Oracle® Solaris 11.2 カスタムインストールイメージの作成

Part No: E53756
2014 年 7 月

ORACLE

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
- **前提知識** – ある程度の Oracle Solaris の使用経験

製品ドキュメントライブラリ

この製品の最新情報や既知の問題は、ドキュメントライブラリ (http://www.oracle.com/pls/topic/lookup?ctx=E56342) に含まれています。

Oracle サポートへのアクセス

Oracle のお客様は、My Oracle Support を通じて電子的なサポートを利用することができます。詳細は、http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info を参照してください。聴覚に障害をお持ちの場合は、http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs を参照してください。

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、7 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、11 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は、事前構成済みの Oracle Solaris イメージを作成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、ISO イメージを構築できます。ISO イメージは、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。

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 イメージは、どちらも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。

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サポートを持つ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。USB イメージを USB フラッシュドライブにコピーするには、`usbcopy` ユーティリティーを使用する必要があります。この `usbcopy` ユーティリティーは、`distribution-constructor` パッケージに付属しています。

Oracle Solaris 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** - 各 Oracle Solaris リリースと一緒に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。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。たとえば、パッケージを追加または削除できます。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については、『Oracle Solaris 11.2 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、14 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を実行するために使用できる SPARC または x86 ISO イメージを作成できます。テキストインストーラは、グラフィックカードを必要としないシステムで使用できます。

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 11.2 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自動インストールツールが含まれています。自動インストーラ (AI) は、1 つ以上の SPARC および x86 システムにネットワーク経由で Oracle Solaris OS を自動インストールするために使用します。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。

自動インストーラの使用の詳細は、『Oracle Solaris 11.2 システムのインストール 』のパート III「インストールサーバーを使用したインストール」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。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。12 ページの「サンプルマニフェストファイル」を参照してください。

各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。

- 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できます。
- 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、12 ページの「イメージのカスタマイズ」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、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。詳細は、24 ページの「カスタムスクリプトの作成と使用」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、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コマンド行ユーティリティーである `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後に、`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

び再開することができます。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、28 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。

SPARC と x86 のアーカイブの違い

x86 イメージのルートアーカイブと SPARC イメージのルートアーカイブは異なります。x86 イメージのルート全体のアーカイブである `boot_archive` は、UFS ファイルシステムであり、`lzma` を使用して圧縮されます。SPARC プラットフォームでは、ルート全体のアーカイブの圧縮はサポートされません。その代わりに、SPARC のルートアーカイブでは DCFS が使用され、各ファイルが個別に圧縮されます。個別に圧縮されたファイルは、マニフェストで特定の処理が必要になる場合があります。手順については、`dc_manifest`(4)のマニュアルページの `<boot_archive_contents>` フィールド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、システム要件を示し、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。

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、次の表に示すシステム要件が必要です。

表 2-1 システム要件

要件	説明
ディスク領域	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推奨最小サイズは 8G バイトです。システムにディストリビューションコンストラクタを使用するための十分なディスク容量があることを確認してください。
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、ネットワーク経由で利用できる 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します。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。 ■ ディストリビューションコンストラクタを使用している場合は、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。 ■ 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。 **注記 -** ディストリビューションコンストラクタを実行するには、`root` 役割になる必要があります。
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。

イメージのカスタマイズ

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。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。`distribution-constructor` 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提供されています。

各マニフェストファイルの要素には、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要素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カスタマイズできます。また、カスタムスクリプトを作成すれば、さらにイメージを変更できます。次に、マニフェストファイルで新しいスクリプトを参照してください。

サンプル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。

表 2-2 サンプルマニフェスト

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	`/usr/share/distro_const/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	`/usr/share/distro_const/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	`/usr/share/distro_const/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	`/usr/share/distro_const/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	`/usr/share/distro_const/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

▼ カスタムイメージの作成方法と構築方法

この手順では、カスタムイメージを作成および構築する一般的なステップについて説明します。

1. **管理者になります。**
   詳細は、『Oracle Solaris 11.2 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. **ディストリビューションコンストラクタツールおよびサンプルマニフェストが含まれる `distribution-constructor` パッケージをインストールします。**

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

3. **サンプルマニフェストをコピーし、新しいファイル名を付けてカスタムのマニフェストファイルを作成します。**
   `distro_const` コマンドを使用してイメージを作成するときに、このマニフェストファイルの名前を指定します。

   注記 - 元のマニフェストファイルおよびデフォルトのスクリプトをコピーするときは、その前に必ずバックアップを作成してください。

4. **マニフェスト要素を編集します。**
   たとえば、マニフェストのターゲット要素を編集すれば、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。さらに、パブリッシャーをチェックして、システムが、イメージの構築に必要なパッケージをダウンロードするために、そのパブリッシャーに連絡できるようにすることができます。必要に応じて、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。
   詳細は、14 ページの「マニフェストの内容の変更」および`dc_manifest`(4)のマニュアルページを参照してください。

5. **(オプション) イメージをさらに変更するには、カスタムスクリプトを作成します。**
   新しいスクリプトを作成する場合は、マニフェストファイル内の実行セクションのスクリプト参照を更新します。
   詳細は、24 ページの「カスタムスクリプトの作成と使用」を参照してください。

6. **`distro_const` ユーティリティーを実行して、イメージを作成します。**
   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。

マニフェストの内容の変更

各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。

次の表で、サンプルマニフェストファイルの主な要素について説明します。

表 2-3 マニフェスト要素

要素	説明
<distro name="Oracle_Solaris_Text_X86" add_timestamp="false">	オプションのタイムスタンプを使用してイメージ名を指定します
<boot_mods>	イメージ用の GRUB メニュー変更を指定します
<target>	イメージが構築される ZFS 構築データセットを定義します
<software name="transfer-ips-install" type="IPS">	インストールするソフトウェアパッケージのソースを指定します
<software_data action="install">	インストールするパッケージを一覧表示します
<software_data action="uninstall">	アンインストールするパッケージを一覧表示します
<software name="set-ips-attributes">	インストールが完了した後、IPS の各種属性を設定します
<software name="ba-init">	ブートアーカイブの内容を指定します **注意 -** 注意して変更してください。ブートアーカイブが正しくない場合は、インストールされたシステムのブートに失敗します。
<execution stop_on_error="true"> <checkpoint name="transfer-ips-install"/>	構築チェックポイントを一覧表示します
<configuration name="pre-pkg-img-mod" type="sysconf" source="/etc/svc/profile/generic_limited_net.xml">	構築中にメディアに適用される SMF サービスを指定します **注意 -** できるだけ変更しないでください。

イメージタイトルの指定

次の要素を使用して、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。

```
<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

このブートメニュー要素はイメージに適用され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。タイムアウト属性は、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">
```

ブートメニュー要素内では、新しい各エントリに対して新しい boot_entry 要素を追加することによって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」の insert_at 属性値に基づいた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。

個々の boot_entry 要素については、次の例を参照してください。

```
<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

target 要素はカスタマイズできます。この要素は、構築に使用する ZFS 構築データセットを定義します。このデータセットは、イメージが作成される場所です。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する必要があります。必要に応じて、構築領域を変更します。

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、サンプルのターゲット要素を示します。

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

パブリッシャーの指定

次の要素には、イメージ構築のためにダウンロードおよび使用するパッケージをディストリビューションコンストラクタが取得できるパブリッシャーを指定します。

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

このソフトウェア名のセクション内にネストされているソース要素内で、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、ここ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。

次の例は、ソフトウェア名要素の内部で見られるサンプルソース要素を示します。

```
<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 11.2 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、ライブメディアイメージの構築に必要なパッケージを一覧表示します。各名前タグは、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>
```

イメージに追加するパッケージがある場合、パッケージごとに名前タグを追加することによってパッケージ名を追加します。

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。他のバージョンが必要な場合、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 - 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。

また、自動インストールサービスのマニフェストファイルでの指定に従って、競合するバージョンを持つ他のパッケージがインストールされている場合は、指定された特定のバージョンのパッケージがインストールされない可能性があります。『Oracle Solaris 11.2 システムのインストール 』の第 9 章「インストールのカスタマイズ」を参照してください。

例 2-1 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、mypublisher という 2 番目のパブリッシャーが指定されます。mypackage1 および mypackage2 という追加パッケージも指定されます。

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

アンインストール属性を持つ `software_data` 要素は、個々のパッケージのアンインストールまたはグループパッケージ定義のアンインストールに使用できます。

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。

アンインストール属性は、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したい場合に特に便利です。アンインストール属性を使用して、最初にグループパッケージ定義を削除することができます。次に、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、ライブメディアインストールイメージを構築することを選択する場合があります。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場合、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。17 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. アンインストール属性を次のように使用して、`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>
```

注記 - グループパッケージに対するアンインストールアクションは、グループパッケージ定義のみアンインストールします。そのグループ内の個々のパッケージは、最初のステップに従ってインストールされたままになります。

---

3. これで個々のパッケージがグループ定義に結合されなくなったため、アンインストール属性を再度使用して Firefox パッケージをアンインストールできます。

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

   また、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もできます。

   ```
   <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パッケージを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後、`software name` 要素がシステムに影響を与えます。

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

ダウンロードおよびインストールする追加パッケージにインストール済みシステムがアクセスできる場所を指定するための、パブリッシャー名とオプションのミラー名のタグを提供します。

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

マニフェストの 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、このセクションに一覧表示された順序で実行されます。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は、各マニフェストに含まれています。

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。

構築領域には、次のディレクトリがあります。

- *ZFS dataset*/build_data/pkg_image
- *ZFS dataset*/build_data/boot_archive

ここで、*ZFS dataset* 変数は、マニフェストのターゲット要素で指定されます。

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。

次の一覧表示は、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に、デフォルトの各チェックポイントの簡単な説明を提供します。

- 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、マニフェストの software_data 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。
- 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。

  このチェックポイントまでのすべての変更は、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 pre-pkg-img-mod チェックポイントの前か直後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するようにします。
- 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。

- 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 ファイルを作成します。
- 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。

各チェックポイントセクションに含まれている特定のフィールドを見ると、各チェックポイント名のタグにはチェックポイントスクリプトの場所を指定する mod-path 属性が含まれています。

一部のデフォルトチェックポイントタグには、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、そのタスクを実行するスクリプトを指定します。チェックポイント例には、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。さらに、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。

各 kargs 一覧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、24 ページの「カスタムスクリプトの作成と使用」を参照してください。

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。28 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。

例 2-2 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 ユーティリティーによって *ZFS dataset* /build_data/pkg_image および *ZFS dataset* /build_data/boot_archive にそれぞれ置き換えられます。この例では、SVR4 パッケージが *ZFS dataset* /build_data/pkg_image 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。チェックポイントには任意の文字列を指定できますが、この例では、checkpoint_class 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 2-3 メディアのハッシュの作成

このチェックポイントを指定すると、distro_const が、DC マニフェストのチェックサムチェックポイントを使用して、メディアのハッシュを自動生成するようにできます。

```
<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` リストには、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` 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。

カスタムスクリプトの作成と使用

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。カスタムスクリプトチェックポイントはいくつでも指定できます。

注記 - スクリプトのサポートは、アプリケーションパッケージに付属している未変更のデフォルトのスクリプトに限定されます。デフォルトのスクリプトをカスタマイズする場合は、まず、元のスクリプトのバックアップを作成してください。

また、マニフェストファイルの実行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ることに注意してください。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。

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実行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。

▼ カスタムスクリプトの作成方法と使用方法

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他の場所に追加します。**

   `root` 役割になるユーザーがこれらのスクリプトを実行できます。

3. **チェックポイントを適切なマニフェストファイルの実行セクションに追加することによって、新しいスクリプトを参照します。**

   新しいチェックポイントの設定先を決定するには、20 ページの「構築チェックポイントの設定」に記載されているデフォルトチェックポイントの説明を確認してください。

   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、マニフェストの実行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完了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す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数値でなく意味のある名前を使用してください。新しいスクリプトが追加された場合に、新しいスクリプトの新しいチェックポイントが使われるため、番号を使用したチェックポイントの秩序は乱れます。

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. **(オプション) 次のように、構築パラメータをチェックポイントの一部として指定します。**

   ここで、引数セクションの構築パラメータとして `{PKG_IMAGE_PATH}` が指定されます。

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

   `{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` ユーティリティーによって *ZFS dataset* `/build_data/pkg_image` および *ZFS dataset* `/build_data/boot_archive` にそれぞれ置き換えられます。

5. **イメージを構築します。**

   イメージは 1 ステップで構築できます。または、構築のステータスをチェックするために、さまざまなチェックポイントで構築を停止および再開できます。

   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。

6. **(オプション) 構築が完了したら、構築プロセスで報告するログファイルを参照できます。**

   構築出力には、ログファイルの場所が表示されます。

♦ ♦ ♦ 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 コマンドを使用して、イメージを 1 ステップで構築できます。あるいは、必要に応じて構築を一時停止したり再開したりすることで、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することができます。

distro_const コマンド

distro_const コマンドの完全な構文は、次のとおりです。

```
distro_const build [-v] [-r checkpoint] [-p checkpoint] [-l] manifest
```

distro_const コマンドのオプションについては、次の表で説明します。

表 3-1 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。

▼ 1 ステップでイメージを構築する方法

1. **管理者になります。**

   詳細は、『Oracle Solaris 11.2 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` パッケージをダウンロードします。**

3. **イメージに対するマニフェストを選択します。**

4. **(オプション) 必要な場合、マニフェストをカスタマイズし、カスタムスクリプトへの参照を追加します。**

5. **オプションを指定せずに基本の `distro_const` コマンドを発行します。**

   ```
   # distro_const build manifest.xml
   ```

   manifest は、イメージの青写真として使用するマニフェストファイルの名前に置き換えてください。

   次に例を示します。

   ```
   # distro_const build /usr/share/distro_const/dc_livecd.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、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。

6. **(オプション) 構築が完了したら、構築プロセスで報告するログファイルを参照できます。**

   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` コマンドでは、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するためのオプションを使用できます。

1. **管理者になります。**

   詳細は、『Oracle Solaris 11.2 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` パッケージをダウンロードします。**

3. **イメージに対するマニフェストを選択します。**

4. **(オプション) 必要な場合、マニフェストをカスタマイズし、カスタムスクリプトへの参照を追加します。**

5. **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確認します。**

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

   このコマンドを実行すると、イメージの構築を一時停止または再開できる有効なチェックポイントが表示されます。このコマンドによって表示されたチェックポイント名を、他のチェックポイント処理コマンドオプションの有効な値として使用してください。

   たとえば、次のコマンドでは、`dc_livecd.xml` という名前のマニフェストファイルに対して使用可能なチェックポイントを確認します。

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

6. **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。**

   ```
   # distro_const build -p checkpoint manifest
   ```

たとえば、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 /usr/share/distro_const/dc_livecd.xml
```

7. **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 -r checkpoint manifest
```

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。

たとえば、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 /usr/share/distro_const/dc_livecd.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。

---

8. **(オプション) 構築が完了したら、構築プロセスで報告するログファイルを参照できます。**

構築出力には、ログファイルの場所が表示されます。

索引