Oracle® Solaris 11.3 カスタムインストールイメージの作成

Part No: E62531
2016 年 11 月

Part No: E62531

ドキュメントのアクセシビリティについて

オラクルのアクセシビリティについての詳細情報は、Oracle Accessibility ProgramのWeb サイト(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc)を参照してください。

Oracle Supportへのアクセス

サポートをご契約のお客様には、My Oracle Supportを通して電子支援サービスを提供しています。詳細情報は(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info) か、聴覚に障害のあるお客様は (http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs)を参照してください。

目次

このドキュメントの使用法

- **概要** – ディストリビューションコンストラクタツールを使用して、カスタムの Oracle Solaris インストールパッケージを構築する方法を説明します。
- **対象読者** – 技術者、システム管理者、および認定サービスプロバイダ
- **必要な知識** – Oracle Solaris システムの管理経験

製品ドキュメントライブラリ

この製品および関連製品のドキュメントとリソースは `http://www.oracle.com/pls/topic/lookup?ctx=E62101-01` で入手可能です。

フィードバック

このドキュメントに関するフィードバックを `http://www.oracle.com/goto/docfeedback` からお聞かせください。

♦ ♦ ♦ 第 1 章

1 カスタムインストールイメージの作成入門

システム管理者およびアプリケーション開発者は、ディストリビューションコンストラクタツールを使用して、カスタムの Oracle® Solaris インストールイメージを構築できます。

- これまでにカスタムインストールイメージを作成したことがない場合は、9 ページの「ディストリビューションコンストラクタとは」からお読みください。
- カスタムイメージをすぐに構築できる場合は、13 ページの「イメージを構築するためのシステム要件」に進んでください。

ディストリビューションコンストラクタとは

ディストリビューションコンストラクタ (DC) は、事前構成済みの Oracle Solaris イメージを構築するためのコマンド行ツールです。このツールは XML マニフェストファイルを入力として受け取り、マニフェストファイルに指定されているパラメータに基づいて、イメージを構築します。

ディストリビューションコンストラクタは、International Organization for Standardization (ISO) によって定義された形式の光学式ディスクのアーカイブファイルで、ディスクイメージとも呼ばれる、ISO イメージを構築できます。また、生成された ISO イメージに基づいて、USB イメージを作成することもできます。

ディストリビューションコンストラクタは、Oracle Solaris オペレーティングシステムによって提供されるドライバがサポートされるさまざまな種類のフラッシュメモリーデバイスで動作する USB イメージを作成します。`usbcopy` コマンドを使用して、USB イメージを USB フラッシュドライブにコピーする必要があります。`usbcopy` コマンドは `distribution-constructor` パッケージで使用できます。

次の点を確認してください。

- イメージの構成に応じて、ISO イメージや USB イメージをブート可能にすることもできます。

- ISO イメージと USB イメージをシステムにインストールしたり、ライブメディア環境で実行したりできます。
- ISO イメージは、CD または DVD に書き込むことができます。
- USB イメージは、フラッシュ ドライブにコピーできます。

Oracle Solaris インストールイメージの種類

ディストリビューションコンストラクタを使用して、次の種類の Oracle Solaris イメージを作成できます。

- **Oracle Solaris x86 ライブメディア** – それぞれの Oracle Solaris リリースで配布されるライブメディアイメージと同等の x86 ISO イメージを作成できます。たとえば、パッケージを追加または削除することで、この ISO イメージの内容をカスタマイズすることもできます。さらに、作成されたブート環境のデフォルト設定を変更して、カスタムの ISO イメージまたは USB イメージを作成することもできます。

  ライブメディアインストールの詳細は、『Oracle Solaris 11.3 システムのインストール』 の 第 3 章,「Live Media の使用」を参照してください。イメージ内容のカスタマイズについては、15 ページの「マニフェストの内容の変更」を参照してください。
- **Oracle Solaris x86 または SPARC テキストインストールイメージ** – Oracle Solaris オペレーティングシステムのテキストインストールの実行に使用できる SPARC または x86 ISO イメージを作成できます。グラフィックスカードを搭載していないシステムでは、テキストインストーラを使用できます。

  注記 - テキストインストールでは、ライブメディアイメージからインストールするときに含まれるすべてのソフトウェアパッケージがインストールされるわけではありません。たとえば、テキストインストーラではデスクトップはインストールされません。テキストインストールが完了したら、`solaris-desktop` パッケージなどの追加パッケージを追加できます。

  テキストインストールの詳細は、『Oracle Solaris 11.3 システムのインストール』 の 第 4 章,「テキストインストーラの使用」を参照してください。
- **自動インストール用の x86 または SPARC ISO イメージ** – Oracle Solaris オペレーティングシステムには、ネットワークを介して 1 つ以上のシステムで Oracle Solaris OS のインストールを自動化する、自動インストーラ (AI) ツールが含まれています。アーキテクチャー、インストールするパッケージ、ディスク容量、およびその他のパラメータは、インストールごとに異なる場合があります。ディストリビューションコンストラクタを使用して、Oracle Solaris OS を SPARC クライアントにインストールするための SPARC AI ISO イメージ、または Oracle Solaris OS を x86 クライアントにインストールするための x86 AI ISO イメージを作成できます。

Automated Installer については、『Oracle Solaris 11.3 システムのインストール』を参照してください。

イメージ作成のプロセス

ディストリビューションコンストラクタは、マニフェストファイルと呼ばれる XML ファイルに指定された設定に基づいてイメージを作成します。DC マニフェストファイルには、ディストリビューションコンストラクタを使用して作成する ISO イメージの内容およびパラメータの仕様が記述されています。ディストリビューションコンストラクタのパッケージには、カスタム x86 ライブメディア ISO、x86 または SPARC 自動インストール ISO イメージ、x86 または SPARC テキストインストール ISO イメージを作成する際に使用できるサンプルマニフェストが含まれます。14 ページの「サンプル DC マニフェストファイル」を参照してください。

各 DC マニフェストファイルのすべてのフィールドには、必要とする種類のイメージを作成するためのデフォルト値が事前設定されています。マニフェストファイル内のフィールドを編集することで、結果として生成されるイメージをさらにカスタマイズできます。次に例を示します。

- DC マニフェストのターゲット要素を編集して、イメージを構築できる構築領域に別の場所を指定できます。
- 指定されたパブリッシャーを確認し、使用中のシステムが、そのパブリッシャーに連絡してイメージの構築に必要なパッケージをダウンロードできるようにすることが可能です。
- ソフトウェア名要素を編集して、別のパブリッシャーおよびリポジトリの場所を指定できます。

手順については、15 ページの「マニフェストの内容の変更」を参照してください。

カスタムスクリプトを作成して、インストールイメージを変更することもできます。その後、DC マニフェストファイルにチェックポイントを追加して、これらのカスタムスクリプトを実行できます。詳細については、25 ページの「インストールイメージを作成する際のカスタムスクリプトの作成および使用」を参照してください。

ディストリビューションコンストラクタのパッケージには、DC マニフェストの仕様を解釈して、イメージを構築するための `distro_const` コマンドも含まれています。マニフェストファイルでイメージの青写真を編集する作業が完了したあとに、`distro_const` コマンドを実行してイメージを作成します。詳細は、第3章「イメージの構築」を参照してください。

`distro_const` コマンドに用意されているオプションを使用して、構築中のイメージのチェックおよびデバッグを行うために、イメージ生成プロセスのさまざまな段階で構築プロセスを停止および再開できます。構築プロセス中に停止および再開するこ

のプロセスをチェックポイント処理と呼びます。チェックポイント処理はオプションです。デフォルトのチェックポイントは、各 DC マニフェストファイルに指定されます。

`distro_const` コマンドを実行したあと、単純なログファイルまたは詳細なログファイルを確認して構築情報を調べることができます。

詳細は、30 ページの「段階的にイメージを構築する方法」を参照する
か、`distro_const(1M)` のマニュアルページを参照してください。

♦ ♦ ♦ 第 2 章

2 カスタムインストールイメージの設計

この章では、イメージを構築するためのシステム要件、および DC マニフェストとスクリプトを作成してカスタムインストールイメージを設計する方法について説明します。ここでは、次のトピックについて説明します。

イメージを構築するためのシステム要件

ディストリビューションコンストラクタを使用するには、システムが、次の表に示すシステム要件を満たしている必要があります。

表 1 システム要件

要件	説明
ディストリビューションコンストラクタのワークスペースのディスク容量	8G バイト
Oracle Solaris オペレーティングシステム	Oracle Solaris オペレーティングシステム (OS) がシステムにインストールされている必要があります。次の点に注意してください。 ■ インストールされている システムはネットワークにアクセスできる必要があります。ディストリビューションコンストラクタは、ネットワーク上の Image Packaging System (IPS) リポジトリにアクセスして、ISO イメージのパッケージを取得できる必要があります。マニフェストファイル内に指定されたリポジトリへのネットワークアクセスが可能である必要があります。 ■ SPARC システムには SPARC イメージのみ、x86 システムには x86 イメージのみを作成できます。

要件	説明
	■ システム上の Oracle Solaris のリリースバージョンは、ディストリビューションコンストラクタで作成しようとしているイメージのリリースバージョンと同じである必要があります。
必要なパッケージ	ディストリビューションコンストラクタツールが含まれる `distribution-constructor` パッケージ。

サンプル DC マニフェストファイル

`distribution-constructor` パッケージには、次の表に示すサンプルマニフェストファイルがあります。すべてのファイルが `/usr/share/distro_const` にインストールされます。

表 2 サンプル DC マニフェスト

マニフェストの種類	マニフェストの場所	説明
x86 ライブメディア ISO イメージ	`dc_livecd.xml`	Oracle Solaris ライブメディアイメージと同等の x86 ISO イメージを作成する場合に使用されます
x86 テキストインストールイメージ	`dc_text_x86.xml`	x86 Oracle Solaris オペレーティングシステムのテキストインストールを実行する際に使用可能な x86 ISO イメージを作成する場合に使用されます
SPARC テキストインストールイメージ	`dc_text_sparc.xml`	SPARC Oracle Solaris オペレーティングシステムのテキストインストールを実行する際に使用可能な SPARC ISO イメージを作成する場合に使用されます
x86 AI ISO イメージ	`dc_ai_x86.xml`	Oracle Solaris OS を x86 クライアントに自動インストールするための x86 自動インストール ISO イメージを作成する場合に使用されます
SPARC AI ISO イメージ	`dc_ai_sparc.xml`	Oracle Solaris OS を SPARC クライアントに自動インストールするための SPARC 自動インストール ISO イメージを作成する場合に使用されます

DC マニフェスト内の環境変数

次の環境変数を DC マニフェスト内で使用できます。

- `PKG_IMAGE_PATH` - インストールイメージの構築プロセス中に、この変数はパッケージイメージへのパスに解決されます。
- `BOOT_ARCHIVE` - インストールイメージの構築プロセス中に、この変数はインストールイメージへのパスに解決されます。たとえば、アーカイブ内の `/etc/system` ファイルへのパスは `{BOOT_ARCHIVE}/etc/system` です。

マニフェストの内容の変更

各 DC マニフェストファイルのすべてのフィールドには、必要とする種類の ISO イメージを作成するためのデフォルト値が事前設定されています。マニフェストファイル内の事前設定フィールドを手動で編集することで、結果として生成されるイメージをさらにカスタマイズできます。

次のセクションでは、サンプルマニフェストファイル内の主な要素の一覧とこれらの要素の説明を示します。

- `distro` 要素は、構築するイメージの名前および HTTP プロキシを提供するための機能を定義します。16 ページの「イメージタイトルの指定」を参照してください。
- `boot_mods` 要素は、ブートエントリがデフォルトであるブートメニューオプション、およびデフォルトエントリがブートされるまでに待機する期間を定義します。16 ページの「イメージタイトルの指定」を参照してください。
- `target` 要素は、インストールイメージの作成時にインストールイメージを保持する ZFS データセットを定義します。17 ページの「構築領域の指定」を参照してください。
- `software` 要素は、インストールイメージとインストールクライアントの両方のパブリッシャー、およびインストールまたはアンインストールするパッケージを定義します。17 ページの「構築領域の指定」を参照してください。
- `execution` 要素は、インストールイメージの構築時に実行されるチェックポイントを定義します。独自のチェックポイントを追加できます。21 ページの「構築チェックポイントの設定」を参照してください。

▼ インストールイメージを構築する際に使用するマニフェストの選択

この手順では、カスタマイズされたマニフェストを作成するために使用できる汎用の DC マニフェストをインストールする一般的な手順について説明します。

1. **管理者になります。**

   詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

2. **`distribution-constructor` パッケージをインストールします。**

   ```
   # pkg install distribution-constructor
   ```

3. **マニフェストファイルの内容を確認します。**

14 ページの「サンプル DC マニフェストファイル」には、パッケージに含まれているデフォルトのマニフェストの一般的な説明が記載されています。

4. **いずれかのサンプルマニフェストのコピーを作成します。**

```
# cp /usr/share/distro_const/dc_livecd.xml /var/tmp/test.xml
```

次の手順 このあとで必要に応じて、次のセクションで説明されているカスタムスクリプトの追加を含め、新しいマニフェストを編集できます。そのあとで、第3章「イメージの構築」で説明されているようにイメージを作成できます。

イメージタイトルの指定

`distro name` 要素を使用して、構築するイメージのカスタム名またはデフォルト名を指定します。次に例を示します。

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

イメージの構築作業を続けて実行して複数の増分イメージを保持する場合、タイムスタンプ変数を `true` に変更すると、タイムスタンプが各イメージの名前に自動的に追加されます。

HTTP プロキシを指定する必要がある場合、プロキシ変数を含む `distro name` 要素のコメントを解除して、プロキシの場所を指定します。

ブートメニューの変更

`boot_mods` 要素は、イメージに適用するブートメニューの変更を指定します。

次の例では、`boot1` というタイトルの特殊なブートメニューがイメージに適用されます。timeout 属性は、デフォルトのブートエントリが自動的に有効にされるまでの時間を指定します。

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

`boot_mods` 要素内で、新しいエントリごとに新しい `boot_entry` 要素を追加することで個々のブートメニューエントリを追加できます。エントリは、各ブートエントリの `start` または `end` という `insert_at` 属性値によって示される順序でブートメニューに順次追加されます。

注記 - 新しいエントリは、既存の `with magnifier` エントリの前に追加します。

次の例は、サンプルの個々の boot_entry 要素を示しています。

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

詳細は、dc_manifest(4) のマニュアルページを参照してください。

構築領域の指定

イメージが作成される ZFS 構築データセットを定義する、target 要素をカスタマイズできます。有効なデータセットの場所を指定する必要があります。システムで保持する必要がある内容が構築によって破棄されないように、デフォルトの構築領域をチェックします。データセットへのパスは、定義済みのプール名に対して相対的です。次の例では、データセットは rpool/dc/sample-dataset-location としてインストールされます。

注記 - ファイルシステム名には、zpool という名前を含めないでください。

次の例では、サンプルの target 要素を示します。

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

ソフトウェアソースの指定

name 属性が transfer-ips-install に設定された software 要素には、インストールイメージの作成時に使用されるパブリッシャーを定義するセクション、およびイメージでインストールまたはアンインストールされるパッケージのリストが含まれています。name 要素が set-ips-attributes 属性に設定された software 要素は、インストール済みのクライアントが使用するように構成されるパブリッシャーを定義します。詳細については、次のトピックを参照してください。

- 18 ページの「インストールイメージの構築時に使用するパブリッシャーの指定」。
- 18 ページの「インストールするパッケージの一覧表示」。

- 20 ページの「アンインストールするパッケージの一覧表示」。
- 21 ページの「インストール対象システムのパブリッシャーの指定」。

インストールイメージの構築時に使用するパブリッシャーの指定

`transfer-ips-install` セクション内にネストされている `source` 要素内で、パブリッシャー名要素と起点名要素を編集し、使用するパブリッシャーとパッケージリポジトリが存在する場所を指定します。リポジトリの場所は、NFS パスまたはローカルディレクトリとすることができます。複数のパブリッシャーを一覧表示できます。ディストリビューションコンストラクタがインストールするパッケージの検出を試みると、この要素に一覧表示されている順序でパブリッシャーが検索されます。

パブリッシャーのミラーを指定する必要がある場合は、ミラー名要素をコメント解除して編集します。

次の例では、サンプルの source 要素を示します。

```
<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"/>
   </publisher>
   <publisher name="publisher3.org">
       <origin name="http://example3.com/dev"/>
   </publisher>
   <publisher name="publisher4">
       <origin name="file:///net/myserver/publisher4/repo"/>
   </publisher>
</source>
```

パブリッシャーの使用方法の詳細は、『Oracle Solaris 11.3 ソフトウェアの追加と更新』を参照してください。

インストールするパッケージの一覧表示

`install` 属性を持つ `software_data` 要素は、使用しているマニフェストに応じて、特定の種類のイメージを構築するためにインストールされるパッケージのセットを一覧表示します。たとえば、`dc_livecd.xml` マニフェストは、Live Media イメージを構築するために必要なパッケージをリストします。各 name 要素は、1 つのパッケージの名前、または多数のパッケージを含む 1 つのグループパッケージの名前を一覧表示します。

```
<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>
```

イメージにパッケージを追加するには、パッケージごとに name 要素を追加します。

デフォルトでは、指定されたリポジトリで利用できる最新のパッケージバージョンがインストールされます。別のバージョンをインストールする場合は、次の形式を使用してパッケージ参照にバージョン番号を追加します。

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

注記 - 特定のバージョンのパッケージを指定した場合に、競合するバージョンの別のパッケージが自動インストールサービスのマニフェストファイルで指定されたとおりにインストールされていると、パッケージはインストールされない可能性があります。『Oracle Solaris 11.3 システムのインストール』 の 第 9 章,「AI クライアントへのカスタマイズの割り当て」を参照してください。

システム上の Oracle Solaris のリリースバージョンは、ディストリビューションコンストラクタで作成しようとしているイメージのリリースバージョンと同じである必要があります。

例 1 パッケージおよび追加パブリッシャーの追加

この例では、2 番目のパブリッシャーである `mypublisher`、追加のパッケージである `mypackage1` および `mypackage2` が指定されています。

構築プロセス中に、パブリッシャーは一覧表示される順序でチェックされます。1 番目のパブリッシャーでパッケージが見つからない場合は、次のパブリッシャーで指定されたパッケージが検索されます。

```
<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>
```

アンインストールするパッケージの一覧表示

`software_data` 要素を `uninstall` 属性とともに使用して、個々のパッケージまたはグループパッケージ定義をアンインストールできます。

注記 - グループパッケージ定義は、そのグループ内の個々のパッケージすべてを、グループとしてのみ実行できる 1 つの単位に結合します。

`uninstall` 属性は、全部のグループパッケージをインストールするが、1 つ以上の個々のパッケージをそのグループから省略する場合に特に便利です。`uninstall` 属性を使用して、グループパッケージ定義を削除してから、グループパッケージの一部としてインストールされた個々のパッケージをアンインストールできます。

たとえば、デフォルトのライブメディアインストールイメージには、デスクトップグループパッケージ内に Firefox ブラウザが含まれています。構築するイメージから Firefox ブラウザを省略するには、次のことを実行します。

1. 通常のライブメディアデスクトップ用のすべてのソフトウェアを含む `solaris-desktop` グループパッケージをインストールします。18 ページの「インストールするパッケージの一覧表示」を参照してください。
2. `solaris-desktop` グループパッケージ定義をアンインストールします。

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

   このアクションにより、グループパッケージ定義のみがアンインストールされます。これは、個々のパッケージはグループ定義に結合されなくなることを意味します。それでも、そのグループ内の個々のパッケージがすべてインストールされます。
3. Firefox パッケージをアンインストールするには、再度 `uninstall` 属性を使用します。

   ```
   <software_data action="uninstall">
      <name>pkg:/web/browser/firefox</name>
   </software_data>
   ```

ヒント - ステップ 2 と 3 を次のようにして 1 つのエントリに組み合わせることができます。

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

アンインストールする追加のパッケージを software_data 要素のアンインストールセクションの終わりに追加します。

インストール対象システムのパブリッシャーの指定

ディストリビューションコンストラクタを使用して作成されたイメージでシステムがインストールされたあと、transfer-ips-install 属性内にネストされた source 要素がシステムに影響を与えます。これは、インストールされているシステムが追加のパッケージにアクセスできる場所を指定するために、パブリッシャー名やオプションのミラー名などの情報を指定します。

この要素では、IPS 属性も設定できます。pkg(1) のマニュアルページの IPS プロパティー情報を参照してください。

構築チェックポイントの設定

execution 要素は、イメージ作成処理中に実行される一連のチェックポイントを一覧表示します。チェックポイントは、それらがこのセクションに一覧表示されている順序で実行されます。それぞれのマニフェストには、デフォルトのインストールイメージの構築に必要なデフォルトのチェックポイントが含まれています。

イメージ作成処理中、チェックポイントはマニフェストに指定されている構築領域の内容を変更します。

特定のチェックポイントで構築処理の一時停止と再開を制御するには、distro_const コマンドオプションを使用します。30 ページの「段階的にイメージを構築する方法」を参照してください。

構築領域には、pkg_image および boot_archive ディレクトリが含まれています。構築プロセス中、最終的なイメージに含められるすべての内容は、pkg_image ディレクトリに追加されます。異なる boot_archive ディレクトリ内にあるファイルは構築プロセス中に使用されてブートアーカイブファイルが作成され、これも pkg_image ディレクトリに追加されます。

各デフォルトのチェックポイントの次の簡単な説明は、ほとんどのマニフェストでチェックポイントが実行される順序で示されています。

- transfer-ips-install – このチェックポイントでは、ディストリビューションコンストラクタは IPS のパブリッシャーと連絡を取り、transfer-ips-install 属性を使用して software 要素に一覧表示されているパッケージをイメージに追加します。

- `set-ips-attributes` – このチェックポイントでは、コンストラクタはインストール対象システムによって使用されるパブリッシャーを設定します。これらの値は、`set-ips-attributes` 属性を指定して `software` 要素で設定されます。このチェックポイントによって設定される値は、自動インストールイメージを作成する場合は関係ありません。AI クライアントは、AI サーバーのパブリッシャーを使用するように構成されます。
- `pre-pkg-img-mod` – このチェックポイントでは、コンストラクタはマニフェストの `configuration` 要素で指定された SMF サービスファイルをイメージにインポートします。また、コンストラクタはイメージを最適化するために一部のファイルを変更します。パッケージに関連するすべての操作およびチェックポイントは、このチェックポイントよりも前に置く必要があります。
- `ba-init` – このチェックポイントでは、コンストラクタはマニフェストの `ba-init` セクションに一覧表示されているファイルをルートアーカイブに取り込みます。これらのファイルは `pkg_image` 領域から `root_archive` 領域にコピーされます。

  このチェックポイントまでに行われたすべての変更は、構築されるイメージと、ルートアーカイブの両方に含められます。カスタムスクリプトからの変更が、ルートアーカイブとイメージの両方に組み込まれていることを確認する場合は、このチェックポイントの前にカスタムスクリプト用の新規チェックポイントを追加してください。
- `ba-config` – このチェックポイントでは、コンストラクタはルートアーカイブにコピーされたファイルに追加の変更を実行します。コンストラクタはルートアーカイブのサイズを最小限に抑えるために、ブートプロセスの後半まで不要な他のファイルへのシンボリックリンクを作成します。
- `ba-arch` – このチェックポイントでは、コンストラクタはルートアーカイブをパックし、`pkg_image` ディレクトリ内でルートアーカイブをファイルとして作成します。またコンストラクタは、構築されるシステムの種類に固有となるルートアーカイブへのすべての最適化を適用します。このチェックポイント以降では、ルートアーカイブはすでにパックされているため、カスタムスクリプトによるブートアーカイブ指定への変更はルートアーカイブに組み込まれません。
- `boot-setup` – このチェックポイントでは、コンストラクタはマニフェストの `boot_entry` セクションで指定されるエントリに基づいて、GRUB2 メニューを設定します。このチェックポイントは、x86 システムのイメージにのみ適用されます。
- `pkg-img-mod` – このチェックポイントでは、コンストラクタは構築されるイメージのメインアーカイブを作成し、`pkg_image` 領域を最適化します。コンストラクタは `pkg_image` ディレクトリ内のファイルを移動し、イメージ用のアーカイブを作成します。`pkg_image` ディレクトリに含まれるすべての内容がイメージに含められます。このチェックポイント以降のすべての追加はイメージに含められません。
- `create-iso` – このチェックポイントは、`pkg_image` ディレクトリに含まれるすべての内容を含む `.iso` ファイルを構築します。
- `create-usb` – このチェックポイントは、生成された `.iso` ファイルから `.usb` ファイルを構築します。

それぞれのチェックポイント要素には、チェックポイントスクリプトがある場所を指定する mod-path 属性が含まれています。また、一部のデフォルトのチェックポイントには、デフォルト値を持つ引数が含まれています。

dc_ai_sparc.xml サンプルマニフェストの次のチェックポイント例では、イメージ構築のためのブートアーカイブを作成し、イメージを構築するスクリプトを指定します。また、引数ごとに特定の値が指定された引数フィールドも含まれています。

```
<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>
```

この例で示すように、kwargs 要素には、構築中にチェックポイントに渡す必要があるキーワード引数が含まれています。kwargs 要素には、チェックポイントに渡される個々のキーワードを指定する際に使用可能な arg name 要素が含まれています。arglist 要素には、チェックポイントに渡される複数の argitem 値の一覧も含まれています。この例には、arglist 要素の未圧縮ファイルの一覧が含まれています。

それぞれの kwargs リスト項目は二重引用符で囲まれています。二重引用符が使用されていない場合、または文字列全体が 1 組の二重引用符で囲まれている場合は、スペースおよび改行を含む文字列全体が 1 つの引数と解釈されます。引数をコンマで区切らないでください。

イメージの構築中に使用されるカスタムスクリプトを作成する場合、スクリプトの場所を指示するチェックポイント要素を追加する必要があります。カスタムスクリプトのチェックポイントには、カスタムスクリプトの場所を示す args 要素のみが必要です。詳細および例については、25 ページの「インストールイメージを作成する際のカスタムスクリプトの作成および使用」を参照してください。

例 2 インストールイメージへの SVR4 パッケージの追加

この例では、新しいチェックポイントがマニフェストに追加されます。この新しいチェックポイントは、イメージに追加される SVR4 パッケージとその場所を一覧表示します。その後、この新しいチェックポイントは実行セクションで参照されます。

最初に、新規 `software` 要素を追加することによって、新しいチェックポイントが作成されます。このチェックポイントには、ソフトウェアタイプとして SVR4、パッケージを検索する場所、およびパッケージをインストールする場所を指定します。

さらに、`software_data` 要素には、インストールされる特定の SVR4 パッケージが一覧表示されます。

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

`{PKG_IMAGE_PATH}` および `{BOOT_ARCHIVE}` の値は、チェックポイントに含まれる場合、`distro_const` コマンドによって、それぞれパッケージイメージとブートアーカイブの構築領域へのパスに置き換えられます。この例では、SVR4 パッケージはパッケージイメージディレクトリにインストールされます。

最後に、この新しいチェックポイントは実行セクションで参照されます。チェックポイント名は任意の文字列ですが、この例では `checkpoint_class` 値は `TransferSVR4` である必要があります。

```
<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"/>
```

ソフトウェア名はチェックポイント名と一致する必要があることに注意してください。この例では、どちらも「transfer-svr4–install」です。

例 3 インストールイメージ内のメディアのハッシュ作成

checksums チェックポイントを指定すると、ユーザーは、distro_const コマンドによって生成されるメディアのハッシュを自動生成できます。

```
<checkpoint name="checksums"
  desc="Checksum calculation for media"
  mod_path="solaris_install/distro_const/checkpoints/checksums"
  checkpoint_class="Checksums">
  <kwargs>
    <arglist name="algorithms">
      <argitem file_path="/tmp/md5sums.txt">md5</argitem>
      <argitem>sha1</argitem>
      <argitem>sha224</argitem>
      <argitem>sha256</argitem>
      <argitem>sha384</argitem>
      <argitem>sha512</argitem>
    </arglist>
  </kwargs>
</checkpoint>
```

arglist 要素には、生成されたメディアのハッシュを生成するために使用されるすべてのアルゴリズムが含まれます。各 argitem はアルゴリズムを指定します。有効なアルゴリズムを確認するには /usr/bindigest -l コマンドを実行します。各 argitem には、そのアルゴリズムによって生成されるハッシュの末尾に追加される、追加ファイルの絶対パスを指定する path 属性を指定できます。アルゴリズムを指定しない場合、デフォルトは md5 です。

イメージの構築中、アルゴリズムごとに、各メディアのチェックサムが格納されたファイルが生成されます。

インストールイメージを作成する際のカスタムスクリプトの作成および使用

ディストリビューションコンストラクタでは、構築するイメージの種類に基づいて、イメージ作成プロセス中にカスタマイズするために使用する追加スクリプトを指定できます。マニフェストファイルはスクリプトを示し、スクリプトは汎用イメージをメディア固有の配布用に変換します。これらのスクリプトは、マニフェストファイルの実行セクションで参照されます。任意の数の custom-script チェックポイントを指定できます。

多くの場合、カスタムスクリプトは、構成ファイルを変更するか、マニフェストを使用して行うことができないその他の変更を行うために使用されます。

マニフェストファイルの execution セクションで指定されているスクリプトは、イメージ作成プロセス中に実行されます。実行セクションは、インストール前処理スクリプトまたはインストール後処理スクリプトを参照しません。

注記 - パッケージからインストールされたスクリプトを変更しないでください。将来のパッケージ更新で問題が発生しないようにするには、作成するスクリプトで変更を行なってください。

独自のカスタムスクリプトを作成する場合は、次の点に注意してください。

- スクリプトには、Python プログラム、シェルスクリプト、またはバイナリを使用できます。
- スクリプトは、マニフェストファイルの execution セクションに記載されている順に実行されます。
- スクリプト (シェルと Python の両方のモジュール) 内で実行されるコマンドの標準出力 (stdout) およびエラー出力 (stderr) が、構築の完了時または試行時に報告するログファイルで取得されます。

▼ インストールイメージを作成する際にカスタムスクリプトを作成および使用する方法

1. **新しいスクリプトを作成します。**

2. **新しいスクリプトをホームディレクトリ、またはシステム上かネットワーク上のほかの場所に追加します。**

   root 役割になるユーザーがこのスクリプトを実行できることを確認します。

3. **マニフェストを変更します。**

   新しいスクリプトを参照する情報をマニフェストの execution セクションに追加します。新しいチェックポイントを追加する場所を確認するには、21 ページの「構築チェックポイントの設定」を参照してください。

   スクリプトには必ずフルパスを指定してください。チェックポイントは、マニフェストの execution セクションに記載されている順に実行されます。

   マニフェストファイルの execution セクションに新しいスクリプトの参照を追加するときには、そのスクリプトのタスクの実行前または実行後にイメージ構築を一時停止するのに使用できるチェックポイント名を指定する必要があります。必要に応じて、チェックポイント名に関連付けたカスタムメッセージを指定することもできます。このメッセージを省略すると、スクリプトのパスがデフォルトのチェックポイントメッセージとして使用されます。構築プロセス中にチェックポイントが実行されると、チェックポイントメッセージが表示されます。

注記 - チェックポイント名には、序数ではなく意味のある名前を使用してください。数値を使用する場合、新しいスクリプトの新しいチェックポイントを追加すると、番号を使用したチェックポイントの順序が乱れます。

次のチェックポイントの例では、`my-script` という名前のカスタムスクリプトが参照されます。

```
<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. **イメージを構築します。**

   イメージを 1 ステップで構築したり、さまざまなチェックポイントで構築を停止および再開して構築のステータスをチェックしたりできます。

   手順については、第3章「イメージの構築」を参照してください。

5. **(オプション) 構築が完了したら、構築プロセスのログファイルを確認します。**

   構築出力には、ログファイルの場所が表示されます。

例 4 チェックポイントでの環境変数の使用

次の例では、イメージディレクトリのパスは `myscript.sh` への引数として使用されています。

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

例 5 カスタム DC マニフェストに短いスクリプトを含める

次のカスタムスクリプトは、どのユーザーも `jack` というユーザーとしてログインできないようにします。これを変更して、パスワードを追加したり行を削除したりすることもできます。このチェックポイントは DC マニフェストの特定の場所に追加されたことに注意してください。スクリプトは、パッケージがインストールされてから boot_archive が作成されるまでの間に実行する必要があります。

```
    <checkpoint name="set-ips-attributes"
        desc="Set post-install IPS attributes"
        mod_path="solaris_install/transfer/ips"
        checkpoint_class="TransferIPS"/>
    </checkpoint>
    <checkpoint name="lock-jack-account"
```

```
        desc="Lock the jack account from login"
        mod_path="solaris_install/distro_const/checkpoints/custom_script"
        checkpoint_class="CustomScript">
        <args>sed 's/jack:.[^:]*:/jack:*LK*:/g' {PKG_IMAGE_PATH}/etc/shadow
> {PKG_IMAGE_PATH}/etc/shadow.new; cp {PKG_IMAGE_PATH}/etc/shadow.new
{PKG_IMAGE_PATH}/etc/shadow; rm {PKG_IMAGE_PATH}/etc/shadow.new</args>
    </checkpoint>
    <checkpoint name="pre-pkg-img-mod"
```

次のスクリプトは、インストールプロセス中にインストールクライアントにアクセスするために使用できる `solaris` ユーザーのパスワードを設定します。

```
    <checkpoint name="set-ips-attributes"
        desc="Set post-install IPS attributes"
        mod_path="solaris_install/transfer/ips"
        checkpoint_class="TransferIPS"/>
    </checkpoint>
    <checkpoint name="solaris-password"
        desc="Set the password for the solaris account used during the installation
 process"
        mod_path="solaris_install/distro_const/checkpoints/custom_script"
        checkpoint_class="CustomScript">
        <args>sed 's/solaris:.[^:]*:/solaris:string:/g' {PKG_IMAGE_PATH}/etc/shadow
> {PKG_IMAGE_PATH}/etc/shadow.new; cp {PKG_IMAGE_PATH}/etc/shadow.new
{PKG_IMAGE_PATH}/etc/shadow; rm {PKG_IMAGE_PATH}/etc/shadow.new</args>
    </checkpoint>
    <checkpoint name="pre-pkg-img-mod"
```

♦ ♦ ♦ 第 3 章

3 イメージの構築

使用するマニフェストファイルの準備が完了し、必要に応じてファイナライザスクリプトのカスタマイズが完了したら、`distro_const` コマンドを実行してイメージを構築できます。

`distro_const` コマンドを使用すると、イメージを 1 ステップで構築したり、必要に応じて構築を停止して再開して、構築プロセス中にイメージの内容を検査してスクリプトをデバッグしたりできます。

この章では、次の手順について説明します。

`distro_const` コマンド

次の表で、`distro_const` コマンドのオプションを説明します。

表 3 `distro_const` コマンドのオプション

コマンドオプション	説明
`distro_const build` *manifest*	指定されたマニフェストファイルを使用して、1 ステップでイメージを構築します。
`distro_const build -v` *manifest*	冗長モード
`distro_const build -l` *manifest*	イメージの構築を一時停止または再開できるすべての有効なチェックポイントを一覧表示します。
`distro_const build -p` *checkpoint manifest*	指定されたチェックポイントでイメージの構築を一時停止します。
`distro_const build -r` *checkpoint manifest*	指定されたチェックポイントからイメージの構築を再開します。
`distro_const build -h`	コマンドのヘルプを表示します。

注記 - `distro_const` コマンドを使用するには、`root` 役割になる必要があります。

インストールイメージの構築

このセクションでは次の作業について説明します。

- 30 ページの「1 ステップでイメージを構築する方法」
- 30 ページの「段階的にイメージを構築する方法」

▼ 1 ステップでイメージを構築する方法

始める前に インストールイメージを構築する前に、使用するマニフェストを選択しておく必要があります。15 ページの「インストールイメージを構築する際に使用するマニフェストの選択」を参照してください。

1. **管理者になります。**

   詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

2. **インストールイメージを構築します。**

   ```
   # distro_const build /var/tmp/test.xml
   ```

   ディストリビューションコンストラクタはイメージに必要なパッケージを取得し、DC マニフェストファイル内で設定する仕様に合わせてイメージを構築します。

3. **(オプション) 構築プロセスのログファイルを表示します。**

   構築出力には、ログファイルの場所が表示されます。

▼ 段階的にイメージを構築する方法

`distro_const` コマンドオプションを使用して、構築するイメージのファイル、パッケージ、およびスクリプトのチェックやデバッグを行うために、イメージ生成プロセスのさまざまなチェックポイントで構築プロセスを停止および再開できます。

始める前に インストールイメージを構築する前に、使用するマニフェストを選択しておく必要があります。15 ページの「インストールイメージを構築する際に使用するマニフェストの選択」を参照してください。

1. **管理者になります。**

   詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

2. **構築を一時停止または再開するために選択できる有効なチェックポイントを表示します。**

```
# distro_const build -l /var/tmp/test.xml


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                       X Boot archive initialization
ba-config                     X Boot archive configuration
ba-arch                       X Boot archive archiving
boot-setup                      Setup LiveCD boot menu
pkg-img-mod                     Package image area modifications
create-iso                      ISO image creation
create-usb                      USB image creation
```

   注記 - このサンプルコマンド出力の「Resumable」列の「X」は、そのチェックポイントから構築を再開できることを示します。

3. **イメージを構築し、指定されたチェックポイントで構築を一時停止します。**

   次のコマンドは、イメージの構築を開始し、`ba-arch` でイメージ領域が変更される前に構築を一時停止します。

```
# distro_const build -p ba-arch /var/tmp/test.xml
```

4. **指定されたチェックポイントからイメージの構築を再開します。**

   注記 - 指定するチェックポイントは、前に構築の実行を停止したチェックポイント、またはそれ以前のチェックポイントにしてください。それより後のチェックポイントは無効です。

   次のコマンドは、`ba-arch` ステージでイメージの構築を再開します。

```
# distro_const build -r ba-arch /var/tmp/test.xml
```

   注記 - `build` コマンドでは、一時停止オプションと再開オプションを組み合わせることができます。

5. **(オプション) 構築プロセスのログファイルを表示します。**

   構築出力には、ログファイルの場所が表示されます。

索引

は

ま