le journal détaillé.
- Le constructeur de distribution transmet cinq arguments initiaux à tous les scripts exécutés. Ces cinq arguments ne sont pas inclus en tant qu'entrées dans le fichier manifest. Le fichier manifest spécifie d'autres arguments transmis après ces cinq arguments. Les arguments initiaux sont les suivants.

TABLEAU 2–4 Arguments des scripts de finalisation

Argument	Description
Nom de fichier du socket de serveur	Le premier argument est le socket du lecteur de fichiers manifest. Cet argument spécifie le socket utilisé avec `/usr/bin/ManifestRead` pour accéder aux données du fichier manifest. Reportez-vous à la section "Utilisation du lecteur de fichiers manifest" à la page 16.

Chemin d'accès à la zone d'image du package	Le deuxième argument est `PKG_IMG_PATH`, qui spécifie le chemin d'accès à la zone où l'image du package est créée. Utilisez cet argument pour localiser un fichier dans la zone d'image du package. L'exemple suivant vérifie si l'utilisateur, "jack", se trouve dans le fichier de mots de passe de la zone d'image du package.
	`PKG_IMG_PATH=$2`
	`/usr/bin/grep jack $PKG_IMG_PATH/etc/passwd >/dev/null`
	`if [[ $? == "0" ]] ; then`
	`  print "Found Jack"`
	`fi`

Répertoire temporaire	Le troisième argument spécifie un répertoire qui est utilisé lors de la création de fichiers temporaires nécessaires au processus de création. Dans l'exemple suivant, vous créez un fichier dans le répertoire temporaire pour générer l'archive d'initialisation.
	`TMP_DIR=$3`
	`/usr/sbin/mkfile $TMP_DIR/boot_archive_archive`
	`/usr/sbin/lofiadm -a $TMP_DIR/boot_archive_archive`

TABLEAU 2–4 Arguments des scripts de finalisation *(Suite)*

Argument	Description

Zone de construction de l'archive d'initialisation	Le quatrième argument est la zone de création de l'archive d'initialisation, où les fichiers d'archive d'initialisation sont rassemblés. Reportez-vous à l'exemple suivant dans le module `boot_archive_configure`, qui ajoute ce fichier, `/etc/nodename`, à l'archive d'initialisation. Ce fichier donne au système le nom d'hôte, « solaris ».
	`BR_BUILD=$4 # Boot archive build area`
	
	`# Set nodename to solaris`
	`echo "solaris" > $BR_BUILD/etc/nodename`

Zone de média	Le cinquième argument indique l'emplacement où le média terminé est consigné. Dans l'exemple suivant, le script `create_iso` utilise cet argument pour placer l'image ISO obtenue.
	`MEDIA_DIR=$5`
	`...`
	`DIST_ISO=${MEDIA_DIR}/${DISTRO_NAME}.iso`
	`...`

Arguments supplémentaires	Une liste des arguments supplémentaires qui sont transmis à un script apparaît dans le fichier manifest avec la balise `<argslist>`. Le premier de ces arguments est transmis comme `arg6`. Entourez chaque élément de la liste de guillemets doubles. Si aucun guillemet n'est utilisé, ou si un ensemble de guillemets encadre l'intégralité de la chaîne, celle-ci, espaces et retours inclus, est interprétée comme un argument. N'utilisez pas de virgules pour séparer les arguments.
	Dans l'exemple suivant du fichier manifest `slim_cd_x86.xml`, deux arguments supplémentaires sont transmis au script `boot_archive_configure`, comme `arg6` et `arg7` :
	`<argslist>`
	`"/usr/share/distro_const/slim_cd/slimcd_generic_live.xml"`
	`".livecd"`
	`</argslist>`
	Une autre moyen d'indiquer d'autres arguments consiste à utiliser des paires clé-valeur. Reportez-vous à la section suivante.

Utilisation du lecteur de fichiers manifest

Le constructeur de distribution transmet le socket du lecteur de fichiers manifest comme premier argument dans les scripts de finalisation. Dans un script shell de finalisation, transmettez cet argument en tant que premier argument à `/usr/bin/ManifestRead` pour accéder aux données du fichier manifest. Dans un module python, transmettez cet argument pour créer une instance d'un objet ManifestRead().

Les exemples de scripts shell suivants appellent `ManifestRead` afin de demander l'élément `name` à partir du fichier manifest. `ManifestRead` renvoie zéro ou plusieurs éléments, chacun sur sa propre ligne. Si `ManifestRead` doit rechercher plusieurs éléments, les lignes renvoyées contiennent l'élément recherché et un résultat.

```
MFEST_SOCKET=$1

VOLNAME=‘/usr/bin/ManifestRead ${MFEST_SOCKET} "name"‘
if [ "XX${VOLNAME}" == "XX" ] ; then
       print -u2 "$0: Error retrieving volume ID"
       exit 1
fi
```

L'exemple suivant montre comment utiliser `ManifestRead` à partir d'un script Python :

```
from osol_install.ManifestRead import ManifestRead

# get the manifest reader object from the socket
manifest_reader_obj = ManifestRead(MFEST_SOCKET)

# get boot_archive compression type

BR_COMPR_TYPE = get_manifest_value(manifest_reader_obj,
   "img_params/output_image/boot_archive/compression/type")
if (BR_COMPR_TYPE == None):
       raise Exception, (sys.argv[0] +
           ": boot_archive compression type missing from manifest")
```

Spécification de paires clé-valeur

Une autre méthode pour transmettre des arguments aux scripts consiste à spécifier une paire clé-valeur. Elle est utile pour transmettre le même argument à plusieurs scripts sans duplication. Un script peut accéder à une valeur codée en spécifiant la clé de `/usr/bin/ManifestRead` à partir du script. Indiquez le socket de serveur comme premier argument, puis précisez les chemins d'accès du nœud aux éléments dont les valeurs sont nécessaires, comme dans les exemples suivants.

EXEMPLE 2–1 Script shell

L'exemple suivant appelle `ManifestRead` à partir d'un script shell pour obtenir une valeur codée.

```
...
  MFEST_SOCKET=$1
  ...
  /usr/bin/ManifestRead -k $MFEST_SOCKET iso_sort
  iso_sort_file=‘/usr/bin/ManifestRead $MFEST_SOCKET iso_sort‘
```

EXEMPLE 2–2 Script Python

L'exemple suivant appelle `ManifestRead` à partir d'un script Python pour obtenir la même valeur codée.

```
from osol_install.ManifestRead import ManifestRead

...
  IS_KEY = True
```

EXEMPLE 2–2 Script Python *(Suite)*

```
iso_sort_file = manifest_reader_obj.get_values("iso_sort", IS_KEY)
fd = open(iso_sort_file,....)
```

Création d'une image

Après avoir configuré le fichier manifest que vous avez choisi d'utiliser et, éventuellement, personnalisé les scripts de finalisation, vous êtes prêt à créer une image en exécutant la commande `distro_const`.

Remarque – Pour exécuter la commande `distro_const`, assumez le rôle root en exécutant la commande `su -`.

Vous pouvez utiliser la commande `distro_const` pour effectuer l'une des opérations suivantes :

- Créer une image en une seule étape.
- Créer une image, mais interrompre et redémarrer le processus de création selon les besoins pour examiner le contenu de l'image et déboguer les scripts de finalisation en cours d'exécution.

La syntaxe complète de cette commande est la suivante :

`Syntax: distro_const build [-R] [-r` *step*`] [-p` *step*`] [-l]` *manifest*

Création d'une image en une seule étape

Pour exécuter la création complète d'une image sans interruption, utilisez la commande `distro_const` de base sans option, comme suit :

`# distro_const build` *manifest*

Remplacez *manifest* par le nom du fichier manifest à utiliser comme empreinte pour votre image. La sous-commande `build` est requise. Lorsque vous tapez cette commande, le constructeur de distribution extrait les packages requis pour l'image et crée l'image selon les spécifications définies dans le fichier manifest.

Création d'une image par étapes

Vous pouvez utiliser les options offertes par la commande `distro_const` pour arrêter et redémarrer le processus de création à différentes étapes du processus de génération d'image,

afin de contrôler et de déboguer votre sélection de fichiers, packages et scripts pour l'image en cours de création. Ce processus d'arrêt et de redémarrage au cours de la création est appelé *création de points de contrôle*.

La création de points de contrôle prend en charge le processus de développement et de débogage des images. Vous pouvez commencer à créer une image, interrompre le processus à tout moment pour examiner le contenu de l'image, puis reprendre la création de l'image. La création de points de contrôle est facultative.

Remarque – La fonction de création de points de contrôle est activée par défaut dans le fichier manifest. Toutefois, un jeu de données ZFS ou un point de montage lié à un jeu de données ZFS, doit être spécifié comme zone de construction.

Vous pouvez également désactiver la création de points de contrôle dans le fichier manifest en définissant le paramètre `checkpoint_enable` sur `false`.

Utilisez les options de création de points de contrôle disponibles dans la commande `distro_const` comme décrit dans les instructions de base ci-dessous. Consultez également la page de manuel `distro_const`(1M).

▼ Création d'une image par étapes à l'aide de l'option de points de contrôle

1 Avant de créer l'image, vérifiez les étapes valides au cours desquelles vous pouvez choisir d'interrompre ou reprendre la création à l'aide de la commande suivante :

```
# distro_const build -l manifest
```

Remarque – La sous-commande `build` est requise.

Un point de contrôle dans le processus de création est créé après chaque exécution du script de finalisation.

Cette commande affiche les points de contrôle valides auxquels vous pouvez interrompre ou reprendre la création d'une image. Utilisez les noms d'étapes fournis par cette commande comme valeurs correctes pour les autres options de commande de création de points de contrôle.

Par exemple, la commande suivante confirme les points de contrôle disponibles, pour un fichier manifest nommé *slim_cd_x86.xml*.

```
# distro_const build -l slim_cd_x86.xml
```

Une fois la commande exécutée, les étapes de création de points de contrôle valides s'affichent. Les étapes de création de points de contrôle sont décrites ci-après.

```
Step           Resumable Description
-------------- --------- -------------
im-pop             X     Populate the image with packages
im-mod             X     Image area modifications
slim-im-mod              Slim CD image area modifications
ba-init                  Boot archive initialization
slim-ba-config           Slim CD boot archive configuration
ba-config                Boot archive configuration
ba-arch                  Boot archive archiving (64-bit)
ba-arch-32               Boot archive archiving (32-bit)
slim-post-mod            Slim CD post boot archive image area modifications
grub-setup               Grub menu setup
post-mod                 Post boot archive image area modification
gen-slim-cont            Generate Slim CD image content list
iso                      ISO image creation
usb                      USB image creation
```

Remarque – Dans le résultat de la commande, une coche dans le champ de reprise indique que vous pouvez reprendre la création à partir de cette étape.

2 Utilisez la commande suivante pour créer une image et pour interrompre la création de l'image à l'étape spécifiée.

```
# distro_const build -p step  manifest
```

Remarque – La sous-commande `build` est requise. Les champs *step* et *manifest* sont obligatoires.

Par exemple, la commande suivante lance la création d'une image et interrompt la construction avant que l'étape *im-mod* ne modifie la zone d'image :

```
# distro_const build -p im-mod slim_cd_x86.xml
```

3 Reprenez la création à partir de la dernière étape exécutée ou d'une étape de votre choix, en utilisant l'une des options suivantes :

- **Utilisez la commande suivante pour reprendre la création de l'image à partir d'une étape spécifiée.**

  ```
  # distro_const build -r step  manifest
  ```

  Remarque – L'étape spécifiée doit être l'étape à laquelle la création antérieure a été interrompue ou une étape précédente. Une étape ultérieure n'est pas valide. Les champs *step* et *manifest* sont obligatoires. La sous-commande `build` est requise.

Par exemple, la commande suivante reprend la création de l'image là où *im-mod* modifie la zone d'image :

```
# distro_const build -r im-mod slim_cd_x86.xml
```

- **Utilisez la commande suivante pour reprendre la création de l'image à partir de la dernière étape exécutée.**

  Remarque – L'argument *manifest* et la sous-commande `build` sont requis.

  ```
  # distro_const build -R manifest
  ```

  Par exemple, la commande suivante reprend la création de l'image là où le processus avait été interrompu.

  ```
  # distro_const build -R slim_cd_x86.xml
  ```

Dépannage du constructeur de distribution

Vérifiez les éléments de dépannage suivants.

Des messages d'erreur s'affichent lorsque je télécharge un package

Assurez-vous que la commande `pkg(1)` sur votre système fonctionne correctement et que votre connexion avec le serveur IPS est stable. Parfois le délai de connexion du serveur IPS arrive à expiration lors du téléchargement d'un grand cluster de packages. Pour effectuer une vérification en dehors de l'environnement du constructeur de distribution, essayez d'imiter ce que le constructeur effectue en termes d'installation des packages. Assumez le rôle root en exécutant la commande `su -`. Essayez ensuite les commandes suivantes, en confirmant leur fonctionnement.

```
pkg image-create -F -a opensolaris.org=http://pkg.opensolaris.org
pkg -R /tmp/test_img install SUNWcsd
pkg -R /tmp/test_ima install SUNWcs
pkg -R /tmp/test_img install slim_install
```

Débogage des fichiers manifest présentant des erreurs de validation

Si un fichier manifest n'est pas validé, comme cela peut être le cas après sa modification, exécutez l'utilitaire `ManifestServ` dans un mode détaillé pour trouver l'erreur.

L'utilitaire `ManifestServ`, `/usr/bin/ManifestServ`, sans aucun argument affiche l'usage suivant :

```
ManifestServ
Usage: /bin/ManifestServ [-d] [-h|-?] [-s] [-t] [-v] [-f <validation_file_base> ]
    [-o <out_manifest.xml file> ] <manifest.xml file>
where:
  -d: turn on socket debug output (valid when -s also specified)
  -f <validation_file_base>: give basename for schema and defval files
      Defaults to basename of manifest (name less .xml suffix) when not provided
  -h or -?: print this message
  -o <out_manifest.xml file>: write resulting XML after defaults and
      validation processing
  -t: save temporary file
      Temp file is "/tmp/<manifest_basename>_temp_<pid>
  -v: verbose defaults/validation output
  -s: start socket server for use by ManifestRead
```

Le constructeur de distribution valide le fichier manifest par rapport à un schéma XML et un fichier manifest par défaut. `ManifestServ` vous permet d'effectuer une validation manuelle, en utilisant un mode détaillé qui affiche l'emplacement des problèmes.

EXEMPLE 2–3 Débogage des erreurs de validation de schéma

L'exemple suivant présente un cas dans lequel le fichier manifest n'est pas validé par rapport au schéma. Le message en gras ci-après indique une erreur de validation du schéma.

```
# distro_const build my_distro.xml
/usr/share/distro_const/DC-manifest.defval.xml validates
/tmp/all_lang_slim_cd_x86_temp_7861.xml:350: element pair:
Relax-NG validity error : Element pair failed to validate attributes
/tmp/my_distro_temp_7861.xml fails to validate
validate_vs_schema: Validator terminated with status 3
validate_vs_schema: Validator terminated abnormally
Error validating manifest against schema
/usr/share/distro_const/DC-manifest.rng

#
```

Exécutez `ManifestServ`, `/usr/bin/ManifestServ` en spécifiant l'option `-t` afin d'enregistrer le fichier temporaire, et l'option `-v` pour obtenir une sortie détaillée contenant le numéro de ligne de l'erreur.

```
# ManifestServ -f /usr/share/distro_const/DC-manifest -t -v manifest_file

ManifestServ -f /usr/share/distro_const/DC-manifest -t -v my_distro.xml
/usr/share/distro_const/DC-manifest.defval.xml validates
Checking defaults for name
Checking defaults for distro_constr_params/distro_constr_flags/stop_on_error
Checking defaults for distro_constr_params/pkg_repo_default_authority/main/url
...
...
(omitted content)
...
```

EXEMPLE 2–3 Débogage des erreurs de validation de schéma *(Suite)*

```
...
/tmp/my_distro_temp_7870.xml:350: element pair: Relax-NG validity error
: Element pair failed to validate attributes
/tmp/all_lang_slim_cd_x86_temp_7870.xml fails to validate
validate_vs_schema: Validator terminated with status 3
validate_vs_schema: Validator terminated abnormally
Error validating manifest against schema /usr/share/distro_const/DC-manifest.rng
Error running Manifest Server
schema_validate: Schema validation failed for DC manifest /tmp/my_distro_temp_7870.xml
```

Le fichier temporaire sera nommé vers la fin de la sortie. Dans l'exemple ci-dessus, le fichier est `/tmp/my_distro_temp_7870.xml`. Selon les messages d'erreur en gras, ouvrez ce fichier et recherchez le problème à la ligne 350. Dans cet exemple, la ligne 350 ressemble à ceci :

```
<key_value_pairs>
    <pair value='/usr/share/distro_const/slim_cd/slimcd_iso.sort' key='iso_sort'/>
    <pair VaLuE='myvalue' key='mykey'/>
</key_value_pairs>
```

L'attribut `VaLuE` est incorrect. Dans cet exemple artificiel, `value` devrait être en minuscules, comme dans la ligne juste au-dessus. Les messages de la deuxième à la dernière ligne indiquent que la validation du schéma a échoué. Le schéma utilisé pour la validation du constructeur de distribution est `/usr/share/distro_const/DC-manifest.xml`. Le schéma indique que les seuls attributs pour `<pair>` sont `<value>` et `<key>`, pas `<VaLuE>`.

Débogage des erreurs de validation sémantique

Une validation sémantique est également réalisée. La validation sémantique contrôle le contenu à la recherche d'erreurs de "signification" et de contexte, et ne contrôle pas uniquement la syntaxe. Par exemple, les scripts de finalisation répertoriés dans un fichier manifest peuvent être validés pour confirmer qu'ils sont des fichiers exécutables.

L'exemple suivant illustre un cas où le fichier manifest ne passe pas la validation sémantique.

```
# distro_const build -l my_distro_sem.xml
/usr/share/distro_const/DC-manifest.defval.xml validates
/usr/share/distro_const/grub_setup.py either doesn't exist
or is not an executable file
validate_node: Content "/usr/share/distro_const/grub_setup.py"
at img_params/output_image/finalizer/script/name did not validate
Error validating manifest tree content
```

La validation sémantique utilise des fonctions afin de mener à bien la validation. Ces fonctions impriment des messages d'erreur expliquant pourquoi le fichier manifest a échoué à la validation. Dans ce cas, le fichier `/usr/share/distro_const/grub_setup.py` est absent et le message d'erreur indique directement ce problème. Dans ce cas, `grub_setup.py` doit être restauré, ou, le cas échéant, la référence à ce fichier doit être supprimée du fichier manifest.

Vous pouvez toujours exécuter `ManifestServ` avec l'option `-v` pour obtenir plus de détails sur la validation sémantique. Toutefois, si elle est spécifiée, l'option de commande `-v` indiquera simplement l'échec parmi de nombreuses réussites, ce qui peut donner une sortie plus difficile à lire.

Vérification des données

Une fois la validation et d'autres traitements préalables terminés, `ManifestServ` demande les données à vider et vérifier. Cette étape est plus utile pour tester le processus utilisant les données que les données elles-mêmes, dans la mesure où les données sont parfaitement visibles dans le fichier manifest même.

Informations supplémentaires relatives à l'installation

Reportez-vous aux ressources supplémentaires suivantes.

TABLEAU 2–5 Documentation relative à l'installation

Document	Thème
Notes de version Oracle Solaris 11 Express	Informations supplémentaires relatives au dépannage
Démarrage d'Oracle Solaris 11 Express	Installation à partir du Live CD ou du programme d'installation en mode texte
Guide du programme d'installation automatisée d'Oracle Solaris 11 Express	Réalisation d'installations automatisées sur plusieurs systèmes
Gestion des environnements d'initialisation avec Oracle Solaris 11 Express	Création et gestion de plusieurs environnements d'initialisation sur votre système

◆ ◆ ◆ CHAPITRE 3

x86 : conception et création d'une machine virtuelle

Vous pouvez utiliser le constructeur de distribution pour concevoir et créer une machine virtuelle Oracle Solaris préinstallée. Consultez les informations suivantes.

Qu'est-ce qu'une machine virtuelle ?

Une *machine virtuelle* est un conteneur logiciel totalement isolé capable d'exécuter son système d'exploitation et ses applications comme s'il s'agissait d'un véritable ordinateur. Une machine virtuelle (VM) se comporte exactement comme un ordinateur physique. Elle contient son propre matériel virtuel : CPU, mémoire RAM, disque dur et carte d'interface réseau (NIC) basés sur du logiciel. La VM créée à l'aide du constructeur de distribution contient un système d'exploitation Oracle Solaris préinstallé.

VirtualBox est une application de virtualisation x86 qui facilite la création de machines virtuelles.

Qu'est-ce que le produit final ?

Lorsque le constructeur de distribution crée une machine virtuelle, celle-ci est exportée dans trois formats par défaut et stockée comme un ensemble de fichiers dans le répertoire `<build_area>/media` sur le système hôte.

Remarque – Un système *hôte* correspond au système sur lequel vous exécutez le constructeur de distribution et créez une machine virtuelle.

Un système *client* correspond au système sur lequel vous déployez une machine virtuelle.

Les fichiers sont stockés dans les sous-répertoires et aux formats suivants :

- `build_area/media/esx` : fichiers de machine virtuelle dans un format adapté aux produits ESX/ESXi de VMware.
- `build_area/media/ovf` : fichiers de machine virtuelle dans un format OVF pour les hyperviseurs qui peuvent importer un tel format.

  Remarque – OVF est un format indépendant des plates-formes, normalisé, pour les machines virtuelles.

Les fichiers contenus dans chaque dossier correspondent à des images de machine virtuelle. Vous pouvez graver les fichiers sur un DVD.

Remarque – Un autre produit final est un fichier XML qui enregistre, pour référence, les paramètres utilisés par le constructeur de distribution pour configurer la machine virtuelle.

Configuration système requise pour la création de machines virtuelles

Afin de pouvoir créer une machine virtuelle, votre système hôte doit disposer de la configuration suivante.

TABLEAU 3–1 Tableau de la configuration système requise

Configuration requise	Description
40 Go d'espace disque disponible	Pour utiliser le constructeur de distribution et les fichiers manifest par défaut, vous avez besoin de 40 Go d'espace disque disponible dans le jeu de données ZFS sur le système hôte où l'image de machine virtuelle sera créée. L'espace réellement nécessaire dépend en partie du nombre de packages que vous choisissez d'installer dans l'image.
4 Go de mémoire	Vous nécessitez environ 4 Go de mémoire sur le système hôte pour pouvoir exécuter un invité VirtualBox, le système d'exploitation et le constructeur de distribution. Ce total inclut environ 1 Go pour l'exécution de la machine virtuelle et environ 3 Go pour l'exécution du système d'exploitation et du constructeur de distribution.

TABLEAU 3–1 Tableau de la configuration système requise *(Suite)*

Configuration requise	Description
Version d'Oracle Solaris	Installez le système d'exploitation (SE) Oracle Solaris x86 sur le système hôte. La version Oracle Solaris sur votre système doit être la même que celle des images d'installation automatisée que vous utilisez avec le constructeur de distribution. **Remarque –** Pour exécuter le constructeur de distribution sur votre système, vous devez assumer le rôle root en exécutant la commande `su -`.
Accès réseau	Le constructeur de distribution utilise également une image du programme d'installation automatisée (IA) et un fichier manifest de client d'installation automatisée afin d'effectuer une installation dans la machine virtuelle. Le système hôte qui exécute le constructeur de distribution doit avoir un accès réseau à un référentiel IPS, tel que `http://pkg.opensolaris.org`, qui est spécifié dans le fichier manifest de client d'installation automatisée. Vous pouvez utiliser le fichier manifest de client d'installation automatisée par défaut ou fournir un fichier manifest de client d'installation automatisée personnalisé comme décrit à la section "Facultatif : personnalisation des spécifications de version" à la page 29. En créant un fichier manifest de client d'installation automatisée personnalisé, vous pouvez modifier les spécifications d'installation, par exemple définir une cible d'installation particulière ou modifier la liste des packages à inclure dans le système d'exploitation.
VirtualBox, version 3.0.12	VirtualBox doit être installé sur le système hôte.
Package `distribution-constructor`	Installez la copie la plus récente du package `distribution-constructor`, qui contient l'application du constructeur de distribution, sur le système hôte. **Remarque –** Vous pouvez utiliser le gestionnaire de packages pour installer le package requis. Le gestionnaire de packages est disponible dans la barre de menu sur le bureau du système d'exploitation Oracle Solaris. Dans la barre de menu, cliquez sur Système>Administration>Gestionnaire de packages. Vous pouvez également utiliser les commandes IPS, telles que la suivante, pour installer ce package : `# pkg install` *distribution-constructor*

Création d'une machine virtuelle

Le processus utilisé pour créer une machine virtuelle est très simple :

1. "Obtention d'une image d'initialisation d'installation automatisée" à la page 28
2. "Facultatif : personnalisation des spécifications de version" à la page 29
3. "Création de la machine virtuelle" à la page 34

Obtention d'une image d'initialisation d'installation automatisée

Une machine virtuelle complète sera créée à l'aide des paramètres spécifiés dans le fichier manifest de la machine virtuelle, `vmc_image.xml`. Il s'agit d'un fichier XML qui se trouve sous `/usr/share/distro_const/vmc/vmc_image.xml`. Le fichier `vmc_image.xml` désigne un jeu par défaut de scripts de finalisation permettant de mettre en oeuvre les paramètres spécifiques au support final pour la machine virtuelle. En outre, une image d'initialisation du programme d'installation automatisée (IA) est utilisée pour créer la machine virtuelle.

Vous devez fournir l'image d'initialisation d'installation automatisée. Vous devez ensuite modifier le fichier manifest de la machine virtuelle afin qu'il désigne cette image.

Téléchargez une image d'installation automatisée et référencez-la dans le fichier manifest de la machine virtuelle comme suit.

Remarque – Vous pouvez créer une image d'installation automatisée personnalisée à l'aide du constructeur de distribution. Cependant, il n'est pas nécessaire de créer une image personnalisée.

▼ Fourniture d'une image d'initialisation

1 **Téléchargez une image d'installation automatisée x86 sur la page `http://www.oracle.com/technetwork/server-storage/solaris11/downloads/index.html`.**

Remarque – L'image d'installation automatisée fait environ 200-250 Mo.

Assurez-vous que la version de l'image d'installation automatisée correspond à la version du système d'exploitation Oracle Solaris que vous prévoyez d'installer dans votre machine virtuelle.

2 **Stockez l'image IA sur votre système de fichiers local.**

L'image d'installation automatisée est un fichier .iso, un ensemble de logiciels fourni dans un seul fichier. Lorsque vous téléchargez l'image, vous l'enregistrez en tant que fichier .iso sur votre système.

Notez le nom et l'emplacement de ce fichier. Vous devrez indiquer ces informations dans le fichier manifest.

3 Modifiez `vmc_image.xml` afin qu'il désigne cette image, comme suit.

Recherchez le champ XML suivant dans le fichier manifest :

```
<script name="/usr/share/distro_const/vmc/prepare_ai_image">
    <checkpoint
        name="prepare-image"
        message="prepare bootable ai image"/>
    <argslist>
    <!-- Path to bootable AI image ISO -->
        "/export/home/name_of_bootable_ai_image"
    <!-- What AI client manifest to use for installation.
         "default" will use the existing AI client manifest
         included on the AI media.  To use a custom
         AI client manifest, provide a path to your custom manifest -->
    "default"
 </argslist>
</script>
```

Le champ ci-dessus fait référence au script de finalisation `prepare_ai_image`. Ce script détecte et prépare l'image d'initialisation spécifiée pour l'installation sur la machine virtuelle et fournit une option permettant de spécifier un fichier manifest de client d'installation automatisée personnalisé.

La ligne suivant ce script fournit un chemin d'accès par défaut à une image d'installation automatisée, à l'aide du substituant `name_of_bootable_ai_image` en lieu et place du nom de fichier actuel.

```
"/export/home/name_of_bootable_ai_image"
```

À l'aide de votre nom de fichier, remplacez cette ligne afin qu'elle désigne l'emplacement sur votre système local de l'image d'installation automatisée que vous avez téléchargée à l'étape précédente. Par exemple,

```
"/export/home/myimage.iso"
```

Remarque – L'emplacement du fichier est indiqué entre guillemets. Ne supprimez pas les guillemets. Ils permettent de délimiter chaque argument dans le champ.

Vous pouvez modifier davantage le champ `prepare_ai_image` afin qu'il désigne un fichier manifest de client d'installation automatisée personnalisé, comme décrit dans la section suivante.

Facultatif : personnalisation des spécifications de version

Vous pouvez choisir de personnaliser d'autres spécifications dans le fichier manifest de la machine virtuelle. Vous pouvez apporter ces changements en modifiant les champs XML du fichier `vmc_image.xml` avant de créer votre machine virtuelle.

Le tableau suivant répertorie les champs de `vmc_image.xml`, fournit les valeurs par défaut de chaque champ et indique quels champs peuvent être modifiés.

Remarque – Certains champs de ce tableau sont également utilisés dans les autres fichiers manifest du constructeur de distribution. La plupart de ces champs sont toutefois propres au fichier manifest de la machine virtuelle.

TABLEAU 3–2 Champs du fichier manifest de la machine virtuelle

Champ du fichier manifest	Description
`<build_area>rpool/dc</build_area>`	Le champ de la zone de construction spécifie la zone dans laquelle la machine virtuelle sera créée. Vous pouvez utiliser la zone par défaut si elle est appropriée pour votre système ou spécifier une autre zone de construction sur le système hôte. **Remarque –** Pour utiliser l'option de création de points de contrôle pour interrompre et reprendre le processus en cours de création, vous devez spécifier un jeu de données ZFS ou un point de montage correspondant à un jeu de données ZFS comme zone de construction. Si le jeu de données ZFS n'existe pas, il est créé au cours du processus. Le pool ZFS que vous spécifiez doit toutefois déjà exister.

`<distro_constr_flags>`
`<stop_on_error>true</stop_on_error>`
`<checkpoint_enable>`
`true`
`</checkpoint_enable>`
	Dans la balise `<distro_constr_flags>`, la valeur true appliquée à `<stop_on_error>` signifie que, si une erreur se produit pendant la création, le constructeur de distribution cesse de fonctionner.
	
	La valeur true appliquée à `<checkpoint_enable>` vous permet d'interrompre la création de machine virtuelle à n'importe quelle étape (ou point de contrôle) et de redémarrer la création au point de contrôle spécifié. Pour plus d'instructions, reportez-vous à la section "Création d'une image par étapes à l'aide de l'option de points de contrôle" à la page 19.

TABLEAU 3–2 Champs du fichier manifest de la machine virtuelle *(Suite)*

Champ du fichier manifest	Description
`<finalizer>`	La section `finalizer` contient une liste de références aux scripts de finalisation exécutés lors de la création de la machine virtuelle. Ceux-ci sont utilisés pour personnaliser l'image et sont exécutés dans l'ordre où ils apparaissent dans le fichier manifest. Comme indiqué dans les entrées de script suivantes, chaque champ du script inclut le champ `checkpoint` obligatoire contenant le nom du point de contrôle. Le point de contrôle indique l'étape de création de la machine virtuelle lorsque ce script est exécuté. Le champ de point de contrôle contient également un message de point de contrôle. Lorsque le script s'exécute, le message s'affiche. Chaque script comprend également le champ `argslist` qui fournit les arguments nécessaires à l'exécution dudit script. Vous pouvez modifier ces arguments comme décrit dans ce tableau. **Remarque** – Les arguments sont encadrés par des guillemets. Ne supprimez pas les guillemets. Ils délimitent chaque argument dans le champ.

TABLEAU 3–2 Champs du fichier manifest de la machine virtuelle *(Suite)*

Champ du fichier manifest	Description

`<script name="/usr/share/distro_const/vmc/prepare_ai_image">`
`<checkpoint`
`name="prepare-image"`
`message="prepare bootable ai image"/>`
`<argslist>`
`<!-- Path to bootable AI image ISO -->`
`"/export/home/name_of_bootable_ai_image"`
`<!-- What AI client manifest to use for installation.`
`"default" will use the existing AI client manifest`
`included on the AI media. To use a custom`
`AI client manifest, provide a path to`
`your custom manifest -->`
`"default"`
`</argslist>`
`</script>`
	Dans la section précédente, vous avez modifié la référence du script `prepare_ai_image` pour désigner l'image d'installation automatisée téléchargée. Dans ce champ, vous pouvez également indiquer un pointeur vers un fichier manifest de client d'installation automatisée personnalisé. Lorsque vous fournissez un fichier manifest de client d'installation automatisée personnalisé, vous pouvez modifier les spécifications d'installation par défaut, par exemple définir une cible d'installation particulière ou modifier la liste des packages à installer sur le système d'exploitation.
	
	Modifiez la variable "`default`" par le chemin d'accès au fichier manifest de client d'installation automatisée personnalisé. Le chemin correspond à un chemin de fichier sur le système sur lequel vous exécutez le constructeur de distribution, comme illustré dans l'exemple suivant.
	
	"`home/user/mymanifest.xml`"
	
	Pour plus d'informations sur la création d'un fichier manifest de client d'installation automatisée personnalisé, reportez-vous à la section relative à la "création d'un fichier manifest d'installation automatisée personnalisé" du *Guide du programme d'installation automatisée d'Oracle Solaris 11 Express*.

TABLEAU 3–2 Champs du fichier manifest de la machine virtuelle *(Suite)*

Champ du fichier manifest	Description

`<script name="/usr/share/distro_const/vmc/create_vm">`
`<checkpoint`
`name="create-vm"`
`message="create and configure virtual machine"/>`
`<argslist>`
`"16000"`
`"1536"`
`"opensolaris"`
`</argslist>`
`</script>`
	Le script `create_vm` crée et configure la machine virtuelle. Le champ `argslist` fournit 3 arguments pour le script. Vous pouvez modifier ces arguments comme suit.
	■ Taille du disque de la machine virtuelle : la valeur par défaut est de 16 000 Mo. Les valeurs possibles sont comprises entre 12 000 et 99 999 999 Mo.
	■ Taille de la mémoire RAM de la machine virtuelle : la valeur par défaut est de 1 536 Mo. Les valeurs possibles sont comprises entre 1 000 et 16 384 Mo.
	■ Type de machine virtuelle : le type par défaut est `opensolaris` pour une machine virtuelle 32 bits. Sinon, utilisez `opensolaris_64` pour une machine virtuelle 64 bits.
	Remarque – Le script `create_vm` et ses arguments ont une incidence spécifique sur la machine virtuelle au cours de sa création. Reportez-vous à la section sur le script `post_install_vm_config` ci-dessous pour connaître les arguments ayant une incidence sur la machine virtuelle après son installation.

`<script name="/usr/share/distro_const/vmc/install_vm">`
`<checkpoint`
`name="install-vm"`
`message="Boot and Install virtual machine"/>`
`</script>`
	Le script `install_vm` initialise et installe la machine virtuelle dans la zone de construction.

`<script name="/usr/share/distro_const/vmc/`
`post_install_vm_config">`
`<checkpoint`
`name="post-config"`
`message="Post installation virtual machine`
`configuration"/>`
`<argslist>`
`"1024"`
`"1"`
`"on"`
`</argslist>`
`</script>`
	Le script `post_install_vm_config` effectue la configuration post-installation sur la machine virtuelle. Le champ `argslist` fournit 3 arguments pour le script. Vous pouvez modifier ces arguments comme suit.
	■ Taille de la mémoire RAM de la machine virtuelle : la valeur par défaut est de 1 024 Mo. Les valeurs possibles sont comprises entre 1 000 et 16 394 Mo.
	■ Nombre de CPU de la machine virtuelle : la valeur par défaut est de 1 CPU. Les valeurs possibles sont comprises entre 1 et 32.
	■ Prise en charge VT-x/AMD-V de la machine virtuelle : elle peut être activée ou désactivée.

TABLEAU 3–2 Champs du fichier manifest de la machine virtuelle *(Suite)*

Champ du fichier manifest	Description

`<script name="/usr/share/distro_const/vmc/export_esx">`
`  <checkpoint`
`  name="export-esx"`
`  message="Export virtual machine for VMWare ESX"/>`
`  <argslist>`
`  "esx"`
`  </argslist>`
`</script>`
	Le script `export_esx` convertit la machine virtuelle en un ensemble de fichiers ESX prêts pour l'importation. Si vous souhaitez que vos fichiers de machine virtuelle soient configurés au format ESX, utilisez ce script. Sinon, mettez ce script en commentaire.
	
	Par exemple, si vous souhaitez mettre en commentaire le champ suivant :
	
	`<sample_script>example</sample_script>`
	
	Entourez-le de balises, comme suit :
	
	`<!--`
	`<sample_script>example</sample_script>`
	`-->`
	
	La machine virtuelle ESX se compose d'un ensemble de fichiers dans le dossier `build_area/media/esx`.
	
	Remarque – Si vous utilisez les scripts `export_esx` et `export_ovf`, des machines virtuelles identiques pour chaque format sont créées.

`<script name="/usr/share/distro_const/vmc/export_ovf">`
`  <checkpoint`
`  name="export-ovf"`
`  message="Export virtual machine in OVF format"/>`
`  <argslist>`
`  "ovf"`
`  </argslist>`
`</script>`
	Le script `export_ovf` convertit la machine virtuelle en un ensemble de fichiers OVF prêts pour l'importation. Si vous souhaitez que vos fichiers de machine virtuelle soient configurés au format OVF, utilisez ce script. Sinon, mettez ce script en commentaire de telle sorte qu'il ne soit pas exécuté.
	
	La machine virtuelle OVF se compose d'un ensemble de fichiers dans le dossier `build_area/media/ovf`.

Création de la machine virtuelle

Une fois que vous avez personnalisé le fichier manifest de la machine virtuelle, vous êtes prêt à créer une machine virtuelle. Exécutez l'utilitaire `distro_const` pour créer la machine virtuelle, en utilisant les points de contrôle facultatifs pour interrompre, tester et reprendre la création selon vos besoins. Pour obtenir des instructions sur l'utilisation de l'utilitaire `distro_const`, reportez-vous à la section “Création d'une image” à la page 18.

Dépannage lors de la création

Le constructeur de distribution utilise le programme d'installation automatisée pour effectuer une installation dans la machine virtuelle. Le programme d'installation automatisée, exécuté dans le client de la machine virtuelle, ne peut plus communiquer la progression à l'hôte de machine virtuelle. Le constructeur de distribution ne peut donc pas suivre précisément la progression de l'installation dans le client de la machine virtuelle. Une fois l'installation terminée, le programme d'installation automatisée désactive la machine virtuelle et le constructeur de distribution reprend sa tâche.

Remarque – Si le programme d'installation automatisée rencontre une erreur qui entraîne un échec de l'installation, la machine virtuelle n'est pas arrêtée. Le constructeur de distribution semble alors se suspendre, alors qu'il attend en réalité que la machine virtuelle s'arrête.

Vous pouvez contrôler la progression de l'installation automatisée en cours d'exécution dans le client de la machine virtuelle. Utilisez l'utilitaire `rdesktop` disponible dans les référentiels IPS Oracle Solaris standard ou utilisez un autre client RDP.

Pour vous connecter à la console d'une installation de machine virtuelle en cours d'exécution, effectuez les opérations suivantes :

1. Si ce n'est pas déjà fait, installez le package `rdesktop`.

   ```
   # pkg install remote-desktop/rdesktop
   ```

2. Connectez la machine virtuelle en cours d'exécution qui semble être bloquée.

   ```
   # rdesktop -a 16 localhost:3389
   ```

Cette commande indique à `rdesktop` de se connecter à la machine locale sur le port 3389, avec une profondeur de couleur de 16 bits pour de meilleures performances.

Le numéro de port utilisé dans cette commande est le numéro de port par défaut utilisé par la première machine virtuelle démarrée sur l'hôte. Si vous exécutez plusieurs appels du constructeur de distribution, recherchez dans le fichier journal détaillé du constructeur de distribution le port sur lequel la machine virtuelle est exécutée. Des lignes similaires à celles qui suivent s'affichent dans le fichier journal :

```
==== install-vm: Boot and Install virtual machine

Invoking: VBoxHeadless startvm OpenSolaris_VM
VirtualBox Headless Interface 3.0.8

Listening on port 3389
```

La dernière ligne du fichier journal indique que la machine virtuelle en question peut être contactée sur le port 3389. Une fois la commande `rdesktop` exécutée, une fenêtre d'interface graphique dans laquelle vous pouvez voir la console pour la machine en cours d'exécution s'ouvre. Vous pouvez vous connecter à la console comme suit :

```
username: jack
password: jack
```

Remarque – Le nom d'utilisateur et le mot de passe par défaut sont « jack » et « jack ». Le mot de passe root par défaut est « solaris ».

Une fois connecté, vous pouvez déboguer l'installation, comme toute autre installation client du programme d'installation automatisée. Reportez-vous à l'annexe A sur le “dépannage des installations automatisées,” dans le *Guide du programme d'installation automatisée d'Oracle Solaris 11 Express* .

Informations complémentaires

Voir “Informations supplémentaires relatives à l'installation” à la page 24.

◆ ◆ ◆ ANNEXE A

Exemples de scripts de finalisation personnalisés

Les scripts de finalisation personnalisés suivants vous permettent d'exécuter des tâches particulières.

Remarque – Vous devrez modifier le fichier manifest afin de désigner les nouveaux scripts personnalisés. Reportez-vous aux instructions de la section "Utilisation de scripts personnalisés" à la page 39.

Exemple de script permettant d'ajouter des packages à une image

Lors de l'assemblage d'une image, vous pouvez faire des essais de fonctionnement de l'image lorsque des packages sont ajoutés ou supprimés d'un jeu de travail. Ce type de tests est pris en charge par le constructeur de distribution. Des packages supplémentaires peuvent être ajoutés à la liste existante dans la section `package` d'un fichier manifest. Les packages à supprimer peuvent être ajoutés à la section `post_install_remove_package`. Après avoir apporté des modifications au fichier manifest pour ajouter ou supprimer des packages, vous devez relancer le processus de création depuis le début et télécharger de nouveau l'ensemble des packages. Cette opération peut prendre du temps. Vous pouvez effectuer ces modifications plus rapidement en utilisant les scripts de finalisation.

Les scripts personnalisés suivants ajoutent un package IPS à l'image à partir d'un autre référentiel spécifié dans le fichier manifest. Le package est ajouté à la zone d'image du package. Le nom du package à ajouter est inclus dans le script. Cet exemple montre également comment utiliser le programme `ManifestRead` pour obtenir des valeurs à partir du fichier manifest.

EXEMPLE A–1 Exemple de script pour l'ajout de packages

```
#!/bin/ksh
#
#
# Name:
# add_my_pkg
#
# Description:
# This script will add the package SUNWcdrw from
# the alternate repository specified in the manifest
# at pkg_repo_addl_authority.
#
# Args:
#
#   5 arguments are passed in by default from the DC.
#
#   MFEST_SOCKET: Socket needed to get manifest data via ManifestRead object
#   PKG_IMG_PATH: Package image area
#   TMP_DIR: Temporary directory
#   BR_BUILD: Area where boot archive is put together (not used in this example)
#   MEDIA_DIR: Area where the media is put (not used)
#
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

if [ "$#" != "5" ] ; then
        print -u2 "Usage: $0: Requires 5 args:"
        print -u2 "    Reader socket, pkg_image area, tmp dir,"

        print -u2 "    boot archive build area, media area"
        exit 1
fi

MFEST_SOCKET=$1

PKG_IMG_PATH=$2
if [ ! -d $PKG_IMG_PATH ] ; then
        print -u2 "$0: Image package area $PKG_IMG_PATH is not valid"
        exit 1
fi

PKGCMD="/bin/pkg"

#Install this package
TEST_PKGS="SUNWcdrw"

#
# You would have specified the additional repository like this in the manifest
#
#
#   <pkg_repo_addl_authority>
#        <main url="http://localhost:10000" authname="localtest"/>
#   </pkg_repo_addl_authority>

# Get the alternate repository URL from the manifest
add_url=/usr/bin/ManifestRead ${MFEST_SOCKET} \
"img_params/pkg_repo_addl_authority/main/url"

# Get the alternate repository authority from the manifest
```

EXEMPLE A–1 Exemple de script pour l'ajout de packages *(Suite)*

```
add_auth=/usr/bin/ManifestRead ${MFEST_SOCKET} \
"img_params/pkg_repo_addl_authority/main/authname"

added_authority=0

#
# Check if authority is already set in the package image area
# if not, add it in
#
${PKGCMD} -R $PKG_IMG_PATH authority $add_auth > /dev/null 2>& 1
if [ $? != 0 ] ; then
        ${PKGCMD} -R $PKG_IMG_PATH  set-authority -O ${add_url} ${add_auth}
        added_authority=1
fi

if [ $? != "0" ] ; then
        print -u2 "$0: Unable to set additional authority"
        exit 1
fi

for t in ${TEST_PKGS} ; do
        pkg_name="pkg://${add_auth}/${t}"
        ${PKGCMD} -R $PKG_IMG_PATH install ${pkg_name}
        if [ $? != "0" ] ; then
                print -u2 "$0: Unable to install ${pkg_name}"

                exit 1
        fi
done

# if we have added the additional authority, unset it so it doesn’t pollute what’s
# originally there
if [ $added_authority == 1 ] ; then
         ${PKGCMD} -R $PKG_IMG_PATH  unset-authority ${add_auth}
fi

exit 0
```

Utilisation de scripts personnalisés

Une fois que vous avez préparé votre script personnalisé, vous devez ajouter une référence au script dans le fichier manifest. Vous êtes ensuite prêt à utiliser le script lors de la création de l'image.

Ajout d'un script à un fichier manifest

Vous devez mettre à jour le fichier manifest pour qu'il désigne les scripts personnalisés que vous souhaitez utiliser. Par exemple, si vous avez créé un script, `/export/home/user1/test_my_pkg`, pour tester votre package personnalisé, vous devez référencer le script comme suit dans la section de finalisation du fichier manifest. Le point de contrôle *my_test* indique où la création va redémarrer.

```
<finalizer>
      <script name="/export/home/user1/test_my_pkg">
           <checkpoint name="my_test" message="Running my test package"/>
      </script>
      <script name="/usr/share/distro_const/pre_boot_archive_pkg_image_mod">
           <checkpoint name="im-mod" message="Image area modifications"/>
      </script>
      ........
</finalizer>
```

Le script personnalisé ci-dessus est conçu pour ajouter ou supprimer des packages de la zone d'image du package. Vous devez donc référencer ce script comme premier script de finalisation dans la section `<finalizer>` du fichier manifest. Incluez dans cette référence un nom de point de contrôle qui indique le point du processus de création d'image à partir duquel vous voulez relancer la création afin de tester le package ajouté.

Remarque – L'exemple de script ci-dessus suppose qu'un autre référentiel a été spécifié dans le fichier manifest, à l'aide du champ `pkg_repo_addl_authority`. Le fichier manifest par défaut est indiqué lorsque le champ `pkg_repo_addl_authority` est mis en commentaire et, par conséquent, inutilisé. Supprimez donc la balise de commentaire de ce champ, et mettez à jour ses valeurs pour fournir une URL et un `authname` valides.

Par exemple, si l'autorité supplémentaire est disponible sur le `localhost` sur le port 10000, avec le nom `localtest`, vous devez modifier le champ `pkg_repo_addl_authority` dans le fichier manifest comme suit :

```
<pkg_repo_addl_authority>
<main
url="http://localhost:10000"
authname="localtest"/>
<mirror url="" />
</pkg_repo_addl_authority>
```

Exécution du script

Une fois que vous avez terminé d'appliquer des modifications dans le fichier manifest pour référencer votre nouveau script, vous pouvez commencer à créer votre image à partir du début. Par la suite, si vous apportez des modifications à votre package, vous n'avez plus besoin de relancer le processus depuis le début. Vous pouvez générer une image avec votre package modifié à partir du point de contrôle qui exécute votre script.

Remarque – Cette option suppose que la zone de construction se trouve dans un système de fichiers ZFS. Sans système ZFS, vous ne pouvez pas utiliser les points de contrôle.

Vous trouverez ci-dessous la séquence de commandes qu'un utilisateur doit exécuter afin de tester plusieurs fois les modifications apportées aux packages.

1. Lancez la création à partir du début après avoir modifié le fichier manifest. Vous n'avez besoin d'effectuer cette étape qu'une seule fois.

   `distro_const build` *my_updated_manifest.xml*
2. Vérifiez les étapes qui peuvent être reprises.

   `distro_const build -l` *my_updated_manifest.xml*
3. Relancez le processus à partir de l'étape de votre modification de package. Vous pouvez redémarrer à partir de cette étape autant de fois que nécessaire au cours du débogage de votre contenu modifié.

   `distro_const build -r` *my_test my_updated_manifest.xml*

Pour en savoir plus sur l'utilisation de la commande `distro_const` avec les options de points de contrôle, comme illustré dans l'exemple, reportez-vous à la section "Création d'une image par étapes" à la page 18.

Exemple de script permettant de tester d'autres packages

Lorsqu'un package est spécifié dans la section `package` d'un fichier manifest, il est installé dans la zone d'image de package par le constructeur de distribution. Si vous disposez d'une autre version de ce package (votre copie privée d'un package, par exemple), vous pouvez la tester à l'aide d'un script de finalisation personnalisé. Le script de finalisation remplacera la version installée précédente du package par une version testée à partir d'un autre référentiel.

Dans le script personnalisé suivant, un serveur de référentiel IPS est en cours d'exécution sur `http://localhost:10000`. Le nom du package à remplacer est transmis sous forme d'argument. Le script désinstalle d'abord la version existante du package, puis installe la version testée à partir de l'autre référentiel. Ce script personnalisé indique également comment transmettre les arguments en tant qu'arguments de script de finalisation. Dans le script, notez les commentaires qui expliquent comment il réalise ces tâches.

EXEMPLE A–2 Exemple de script permettant de tester des packages

```
#!/bin/ksh
#
#
# Name:
# test_my_pkg
#
# Description:
# This script will build an image using my test package
# from my local repository.
#
# Args:
#
#   These Arguments are passed in by default from the DC.
#
#   MFEST_SOCKET: Socket needed to get manifest data via ManifestRead object \
```

EXEMPLE A–2 Exemple de script permettant de tester des packages *(Suite)*

```
# (not used in this example)
#   PKG_IMG_PATH: Package image area
#   TMP_DIR: Temporary directory
#   BR_BUILD: Area where boot archive is put together (not used in this example)
#   MEDIA_DIR: Area where the media is put (not used)
#   TEST_PKG: Package to replace with one from the test repo
#
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

if [ "$#" != "6" ] ; then
        print -u2 "Usage: $0: Requires 6 args:"

        print -u2 "    Reader socket, pkg_image area, tmp dir,"
        print -u2 "    boot archive build area, media area, pkg_name"
        exit 1
fi


PKG_IMG_PATH=$2
if [ ! -d $PKG_IMG_PATH ] ; then
        print -u2 "$0: Image package area $PKG_IMG_PATH is not valid"
        exit 1
fi

PKGCMD="/bin/pkg"

#The test packages are passed in as arguments to this finalizer script
#You would have specified the argument like this in the finalizer section
#to pass in the argument
#
#
#  <finalizer>
#       <script name="/my/update_my_pkg_test">
#               <checkpoint name="update-pkg" message= \
"Replaces an existing package with my own"/>
#                   <argslist>
#                      "SUNWcdrw"
#                   </argslist>
#        </script>
#  </finalizer>
#
TEST_PKG=$6

# Assume that my test package resides in
#a repository running on port 10000 of the localhost.

# Specify  alternate repository URL
add_url="http://localhost:10000"

# Specify alternate repository authority
add_auth="MY_TEST"
# Check if authority is already set in the package image area, if not,
# add it in

added_authority=0
```

EXEMPLE A–2 Exemple de script permettant de tester des packages *(Suite)*

```
${PKGCMD} -R $PKG_IMG_PATH authority $add_auth > /dev/null 2>& 1
if [ $? != 0 ] ; then
        ${PKGCMD} -R $PKG_IMG_PATH  set-authority -O ${add_url} ${add_auth}
        if [ $? != "0" ] ; then
             print -u2 "$0: Unable to set additional authority"
             exit 1
        fi
        added_authority=1
fi


if [ $? != "0" ] ; then
        print -u2 "$0: Unable to set additional authority"
        exit 1
fi

# Remove the package that's currently in the package image area.
${PKGCMD} -R $PKG_IMG_PATH uninstall ${TEST_PKG}
if [ $? != "0" ] ; then
        print -u2 "$0: Unable to uninstall ${TEST_PKG}"
        exit 1
fi

# Install the package from test repo
pkg_name="pkg://${add_auth}/${TEST_PKG}"

${PKGCMD} -R $PKG_IMG_PATH install ${pkg_name}
if [ $? != "0" ] ; then
        print -u2 "$0: Unable to install ${pkg_name}"
        exit 1
fi

# if we have added the additional authority, unset it so it doesn't pollute what's
# originally there
if [ $added_authority == 1 ] ; then
         ${PKGCMD} -R $PKG_IMG_PATH  unset-authority ${add_auth}
fi

exit 0
```

Exemple de script permettant d'ajouter un package SVR4 à une image

Si vous disposez d'un package au format SVR4, vous pouvez le tester dans une image avant de procéder à sa conversion en package IPS. Vous pouvez utiliser un script de finalisation pour ajouter le contenu d'un package SVR4 à l'image. Reportez-vous au script personnalisé suivant et notez les commentaires dans le script expliquant la manière dont le script réalise cette tâche.

Remarque – Vous devrez modifier ce script pour inclure le chemin de votre package SVR4. Reportez-vous aux commentaires dans le script.

EXEMPLE A–3 Exemple de script permettant l'ajout de packages SVR4

```
#!/bin/ksh
#
#
# Name:
# add_my_svr4_pkg
#
# Description:
# This script will build an image using an SVR4 package
# located at a user specified file path.
#
# Note:
# The user must modify this script and provide a valid
# path to an SVR4 package in the PKG_PATH variable.
#
# #
# Args:
#
#   5 arguments are passed in by default from the DC.
#
#   MFEST_SOCKET: Socket needed to get manifest data via ManifestRead object (not used)
#   PKG_IMG_PATH: Package image area
#   TMP_DIR: Temporary directory
#   BR_BUILD: Area where boot archive is put together (not used in this example)
#   MEDIA_DIR: Area where the media is put (not used)
#
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

if [ "$#" != "5" ] ; then
        print -u2 "Usage: $0: Requires 5 args:"

        print -u2 "    Reader socket, pkg_image area, tmp dir,"
        print -u2 "    boot archive build area, media area"
        exit 1
fi

PKG_IMG_PATH=$2
if [ ! -d $PKG_IMG_PATH ] ; then
        print -u2 "$0: Image package area $PKG_IMG_PATH is not valid"
        exit 1
fi

TMP_DIR=$3

#
# Install an SVR4 packages into the package image area
#

#create an admin file for non-interactive pkgadd's

ADMIN_FILE=${TMP_DIR}/admin.$$
cat << \ADMIN_EOF > $ADMIN_FILE
```

EXEMPLE A–3 Exemple de script permettant l'ajout de packages SVR4 *(Suite)*

```
mail=
instance=unique
partial=nocheck
runlevel=nocheck
idepend=nocheck
rdepend=nocheck
space=nocheck
setuid=nocheck
conflict=nocheck
action=nocheck
networktimeout=60
networkretries=3
authentication=quit
keystore=/var/sadm/security
proxy=
basedir=default
ADMIN_EOF

#
# Path to your new packages
#
PKG_PATH=<User-specified path for SVR4 package>

#
# Test package name
#
SVR4_TEST_PKG=SUNWmy-test

/usr/sbin/pkgadd -n -a ${ADMIN_FILE} -d $PKG_PATH -R ${PKG_IMG_PATH} ${SVR4_TEST_PKG}

if [ $? != "0" ] ; then
        echo "installing package failed"

        exit 1
fi

/bin/rm ${ADMIN_FILE}

exit 0
```