Oracle® Solaris 11 インストールマニュアルページ

Part No: E25756
2012 年 3 月、E25756-03

ORACLE

120411@25097

目次

はじめに

このドキュメントは、Oracle Solaris 11 システムインストールツールのマニュアルページを提供します。

概要

次に、マニュアルページの各セクションと、そこに含まれている情報について簡単に説明します。

- セクション 1 では、オペレーティングシステムで利用可能なコマンドについて説明します。
- セクション 1M では、システムの保守および管理目的で主に使用するコマンドについて説明します。
- セクション 5 には、文字セット表などのその他のドキュメントが含まれています。

以下に、マニュアルページの一般的な形式を示します。一般に、各マニュアルセクションのマニュアルページはこの順序に従いますが、必要な見出しのみが含まれています。たとえば、レポートするバグがない場合は、「使用上の留意点」セクションはありません。一般的なマニュアルページの詳細については、`man` コマンドを参照してください。

名前
: このセクションでは、ドキュメント化されたコマンドまたは関数の名前に続いて、その動作について簡単な説明を示します。

形式
: このセクションでは、コマンドまたは関数の構文を示します。コマンドまたはファイルが標準パスに存在しない場合は、そのフルパス名が表示されます。異なる引数順序が必要でないかぎり、オプションおよび引数はアルファベット順 (1 番目に単一文字の引数、次に引数が付いたオプション) で表示されます。

  このセクションでは、次の特殊文字が使用されています。

  [ ]
  : 角括弧。角括弧で囲まれたオプションまたは引数は省略可能です。角括弧を省略した場合は、引数を指定する必要があります。

. . .	省略記号。前の引数に複数の値を指定したり、前の引数を複数回指定したりできます (例: "`filename...`")。
|	区切り文字。この文字で区切られた引数のうち 1 つのみを一度に指定できます。
{ }	中括弧。中括弧で囲まれたオプション・引数は相互に依存しており、1 つの組として取り扱う必要があります。

プロトコル　このセクションは、サブセクション 3R でのみ発生し、プロトコルの説明ファイルを示します。

機能説明　このセクションでは、サービスの機能および動作を定義します。つまり、コマンドの動作について簡潔に説明しています。オプションについて説明したり、例を示したりはしていません。対話型のコマンド、サブコマンド、要求、マクロ、および関数については、「使用法」で説明されています。

オプション　このセクションでは、各オプションの動作についての簡潔なサマリーとともに、コマンドのオプションを一覧表示します。オプションは文字順および「形式」セクションで表示される順序で一覧表示されます。オプションに指定可能な引数については「オプション」で説明され、該当する場合はデフォルト値が示されます。

オペランド　このセクションでは、コマンドのオペランドを一覧表示し、コマンドの動作への影響について説明します。

出力　このセクションでは、コマンドで生成される出力 (標準出力、標準エラー、または出力ファイル) について説明します。

戻り値　マニュアルページに値を戻す関数について記載されている場合、このセクションでは、これらの値を一覧表示し、値が戻される条件について説明します。関数が定数値 (0 や -1 など) のみを戻すことができる場合は、これらの値がタグ付きの段落に一覧表示されます。それ以外の場合は、単一の段落で各関数の戻り値について説明します。void が宣言された関数は値を戻さないため、「戻り値」では説明しません。

エラー　大部分の関数は失敗時に、エラーの原因を示すエラーコードをグローバル変数 `errno` に配置します。このセクションでは、関数が生成できるすべてのエラーコードをアルファベット順に一覧表示し、各エラーが発生する条件について説明します。複数の条件で同じエラーが発生する可能性がある場合は、エラーコードの下にある個別の段落で各条件について説明します。

使用法	このセクションでは、詳細な説明を必要とする特別な規則、機能、およびコマンドを一覧表示します。ここで一覧表示するサブセクションを使用して、組み込み機能について説明します。 コマンド 修飾子 変数 式 入力文法
使用例	このセクションでは、コマンドまたは関数の使用例または使用方法を示します。可能なかぎり、コマンド行エントリおよびマシン応答を含む完全な例が表示されます。例が示される場合は、常にプロンプトは `example%` と表示され、ユーザーがスーパーユーザーになる必要がある場合は、`example#` と表示されます。例に続いて、説明、変数の置換規則、または戻り値が表示されます。ほとんどの例は、「形式」、「機能説明」、「オプション」、および「使用法」に記載されたコンセプトを例証しています。
環境変数	このセクションでは、コマンドまたは関数によって影響を受ける環境変数を一覧表示したあとに、その影響について簡潔に説明します。
終了ステータス	このセクションでは、コマンドが呼び出し元のプログラムまたはシェルに戻す値、およびそれらの値が戻される条件を一覧表示します。通常、正常に完了した場合はゼロが戻され、さまざまなエラー状況の場合はゼロ以外が戻されます。
ファイル	このセクションでは、マニュアルページで参照されるすべてのファイル名、対象のファイル、およびコマンドによって作成または要求されるファイルを一覧表示します。各ファイルのあとに、説明的なサマリーまたは説明が続きます。
属性	このセクションでは、属性タイプおよび対応する値を定義して、コマンド、ユーティリティー、およびデバイスドライバの特性を一覧表示します。詳細については、`attributes(5)` のマニュアルページを参照してください。
関連項目	このセクションでは、その他のマニュアルページ、Oracle のドキュメント、および社外出版物への参照を一覧表示します。
診断	このセクションでは、エラーが発生する条件についての簡単な説明とともに、診断メッセージを一覧表示します。

警告	このセクションでは、作業状況に著しく影響を与える可能性のある特別な条件についての警告を一覧表示します。これは診断の一覧ではありません。
注意事項	このセクションでは、ページのどこにも属さない追加情報を一覧表示します。これはユーザーへの余談という形式を取り、特別に関心がある点について扱います。重大な情報については、ここでは扱いません。
使用上の留意点	このセクションでは、既知のバグについて説明し、可能な場合は回避方法を示します。

参 照

システム管理コマンド

名前　　aimanifest – Automated Installer (AI) を使用した XML ファイルの変更

形式　　`/usr/bin/aimanifest [-h]`

`aimanifest add [-r]` *path value*

`aimanifest get [-r]` *path*

`aimanifest set [-r]` *path value*

`aimanifest load [-i]` *filename*

`aimanifest validate`

機能説明　　`aimanifest` コマンドは、新しい XML マニフェストを作成するか、既存の XML マニフェストを変更します。`aimanifest` は DTD 定義への有効な `!DOCTYPE` 参照を含む XML ファイルで使用できますが、本来は Automated Installer (AI) によって使用される派生マニフェストを作成するためのものです。AI 派生マニフェストについては、『Oracle Solaris 11 システムのインストール』を参照してください。

`aimanifest` コマンドは、マニフェストを作成するために複数回呼び出すことができます。`AIM_MANIFEST` 環境変数は、変更する `aimanifest` のマニフェストの場所を指定します。`AIM_MANIFEST` を設定する必要があります。`aimanifest` コマンドを `load`、`add`、または `set` サブコマンドオプションとともに呼び出すたびに、`AIM_MANIFEST` ファイルが開かれ、変更され、保存されます。

`aimanifest` コマンドが変更できる `AIM_MANIFEST` ファイルには、少なくとも次の両方の要素が含まれている必要があります。

- 作成している XML マニフェストで有効な DTD への `!DOCTYPE` 参照。
- このマニフェストのルート要素。

AI が派生マニフェストスクリプトを実行しているときのように、空の `AIM_MANIFEST` ファイルで作業を始める場合、最初の `aimanifest` コマンドで `load` サブコマンドを指定し、少なくとも最低限必要な `AIM_MANIFEST` ファイルを読み込む必要があります。マニフェストを変更するための以降の `aimanifest` コマンドでは、DTD を使用して、作成中のマニフェスト内で要素を追加する場所を決定します。

エラーや情報メッセージを標準出力と標準エラー出力に表示するだけでなくファイルに保存するには、`AIM_LOGFILE` 環境変数をログファイルの場所に設定します。情報はログファイルに追加されます。ログファイルは消去されません。

オプション　　`aimanifest` コマンドには次のオプションがあります。

`-h、--help`　　使用法のヘルプメッセージを表示します。

`aimanifest` コマンドの `add`、`get`、`set` サブコマンドには次のオプションがあります。

`-r、--return-path`　　この `aimanifest` コマンドが作成または処理する XML 要素のパスを返します。この返されたパスは、ノード ID のチェーンです。この返されたパスの値を保存し

て、aimanifest への以降の呼び出しで使用できます。XML 要素と属性の値を使用してパスを指定するよりも、-r オプションによって返されたパスを使用する方が信頼性が高くなります。これは、AI マニフェストが構築されているときに値が変わる可能性があるためです。-r オプションによって返されるパスについては、「戻りパス」のセクションを参照してください。

aimanifest コマンドの load サブコマンドには、次のオプションがあります。

-i、--incremental　新しいデータを追加する前に AIM_MANIFEST データを消去しません。

サブコマンド　サポートされているサブコマンドは次のとおりです。

add [-r | --return-path] *path value*
新しい要素を XML マニフェストに追加します。値 *value* を使用して、*path* で新しい要素を追加します。*path* については、「オペランド」のセクションを参照してください。*path* が属性 (@*attr*) 内で終了する場合、新しい要素は *attr* 属性を持ち、*value* はその属性の値になります。

path 内で親/子関係を検査する場合を除き、検証は行われません。

-r オプションは、新しく追加されたノードへのパスを返します。詳細は、「戻りパス」のセクションを参照してください。

親パスが AIM_MANIFEST ファイルの要素に一致する場合は、1 つの要素のみに一致する必要があります。新しい要素が、一致する親要素の子として作成されます。このセクションの「例 2: 値を持つパス」に示されているように、パスで要素と属性の値を指定して、一意の親要素に一致させることができます。

親パスが AIM_MANIFEST ファイルの要素に一致しない場合、必要に応じて新しい要素が作成され、新しい子要素が新しい親に追加されます。追加された要素へのパスは、次の規則に従って、既存の要素から分割されます。

- 分割は、値を指定するパスのすべての部分より後ろで実行されます。
- 分割は、同じタグが付けられた複数の関連要素が DTD によって許可されている最初の場所で、値を指定するパスのすべての部分より後ろで実行されます。

この XML マニフェストスキーマを使用して、次の例を分析します。

- マニフェストは、単一の A ノードで開始されます。
- A ノードは、B ノードの子を 1 つだけ持つことができます。
- B ノードは、C ノードの子を複数持つことができます。
- C ノードは、D ノードの子を複数持つことができます。

例**1:** 単一のパス。AI マニフェストは次のように 1 つの A ノード、1 つの B ノード、1 つの C ノードを持ちます: `/A/B/C`。`add` サブコマンドは、パス `/A/B/C/D` で発行されます。この場合、C ノードは同じタグが付けられた兄弟を持つことができるパスに属する最初のノードであるため、新しい C ノードが作成されます。新しい D ノードは、子として新しい C ノードに追加されます。その結果、マニフェストの構造は `/A/B/{C,C/D}` になります。異なる値の D に対して同じコマンドを発行すると、次のように 3 つの C ノードになります: `/A/B/{C,C/D,C/D}`。

例**2:** 値を持つパス。AI マニフェストは、1 つの A ノード、1 つの B ノード、2 つの C ノードを持ちます。1 つの C ノードのみが値 1 を持つため、マニフェストの構造は `/A/B/{C,C=1}` になります。`add` サブコマンドは、パス `/A/B/C=1/D` と値 10 で発行されます。この場合、C に対して値 1 を指定することで一意のノードが識別されるため、新しい C ノードは追加されません。また、値が指定されるブランチの場所またはその前でパスを分割することはできません。このパスを分割できる 1 番目の場所は D です。値が 10 の新しい D ノードは、値が 1 の C ノードの子として追加されます。その結果、マニフェストの構造は `/A/B/{C,C=1/D=10}` になります。D に対して値 20 を持つ同じコマンドを発行すると、`/A/B/{C,C=1/{D=10,D=20}}` になります。

`get [-r | --return-path]` *path*
要素または属性の値を取得します。空の要素または属性の値に対しては、空の文字列 ("") が表示されます。*path* は、一意の既存の要素または属性に一致している必要があります。*path* については、「オペランド」のセクションを参照してください。

`-r` オプションは、2 番目に返された文字列として、アクセス先ノードへのパスを返します。詳細は、「戻りパス」のセクションを参照してください。

`set [-r | --return-path]` *path value*
既存の要素または属性の値を変更するか、既存の要素の新しい属性を作成します。検証は行われません。

既存の要素の値を変更するときに、*path* は一意の既存の要素に一致する必要があります。同じタグが付けられた兄弟を要素が持っている場合は、要素の値または属性を使用するかターゲット要素の子要素を使用してパスを一意にします。「Path オペランド」のセクションを参照してください。

属性の値を設定するときに、その属性が存在している必要はありませんが、その属性が所属する要素が存在している必要があります。

`-r` オプションは、変更された要素へのパスを返します。詳細は、「戻りパス」のセクションを参照してください。

`load [-i | --incremental]` *filename*
XML マニフェストまたは部分的な XML マニフェストをファイル *filename* から読み込みます。要素の親/子関係を検査する場合を除き、検証は行われません。

-i オプションが指定されていない場合、既存の XML データはすべて上書きされます。AIM_MANIFEST ファイルのすべてのデータは、*filename* ファイルの内容に置き換えられます。*filename* ファイルに DTD への !DOCTYPE 参照を含めて、以降の aimanifest コマンドがファイルを変更できるようにする必要があります。

-i オプションが指定されている場合、新しいデータを追加する前に AIM_MANIFEST データを消去しないでください。その代わりに、既存の XML データを使用して、少しずつ新しいデータを挿入またはマージしてください。AIM_MANIFEST の !DOCTYPE 参照によって指定された DTD は、*filename* データを統合する方法と場所を決定するために使用されます。!DOCTYPE 参照が見つからない場合は、/usr/share/install/ai.dtd の AI マニフェスト DTD が使用されます。*filename* のデータを DTD で調整できない場合は、ゼロ以外のエラーステータスが返されます。

次の事項は、新しいデータが AIM_MANIFEST マニフェストに挿入される場所に影響します。

- AIM_MANIFEST データパスと *filename* データパスの先頭近くにある要素のタグが一致している度合い
- それらの AIM_MANIFEST データ要素で許可されている子要素の種類
- 同じタグが付けられた兄弟要素が許可されている場所
- 子を持たない AIM_MANIFEST データノードが存在している場所

filename データの各要素が処理されるときに次のすべての条件が真である場合、通常は、新しいノードが AIM_MANIFEST データのこの要素に対して作成されることはありません。その代わりに、既存のノードが新しいデータに置き換えられます。

- 両方のセットのデータに、タグと場所が同じノードが含まれます。
- AIM_MANIFEST 内で !DOCTYPE 参照によって指定される DTD は、同じタグが付けられた兄弟要素としてこの両方のノードが共存することを許可しません。
- *filename* データ要素は子を持ちます。

要素が *filename* から挿入されるときに、新しいノードの作成が開始される場所の AIM_MANIFEST データルートにできるかぎり近い位置で分割が実行されます。分割の最初の新しいノードは、同じタグが付けられた兄弟要素が許可されているもっとも早い場所に作成されます。または、同じタグが付けられた要素が AIM_MANIFEST 内に存在しない場合には、もっとも早い適切な地点に作成されます。

この XML マニフェストスキーマを使用して、次の例を分析します。

- マニフェストは、単一の A ノードで開始されます。
- A ノードは、B ノードの子を 1 つだけ持つことができます。
- B ノードは、C ノードの子を複数持つことができます。
- B ノードは、E ノードの子を 1 つだけ持つことができます。

例**1:** 同じタグが付けられた要素の挿入。AIM_MANIFEST の内容が /A/B/C1/D1 で *filename* の内容が /A/B/C2/D2 の場合、load -i コマンド後の AIM_MANIFEST ファイルの内容は /A/B/{C1/D1,C2/D2} です。C ノードは、新しいノードを追加できる最初の場所です。*filename* データの C ノードが、AIM_MANIFEST データ内の既存の C ノードの後に追加されます。2 つの A 要素が異なる値を持っているか、2 つの B 要素が異なる値を持っている場合、*filename* 要素の値は AIM_MANIFEST 要素の値を置き換えます。2 つの A 要素が異なる属性を持っているか、2 つの B 要素が異なる属性を持っている場合、属性の値はマージされます。

- AIM_MANIFEST ファイルと *filename* ファイルの両方に存在する A と B の属性は、マージされたファイルでは *filename* ファイルの値になります。
- AIM_MANIFEST ファイルまたは *filename* ファイルの両方ではなくどちらかにのみ存在する A と B の属性は、マージされたファイル内ではすべて保持されます。

例 **2:** 異なるタグが付けられた要素の挿入。AIM_MANIFEST の内容が /A/B/C/D で *filename* の内容が /A/B/E/F の場合、load -i コマンド後の AIM_MANIFEST ファイルの内容は /A/B/{E/F,C/D} です。E ノードは、DTD によって許可されている最初の場所に追加されます。要素 A と要素 B の値は *filename* の値で、A と B の属性は、上の例 1 で説明されているように *filename* から AIM_MANIFEST にマージされます。

場合によっては、正しいマージ場所を決定できないことがあります。これは、マージするノードをたどるために必要な兄弟がまだ追加されていない場合に起こります。この問題を回避するには、複数のノードまたはサブツリーを、DTD によって指定された順序で共通の親ノードに追加します。新しい兄弟間で正しい場所が決定できない場合、ノードはそれらの兄弟のリストの最後に配置されます。

validate
: !DOCTYPE 文で参照されている DTD に対して AIM_MANIFEST マニフェストを検証します。エラーは標準エラー出力に出力されます。検証が失敗した場合、ゼロ以外のエラーステータスが返されます。

オペランド　次のオペランドは必須です。

Filename オペランド　load サブコマンドには *filename* オペランドが必要です。このオペランドは、AIM_MANIFEST マニフェストに読み込む完全なマニフェストまたは部分的なマニフェストの名前です。

Value オペランド　add と set サブコマンドには、*value* オペランドが必要です。*value* オペランドは、*path* オペランドによって指定された要素または属性の有効な値です。

Path オペランド　aimanifest コマンドの add、get、set サブコマンドには、*path* オペランドが必要です。パスは、要素と属性の XML 階層内のノードを定義します。

XML 要素の階層構造は、XML ツリーとも呼ばれます。次の部分的な AI マニフェストでは、auto_install 要素はツリーのルートで、ai_instance 要素と software 要素はブランチ、またはサブツリーのルートです。

```
<auto_install>
  <ai_instance>
    <software type="IPS"/>
  </ai_instance>
</auto_install>
```

aimanifest パス構文では、スラッシュ文字 (/) を使用してツリー構造のブランチを示します。現在の例では、software 要素へのパスは /auto_install/ai_instance/software です。

属性は 1 つの要素にバインドされます。aimanifest パス構文では、アットマーク記号 (@) を使用して属性名を示します。software 要素の type 属性へのパスは、/auto_install/ai_instance/software@type です。

aimanifest の *path* オペランドは、単一の要素に対応している必要があります。必要に応じて、要素と属性の値を含めてパスを一意にします。たとえば、次の部分的な AI マニフェストで定義された 2 番目のスライスに対してサイズを指定するには、パス /auto_install/ai_instance/target/disk/slice[@name="4"]/size@val を使用して、サイズを指定するスライスを特定します。

```
<auto_install>
  <ai_instance>
    <target>
        <disk>
          <slice name="0"/>
          <slice name="4"/>
        </disk>
    </target>
  </ai_instance>
</auto_install>
```

相対パスは許可されています。前の段落で示された slice のパスの指定は、4 の name 属性値を持つ slice が 1 つしかないため、ai_instance、target、disk、または slice で開始できます。たとえば、パス slice[@name="4"]/size@val を使用できます。

パス内の値にスラッシュ文字が含まれる場合、/name="pkg:/entire" のように、その値を一重引用符または二重引用符で囲む必要があります。

aimanifest の呼び出しがシェルスクリプト内で行われる場合、引用符を含む値に対してさらに特別な処理が必要になる場合があります。シェルスクリプト内では、aimanifest パス値の引用符の前にバックスラッシュ文字 (\) を追加してエスケープし、シェルが引用符を削除または解釈しないようにすることが必要になる場合があります。使用しているシェルの規則を確認してください。次の例は、ksh93 スクリプト内のスラッシュ文字が含まれる値を示しています。

```
/usr/bin/aimanifest get software_data[name=\"pkg:/entire\"]@action
```

このマニュアルページでは、aimanifest がスクリプトまたは特定のシェル内で呼び出されることを前提としないため、このマニュアルページ内のほとんどの例でバックスラッシュエスケープ文字を省略しています。AI 派生マニフェストスクリプトについては、『Oracle Solaris 11 システムのインストール』を参照してください。

次のブランチの形式は、要素または要素属性へのパスを作成する方法を示しています。

/A
: A は要素のタグ名で、/auto_install のようになります。このブランチ指定は、単純ブランチとも呼ばれます。単純ブランチのみを持つパスは、単純パスと呼ばれます。

/A=*value*
: A は要素のタグ名、*value* はその要素の値で、/name="pkg:/entire" のようになります。

/A[B/C=*value*]
: A は要素、B は A の子である要素、C は B の子である要素、*value* は C 要素の値です。このパス形式は、値が *value* の孫要素 C を持つ A 要素を指定します。たとえば、AI マニフェストに複数のソフトウェアセクションがある場合、この形式を使用すると、次のパスのように、パッケージ pkg:/entire をインストールするソフトウェアセクションに対する操作が可能になります。

  ```
  software[software_data/name="pkg:/entire"]
  ```

/A[@Aattr=*value*]
: A は要素、Aattr は A の属性、*value* は Aattr 属性の値です。このパス形式は、値が *value* の属性 Aattr を持つ A 要素を指定します。たとえば、AI マニフェストが複数のスライスを定義している場合、この形式を使用すると、slice[@name="4"] のように、4 の name 値を持つスライスに対する操作が可能になります。

/A[B/C@Cattr=*value*]
: A は要素、B は A の子、C は B の子、Cattr は C の属性、*value* は Cattr 属性の値です。このパス形式は、値が *value* の属性 Cattr を持つ孫要素 C を持つ A 要素を指定します。たとえば、AI マニフェストに複数のソフトウェアセクションがある場合、この形式を使用すると、パス software[source/publisher@name="solaris"] のように、名前の値が solaris のパブリッシャーセクションを持つソフトウェアセクションに対する操作が可能になります。

/A[1]
: /A[1] は、マニフェスト内の A 要素の最初のインスタンスを指定します。たとえば、AI マニフェストに複数のソフトウェアセクションがある場合、この形式を使用すると、/auto_install[1]/ai_instance[1]/software[2] のように、2 番目のソフトウェアセクションに対する操作が可能になります。

これは、-r オプションによって返されるパスの形式です。「戻りパス」のセクションを参照してください。

/A@Aattr
このパスは、A 要素の Aattr 属性を指定します。このパスは、A 要素ではなく Aattr 属性を指定します。この形式を使用して、Aattr 属性を設定または取得します。

/A[B/C=*value*]@Aattr
このパスは、値が *value* の孫属性 C を持つ A 要素の Aattr 属性を指定します。

/A[B/C@Cattr=*value*]@Aattr
このパスは、値が *value* の Cattr 属性を持つ孫要素 C を持つ A 要素の Aattr 属性を指定します。

/A/B=*value*@Battr
このパスは、値が *value* の B 要素の Battr 属性を指定します。B 要素は A 要素の子です。

戻りパス

-r オプションを使用すると、add、get、set サブコマンドは、そのサブコマンドによって作成またはアクセスされた要素のアドレスを返します。この返されたアドレスは、ノード ID のチェーンの形式になります。この返されたアドレスを使用すると、同じ要素に関連付けられた値が変更された場合であっても、その値に再度アクセスできます。

次の例は、-r オプションによって返されたアドレスが、要素と属性の値を指定するパスよりはるかに簡単に使用できることを示しています。次のノードツリーで開始します。

```
auto_install
     |
ai_instance
     |
   target
     |
   disk
 attribute: whole_disk=true
     |
 disk_name
 attribute: name=data1
 attribute: name_type=volid
```

name 属性値が data2 で name_type 属性値が volid の新しい disk ノードを追加します。

```
auto_install
     |
ai_instance
     |
```

```
            target
              |
  |-----------|-----------|
disk                    disk
whole_disk=true         whole_disk=true
    |                       |
disk_name               disk_name
name=data1              name=data2
name_type=volid         name_type=volid
```

1 つの属性を持つ新しい disk_name 要素は、単一のコマンドで簡単に追加できます。2 番目の属性と 3 番目の属性を追加するには、変更する disk_name 要素を指定する必要があります。同じノードに複数回アクセスするための次の 2 つの方法を比較します。

値を使用したパスの指定

この例のコマンドは、値を使用してパスを指定します。最初のコマンドで一意の値を割り当て、以降のコマンドではその値を使用して一意のパスを指定できるようにする必要があります。この方法では、値が変更された場合に正しくない結果が生成される可能性があります。

```
$ aimanifest add target/disk/disk_name@name data2
$ aimanifest set \
> target/disk/disk_name[@name=data2]@name_type volid
$ aimanifest set \
> target/disk[disk_name@name=data2]@whole_disk true
```

戻りパスを使用したパスの指定

同じノードに複数回アクセスするもっとも信頼性の高い方法は、新しい disk_name 要素へのパスを保存し、その保存したパスを以降のアクセスで使用する方法です。

```
$ NewDisk=$(aimanifest add -r target/disk@whole_disk true)
$ aimanifest add ${NewDisk}/disk_name@name data2
$ aimanifest add ${NewDisk}/disk_name@name_type volid
```

-r オプションによって $NewDisk に返されるパスは、ノードを ID で表しており、値を持ちません。

```
$ aimanifest add -r target/disk/@whole_disk true
/auto_install[1]/ai_instance[1]/target[1]/disk[2]
```

使用例

これらの例を試すには、AIM_MANIFEST を設定する必要があります。

```
$ export AIM_MANIFEST=/tmp/aimtest.xml
```

aimanifest コマンドが変更できる AIM_MANIFEST ファイルには、少なくとも次の両方の要素が含まれている必要があります。

- 作成している XML マニフェストで有効な DTD への !DOCTYPE 参照。
- このマニフェストのルート要素。

次の例は、AI マニフェスト用の最低限の AIM_MANIFEST マニフェストファイルを示しています。

```
<!DOCTYPE auto_install SYSTEM "file:///usr/share/install/ai.dtd">
<auto_install/>
```

通常は、既存の有効な AI マニフェストに対して動作する派生マニフェストスクリプト内で aimanifest コマンドを使用します。これらの例を試すには、/usr/share/auto_install/manifest/default.xml をコピーしてから、AIM_MANIFEST を定義してこのコピーを参照します。このコピーが書き込み可能になっていることを確認してください。

例 1 auto_reboot の属性の設定

```
$ aimanifest set /auto_install/ai_instance@auto_reboot false
```

例 2 auto_reboot の値の取得

```
$ aimanifest get /auto_install/ai_instance@auto_reboot
false
```

例 3 値のパスを使用したパブリッシャーの追加

この例のパッケージリポジトリは、file:///net/host2/export/extras_repo にあるファイルリポジトリです。パブリッシャーは extras です。

```
$ aimanifest add \
> software/source/publisher@name extras
$ aimanifest add \
> software/source/publisher[@name=extras]/origin@name \
> file:///net/host2/export/extras_repo
$ aimanifest set \
> software[source/publisher@name=extras]@name extras
$ aimanifest set \
> software[source/publisher@name=extras]@type IPS
```

これらの aimanifest コマンドは、次の AI マニフェストエントリを生成します。software 要素は、同じタグが付けられた兄弟が許可されているパスの最初の要素であるため、XML コードのこのセクションは、出力ファイル内にすでに存在する最後の software セクションに続きます。

```
<software name="extras" type="IPS">
  <source>
    <publisher name="extras">
      <origin name="file:///net/host2/export/extras_repo"/>
    </publisher>
  </source>
</software>
```

例 4 戻りパスを使用したパブリッシャーの追加

この例は前の例と同じですが、同じ結果を得るために異なる方法を使用します。

例 4 戻りパスを使用したパブリッシャーの追加 (続き)

```
$ SW_PATH=$(aimanifest add -r \
> /auto_install/ai_instance/software@name extras)
$ aimanifest set ${SW_PATH}@type IPS
$ PUB_PATH=$(aimanifest add ${SW_PATH}/source/publisher@name extras)
$ aimanifest add \
${PUB_PATH}/origin@name file:///net/host2/export/extras_repo)
```

例 5 マニフェストフラグメントの追加によるパブリッシャーの追加

この例は前の例と同じですが、同じ結果を得るために 3 番目の方法を使用します。

次の内容を持つ extras.xml という名前のファイルを作成します。

```
<auto_install>
  <ai_instance>
    <software name="extras" type="IPS">
      <source>
        <publisher name="extras">
          <origin name="file:///net/host2/export/extras_repo"/>
        </publisher>
      </source>
    </software>
  </ai_instance>
</auto_install>
```

ソフトウェアセクションのみが必要な場合でも、auto_install 要素と ai_instance 要素も含める必要があります。*path* オペランドで必要となるすべてのものを含める必要があります。読み込まれたファイルが auto_install 要素または ai_instance 要素の属性を指定している場合、それらの属性値は既存の値を置き換えるか、または追加されます。

次のコマンドを使用して、この software セクションを AIM_MANIFEST マニフェストに追加します。

```
$ aimanifest load -i extras.xml
```

例 6 値のパスを使用したパッケージの追加

この例では、パス内の値としてパブリッシャー名を指定することによって、extras という名前の publisher 要素を持つ software 要素にパッケージを追加します。この例では、相対パス指定の使用についても示しています。

```
$ aimanifest add \
> software[source/publisher@name=extras]/software_data/name \
> pkg:/system/utils
```

この aimanifest コマンドは、次の software_data セクションを追加します。

例 6　値のパスを使用したパッケージの追加　　(続き)

```
<software name="extras" type="IPS">
  <source>
    <publisher name="extras">
      <origin name="file:///net/host2/export/extras_repo"/>
    </publisher>
  </source>
  <software_data>
    <name>pkg:/system/utils</name>
  </software_data>
</software>
```

例 7　戻りパスを使用したパッケージの追加

この例は前の例と同じですが、同じ結果を得るために異なる方法を使用します。この例では、パス内の値としてパブリッシャーの名前を指定する代わりに、「戻りパスを使用したパブリッシャーの追加」の例で SW_PATH に保存した software 要素へのパスを使用します。

```
$ aimanifest add ${SW_PATH}/software_data/name pkg:/system/utils
```

例 8　マニフェストの検証

AIM_MANIFEST マニフェストを検証します。

```
$ aimanifest validate
```

終了ステータス

次の終了ステータスが返されます。

0　　コマンドは正常に処理されました。

>0　　エラーが発生した。

ファイル

AIM_MANIFEST　　この環境変数の値は、構築されている AI マニフェストの場所です。

AIM_LOGFILE　　この環境変数の値は、aimanifest の動作のログファイルの場所です。

属性

属性についての詳細は、マニュアルページの attributes(5) を参照してください。

属性タイプ	属性値
使用条件	system/install/auto-install/auto-install-common
インタフェースの安定性	不確実

関連項目 installadm(1M), pkg(1)

『Oracle Solaris 11 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」

名前　distro_const – Oracle Solaris のイメージとメディアを作成するユーティリティー

形式

```
/usr/bin/distro_const

distro_const --help

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

機能説明

`distro_const` コマンドでは、指定したマニフェストファイルをイメージの青写真として使用してイメージを作成できます。

x86 システムまたは SPARC システムのどちらかで Oracle Solaris オペレーティングシステムをインストールするために使用できるテキストインストーライメージを作成できます。

または、Oracle Solaris オペレーティングシステムが含まれる LiveCD イメージに相当する ISO イメージを作成することもできます。

あるいは、SPARC クライアントに Oracle Solaris OS をネットワークインストールするために使用できる SPARC AI ISO イメージか、x86 クライアントに Oracle Solaris OS をネットワークインストールするために使用できる x86 AI ISO イメージを作成できます。

カスタム ISO イメージを作成することもできます。

基本の `distro_const` コマンドをオプションなしで実行すると、1 回のステップでフルイメージが作成されます。

コマンドオプションを使用すると、さまざまな「チェックポイント」でイメージ作成処理を一時停止および再開できるため、各段階でイメージの状態を確認したり、バグの有無を確認したりできます。チェックポイントを使用することで、すでに少なくとも一度は実行された時間のかかるステップを省略できるため、構築にかかる時間が短縮されます。

注 – `distro_const` コマンドを実行するには、ルートの役割になるか、ルートの権限を持っている必要があります。

ディストリビューションコンストラクタを使用する場合、SPARC システムでは SPARC イメージのみを作成できます。同様に、x86 システムでは x86 イメージのみを作成できます。また、システム上のオペレーティングシステムのリリースバージョンが、構築しているイメージと同じリリースバージョンである必要があります。

サブコマンド　`distro_const` コマンドには、次に示されているサブコマンドとオプションがあります。「使用例」のセクションも参照してください。

`--help`
　使用法を表示します。

build
`distro_const build` *manifest*

サブコマンド「build」は必須です。

指定したマニフェストファイルをイメージの青写真として使用して、フルイメージを作成します。マニフェスト名は必須です。

`-v distro_const build - v`
詳細表示モードを指定します。

`-l distro_const build [- l]` *manifest*
イメージの構築の一時停止または再開を選択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。このコマンドオプションは、有効なチェックポイントの有無をマニフェストファイルに問い合わせます。このコマンドによって表示された名前を、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。build サブコマンドは必須です。

チェックポイントの値は、マニフェストファイル内のエントリによって決まります。

`-p distro_const build [- p` *checkpoint*`]` *manifest*
イメージを構築しますが、指定したチェックポイント名でイメージの構築を一時停止します。有効な名前を検索するには、-l オプションを使用します。-r オプションと -p オプションを組み合わせることができます。チェックポイント名とマニフェスト名は必須です。build サブコマンドは必須です。

`-r distro_const build [- r` *checkpoint*`]` *manifest*
指定したチェックポイント名からイメージの構築を再開します。指定する名前は、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。どのチェックポイントが再開可能かを判定するには、-l オプションを使用します。-p オプションは、-r オプションと組み合わせることができます。チェックポイント名とマニフェスト名は必須です。build サブコマンドは必須です。

`-h distro_const [- h]`
コマンドの使用法を表示します。

使用例

例 1 チェックポイントのオプションを使用したイメージの作成

1. 利用可能なチェックポイントを確認します。この例のマニフェスト名は `dc_livecd.xml` です。

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

有効なチェックポイント名が、このサンプル出力のように表示されます。

```
  Checkpoint            Resumable  Description
```

例 1 チェックポイントのオプションを使用したイメージの作成 (続き)

```
-----------------------------------------
transfer-ips-install X Transfer pkg contents from IPS
set-ips-attributes   X Set post-install 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 archival
grub-setup           X Set up GRUB menu
pkg-img-mod          X Pkg image area modification
create-iso             ISO media creation
create-usb             USB media creation
```

2. イメージの構築を開始し、ba-init チェックポイントで一時停止します。

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

3. ba-init チェックポイントから構築を再開します。イメージの作成を完了します。

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

例 2 イメージを 1 ステップで作成

一時停止せずにイメージの完全な構築を実行するには、チェックポイントオプションを指定せずに基本の `distro_const` コマンドを使用します。マニフェストファイル名は dc_livecd.xml です。

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

属性

属性についての詳細は、マニュアルページの `attributes(5)` を参照してください。

属性タイプ	属性値
使用条件	install/distribution-constructor
インタフェースの安定性	開発中

関連項目

`dc_manifest(4)`

現在のリリースの OTN ドキュメントライブラリの『カスタム Oracle Solaris 11 インストールイメージの作成』。

名前　　installadm – ネットワーク上の自動インストールの管理

形式

```
/usr/bin/installadm [-h|--help]

installadm help [subcommand]

installadm create-service
    [-n|--service svcname]
    [-t|--aliasof existing_service]
    [-p|--publisher prefix=origin]
    [-a|--arch architecture]
    [-s|--source FMRI_or_ISO]
    [-b|--boot-args boot_property=value,...]
    [-i|--ip-start dhcp_ip_start]
    [-c|--ip-count count_of_ipaddr]
    [-B|--bootfile-server server_ipaddr]
    [-d|--imagepath imagepath]
    [-y|--noprompt]

installadm set-service
    -o|--option prop=value svcname

installadm rename-service svcname newsvcname

installadm enable svcname

installadm disable svcname

installadm delete-service
    [-r|--autoremove] [-y|--noprompt] svcname

installadm list
    [-n|--service svcname]
    [-c|--client] [-m|--manifest] [-p|--profile]

installadm create-manifest -n|--service svcname
    -f|--file manifest_or_script_filename
    [-m|--manifest manifest_name]
    [-c|--criteria criteria=value|list|range... |
    -C|--criteria-file criteriafile]
    [-d|--default]

installadm update-manifest -n|--service svcname
    -f|--file manifest_or_script_filename
    [-m|--manifest manifest_name]

installadm delete-manifest
    -m|--manifest manifest_name
    -n|--service svcname

installadm create-profile -n|--service svcname
    -f|--file profile_filename...
    [-p|--profile profile_name]
    [-c|--criteria criteria=value|list|range... |
    -C|--criteria-file criteriafile]
```

```
installadm delete-profile -p|--profile profile_name...
    -n|--service svcname

installadm export -n|--service svcname
    -m|--manifest manifest_name...
    -p|--profile profile_name...
    [-o|--output pathname]

installadm validate -n|--service svcname
    -P|--profile-file profile_filename... |
    -p|--profile profile_name...

installadm set-criteria
    -m|--manifest manifest_name
    -p|--profile profile_name...
    -n|--service svcname
    -c|--criteria criteria=value|list|range... |
    -C|--criteria-file criteriafile |
    -a|--append-criteria criteria=value|list|range...

installadm create-client
    [-b|--boot-args property=value,...]
    -e|--macaddr macaddr -n|--service svcname

installadm delete-client macaddr
```

機能説明

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。

ネットワーク経由で AI を使用するために必要なマシントポグラフィーは、インストールサーバー、DHCP サーバー (インストールサーバーと同じシステムでも可)、およびインストールクライアントです。インストールサーバーでは、AI ブートイメージ (クライアントがネットワーク経由でブートするために提供されます)、入力仕様 (AI マニフェストおよび派生したマニフェストスクリプト、このうち 1 つがクライアントに選択されます)、サービス管理機構 (SMF) 構成プロファイル (このうちゼロ個以上がクライアントに選択されます) が含まれるようにインストールサービスが設定されます。

AI ブートイメージの内容は、パッケージ `install-image/solaris-auto-install` として公開され、`create-service` サブコマンドでインストールされます。`create-service` サブコマンドを使用すると、AI ブートイメージを作成する AI ISO ファイルを受け入れて、展開することもできます。

インストールサービスはデフォルトの AI マニフェストを使用して作成されますが、カスタマイズされたマニフェストまたは派生したマニフェストスクリプト (今後は「スクリプト」と呼びます) は、`create-manifest` サブコマンドを使用してインストールサービスに追加できます。マニフェストおよび派生したマニフェストスクリプトの作成方法についての詳細は、『Oracle Solaris 11 システムのインストール』を参照してください。`create-manifest` サブコマンドを使用すると、インストールクライアントに選択するマニフェストまたはスクリプトを決定する際に使

用される条件を指定することもできます。すでにマニフェストまたはスクリプトに関連付けられている条件は、`set-criteria` サブコマンドを使用すれば変更できます。

マニフェストには、ターゲットデバイス、パーティション情報、パッケージ一覧、その他のパラメータなどの情報が含まれています。スクリプトには、実行中の AI クライアントシステムを照会し、検索される情報に基づいてカスタムマニフェストを作成するコマンドが含まれています。AI がスクリプトで起動されると、マニフェストを生成するために、そのスクリプトが最初のタスクとして実行されます。

クライアントがブートされると、クライアントのマシン条件に一致するマニフェストまたはスクリプトの検索が開始されます。一致するマニフェストまたはスクリプトが見つかると、一致するマニフェストファイルの仕様、または一致するスクリプトから派生したマニフェストファイルの仕様に従って、Oracle Solaris リリースを使用してクライアントがインストールされます。各クライアントは、1 つのマニフェストまたはスクリプトしか使用できません。

各サービスには、1 つのデフォルトマニフェストまたはスクリプトが含まれています。デフォルトは、インストールされるシステムに一致するマニフェストまたはスクリプトの条件が他にない場合に使用されます。どのマニフェストまたはスクリプトでも、デフォルトとして指定できます。 デフォルトのマニフェストまたはスクリプトに関連付けられた条件は非アクティブになり、マニフェストまたはスクリプトの選択時には考慮されません。後で別のマニフェストまたはスクリプトをデフォルトにすると、前にデフォルトだったマニフェストまたはスクリプトの条件が再びアクティブになります。条件が関連付けられていないマニフェストまたはスクリプトは、デフォルトのマニフェストまたはスクリプトとしてのみ使用できます。条件がないマニフェストまたはスクリプトは、別のマニフェストまたはスクリプトがデフォルトに指定されると非アクティブになります。

システム構成プロファイルはマニフェストおよびスクリプトを補完するものであり、インストールの仕様も含まれています。特に、ユーザー名、ユーザーパスワード、タイムゾーン、ホスト名、IP アドレスなどの情報を指定する際に使用されます。 プロファイルには、インストールサーバー環境から、または `create-profile` サブコマンドで指定された条件から構成パラメータを取得する変数を含めることができます。このように、単一のプロファイルファイルで、さまざまなクライアント上にさまざまな構成パラメータを設定できます。「使用例」のセクションを参照してください。

システム構成プロファイルは `smf`(5) で処理され、ドキュメント形式 `service_bundle`(4) に準拠しています。システム構成プロファイルについての詳細は、`sysconfig`(1M) および『Oracle Solaris 11 システムのインストール』を参照してください。各クライアントは、任意の数のシステム構成プロファイルを使用できます。 特定の SMF プロパティーは、クライアントシステムごとに 1 回しか指定できません。

特定のクライアントが特定のインストールサービスを使用する場合は、create-client サブコマンドを使用してクライアントにサービスを関連付けることができます。create-client を使用すると、既存のクライアントを変更することもできます。

installadm ユーティリティーを使用すると、次のタスクを実行できます。

- インストールサービスおよびエイリアスの設定
- インストールイメージの設定
- クライアントの設定または削除
- マニフェストおよびスクリプトの追加、更新、または削除
- マニフェストまたはスクリプトに対する条件の指定または変更
- マニフェストおよびスクリプトのエクスポート
- システム構成プロファイルの追加または削除
- プロファイルの検証
- プロファイルに対する条件の指定または変更
- プロファイルのエクスポート
- インストールサービスの有効化または無効化
- インストールサービスの一覧表示
- インストールサービスのクライアントの一覧表示
- インストールサービスのマニフェストおよびスクリプトの一覧表示
- インストールサービスのプロファイルの一覧表示

インストールサーバー構成のプロパティー

SMF サービス svc:/system/install/server:default の次の 3 つのプロパティーは、インストールサーバーを構成する際に使用されます。

all_services/networks
: all_services/exclude_networks プロパティーの設定に応じて、許可または却下する CIDR 形式のネットワーク (たとえば、192.168.56.0/24) の一覧。

  このネットワーク一覧を使用して、このインストールサーバーで処理されるクライアントを指定します。デフォルトでは、AI インストールサーバーがマルチホームになっている場合は、そのサーバーが接続されているすべてのネットワーク上のインストールクライアントを処理するよう構成されます。

all_services/exclude_networks
: ブール値。true の場合、このインストールサーバーによる処理対象から、all_services/networks プロパティーで指定されたネットワークを除外します。false の場合、all_services/networks プロパティーで指定されたネットワークを含めます。

all_services/port
: AI インストールサービス Web サーバーをホストするポートを指定します。デフォルトでは、Web サーバーはポート 5555 でホストされます。

  デフォルトとは異なるポート番号を使用する場合は、インストールサービスを作成する前に port プロパティーをカスタマイズしてください。

オプション　　installadm コマンドには次のオプションがあります。

-h、--help　　使用法のヘルプメッセージを表示します。

サブコマンド　　stmfadm コマンドには、次のサブコマンドを指定できます。後述の「使用例」のセクションを参照してください。

help [*subcommand*]
installadm ユーティリティーの構文を表示します。

subcommand　　指定されたサブコマンドの構文のみを表示します。

create-service [-n|--service *svcname*] [-t|--aliasof *existing_service*] [-p|--publisher *prefix=origin*] [-a|- -arch *architecture*] [-s|- -source *FMRI_or_ISO*] [-b|- -boot-args *boot_property=value*,...] [-i|--ip-start *dhcp_ip_start*] [-c|--ip-count *count_of_ipaddr*] [-B|--bootfile-server *server_ipaddr*] [-d|--imagepath *imagepath*] [-y|--noprompt]
このサブコマンドは、指定された *imagepath* ディレクトリにネットワークブートイメージ (ネットイメージ) を設定し、ネットイメージからブートされたクライアントのインストール方法が指定されたインストールサービスを作成します。

AI ブートイメージの内容は、パッケージ install-image/solaris-auto-install として公開されています。 -s オプションが指定されない場合は、そのパッケージのインスタンスが記載されているシステムのパブリッシャー設定一覧の 1 番目のパブリッシャーからパッケージがインストールされます。 -s オプションは、pkg の指定をフル FMRI またはイメージ ISO ファイルの場所として受け入れます。ネットイメージは最終的に *imagepath* に配置されます。ネットイメージを使用すると、クライアントインストールを実行できます。

次のような仕様があります。

- 指定されたアーキテクチャーの 1 番目のインストールサービスがインストールサーバーに作成されると、そのサービスのエイリアス (default-i386 または default-sparc) が自動的に作成されます。このデフォルトサービスは、明示的に create-client サブコマンドを使用してインストールサーバーに追加されなかったアーキテクチャーのクライアントへのすべてのインストールで使用されます。default-*arch* サービスによってエイリアスが設定されたサービスを変更するには、set-service サブコマンドを使用します。

  default- *arch* エイリアスが新しいインストールサービスに変更されたときに、ローカル ISC DHCP 構成が見つかった場合は、このデフォルトエイリアスブートファイルが、そのアーキテクチャーのデフォルト DHCP サーバー全体のブートファイルとして設定されます。
- クライアントがそのアーキテクチャーのデフォルト以外のインストールサービスを使用する場合は、create-client サブコマンドを使用してクライアント固有の構成を作成する必要があります。

- -i オプションと -c オプションを使用するときに、まだ DHCP サーバーが構成されていない場合は、ISC DHCP サーバーが構成されます。

  ISC DHCP サーバーがすでに構成されている場合は、その DHCP サーバーが更新されます。

  -i と -c 引数が指定され、DHCP が構成されている場合でも、作成されるインストールサービスと IP 範囲との結合は存在しません。-i と -c が渡され、IP 範囲が設定されている場合は、必要に応じて新規 DHCP サーバーが作成され、その DHCP サーバーは使用するすべてのインストールサービスおよびすべてのクライアントで起動され、実行された状態のままです。DHCP サーバーに指定されたネットワーク情報には、作成されるサービスとの特定の関係はありません。
- インストールサーバーが直接接続され、インストールサーバーがマルチホームになっているサブネット上に、要求された IP 範囲がない場合は、-B オプションを使用して、ブートファイルサーバーのアドレス (通常はこのシステム上の IP アドレス) を指定します。このアドレスは、複数の IP アドレスがインストールサーバーに構成され、DHCP リレーが採用されている場合にのみ必要です。その他のすべての構成では、これはソフトウェアで自動的に決定できます。

-n|--service *svcname*
省略可能: システムで生成されたサービス名の代わりに、このインストールサービス名を使用します。*svcname* は、英数字、アンダースコア (_)、ハイフン (-) で構成できます。*svcname* の最初の文字をハイフンにすることはできません。

-n オプションを指定しない場合は、サービス名が自動的に生成されます。

-t|--aliasof *existing_service*
省略可能: この新規サービスは *existing_service* の代替名です。

-a|--arch *architecture*
省略可能: 特定のバリアントのアーキテクチャーを選択します。 正当な値は i386 または sparc です。 これが指定されない場合は、サーバーのアーキテクチャーに対応するバリアントが選択されます。

-a オプションは、-s 引数が pkg パッケージである場合にのみ適用されます。

-p|--publisher *prefix=origin*
省略可能: クライアントイメージをインストールする pkg パブリッシャー (形式 *prefix=origin*)。

-p オプションを指定しない場合は、パッケージのインスタンスが記載されたシステムのパブリッシャー設定一覧の 1 番目のパブリッシャーが使用されます。

`-s|--source` *FMRI_or_ISO*
省略可能: ネットイメージのデータソースを指定します。指定可能な値は次のいずれかです。

- `pkg`(5) パッケージのフル FMRI。
- AI ISO イメージへのパス。

`-s` が指定されない場合は、使用される `install-image/solaris-auto-install` パッケージは次のいずれかです。

- `-p` パラメータで指定されたパブリッシャー。
- パッケージのインスタンスが記載されたシステムのパブリッシャー設定一覧の 1 番目のパブリッシャー。

`-b|--boot-args` *boot_property=value*`,...`
省略可能: x86 クライアント専用。 サービスイメージのサービス固有の `menu.lst` ファイルにプロパティー値を設定します。このオプションを使用して、このサービスに固有のブートプロパティーを設定します。このオプションでは、コンマで区切った *boot_property= value* のペアを複数指定できます。

`-i|--ip-start` *dhcp_ip_start*
省略可能: ローカル DHCP 構成に追加される範囲内の開始 IP アドレスを指定します。IP アドレスの数は、`-c` オプションで指定されます。 ローカル ISC DHCP 構成が存在しない場合は、ISC DHCP サーバー が起動されます。

`-c|--ip-count` *count_of_ipaddr*
省略可能: DHCP 構成の IP アドレスの総数を *count_of_ipaddr* の値と同じ数に設定します。最初の IP アドレスは、`-i` オプションで指定される *dhcp_ip_start* の値です。

`-B|--bootfile-server` *server_ipaddr*
省略可能: クライアントがブートファイルを要求するブートサーバーの IP アドレスを指定します。 この IP アドレスをその他の方法で決定できない場合にのみ必要です。

`-d|--imagepath` *imagepath*
省略可能: ネットイメージを作成するパスを指定します。 これが指定されない場合は、デフォルトの場所 (`/export/auto_install/`*svcname*) が使用されます。`-y` も指定しなければ、確認プロンプトが表示されます。

`-y|--noprompt`
省略可能: 確認プロンプトを非表示にして、指定されたオプションおよびデフォルト値を使用したサービスの作成を続行します (`-d` を参照)。

`set-service -o|--option` *prop=value svcname*

`-o|--option` *prop=value*
設定するプロパティーおよび値を指定します。

prop=*value* に指定可能な値は次のとおりです。

- `aliasof=`*existing_service*

  svcname を *existing_service* のエイリアスにします。

- `default-manifest=`*manifest_name*

  指定されたサービスにすでに登録されている特定のマニフェストまたはスクリプトを、そのサービスのデフォルトマニフェストまたはスクリプトに指定します。このサービスに登録されているマニフェストおよびスクリプトの一覧を表示するには、次のコマンドを使用します。

  ```
  $ installadm list -n svcname -m
  ```

svcname
必須: 設定するプロパティーを含むインストールサービスの名前を指定します。

`rename-service` *svcname newsvcname*
インストールサービス *svcname* の名前を *newsvcname* に変更します。*newsvcname* は、英数字、アンダースコア (_)、ハイフン (-) で構成できます。*newsvcname* の最初の文字をハイフンにすることはできません。

`enable` *svcname*
svcname インストールサービスを有効化します。

`disable` *svcname*
svcname インストールサービスを無効化します。

`delete-service [-r|--autoremove ] [-y|--noprompt]` *svcname*
インストールサービスを削除します。次の操作を実行します。

- このインストールサービスのマニフェスト、プロファイル、クライアント構成ファイル、および Web サーバー構成を削除します。
- サービスのインスタンス化に使用されたイメージを削除します。
- サービスがデフォルトのエイリアスであり、ローカル ISC DHCP 構成が存在する場合は、このサービスに関連付けられたブートファイルが ISC DHCP 構成から削除されます。

`-r|--autoremove`
これが指定された場合は、このサービスに割り当てられたクライアント、およびこのサービスにエイリアスが設定されたサービスも削除されます。

`-y|--noprompt`
確認プロンプトを非表示にして、サービスの削除を続行します。

svcname
必須: 削除するインストールサービス名を指定します。

`list [-n|--service` *svcname*`] [-c|--client] [-m|- -manifest] [-p|--profile]`
サーバーで有効化されているすべてのインストールサービスを一覧表示します。

`-n|--service` *svcname*
省略可能: ローカルサーバーにある特定のインストールサービスに関する情報を一覧表示します。

- `-c` オプションが指定された場合は、インストールサービスに関連付けられたクライアント情報を一覧表示します。
- `-m` オプションが指定された場合は、インストールサービスに関連付けられたマニフェストおよびスクリプトを一覧表示します。
- `-p` オプションが指定された場合は、インストールサービスに関連付けられたプロファイルを一覧表示します。

`-c|--client`
省略可能: ローカルサーバーにあるインストールサービスのクライアントを一覧表示します。

`-m|--manifest`
省略可能: ローカルサーバーにあるインストールサービスに関連付けられたマニフェストおよびスクリプトを一覧表示します。

`-n` が指定されない場合は、サービスごとに簡易一覧を表示します。この一覧には、デフォルトのマニフェストまたはスクリプト、および関連付けられた条件があるデフォルト以外のすべてのマニフェストおよびスクリプトが含まれます。条件は一覧表示されません。

`-n` が指定された場合は、指定されたサービスのすべてのマニフェストおよびスクリプトを、マニフェストごとの条件を含む完全な一覧形式を使用して表示します。アクティブでない (関連付けられた条件がなく、デフォルトとして指定されていない) マニフェストは、非アクティブとマークされます。デフォルトのマニフェストに関連付けられた条件は、非アクティブとマークされます。

`-p|--profile`
省略可能: ローカルサーバーにあるインストールサービスに関連付けられたプロファイルを一覧表示します。

`-n` が指定されない場合は、プロファイル名を含むサービスごとに簡易一覧を表示します。

`-n` が指定された場合は、要求されたサービスのプロファイルをその条件とともに表示します。

`create-manifest -n|--service` *svcname* `-f|--file` *manifest_or_script_filename* `[-m|--manifest` *manifest_name* `] [-c|--criteria` *criteria* `=`*value*`|`*list*`|` *range*`... | -C|--criteria-file` *criteriafile*`] [-d|--default]`

特定のインストールサービスのマニフェストまたはスクリプトを作成します。したがって、サービスの作成とは別に、マニフェストまたはスクリプトをネットワーク上で使用可能にします。デフォルト以外のマニフェストまたはスクリプトは、条件が関連付けられている場合にのみ使用できます (アクティブにすることができます)。条件はコマンド行 (`-c`) または条件 XML ファイル (`-C`) 経由で入力できます。`-d` オプションとともに指定された条件は、マニフェストまたはスクリプトがデフォルトとして指定されなくなるまで、一時的に無視されます。

マニフェストの名前は次の順序で決定されます。

1. `-m` オプションで指定された *manifest_name* (存在する場合)。
2. `ai_instance` `name` 属性の値 (マニフェストに存在する場合)。
3. マニフェストまたはスクリプトファイル名のベース名。

`-n|--service` *svcname*
必須: このマニフェストまたはスクリプトが関連付けられるインストールサービスの名前を指定します。

`-f|--file` *manifest_or_script_filename*
必須: 追加するマニフェストまたはスクリプトのパス名を指定します。

`-m|--manifest` *manifest_name*
省略可能: マニフェストまたはスクリプトの AI インスタンス名を指定します。マニフェストの `ai_instance` 要素の `name` 属性を *manifest_name* に設定します。 マニフェストまたはスクリプトは、後続の `installadm` コマンドおよび `installadm list` 出力の *manifest_name* で参照されます。

`-c|--criteria` *criteria=value*`|`*list*`|` *range*`...`
省略可能: 追加されたマニフェストまたはスクリプトに関連付けられる条件を指定します。後述の「条件」のセクションを参照してください。デフォルトのマニフェストを公開すると、条件は登録されますが、マニフェストまたはスクリプトがデフォルトに指定されなくなるまで非アクティブのままです。`-c` オプションは複数回指定できます。

`-C|--criteria-file` *criteriafile*
省略可能: 追加されたマニフェストまたはスクリプトに関連付けられる条件を含む条件 XML ファイルのパス名を指定します。 デフォルトのマニフェストまたはスクリプトを公開すると、条件は登録されますが、マニフェストまたはスクリプトがデフォルトに指定されなくなるまで非アクティブのままです。

`-d|--default`
省略可能: このマニフェストまたはスクリプトがサービスの新規デフォルトマニフェストまたはスクリプトであることを指定します。マニフェストまたはスクリプトがデフォルトに設定されなくなるまで、指定された条件は無視されます。

`update-manifest -n|--service` *svcname* `-f|--file` *manifest_or_script_filename* `[-m|--manifest` *manifest_name* `]`
特定のインストールサービスに関連付けられた特定のマニフェストまたはスクリプトを更新します。更新後もマニフェストまたはスクリプトの条件やデフォルトステータスはそのままです。

マニフェストの名前は次の順序で決定されます。

1. `-m` オプションで指定された *manifest_name* (存在する場合)。
2. `ai_instance name` 属性の値 (変更されたマニフェストに存在する場合、または既存のマニフェストの `ai_instance name` 値に一致する場合)。
3. マニフェストまたはスクリプトファイル名のベース名 (既存マニフェストの `ai_instance name` 属性値に一致する場合)、または `installadm list` で指定された名前 (既存スクリプトの名前に一致する場合)。

置換用のマニフェストまたはスクリプトは *manifest_or_script_filename* で指定されます。

`-n|--service` *svcname*
必須: このマニフェストまたはスクリプトが関連付けられるインストールサービスの名前を指定します。

`-f|--file` *manifest_or_script_filename*
必須: 置換用のマニフェストまたはスクリプトのパス名を指定します。

`-m|--manifest` *manifest_name*
省略可能: 置換用のマニフェストまたはスクリプトの AI インスタンス名を指定します。

`delete-manifest -m|--manifest` *manifest_name* `-n|--service` *svcname*
特定のインストールサービスで公開されたマニフェストまたはスクリプトを削除します。 デフォルトのマニフェストまたはスクリプトは削除できません。

`-m|--manifest` *manifest_name*
必須: `installadm list` に `-n` オプションを付けてマニフェストまたはスクリプトの AI インスタンス名を出力として指定します。

`-n|--service` *svcname*
必須: このマニフェストが関連付けられたインストールサービスの名前を指定します。

`create-profile -n|--service` *svcname* `-f|--file` *profile_filename* `... [-p|--profile` *profile_name* `] [-c|--criteria` *criteria =value|list|* *range*`... | -C|--criteria-file` *criteriafile*`]`
特定のインストールサービスのプロファイルを作成します。条件はオプションで、コマンド行 (`-c`) で入力するか、または条件 XML ファイル (`-C`) 経由でプロファイルに関連付けることができます。条件なしで作成されたプロファイルは、サービスのすべてのクライアントに関連付けられます。

プロファイルの名前は次の順序で決定されます。

1. - p オプションで指定された *profile_name* (存在する場合)。
2. プロファイルファイル名のベース名。

プロファイル名は AI サービスで一意である必要があります。 複数の -f オプションを使用して、同じ条件を持つ複数のプロファイルを作成すると、-p オプションは無効になり、プロファイルの名前はファイル名から派生します。

`-n|--service` *svcname*
必須: 更新されるインストールサービスの名前を指定します。

`-f|--file` *profile_filename*`...`
必須: プロファイルを追加するためのファイルのパス名を指定します。複数のプロファイルを指定できます。

`-p|--profile` *profile_name*
省略可能: 作成されるプロファイルの名前を指定します。単一プロファイルの作成にのみ有効です。

`-c|--criteria` *criteria*=*value*|*list*| *range*`...`
省略可能: プロファイルに関連付ける条件を指定します。後述の「条件」のセクションを参照してください。複数の -c オプションを指定できます。

`-C|--criteria-file` *criteriafile*
省略可能: 指定されたプロファイルに関連付けられる条件を含む条件 XML ファイルのパス名を指定します。

`delete-profile -p|--profile` *profile_name*`... -n|--service` *svcname*
profile_name プロファイルを *svcname* インストールサービスから削除します。

`-p|--profile` *profile_name*`...`
必須: 削除するプロファイルの名前を指定します。複数の -p オプションを指定できます。

`-n|--service` *svcname*
必須: 削除するプロファイルのインストールサービスの名前を指定します。

`export -n|--service` *svcname* `-m|--manifest` *manifest_name*`... -p|--profile` *profile_name*`... [-o|--output` *pathname*`]`
指定されたマニフェスト/スクリプト、またはサービスに属するプロファイル (あるいはその両方) を表示 (エクスポート) します。1 つ以上のマニフェスト/スクリプトまたはプロファイルを指定する必要があります。-o オプションでファイルまたはディレクトリにリダイレクトされない限り、標準出力に表示されます。

`-n|--service` *svcname*
必須: エクスポートするマニフェストまたはプロファイルに関連付けられたインストールサービスを指定します。

`-m|--manifest` *manifest_name*...
エクスポートするマニフェストまたはスクリプトの AI インスタンス名を指定します。複数の `-m` オプションを指定できます。

`-p|--profile` *profile_name*...
エクスポートするプロファイルの名前を指定します。複数の `-p` オプションを指定できます。

`-o|--output` *pathname*
省略可能: 出力をリダイレクトします。複数のマニフェスト、スクリプト、またはプロファイル (あるいは両方) が要求される場合は、*pathname* にディレクトリを指定する必要があります。1 つのマニフェスト、スクリプト、またはプロファイルのみが要求される場合は、*pathname* にファイルを指定できます。

`validate -n|--service` *svcname* `-P|--profile-file` *profile_filename*... |
`-p|--profile` *profile_name*...
指定されたプロファイルを検証します。 `validate` サブコマンドを使用すると、データベース (`-p`) 内のプロファイルを検証したり、データベース (`-P`) への入力前に開発中のプロファイルを検証したりできます。

`-n|--service` *svcname*
必須: プロファイルが関連付けられたサービスを指定します。

`-P|--profile-file` *profile_filename*...
検証する外部プロファイルファイルを指定します。

`-p|--profile` *profile_name*...
検証するプロファイルの名前を指定します。

`set-criteria -m|--manifest` *manifest_name* `-p|--profile` *profile_name*...
`-n|--service` *svcname* `-c|--criteria` *criteria=value*|*list*| *range*... |
`-C|--criteria-file` *criteriafile* | `-a|--append-criteria` *criteria=value*|*list*| *range*...
公開済みのマニフェスト/スクリプトまたはプロファイル (あるいは両方) の条件を更新します。条件はコマンド行または条件 XML ファイル経由で指定できます。相互に排他的なオプション `-a`、`-c`、または `-C` のいずれかを使用して、条件を指定する必要があります。

有効な条件については、`create-manifest` サブコマンドで説明します。

`-m|--manifest` *manifest_name*
マニフェストまたはスクリプトの AI インスタンス名を指定します。

`-p|--profile` *profile_name*...
プロファイルの名前を指定します。任意の数のプロファイルを指定できます。

`-n|--service` *svcname*
必須: このマニフェスト/スクリプトまたはプロファイルに関連付けられたインストールサービスの名前を指定します。

`-c|--criteria` *criteria=value|list| range...*
マニフェスト/スクリプトまたはプロファイルに対する既存の条件をすべて置換するための条件を指定します。後述の「条件」のセクションを参照してください。

`-C|--criteria-file` *criteriafile*
マニフェスト/スクリプトまたはプロファイルに対する既存の条件をすべて置換するための条件を含む条件 XML ファイルのパス名を指定します。

`-a|--append-criteria` *criteria=value|list| range...*
マニフェスト/スクリプトまたはプロファイルに対する既存の条件に追加するための条件を指定します。後述の「条件」のセクションを参照してください。指定済みの *criteria* がすでに存在する場合は、その条件の *value|list| range* は *value |list|range* で置換されます。

`create-client [-b|--boot-args` *property=value*`,...] - e|--macaddr` *macaddr*
`-n |--service` *svcname*
`create-service` サブコマンドで使用されるデフォルト設定とは異なるカスタムクライアント設定を行うために、指定されたクライアントに省略可能な設定タスクを実行します。 ユーザーはクライアントにデフォルト以外のサービス名およびブート引数を指定できます。既存クライアントの変更にも使用できます。

クライアントが x86 システムであり、ローカル ISC DHCP 構成が存在する場合は、ISC DHCP 構成でクライアントが構成されます。

`-b|--boot-args` *property=value*`,...`
省略可能: x86 クライアント専用。 `/etc/netboot` にあるクライアント固有の `menu.lst` ファイルにプロパティー値を設定します。このオプションは、対象クライアントに固有のブートプロパティーを設定するのに使用します。このオプションでは、*property=value* のペアを複数指定できます。

`-e|--macaddr` *macaddr*
必須: クライアントの MAC アドレスを指定します。

`-n|--service` *svcname*
必須: クライアントインストールのインストールサービスを指定します。

`delete-client` *macaddr*
`create-client` サブコマンドを使用して事前に設定された既存クライアントの特定のサービス情報を削除します。

クライアントが x86 システムであり、ローカル ISC DHCP 構成が存在する場合は、 ISC DHCP 構成でクライアントが構成解除されます。

macaddr　必須: 削除するクライアントの MAC アドレスを指定します。

条件　マニフェスト、スクリプト、およびプロファイルを使用すれば、特定の特性または条件に従って AI クライアントを個別に構成できます。 特定のクライアントには、1

つのマニフェストまたはスクリプトのみを関連付けることができます。特定のクライアントには、任意の数のプロファイルを関連付けることができます。

条件の値は、起動中に AI クライアントによって決定されます。

次の AI クライアントシステムは、特に注記がない限り、マニフェスト/スクリプトおよびプロファイルの両方に指定できます。

コマンド行で条件を指定する方法については、「使用例」のセクションを参照してください。条件ファイルの作成についての詳細は、『Oracle Solaris 11 システムのインストール』を参照してください。

条件	説明
arch	uname -m ごとのアーキテクチャー。
cpu	uname -p ごとの CPU クラス。
hostname	割り当てられたホスト名。プロファイルでのみ使用可能です (マニフェストでは不可)。
ipv4	IP バージョン 4 のネットワークアドレス。
mac	コロン (:) で区切られた 16 進数の MAC アドレス。
mem	prtconf(1M) ごとのメモリーサイズ (MB)。
network	IP バージョン 4 のネットワーク番号。
platform	uname -i ごとのプラットフォーム名。
zonename	zones(5) ごとのゾーン名。

ipv4、mac、mem、および network の指定は、ハイフン (-) で区切られた範囲値で表現できます。範囲の一端に制限なしを指定するには、unbounded を使用します。

範囲として指定できない条件は、空白で区切られた値の一覧として指定できます。

使用例

例 1　ISO ファイルから新規 x86 インストールサービスを設定する

インストールサーバーおよび x86 インストールサービスをはじめて設定します。DHCP サーバーを構成するために、コマンドには開始 IP アドレスおよび IP アドレスの合計数が含まれます。

```
# installadm create-service -n sol-11-i386-svc \
-s /export/isos/sol-11-i386.iso \
-i 172.0.0.10 -c 10 -d /export/images/soli386
```

例 1　ISO ファイルから新規 x86 インストールサービスを設定する　　(続き)

AI ISO イメージは /export/isos/sol-11-i386.iso にあります。コマンドは、AI ISO イメージに基づいた /export/images/soli386 にあるネットイメージおよびインストールサービスを設定します。 このネットイメージを使用すると、クライアントインストールを実行できます。

開始 IP アドレス 172.0.0.10 および 10 個の IP アドレスがローカル ISC DHCP 構成に追加されます。 ローカル ISC DHCP 構成が存在しない場合は、ISC DHCP サーバー が起動されます。

これは最初に作成される i386 サービスであるため、default-i386 サービスが自動的に作成され、このサービスにエイリアスが設定されます。default-i386 エイリアスは稼働中であり、PXE 経由でブートされたクライアントは default-i386 サービスからブートおよびインストールします。

例 2　ISO ファイルから新規 SPARC インストールサービスを設定する

SPARC インストールサービスをはじめて設定します。

```
# installadm create-service -n sol-11-sparc-svc \
-s /export/isos/sol-11-sparc.iso \
-d /export/images/solsparc
```

AI ISO イメージは /export/isos/sol-11-sparc.iso にあります。コマンドは、AI ISO イメージに基づいた /export/images/solsparc にあるネットイメージおよびインストールサービスを設定します。 このネットイメージを使用すると、クライアントインストールを実行できます。

これは最初に作成される SPARC サービスであるため、default-sparc サービスが自動的に作成され、このサービスにエイリアスが設定されます。default-sparc エイリアスは稼働中であり、PXE 経由でブートされたクライアントは default-sparc サービスからブートおよびインストールします。

例 3　パッケージリポジトリから i386 インストールサービスを設定する

```
# installadm create-service -y -n mysvc
```

i386 インストールサーバーでは、このコマンドはデフォルトのイメージの場所 (/export/auto_install/mysvc) に、i386 ネットイメージおよび mysvc というインストールサービスを設定します。 -y オプションは、デフォルトの場所が受け入れ可能かどうかを確認します。アーキテクチャーが指定されていない場合は、作成されるサービスがインストールサーバーと同じアーキテクチャーのものになります。 このコマンドでは、インストールサーバーの pkg publisher 一覧にあるパッケージリポジトリに install-image/solaris-auto-install パッケージが含まれることが前提となっています。

このサーバーへの SPARC サービスの作成を指定するには、-a オプションを使用します。

例 3 パッケージリポジトリから i386 インストールサービスを設定する (続き)

solaris-auto-install パッケージのソースを指定するには、-p オプションを使用します。 たとえば、http://example.company.com:4281 にある ai-image リポジトリを solaris-auto-install パッケージのソースとして指定するには、次のコマンドを使用します。

```
# installadm create-service -y -n mysvc \
-p ai-image=http://example.company.com:4281
```

例 4 クライアントをインストールサービスに関連付ける

クライアントを特定のインストールサービスに関連付けるには、次のサンプルコマンドを使用します。インストールサービスはすでに存在する必要があります。

```
# installadm create-client -b "console=ttya" \
-e 0:e0:81:5d:bf:e0 -n my-i386-service
```

この例のコマンドは、MAC アドレスが 0:e0:81:5d:bf:e0 のシステムにクライアント固有の設定を作成します。このクライアントは、事前に設定された my-i386-service というインストールサービス、およびそのサービスに関連付けられたネットイメージを使用します。このコマンドは、/etc/netboot にあるクライアント固有の menu.lst ファイルにブートプロパティー console=ttya を設定します。

例 5 デフォルトサービスを変更せずに新規インストールサービスを追加する

既存のサービスを保持し、既存のデフォルトを変更しないで、my-sparc-service という新規サービスを追加するには、次のサンプルコマンドを使用します。

```
# installadm create-service -n my-sparc-service \
-s /export/isos/mysparc.iso \
-d /export/ai/mysparc-image
```

例 6 新規インストールサービスを追加してデフォルトサービスを変更する

既存のサービスを保持し、新規サービスを SPARC クライアントのデフォルトにして、my-sparc-service という新規サービスを追加するには、次の 2 つのサンプルコマンドを使用します。

```
# installadm create-service -n my-sparc-service \
-s /export/isos/mysparc.iso \
-d /export/ai/mysparc-image
# installadm set-service \
-o aliasof=my-sparc-service default-sparc
```

例 7 カスタムのデフォルト AI マニフェストをインストールサービスに追加する

新規マニフェストを service_092910 インストールサービスに追加して、サービスのデフォルトマニフェストにするには、次のサンプルコマンドを使用します。マニフェストデータは my_manifest.xml にあります。将来の installadm コマンドでは、このマニフェストは my_manifest として参照されます。

例 7　カスタムのデフォルト AI マニフェストをインストールサービスに追加する　(続き)

```
# installadm create-manifest -d -f my_manifest.xml \
-m my_manifest -n service_092910
```

例 8　派生したマニフェストスクリプトをインストールサービスに追加する

my_script という派生したマニフェストスクリプトを service_092910 という既存のインストールサービスに追加するには、次のサンプルコマンドを使用します。スクリプトはマニフェストと同じ方法で追加されます。

```
# installadm create-manifest -f my_script.py \
-m my_script -n service_092910
```

派生したマニフェストスクリプトの作成方法についての詳細は、『Oracle Solaris 11 システムのインストール』を参照してください。

例 9　インストールサービスのデフォルト AI マニフェストを置換する

既存のインストールサービス service_092910 のデフォルトマニフェストを、すでに my_manifest としてサービスに追加されているカスタムマニフェストに置換するには、次のサンプルコマンドを使用します。マニフェストは -m my_manifest を create-manifest サブコマンドに指定することによって、サービスに追加されました。

```
# installadm set-service -o default-manifest=my_manifest \
service_092910
```

例 10　インストールサービスを一覧表示する

ローカルサーバーにあるインストールサービスを一覧表示するには、次のサンプルコマンドを使用します。

```
$ installadm list
Service Name     Alias Of         Status  Arch  Image Path
------------     --------         ------  ----  ----------
default-i386     sol-11-i386-svc  on      x86   /export/images/soli386
default-sparc    sol-11-sparc-svc on      Sparc /export/images/solsparc
sol-11-i386-svc  -                on      x86   /export/images/soli386
sol-11-sparc-svc -                on      Sparc /export/images/solsparc
```

例 11　インストールサービスに関連付けられたクライアントを一覧表示する

ローカルサーバーにある特定のインストールサービスのクライアントを一覧表示するには、次のサンプルコマンドを使用します。

```
$ installadm list -c -n my-x86-service
Service Name   Client Address    Arch  Image Path
------------   --------------    ----  ----------
my-x86-service 01:C2:52:E6:4B:E1 i386  /export/images/myimage
```

例 12　インストールサービスに関連付けられたマニフェストを一覧表示する

ローカルサーバーにある特定のインストールサービスに関連付けられたマニフェストおよびスクリプトを一覧表示するには、次のサンプルコマンドを使用します。

```
$ installadm list -m -n my-x86-service
Manifest      Status    Criteria
--------      ------    --------
manifest2               arch = i86pc
                        mem  = 4096 MB - unbounded

sparc_setup             arch = sun4v

new_default   Default   (Ignored: mem = 2048 MB - 4095 MB)

orig_default  Inactive  None
```

この例では、次の出力が表示されます。

- 条件付きのデフォルト以外のマニフェスト (manifest2)
- 条件付きのデフォルト以外のスクリプト (sparc_setup)
- 無視される条件付きのデフォルトマニフェスト (new_default)
- 条件がないために非アクティブとマークされているデフォルト以外のマニフェスト (orig_default)

例 13　プロファイルを一覧表示する

ローカルサーバーにあるプロファイルを一覧表示するには、次のサンプルコマンドを使用します。

```
$ installadm list -p
Service Name Profile
------------ -------
sparc2       myprofile.xml
             myprofile2.xml
svc0817      profile3
svc0819      profile4.xml
             newprofile
             foo.xml
```

例 14　名前なしのカスタム AI マニフェストをインストールサービスに追加する

/export/my_manifest.xml のマニフェストを、MAC アドレスが aa:bb:cc:dd:ee:ff に等しいという条件付きで svc1 に追加するには、次のサンプルコマンドを使用します。

```
# installadm create-manifest -f /export/my_manifest.xml \
-n svc1 -c MAC="aa:bb:cc:dd:ee:ff"
```

例 14 名前なしのカスタム AI マニフェストをインストールサービスに追加する (続き)

この例では、マニフェストに名前属性が含まれないため、マニフェスト名はファイル名から取得されます。

```
$ installadm list -m -n svc1
Manifest       Criteria
--------       --------
my_manifest    mac  = AA:BB:CC:DD:EE:FF
```

例 15 カスタム名付きのカスタム AI マニフェストをインストールサービスに追加する

/export/my_manifest.xml のマニフェストを、IPv4 範囲が 10.0.2.100 - 10.0.2.199 という条件付きで svc1 に追加するには、次のサンプルコマンドを使用します。

```
# installadm create-manifest -f /export/my_manifest.xml \
-n svc1 -m chosen_name \
-c IPV4="10.0.2.100-10.0.2.199"
```

この例では、マニフェスト名は -m オプションから取得されます。

```
$ installadm list -m -n svc1
Manifest       Criteria
--------       --------
chosen_name    ipv4 = 10.0.2.100 - 10.0.2.199
```

例 16 マニフェストに指定された名前付きのカスタム AI マニフェストを追加する

/export/manifest3.xml のマニフェストを、メモリーが 2048 MB 以上で、アーキテクチャーが i86pc であるという条件付きで svc1 に追加するには、次のサンプルコマンドを使用します。

```
# installadm create-manifest -f /export/manifest3.xml \
-n svc1 -c MEM="2048-unbounded" -c ARCH=i86pc
```

この例では、次の部分的なマニフェストで示すように、マニフェスト名はマニフェストの ai_instance 要素の name 属性から取得されます。

```
<auto_install>
    <ai_instance name="my_name" />
</auto_install>
```

```
$ installadm list -m -n svc1
Manifest    Criteria
--------    --------
my_name     arch = i86pc
            mem  = 2048 MB - unbounded
```

例 17 システム構成プロファイルをインストールサービスに追加する

/export/profile4.xml のプロファイルを、ホスト名のいずれかが myhost1、host3、または host6 であるという条件付きで svc1 に追加するには、次のサンプルコマンドを使用します。

```
# installadm create-profile -f /export/profile4.xml \
-n svc1 -p profile4 -c hostname="myhost1 host3 host6"
$ installadm list -p -n svc1
Profile          Criteria
-------          --------
profile4         hostname = myhost1 host3 host6
```

例 18 すべてのクライアントのシステム構成プロファイルを追加する

条件を指定しない場合、プロファイルは指定されたインストールサービスを使用するすべてのクライアントによって使用されます。次の例では、作成されたプロファイルが svc1 サービスを使用するすべてのクライアントによって使用されます。

```
# installadm create-profile -f /export/locale.xml -n svc1
$ installadm list -p -n svc1
Profile          Criteria
-------          --------
profile4         hostname = myhost1 host3 host6
locale
```

例 19 置換タグ付きのシステム構成プロファイルを追加する

プロファイルでは、置換タグを使用できます。置換タグは、ユーザーの環境 (environ(4) を参照) から、または create-profile サブコマンドの -c オプションで指定された条件から取得されたカスタムクライアント構成情報のためのプレースホルダとして機能します。置換タグを使用すると、さまざまなシステム向けにプロファイルファイルを再利用できます。次の例では、各プロファイルが -c 条件オプションから取得された hostname 値で格納されます。

```
# installadm create-profile -p myhost1_hostname \
-f /export/hostname.xml -n svc1 -c hostname=myhost1
# installadm create-profile -p myhost2_hostname \
-f /export/hostname.xml -n svc1 -c hostname=myhost2
$ installadm list -p -n svc1
Profile              Criteria
-------              --------
myhost1_hostname     hostname = myhost1
myhost2_hostname     hostname = myhost2
```

hostname.xml ファイルには次の行が含まれます。

```
<propval name="nodename" value="{{AI_HOSTNAME}}"/>
```

例 19　置換タグ付きのシステム構成プロファイルを追加する　　(続き)

create-profile コマンドは、myhost1_hostname プロファイルに次の行が含まれるように置換します。

```
<propval name="nodename" value="myhost1"/>
```

同じ hostname.xml 入力ファイルを使用すると、 myhost2_hostname プロファイルに次の行が含まれます。

```
<propval name="nodename" value="myhost2"/>
```

hostname 条件が使用され、プロファイルに代入されたため、置換タグ {{AI_HOSTNAME}} は create-profile 呼び出しごとに異なる値で置換されます。置換タグの使用についての詳細は、『Oracle Solaris 11 システムのインストール』を参照してください。

例 20　条件を既存のマニフェストに追加する

メモリーが 4096 MB 以上であるという条件を svc1 の manifest2 の条件に追加するには、次のサンプルコマンドを使用します。

```
# installadm set-criteria -m manifest2 -n svc1 \
-a MEM="4096-unbounded"
```

例 21　既存のマニフェストに対する条件を置換する

svc1 の manifest2 の条件を /tmp/criteria.xml ファイルに指定された条件で置換するには、次のサンプルコマンドを使用します。

```
# installadm set-criteria -m manifest2 -n svc1 \
-C /tmp/criteria.xml
```

条件 XML ファイルの内容についての詳細は、『Oracle Solaris 11 システムのインストール』を参照してください。

例 22　開発中にプロファイルファイルを検証する

myprofdir/myprofile.xml および herprofdir/herprofile.xml ファイルに格納されたプロファイルを開発中に検証するには、次のサンプルコマンドを使用します。

```
# installadm validate -P myprofdir/myprofile.xml \
-P herprofdir/herprofile.xml -n svc1
```

例 23　プロファイルの内容をエクスポートする

サービス svc1 のプロファイル myprofile.xml をエクスポートするには、次のサンプルコマンドを使用します。

```
$ installadm export -p myprofile -n svc1
```

例 24　既存の AI マニフェストの内容を置換する

マニフェスト名 (または AI インスタンス名) が spec のサービス svc2 のマニフェストを、/home/admin/new_spec.xml ファイルのマニフェストの内容に置換するには、次のサンプルコマンドを使用します。

```
# installadm update-manifest -n svc2 \
-f /home/admin/new_spec.xml -m spec
```

例 25　既存の AI マニフェストをエクスポートして更新する

サービス svc2 の spec という既存のマニフェストのデータをエクスポートし、そのマニフェストを変更済みの内容で更新するには、次のサンプルコマンドを使用します。

```
$ installadm export -n svc2 -m spec -o /home/admin/spec.xml
```

/home/admin/spec.xml の変更を行います。

```
# installadm update-manifest -n svc2 \
-f /home/admin/spec.xml -m spec
```

終了ステータス

次の終了ステータスが返されます。

0　　コマンドは正常に処理されました。

>0　　エラーが発生した。

属性

属性についての詳細は、マニュアルページの attributes(5) を参照してください。

属性タイプ	属性値
使用条件	install/installadm
インタフェースの安定性	不確実

関連項目

aimanifest(1M), sysconfig(1M), dhcp(5), dhcpd(8), smf(5), service_bundle(4), ai_manifest(4), environ(5)

『Oracle Solaris 11 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」

『Oracle Solaris 10 JumpStart から Oracle Solaris 11 自動インストーラへの移行』

名前

js2ai – 自動インストーラ (AI) で使用するための JumpStart ルールおよびプロファイルの変換

形式

```
js2ai [-h | --version]

js2ai -r | -p profile_name [-d jumpstart_dir]
    [-D destination_dir] [-lSv]

js2ai -s [-d jumpstart_dir]
    [-D destination_dir] [-Sv]

js2ai -V manifest
```

機能説明

`js2ai` は、Oracle Solaris 10 JumpStart の `rules`、プロファイル、および `syscfg` 構成ファイルを、自動インストーラ (AI) と互換性がある形式に変換するためのユーティリティーです。このユーティリティーは最適な方法で、AI コンテキストに変換可能な JumpStart キーワードを変換します。この変換で JumpStart と完全に 1 対 1 で対応するものは作成されませんが、あとで JumpStart 構成ファイルから収集された情報に基づいて、完全な AI 構成設定を作成するためのテンプレートとして使用できる AI マニフェストおよびシステム構成プロファイルのエントリが提供されます。

`js2ai` を使用すると、次の操作を実行できます。

- 現在の作業用ディレクトリにおける `rules` ファイルおよび関連付けられたプロファイルの処理。
- 指定されたディレクトリにおける `rules`ファイルおよび関連付けられたプロファイルの処理。
- 特定のプロファイルまたは `sysidcfg` ファイルの処理。
- 生成された出力ファイルの特定のディレクトリへの送信。`js2ai` 出力ファイルについての詳細は、「使用例」および「ファイル」のセクションを参照してください。

ルールのキーワードを変換する

表 1　JumpStart ルールのキーワード変換

JumpStart ルールのキーワード	AI 条件のキーワード
arch	cpu
hostaddress	ipv4
karch	arch
memsize	mem
model	platform
network	ipv4

js2ai でサポートされていない JumpStart ルールのキーワード:

```
any             installed
disksize        osname
domainname      probe
hostname        totaldisk
```

プロファイルのキーワードを変換する

表 2　JumpStart プロファイルのキーワード

JumpStart プロファイルのキーワード	注意事項
boot_device	rootdisk は、事前に root_device キーワードで設定されていない場合、指定されたデバイスに設定されます。
fdisk	disk_name の値はデバイスにする必要があります。all のデバイスはサポートされていません。fdisk タイプは solaris にする必要があります。サイズ 0 または delete はサポートされていません。 partitioning が default であり、rootdisk が設定されていない場合は、検出された 1 番目の fdisk solaris パーティションが rootdisk として使用されます。
filesys	指定されたマウントポイントが / または swap の場合は、ローカルおよびミラー化されたファイルシステムがサポートされます。 サイズの検証は実行されません。このマニフェストを使用したインストールに成功するには、生成された AI マニフェストに指定されたサイズを調整する必要がある場合があります。
install_type	値 initial_install のみがサポートされています。
locale	変換は実行されません。指定されたロケールが Oracle Solaris 11 でサポートされていることを確認してください。
パッケージ	指定されたパッケージを Oracle Solaris 11 のパッケージに変換しようと試みられます。パッケージの場所の指定はサポートされていません。パッケージの検索には、非常に長い時間がかかる可能性があります。プロファイルにパッケージの長形式のリストが含まれる場合は、変換プロセス中に --local フラグを使用できます。
partitioning	サポートされているタイプは default および explicit です。JumpStart とは異なり、partitioning default が指定された場合は、js2ai で認識されるディスクのみが使用されます。ディスクがどのキーワードでも指定されていない場合は、生成されたプロファイルによって、使用するディスクを選択するように AI に通知されます。

表 2 JumpStart プロファイルのキーワード *(続き)*

JumpStart プロファイルのキーワード	注意事項
プール	プロファイルでプールが指定されている場合は、指定されたデバイスを使用して ZFS ルートプールが作成されます。ZFS ルートプールで使用するデバイスを決定する際には、`pool` キーワードが他のすべてのキーワードに優先されます。 プールサイズ、スワップワイズ、またはダンプサイズの検証は実行されません。このマニフェストを使用したインストールに成功するには、生成された AI マニフェストでこれらのサイズを調整する必要がある場合があります。
`root_device`	`rootdisk` は指定されたデバイスに設定されます。
`system_type`	値 `standalone` のみがサポートされています。
`usedisk`	指定されたデバイスは、変換中に `any` または `rootdisk` デバイスを解決するために使用される場合があります。ZFS ルートプールがミラー化されていない場合は、この目的で使用されない指定されたデバイスが、そのプールに追加されます。

`js2ai` でサポートされていない JumpStart プロファイルのキーワード:

```
archive_location        geo
backup_media            layout_constraint
bootenv                 local_customization
client_arch             metabd
client_root             no_master_check
client_swap             no_content_check
cluster                 num_clients
dontuse                 patch
forced_deployment
```

プロファイルの変換中にシステムのルートディスクを決定する方法

プロファイルの変換プロセス中は、`js2ai` はプロファイルが参照する実際のシステムにアクセスしないため、`js2ai` はできる限り JumpStart と一致するプロセスを使用して、変換中にルートディスクを決定しようとします。

`js2ai` ツールは次のステップを実行して、ルートディスクで使用するデバイスを決定します。

手順	アクション
1	プロファイルで `root_device` キーワードが指定されている場合、`js2ai` は `rootdisk` をスライスが存在するデバイスに設定します。

手順	アクション
2	rootdisk が設定されておらず、プロファイルで boot_device キーワードが指定されている場合、js2ai は rootdisk をブートデバイスに設定します。
3	rootdisk が設定されておらず、partitioning default が指定され、solaris fdisk エントリが指定されている場合、js2ai は rootdisk を指定された disk_name に設定します。
4	rootdisk が設定されておらず、プロファイルで filesys *cwtxdysz size* / エントリが指定されている場合、js2ai は rootdisk をそのエントリで指定された *cwtxdysz* ディスクに設定します。
5	rootdisk が設定されておらず、プロファイルで usedisk *disk_name* エントリが指定されている場合、js2ai は rootdisk をそのエントリで指定された *disk_name* ディスクに設定します。
6	rootdisk が設定されておらず、プロファイルで次の指定が見つかった場合 (*size* は 0 または delete でなく、*disk_name* は all でない)、rootdisk はこの *disk_name* に設定されます。 `fdisk` *disk_name* `solaris` *size*
7	rootdisk が設定されていない場合、デバイスが rootdisk として指定されたオカレンスで変換エラーが生成されます。

プロファイルの変換中に任意のデバイスを変換する方法

js2ai ツールは次のステップを実行して、any キーワードが指定されているときに使用するデバイスを決定します。

手順	アクション
1	any デバイスが指定されていて、キーワードアクション (ミラー化されていない pool、または / マウントポイントを使用した filesys) が指定されている場合、any デバイスが rootdisk に設定されます (rootdisk が設定されている場合)。
2	any デバイスが変換されておらず、usedisk 文がプロファイル内に存在する場合、any デバイスが usedisk 文で指定されたデバイスに設定されます。
3	any デバイスが変換されておらず、any デバイスが指定されたアクションによって ZFS ルートプールが作成される場合、AI はそのデバイスを選択します。ミラー化されたプールが指定されている場合、これは適用できません。

プロファイルの変換中に **ZFS** ルートプールを決定する方法

js2ai ツールは次のステップを実行して、ZFS ルートプールで使用するデバイスを決定します。ZFS ルートプールが決定されると、その後に検出される定義がすでに決定されている ZFS ルートプールと競合する場合、エラーとしてフラグが設定されます。

手順	アクション
1	プロファイルで pool キーワードが指定されている場合、js2ai は ZFS ルートプールを pool キーワードで指定されたデバイスに設定します。
2	ZFS ルートプールが決定されておらず、プロファイルでマウントポイント / を使用して filesys が指定された場合、ZFS ルートプールは指定されたデバイスを使用して作成されます。
3	ZFS ルートプールが決定されておらず、プロファイル内のすべてのキーワードが処理され、rootdisk が設定されている場合、ZFS ルートプールは rootdisk デバイスを使用して作成されます。
4	ZFS ルートプールが決定されておらず、パーティションタイプが default の場合、AI は ZFS ルートプールで使用するデバイスを選択します。
5	ZFS ルートプールが決定されておらず、処理中にエラーが発生しなかった場合、AI は ZFS ルートプールで使用するデバイスを選択します。
6	ZFS ルートプールがミラー化されたプールではなく、指定された 1 つ以上の usedisk デバイスが rootdisk デバイスまたは any デバイスの変換で使用されていない場合、これらのディスクが ZFS ルートプールに追加されます。

sysidcfg キーワードを変換する

表 3　JumpStart sysidcfg キーワード

sysidcfg キーワード	注意事項
keyboard	変換は実行されません。sysidcfg ファイルで指定されたキーボードが Oracle Solaris 11 でサポートされていることを確認してください。
name_service	値 None、DNS、NIS 、および LDAP がサポートされています。NIS+ ネームサービスは NIS として変換されます。
network_interface	単一のインタフェースのみがサポートされています。PRIMARY のサポートは制限されています。sysidcfg ファイルで指定された 1 番目のインタフェースのみが処理されます。
root_password	変換は必要ありません。
security_policy	値 None がサポートされています。
service_profile	値 limited_net がサポートされています。
system_locale	変換は実行されません。sysidcfg ファイルで指定されたロケールが Oracle Solaris 11 でサポートされていることを確認してください。
terminal	変換は実行されません。sysidcfg ファイルで指定された端末タイプが Oracle Solaris 11 でサポートされていることを確認してください。
timeserver	値 localhost がサポートされています。
timezone	変換は必要ありません。

js2ai でサポートされていない JumpStart sysidcfg キーワード:

```
nfs4_domain
```

オプション

`js2ai` コマンドには次のオプションがあります。これらのオプションの使用については、「使用例」のセクションで説明します。

`-h`、`--help`
使用法のヘルプメッセージを表示します。

`--version`
`js2ai` ユーティリティーのバージョン番号を表示します。

`-d` *jumpstart_dir*、`- -dir` *jumpstart_dir*
`rules` ファイルおよびプロファイルファイル、または `sysidcfg` ファイルの場所を指定します。

`-D` *destination_dir*、`- -dest` *destination_dir*
出力ファイルの場所を指定します。

`-l`、`--local`
JumpStart プロファイルでの `package` キーワード値に相当する Image Packaging System (IPS) の値を検索する場合は、IPS パッケージリポジトリのパッケージではなく、ホストシステムにインストールされている IPS パッケージを検索します。

`-p` *profile_name*、`- -profile` *profile_name*
指定された JumpStart プロファイルを変換し、処理されたプロファイルに対応するマニフェストを生成します。この場合、条件ファイルは必要ないか、または生成されません。

`-r`、`--rule`
ルールおよび関連付けられたプロファイルを変換し、処理されたプロファイルごとにマニフェストを生成します。

`-s`、`--sysidcfg`
`sysidcfg` ファイルを処理し、その結果を `sc_profile.xml` に出力します。

`-S`、`--skip`
検証をスキップします。

`-v`、`--verbose`
処理中に発生したアクションに関する詳細を表示します。

`-V` *filename*
指定した AI マニフェストファイルまたは SMF システム構成プロファイルファイルを検証します。AI 条件の検証はサポートされていません。

エラーレポート

`js2ai` ツールは、変換中に 1 つ以上のエラーが発生するとエラーレポートを生成します。

```
# js2ai -r
                    Process  Unsupported  Conversion  Validation
Name      Warnings  Errors   Items        Errors      Errors
```

```
--------  --------  -------  -----------  ----------  ----------
rules            0         0           2           0           -
profile1         0         0           0           2           1

Conversion completed. One or more failures occurred.
For errors see ./js2ai.log
```

レポートには、`js2ai` でエラーが発生したファイルごとに 1 つのエントリが含まれます。エラーが発生しない場合でもエラーレポートを生成するには、`-v` または `--verbose` を指定します。

レポートでは、どのファイルでどのタイプのエラーが発生したのかが報告されます。定義されるエラーのタイプは、警告、処理エラー、サポート外項目、変換エラー、および検証エラーの 5 つです。

警告
: このメッセージの項目は修正する必要がありません。たとえば、ホスト名や root パスワードなどの情報が指定されなかったため、デフォルト値が使用される旨の警告メッセージを受信する場合があります。

処理エラー
: このエラーは、`js2ai` がファイルまたはファイル内の行を処理できなくなる問題を指します。通常、処理エラーはファイルに構文エラーがある場合に発生します。

サポート外項目
: この項目は、`js2ai` でサポートされていない行を指します。キーワードに関連付けられた値を変更すると、このエラーが発生しなくなる場合があります。

変換エラー
: このエラーは、`js2ai` が行を処理できなくなる状況を指します。これらのエラーを手動で修正するか、または問題のある行をファイルから削除するようにしてください。

検証エラー
: このエラーは、AI で使用されるスキーマ定義に対して生成されたマニフェストを検証したときに発生するエラーを指します。マニフェストが AI で使用される前に、これらのエラーを修正する必要があります。

`js2ai.log` ファイルには、どの行でどのエラーが発生したのかが示されます。

```
# cat js2ai.log
rules: line 4: unsupported keyword: disksize
rules: line 4: unsupported keyword: installed
net924_sun4c: line 4: unsupported keyword: cluster
net924_sun4c: line 5: unsupported keyword: num_clients
net924_sun4c: line 6: unsupported keyword: client_swap
net924_sun4c: line 7: unsupported keyword: client_arch
upgrade: line 1: unsupported value for 'install_type' specified: upgrade
```

マニフェストの検証エラーが発生した場合、次の例で示すように、`js2ai.log` ファイルには検証エラーを含むログファイルへのポインタが含まれます。

```
Validation Errors:
    profile1: manifest validation of
    ./AI_profile1/profile1.xml failed.
    For details see ./AI_profile1/profile_validation.log
```

変換の方針　ルールおよびプロファイルの変換で推奨される方針

JumpStart と AI との間には 1 対 1 の変換は存在しません。次のステップは、変換を実行するための一般的な手順を提供します。

1. `js2ai` ユーティリティーは、発生したエラーへのフラグ設定を試みます。ただし、`js2ai` では、変換されるルール、プロファイル、および `sysidcfg` ファイルが有効であることが前提となっています。
2. `rules`、プロファイル、および `syscfg` 構成ファイルの JumpStart 構成ディレクトリを、`install/installadm` パッケージがインストールされている Oracle Solaris 11 システムにコピーします。
3. ステップ 2 で Oracle Solaris 11 システムにコピーした JumpStart 構成ディレクトリで、`js2ai` 変換ツールを実行します。

   ```
   # js2ai -rS
   ```

   このコマンドは、`rules` ファイルおよび `rules` ファイルによって参照されるプロファイルで変換操作を実行します。`rules` ファイルで参照される各プロファイルは、AI クライアントプロビジョニングマニフェスト (`/usr/share/auto_install/manifest/default.xml`) に対して処理されます。このステップでは、JumpStart `rules` ファイルで指定されたプロファイルごとに、`AI_`*profile_name* という名前のディレクトリを作成します。`AI_`*profile_name* ディレクトリには、変換されたプロファイルごとに 1 つ以上の AI マニフェストが *profile_name*`${arch}.xml` の形式で含まれています。詳細は、「ファイル」のセクションを参照してください。

   `-S` オプションは検証処理をスキップします。検証はステップ 5 で実行されます。
4. 「Successfully completed conversion」というメッセージが出力された場合は、ステップ 5 にスキップします。それ以外の場合は、`js2ai.log` ファイルを検査して、次のステップに従います。
   a. 処理エラーを修正します。
   b. `rules` ファイルおよびプロファイルファイルから、サポート外項目として一覧表示されたすべての行を削除します。
   c. 変換エラーを検査して、可能な場合はエラーを修正します。それ以外の場合は、エラーの原因となっている行を削除します。
   d. 警告メッセージを検査して、修正が必要ないことを確認します。

e. 処理中のエラー、サポート外項目、および変換エラーが報告されなくなるまで、ステップ 3 を繰り返します。

5. -S オプションを指定せずに js2ai を再実行します。

```
# js2ai -r
```

処理されたプロファイルのいずれかで検証エラーが発生した場合は、生成された AI マニフェストを手動で修正する必要があります。失敗の詳細について、js2ai.log ファイルを検査します。AI マニフェストについての詳細は、AI のドキュメントを参照してください。

6. この JumpStart 構成に関連付けられた sysidcfg ファイルを変換します。

sysidcfg ファイルごとに、次のコマンドを実行します。

```
# js2ai -sS -d sysidcfg_dir
```

このステップは、処理された sysidcfg ファイルごとに、sc_profile.xml という名前の AI システム構成プロファイルファイルを、js2ai コマンドが起動されたディレクトリに作成します。sc_profile.xml ファイルに別のディレクトリを指定するには、-D オプションを使用します。

7. 「変換は正常に完了しました」というメッセージが出力された場合は、ステップ 8 にスキップします。それ以外の場合は、js2ai.log ファイルを検査して、次のステップに従います。

a. 処理エラーを修正します。

b. sysidcfg ファイルから、サポート外項目として一覧表示されたすべての行を削除します。

c. 変換エラーを検査して、可能な場合はエラーを修正します。それ以外の場合は、エラーの原因となっている行を削除します。

d. 警告メッセージを検査して、修正が必要ないことを確認します。

e. 処理中のエラー、サポート外項目、および変換エラーが報告されなくなるまで、ステップ 6 を繰り返します。

8. -S オプションを指定せずに js2ai を再実行します。

```
# js2ai -s -d sysidcfg_dir
```

処理された sysidcfg ファイルのいずれかで検証エラーが発生した場合は、結果となる AI システム構成プロファイルを手動で修正する必要があります。失敗の詳細について、js2ai.log ファイルを検査します。システム構成プロファイルについては、AI のドキュメントを参照してください。

9. js2ai 変換プロセスが完了しました。結果となる条件、AI マニフェスト、およびシステム構成プロファイルファイルの手動検証を実行します。Oracle Solaris 11 インストールのディスク容量の要件は、Oracle Solaris 10 インストールで必要なディスク容量とは異なります。AI マニフェスト内で割り当てられたディスク容量が Oracle Solaris 11 の要件を満たしていることを確認します。

10. 新しく生成されたファイルを使用するように AI を構成します。新しく生成された条件、AI マニフェスト、およびシステム構成プロファイルファイルを既存の AI インストールサービスに追加します。

    マニフェストを選択するための条件付きで各 AI マニフェストを追加するには、`create-manifest` サブコマンドを指定した `installadm` コマンドを使用します。各クライアントは、1 つの AI マニフェストのみを使用できます。

    ```
    # installadm create-manifest -n ai_service_name \
    -f manifest_file -m manifest_name \
    -C criteria_file
    ```

    構成プロファイルを選択するための条件付きで各プロファイルを追加するには、`create-profile` サブコマンドを使用します。各クライアントは、1 つ以上のシステム構成プロファイルを使用できます。

    ```
    # installadm create-profile -n ai_service_name \
    -f profile_file -p profile_name \
    -C criteria_file
    ```

    AI インストールサービスの構成についての詳細は、AI のドキュメントおよび `installadm`(1M) のマニュアルページを参照してください。

使用例

例 1　JumpStart 構成を処理する

次のコマンドは、現在のディレクトリで JumpStart のルールおよびプロファイルを処理します。出力は、このディレクトリにも配置されます。

```
# js2ai -r
```

例 2　特定の JumpStart ディレクトリを処理する

次のコマンドは、指定されたディレクトリから JumpStart のルールおよびプロファイルを処理し、同じディレクトリに出力ファイルを配置します。

```
# js2ai -r -d /export/jumpstart
```

出力ファイルについての詳細は、例 4 および「ファイル」のセクションを参照してください。

例 3　特定の JumpStart ディレクトリおよび個別のインストール先ディレクトリでプロファイルを処理する

次のコマンドは、`/export/jumpstart` ディレクトリから JumpStart の `rules` ファイルおよびプロファイルファイルを処理し、`/export/output` に出力ファイルを配置します。

```
# js2ai -p profile1 -d /export/jumpstart -D /export/output
```

例 4　指定されたルールおよびそのプロファイルの入力例および生成された出力

ルール:

例 4 指定されたルールおよびそのプロファイルの入力例および生成された出力 (続き)

```
arch sparc && karch sun4u && \
    model 'SUNW,Serverblade1'  -   profile    -
```

プロファイル:

```
install_type    initial_install
pool mypool auto auto auto c1t0d0s0
```

変換コマンド:

```
# js2ai -r -d /jumpstart -D /tmp/output
```

出力ファイル:

```
/tmp/output/AI_profile/profile.x86.xml
/tmp/output/AI_profile/profile.sparc.xml
/tmp/output/AI_profile/criteria-1.xml
```

rules ファイルに CPU タイプが SPARC と指定されている場合でも、2 つのマニフェストファイル (SPARC と x86 で 1 つずつ) が作成されます。変換プロセス中は、ルールとプロファイルは相互に独立して処理されます。

例 5 生成されたファイルを AI インストールサービスに追加する

この例では、例 4 で生成されたファイルを使用して、マニフェストおよび条件を既存のサービスに追加します。

ファイル:

```
/tmp/output/AI_profile/profile.sparc.xml
/tmp/output/AI_profile/criteria-1.xml
```

installadm コマンド:

```
# installadm create-manifest -n svc-name \
-f /tmp/output/AI_profile/profile.sparc.xml \
-m sparc_profile \
-C /tmp/output/AI_profile/criteria-1.xml
```

例 6 sysidcfgファイルを処理する

次のコマンドは、現在のディレクトリで sysidcfg ファイルを処理し、同じディレクトリに結果となる SMF システム構成プロファイルを sc_profile.xml として出力します。

```
# js2ai -s
```

終了ステータス　次の終了ステータスが返されます。

`0`　すべてのファイルが正常に処理されました。

`>0`　エラーが発生した。

ファイル

output_directory`/AI_${`*profile_name*`}`
プロファイルに関連付けられた新しい AI 構文に変換されたすべての対応するファイルが含まれるディレクトリ。

output_directory`/AI_${`*profile_name*`}.${`*arch*`}.xml`
プロファイルを変換した結果として作成されるマニフェストファイル。 `${`*arch*`}` には、3 つの値 `sparc`、`x86`、または `generic` のいずれかを指定できます。`${`*profile_name*`}.generic.xml` 形式のマニフェストファイルを使用して、x86 と SPARC の両方のシステムをインストールできます。

output_directory`/AI_${`*profile_name*`}/criteria-`*rule_number*`.xml`
生成された `criteria-`*rule_number*`.xml` ファイルは `rules` ファイル内のルールに対応し、*rule_number* は `rules` ファイル内での場所に基づいたルール番号です。この条件ファイルは、`installadm` コマンドの `-C` オプションで使用できます。

複数のルールで同じプロファイルを指定できるため、各ディレクトリに複数の条件ファイルが存在できますが、 `${`*profile_name*`}` のインスタンスは `1` つのみです。各出力ディレクトリには、`${`*arch*`}.xml` ファイルが存在する必要があります。

注– `-p` オプションが使用される場合は、処理されるプロファイルに対して条件ファイルが生成されません。条件ファイルは、`-r` オプションを指定して使用された場合にのみ生成されます。

output_directory`/js2ai.err`
このファイルには、処理中に発生した予期しない状況のスタックトレースが含まれています。通常、このファイルは作成されません。

output_directory`/js2ai.log`
このファイルには、処理されたファイルおよび処理中に見つかったエラーのログが含まれています。

output_directory`/sc_profile.xml`
このファイルは、`-s` オプションを使用して `sysidcfg` ファイルを変換する際に生成される SMF システム構成プロファイルです。

属性　属性についての詳細は、マニュアルページの `attributes(5)` を参照してください。

属性タイプ	属性値
使用条件	`install/js2ai`
インタフェースの安定性	不確実

関連項目

installadm(1M), pkg(1)

『Oracle Solaris 10 JumpStart から Oracle Solaris 11 自動インストーラへの移行』

『Oracle Solaris 11 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」

参 照

ファイル形式

名前　ai_manifest – 自動インストールマニフェストファイル形式

形式　`/usr/share/install/ai.dtd.1`

機能説明　自動インストーラ (AI) は、カスタマイズ可能でハンズフリーの Oracle Solaris のインストールメカニズムを提供し、インストールパラメータの説明として XML ベースのファイル形式を使用します。このインストールパラメータファイルは AI マニフェストと呼ばれます。インストールは、ディスクの配置やシステムにインストールされるソフトウェアなど、さまざまな方法でカスタマイズできます。

AI マニフェストには、次のセクションがあります。

- 自動インストール設定。インストール時に使用される設定を指定します。
- ディスクの配置。インストールのディスクの配置を指定します。
- ソフトウェア。インストールするソフトウェアパッケージを指定します。
- ブート構成 (x86 のみ)。GRUB ブートメニューの構成方法を指定します。
- その他の構成。システムにインストールするその他の構成コンポーネントを指定します。

これらのセクションについて、次により詳しく説明します。

新しい AI マニフェストを作成するには、テンプレートまたは関連インストールサービスイメージからのデフォルトのマニフェストのコピーを使用します。たとえば、インストールサービスイメージが *imagepath* にある場合、次のファイルを使用できます。

imagepath`/auto_install/manifest/default.xml`
　このインストールサービスの元のデフォルトの AI マニフェスト。

imagepath`/auto_install/manifest/ai_manifest.xml`
　カスタマイズの例を含む注釈付きのサンプル AI マニフェスト。

`installadm export` コマンドを使用して、インストールサービスにすでに存在するマニフェストのコピーを取得できます。

AI マニフェストは、`zoneadm install` コマンドを使用した非大域ゾーンのインストールにも使用されます。このコマンドに AI マニフェストファイルを渡して、ゾーンのインストールをカスタマイズできます。非大域ゾーンのインストールには、AI マニフェストの指定のサブセットのみが適用されます。これらの指定について、次のセクションで説明します。

AI マニフェストを補足して、Service Management Facility (SMF) 構成プロファイルがあります。これらのプロファイルは、ホスト名、ネットワーク、ルートおよび初期ユーザーアカウント設定などのインストールされるシステムのシステム構成を指定します。

インストールサービス、AI マニフェスト、および構成プロファイルの詳細については、installadm(1M) マニュアルページおよび『Oracle Solaris 11 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。構成プロファイルのファイル形式については、smf(5)を参照してください。

自動インストール設定

ai_instance 要素には次の属性があります。

name　このマニフェストインスタンスの名前。

http_proxy　インストール時にリモートファイルにアクセスするために使う HTTP プロキシ。インストール時にアクセスされるリモートファイルの例には、Image Packaging System (IPS) パッケージリポジトリ内のソフトウェアパッケージがあります。http_proxy の値は、http://myproxy.mycompany.com:8080/ などの HTTP URI です。

この属性は、非大域ゾーンのインストール時に適用できず、指定した場合には無視されます。

auto_reboot　インストール後に自動的にリブートするかどうかを指定するフラグ。auto_reboot のデフォルト値は false です。auto_reboot が false の場合、インストールは、手動のリブートを待機します。

auto_reboot が true の場合、インストールの成功時に、新しくインストールされたブート環境でマシンが自動的にリブートします。

この属性は、非大域ゾーンのインストール時に適用できず、指定した場合には無視されます。

次の例に ai_instance 要素の使用方法を示します。

```
<auto_install>
    <ai_instance name='default' auto_reboot='true'
      http_proxy='http://myproxy.mycompany.com:8080/'>
    <!-- target and software sections -->
    </ai_instance>
</auto_install>
```

ディスクの配置

AI では、インストール先の完全な自動選択から、ディスクの配置の詳細な制御まで、幅広くディスクを指定できます。

target 要素はディスクの配置を指定します。target 要素を指定していない場合のデフォルトのディスクの配置は、次の特性があります。

- 1 つのディスク全体が、Oracle Solaris OS のインストールに使われます。このディスクは通常ブートディスクまたは最初のディスクです。
- x86 の場合、ディスクのすべての内容を使用する fdisk パーティションが割り当てられます。fdisk パーティションの詳細については、fdisk(1M) マニュアルページを参照してください。
- SPARC のディスクのフルサイズであり、x86 の fdisk パーティションのフルサイズである単一スライス 0 が割り当てられます。
- 完全なスライス 0 を使用する単一のルートプールが作成されます。
- 領域を使用できる場合、スワップボリュームとダンプボリュームがルートプールに作成されます。

target 要素は次の構造になります。

```
<!-- zero or one target element -->
<target>
    <!-- zero or more disk elements -->
    <disk ...>
    </disk>
    <logical ...>
        <!-- zero or more zpool elements -->
        <zpool ...>
        </zpool>
    </logical>
</target>
```

target 要素の子要素を使用して、ディスクと論理配置を指定できます。

ディスクの指定は、非大域ゾーンのインストール時に適用できず、指定した場合には無視されます。

一部のディスクの配置要素には、size サブ要素があります。size 要素は次の形式になります。

```
<size val="size" start_sector="start_sector"/>
```

start_sector 値は、新しいパーティションやスライスの目的の開始セクターを指定する数値です。*start_sector* 属性が省略されている場合、インストーラは指定された *size* を格納するために十分な大きさの最初の場所を検索します。

size の値は、次のいずれかの接尾辞の付いた数値です。

- s または sec: セクター
- b: バイト
- k または kb: キロバイト (2^{10})

- m または mb: メガバイト (2^{20})
- g または gb: ギガバイト (2^{30})
- t または tb: テラバイト (2^{40})
- p または pb: ペタバイト (2^{50})
- e または eb: エクサバイト (2^{60})
- z または zb: ゼタバイト (2^{70})

このセクションの残りでは、disk 要素と logical 要素について詳しく説明します。

インストールの場所

Oracle Solaris OS をインストールするクライアント上の場所を指定しない場合、AI はそのクライアントのデフォルトの場所を選択します。

インストールのデフォルトの場所は、各クライアント上で見つかった、サイズの要件を満たす最初のディスクです。ディスクのサイズが必要なサイズ以上である場合、インストーラは、そのディスクをインストールの場所として選択します。ディスクのサイズが必要なサイズ未満である場合、インストーラは、次のディスクをチェックします。サイズの要件を満たすディスクが見つからない場合、そのクライアントに対する自動インストールは失敗します。/system/volatile/install_log にあるインストールログに、そのシステムのディスクの選択プロセスの詳細が示されます。

target セクションの disk セクションはインストールの場所を指定します。

ディスクの指定は、非大域ゾーンのインストール時に適用できず、指定した場合には無視されます。

ディスクは、次のいずれかの選択条件のタイプを使用して選択できます。

- グループ 1: ディスク名または IP アドレスなどの決定的条件。次の「ターゲットデバイス名」で説明するように <disk_name> サブ要素を使用するか、次の「ISCSI ターゲットデバイス」で説明するように <iscsi> サブ要素を使用します。
- グループ 2: ディスクサイズやベンダーなどの非決定的条件。次の「ターゲットデバイスのプロパティー」で説明するように <disk_prop> サブ要素を使用します。
- グループ 3: boot_disk キーワードなどのキーワード条件。次の「ターゲットデバイスのキーワード」で説明するように <disk_keyword> サブ要素を使用します。

これら 3 つのグループのうち 1 つのグループからの条件だけを指定できます。グループ 2 選択条件を使用する場合、複数の条件を指定できます。たとえば、サイズとベンダーの両方を指定できます。グループ 1 選択条件を使用する場合、それらの条件のうち 1 つだけを指定できます。

ターゲットデバイス名

disk_name 要素を使用して、iSCSI デバイスでないデバイスのターゲットデバイス名を指定します。disk_name 要素には次の属性があります。

name　name 属性はターゲットデバイスの名前を指定します。

name_type　name_type 属性はターゲットデバイス名のタイプを指定します。name_type 属性は次のいずれかの値を持ちます。

ctd: コントローラターゲットディスク名
これは c0t0d0 などの CTD 名か、c0t2000002037CD9F72d0 などの MPXIO 名です。この名前のタイプは、一般に format(1M) コマンドを実行する場合に見られます。

```
<disk_name name="c0t0d0" name_type="ctd"/>
```

これは、name_type 属性が省略された場合のデフォルトのターゲットデバイス名のタイプです。

volid: ボリューム識別子
これは、format (1M) コマンドによって設定可能なボリューム識別子です。

```
<disk_name name="MY_BOOT_DISK" name_type="volid"/>
```

devpath: デバイスパス
これは、/devices ディレクトリに相対的なデバイスパスです。

```
<disk_name
  name="/devices/pci@0,0/pci10de,375@f/pci108e,286@0/disk@0,0"
  name_type="devpath"/>
```

devid: デバイス識別子
これは、-iEn オプションを使用した iostat(1M) コマンドからの出力の「Device Id」に見つかるデバイス識別子です。

```
<disk_name
  name="id1,sd@TSun_____STK_RAID_INT____F0F0F0"
  name_type="devid"/>
```

receptacle: 受容体識別子
これは、-o cR オプションを使用した croinfo(1M) コマンドからの出力で見つかる CRO (シャーシ、受容体、占有装置) 構成からの受容体値です。

```
<disk_name name="SYS/1" name_type="receptacle"/>
```

iSCSI ターゲットデバイス

インストールターゲットとして iSCSI ディスクを指定するには、iscsi 要素を使用します。iscsi 要素には次の属性があります。

source　source 属性は、iSCSI 構成データのソースを指定します。source 属性は、次のいずれかの値を持ちます。

manifest
この値は、この AI マニフェストを示します。これは、source 属性に値が指定されていない場合のデフォルトです。

source 属性が省略されているか、source 属性の値が manifest である場合、target_lun 属性と target_ip 属性が指定されている必要があります。

dhcp
この値は、DHCP rootpath パラメータに情報を指定して、iSCSI 情報が提供される DHCP の使用を示します。

source 属性の値が dhcp の場合、ほかの iscsi 属性を指定しないでください。

```
<iscsi source="dhcp"/>
```

target_name　target_name 属性は、次の例に示すように、iSCSI ターゲットの IQN (iSCSI Qualified Name) または EUI (Extended Unique Identifier) を指定します。

```
iqn.1986-03.com.sun:02:a4a694bc-6de2-ee50-8979-e25ba29acb86
```

target_name 属性が指定されていない場合、AI は iscsiadm(1M) を sendtargets モードで使用します。

```
<iscsi target_lun="0" target_ip="192.168.1.34"/>
```

target_name 属性が指定されていない場合、AI は静的検出を使用します。

```
<iscsi target_name="iqn.1986-03.com.sun:02:a4a694bc-6de2-ee50-8979-e25ba29acb86"
    target_lun="0" target_ip="192.168.1.34"/>
```

target_lun　iSCSI ターゲットで複数の LUN を提供している場合、target_lun に整数値を指定して、使用する LUN を指定します。LUN 番号は 0 からインデックス付けされます。最初の LUN を指定するには、0 の target_lun 値を指定します。

提供されている LUN が 1 つだけの場合は、この属性を省略できます。

target_port　指定しない場合、デフォルトの 3260 (iSCSI 標準ポート) の target_port が使用されます。この属性を使用して、代替のポート番号を指定できます。

target_ip　この属性の値はサーバーの IP アドレスです。

```
<iscsi target_lun="0" target_ip="192.168.1.34"/>
```

ターゲットデバイスプロパティー

ターゲットデバイスのプロパティーを指定するには、disk_prop 要素を指定します。複数のプロパティーを指定できます。AI は指定された条件に基づいて、もっとも一致するものを見つけようとします。

disk_prop 要素の属性を使用して、ターゲットプロパティーを指定します。disk_prop 要素には次の属性があります。

dev_type: デバイスタイプ
: ターゲットディスクのタイプ。取り得る値には、SCSI、ATA、および USB があります。この値では、大文字と小文字は区別されません。

dev_vendor: デバイスベンダー
: format(1M) コマンドの inquiry メニューオプションによって表示されるベンダー。

```
<disk_prop dev_vendor="Sun"/>
```

dev_chassis: デバイスシャーシ
: -o cA オプションを使用した croinfo(1M) コマンドからの出力で見つかる CRO (シャーシ、受容体、占有装置) 構成からのシャーシ値。

```
<disk_prop dev_chassis="SYS"/>
```

dev_size: デバイスサイズ
: ディスクの最小サイズ。値は数字とサイズの単位です。

```
<disk_prop dev_size="100gb"/>
```

disk_prop 要素を使用して、ディスク検索をさらに制限するために、複数の属性を同時に指定できます。次の例では、ディスクの選択を 100G バイト以上のサイズの Hitachi ドライブに制限しています。

```
<disk_prop dev_vendor="HITACHI" dev_size="100gb"/>
```

ターゲットデバイスのキーワード

disk_keyword 要素を使用して、システムのブートディスクをターゲットディスクとして指定できます。

```
<disk_keyword key="boot_disk"/>
```

key 属性でサポートされている値は boot_disk のみです。

ディスク、パーティション、スライス全体

ディスクを配置するもっとも簡単な方法は、whole_disk 属性を true に設定して、インストールにディスク全体を使用することです。

もっと複雑なディスクの配置の場合は、パーティション (x86 システムの場合のみ) とスライスを指定できます。

disk 要素には次の属性があります。

whole_disk　この属性のデフォルト値は false です。whole_disk が false の場合、パーティションまたはスライスを定義する必要があります。既存のパーティションまたはスライスの action 属性に delete 値を指定して、それらを削除しないかぎり、パーティションまたはスライスが保持されます。

whole_disk が true の場合、既存のパーティションまたはスライスの action 属性に preserve 値を指定して、それらを保持しないかぎり、パーティションまたはスライスが削除されます。

次の例では、インストールにディスク全体を使用することを指定しています。

```
<disk whole_disk="true">
  <disk_name name="c0t0d0" name_type="ctd"/>
</disk>
```

in_zpool　in_zpool 属性は、AI マニフェストの logical セクションに定義されている ZFS プールにこのディスクをリンクします。in_zpool 属性の値は、対応する zpool 要素の name 属性の値に一致している必要があります。

ここで、in_zpool 属性を指定した場合、下位パーティションまたはスライスに in_zpool を指定しないでください。

in_vdev　in_vdev 属性は、AI マニフェストの logical セクションに定義されている仮想デバイスにこのディスクをリンクします。in_vdev 属性の値は対応する vdev 要素の name 属性の値に一致している必要があります。

ここで、in_vdev 属性を指定した場合、下位パーティションまたはスライスに in_vdev を指定しないでください。

パーティション

パーティションは x86 システムにインストールする場合にのみ指定できます。SPARC システムでパーティションを指定すると、インストールが失敗します。partition 要素には次の属性があります。

name　name 属性は fdisk パーティション番号です。値 1、2、3、および 4 はプライマリパーティションです。いずれかのプライマリパーティションが拡張パーティションの場合、値 5 から 32 を論理パーティションに指定できます。

指定した action が use_existing_solaris2 でないかぎり、name 属性が必要です。

action　action 属性は、次のいずれかの値を持ちます。

create
これはパーティションのデフォルトのアクションです。create アクションは、指定した名前でパーティションを作成するようにインストーラに指示します。同じ名前のパーティションがすでに存在する場合、その既存のパーティションが最初に削除されます。

delete
delete アクションは、名前付きのパーティションを削除するようにインストーラに指示します。名前付きのパーティションが存在しない場合、delete アクションはスキップされ、警告メッセージが出力されます。

preserve
preserve アクションは、名前付きパーティションを変更しないでおくようにインストーラに指示します。このアクションは、一般に、別のオペレーティングシステムが同じディスク上の別の場所にインストールされている場合に使われます。

use_existing_solaris2
use_existing_solaris2 アクションは、既存の Solaris2 パーティションを使うようにインストーラに指示します。インストーラは、既存の Solaris2 パーティションを検索します。

use_existing_solaris2 を指定した場合、name 属性と part_type 属性が無視されます。

part_type　part_type は fdisk パーティションタイプです。デフォルト値は 191 で、Solaris2 パーティションのパーティションタイプです。可能なパーティションタイプの詳細については、fdisk(1M) コマンドを参照してください。

in_zpool　in_zpool 属性は、AI マニフェストの logical セクションに定義されている ZFS プールにこのパーティションをリンクします。in_zpool 属性の値は、対応する zpool 要素の name 属性の値に一致している必要があります。

in_zpool 属性を指定する場合、関連 disk 要素または下位の slice 要素に in_zpool を指定しないでください。

in_vdev　in_vdev 属性は、AI マニフェストの logical セクションに定義されている仮想デバイスにこのパーティションをリンクします。in_vdev 属性の値は対応する vdev 要素の name 属性の値に一致している必要があります。

in_vdev 属性を指定する場合、関連 disk 要素または下位の slice 要素に in_vdev を指定しないでください。

パーティションには、パーティションのサイズを指定する size サブ要素を指定できます。size 要素の使用方法の詳細については、「ディスクの配置」の初めを参照してください。

次の例では、デフォルトの属性値を使用して、10G バイトの Solaris2 パーティションを作成しています。

```
<disk>
    <disk_name name="c0t0d0" name_type="ctd"/>
    <partition name="1">
        <size val="10gb"/>
    </partition>
</disk>
```

サイズを指定しない場合、親要素のサイズが使われます。

preserve、delete、および use_existing_solaris2 アクションでは size の指定は必要ありません。

スライス

x86 システムの場合、パーティション定義内に、スライスが含まれている必要があります。

slice 要素には次の属性があります。

name　name 属性はスライス番号です。値は 0 - 7 です。

action　action 属性は、次のいずれかの値を持ちます。

- create　これはスライスのデフォルトのアクションです。create アクションは、指定した名前でスライスを作成するようにインストーラに指示します。同じ名前のスライスがすでに存在する場合、その既存のスライスが最初に削除されます。
- delete　delete アクションは、名前付きのスライスを削除するようにインストーラに指示します。名前付きのスライスが存在しない場合、delete アクションはスキップされ、警告メッセージが出力されます。
- preserve　preserve アクションは、名前付きスライスを変更しないでおくようにインストーラに指示します。このアクションは、一般にデータが前のインストールから存在する場合に使用します。

is_swap　この属性のデフォルト値は false です。is_swap が false の場合、インストーラはルートプールにスワップボリュームを作成します。

is_swap が true の場合、名前付きスライスがスワップデバイスとして使われます。is_swap が true の場合、in_zpool 属性または in_vdev 属性を使用しないでください。

force　この属性のデフォルト値は false です。

force が true の場合、インストーラはすでに使用中の可能性があるスライス (たとえば、既存の ZFS ストレージプールで使われているスライスなど) を無視して、名前付きのスライスに指定されたアクションを実行し続けます。

in_zpool　in_zpool 属性は、AI マニフェストの logical セクションに定義されている ZFS プールにこのスライスをリンクします。in_zpool 属性の値は、対応する zpool 要素の name 属性の値に一致している必要があります。

in_zpool 属性を指定する場合、関連 partition 要素または disk 要素に in_zpool を指定しないでください。

in_vdev　in_vdev 属性は、AI マニフェストの logical セクションに定義されている仮想デバイスにこのスライスをリンクします。in_vdev 属性の値は対応する vdev 要素の name 属性の値に一致している必要があります。

in_vdev 属性を指定する場合、関連 partition 要素または disk 要素に in_vdev を指定しないでください。

スライスには、スライスのサイズを指定する size サブ要素を指定できます。size 要素の使用方法の詳細については、「ディスクの配置」の初めを参照してください。サイズを指定しない場合、親要素のサイズが使われます。

次の例では、SPARC システムに、デフォルトの属性値を使用した 20G バイトのスライスと、4G バイトのスワップスライスを作成しています。

```
<disk>
    <disk_name name="c0t0d0" name_type="ctd"/>
    <slice name="0">
        <size val="20gb"/>
    </slice>
    <slice name="1" is_swap="true">
        <size val="4gb"/>
    </slice>
</disk>
```

次の例は、x86 システムの例と同じです。

```
<disk>
    <disk_name name="c0t0d0" name_type="ctd"/>
    <partition name="1">
        <slice name="0">
```

```
            <size val="20gb"/>
        </slice>
        <slice name="1" is_swap="true">
            <size val="4gb"/>
        </slice>
    </partition>
</disk>
```

スワップとダンプ　スワップスライスは、上述の「スライス」に示すように、slice 要素の is_swap 属性を true に設定して明示的に定義できます。

プール内のボリュームは、上述の「ZFS ボリューム」に示すように、zvol 要素の use 属性を swap または dump に設定して、スワップボリュームまたはダンプボリュームとして明示的に定義できます。

デフォルトで、スワップボリュームとダンプボリュームは、領域が使用できれば自動的に作成されます。

メモリーが少ないシステムでは、ボリュームは少量のメモリーオーバーヘッドが発生するため、スワップスライスの方がスワップボリュームより好ましい場合があります。

スワップまたはダンプを明示的に指定し、スワップボリュームまたはダンプボリュームが自動的に作成されないようにする場合、logical 要素の次の属性を true に設定します。

noswap　この属性のデフォルト値は false です。noswap が false の場合、領域があれば、インストーラは自動的にルートプールにスワップボリュームを作成します。

noswap が true の場合、スワップボリュームは自動的に作成されません。

nodump　この属性のデフォルト値は false です。nodump が false の場合、領域があれば、インストーラは自動的にルートプールにダンプボリュームを作成します。

nodump が true の場合、ダンプボリュームは自動的に作成されません。

ZFS ストレージプール　target セクションの logical セクションを使用して、任意の数の ZFS ストレージプールを指定します。

logical 要素の zpool サブ要素を使用して、複数のプールを定義できます。これらのプールのうち 1 つだけをルートプールにできます。複数のルートプールを定義すると、インストールが失敗します。

zpool 要素で、AI マニフェストにルートプールを定義し、ターゲットディスク、パーティション、またはスライスを定義しない場合、上述の「インストールの

場所」で説明されているとおりに、インストーラがターゲットを選択します。この選択は自動的にルートプールに割り当てられます。

ターゲットディスク、パーティション、またはスライスを AI マニフェストに指定している場合、zpool をこれらのディスク、パーティション、またはディスクの少なくとも 1 つに関連付ける必要があります。この関連付けを作成するには、disk 要素、partition 要素、または slice 要素の in_zpool 属性を使用します。

zpool 要素には次の属性があります。

name　これは新しいプールの名前です。この値は、zpool create コマンドに渡すことができる名前にする必要があります。

この名前は、ディスク、パーティション、またはスライスを zpool の構成要素デバイスとして定義する場合に、disk、partition、または slice 要素の in_zpool 属性の値として使用されることがあります。

action　action 属性は、次のいずれかの値を持ちます。

create
これは、zpool のデフォルトのアクションです。create アクションは、指定した名前でプールを作成するようにインストーラに指示します。

delete
delete アクションは、名前付きのプールを削除するようにインストーラに指示します。

preserve
preserve アクションは、名前付きプールを変更しないでおくようにインストーラに指示します。このアクションは、非ルートプールにのみ指定できます。

action 属性の値は、次の場合に、preserve にする必要があります。

- 下位のいずれかの filesystem の action 属性の値が preserve である。
- 下位のいずれかの zvol の action 属性の値が preserve である。
- 下位のいずれかの zvol の action 属性の値が use_existing である。

use_existing
use_existing アクションは、既存のルートプールをインストールするようにインストーラに指示します。既存のボリュームまたはファイルシステム (データセット) が保持されます。

`is_root` この属性のデフォルト値は `false` です。`is_root` が `false` の場合、データプールが定義されます。

`is_root` が `true` の場合、名前付きプールに新しいブート環境が作成されます。

`mountpoint` `mountpoint` 属性は、プールの最上位ファイルシステムのマウントポイントを指定します。デフォルトのマウントポイントは */poolname* です。マウントポイントは絶対パスにする必要があります。

新しいプールの ZFS プロパティーを設定するには、`pool_options` 要素を使用します。同様に、自動作成された ZFS データセットの ZFS プロパティーを設定するには、`dataset_options` 要素を使用します。`pool_options` 要素と `dataset_options` 要素には、`option` サブ要素があります。各 `option` 要素には、`name` 属性と `value` 属性があります。これらの名前と値のペアで設定されたプロパティーは、`zpool`(1M) コマンドで適用されるものと同じ制限を受けます。次の例に、これらのプロパティーの設定方法を示します。

```
<logical>
    <zpool name="rpool" is_root="true">
        <pool_options>
            <option name="listsnaps" value="on"/>
            <option name="delegation" value="off"/>
        </pool_options>
        <dataset_options>
            <option name="atime" value="on"/>
            <option name="compression" value="on"/>
        </dataset_options>
    </zpool>
</logical>
```

プールには、任意の数の仮想デバイス冗長グループ (`vdev` 要素)、ZFS データセット (`filesystem` 要素)、または ZFS ボリューム (`zvol` 要素) を定義できます。プールにブート環境 (`be` 要素) を指定できます。次のセクションで、`vdev`、`filesystem`、`zvol`、および `be` 要素について説明します。

仮想デバイス冗長グループ

`zpool` のサイズまたは構造を定義するには、`vdev` 要素を使用します。それぞれ異なる冗長タイプの複数の `vdev` 要素を指定できます。

`zpool` に複数の `vdev` 要素が含まれている場合、`in_zpool` 属性で定義されている `disk`、`partition`、または `slice` 要素で、`in_vdev` 属性を使用する必要があります。

`vdev` 名が AI マニフェスト全体で一意の場合は、ディスク、パーティション、またはスライスの `in_zpool` 属性を省略できます。

zpool に含まれる vdev 要素が 1 つだけの場合、disk、partition、または slice の in_vdev 属性を省略できます。

vdev 要素には次の属性があります。

name　これは新しい vdev の名前です。

この名前は、ディスク、パーティション、またはスライスを vdev の構成要素デバイスとして定義する場合に、disk、partition、または slice 要素の in_vdev 属性の値として使用する必要があります。

redundancy　redundancy 属性は、次のいずれかの値を持ちます。

mirror
これがデフォルト値です。redundancy が mirror であるか、指定されていない場合、含まれるすべてのデバイスが相互にミラーであるとみなされます。

raidz、raidz1、raidz2、raidz3
これらのいずれかの値を持つグループ内のデバイスは、RAIDZ グループの定義に使われます。

spare
このグループ内のデバイスは、障害発生時のホットスペアとみなされます。

cache
このグループ内のデバイスは、プールのキャッシュを提供します。

log、logmirror
このグループ内のデバイスはログに使用されます。logmirror を指定した場合、デバイスはミラーです。

none
redundancy が none の場合、冗長性は定義されません。複数のデバイスがこのグループに含まれている場合、これらのデバイスはストライプ化されます。

ルートプールは、次のいずれかの構成としてのみ定義できます。

- デバイスが 1 つで redundancy タイプが none。この構成では、複数のデバイスがサポートされません。
- デバイスが複数で redundancy タイプが mirror。

vdev にデバイスを追加するには、disk、partition、または slice 要素の in_zpool 属性と in_vdev 属性を使用します。次の例では、2 つのディスクでミラー化される rpool というルートプールを指定します。

```
<disk whole_disk="true" in_zpool="rpool" in_vdev="mirrored">
    <disk_name name="c0t0d0" name_type="ctd"/>
</disk>
<disk whole_disk="true" in_zpool="rpool" in_vdev="mirrored">
    <disk_name name="c1t0d0" name_type="ctd"/>
</disk>
<logical>
    <zpool name="rpool" is_root="true">
        <vdev name="mirrored" redundancy="mirror"/>
    </zpool>
</logical>
```

参照しているプールや仮想デバイスが明確な場合、in_zpool 属性または in_vdev 属性のいずれかを省略できます。

ファイルシステム (データセット)

ZFS プール内に ZFS ファイルシステムまたはデータセットを定義するには、filesystem 要素を使用します。

filesystem 要素には次の属性があります。

name
: これは、zpool に対して相対的な、新しい filesystem の名前です。たとえば、rpool という名前の zpool 内の filesystem の名前が export である場合、ZFS データセット名は rpool/export になります。

  filesystem の in_be 属性が true の場合、この名前はブート環境のルートデータセットに相対的になります。

action
: action 属性は、次のいずれかの値を持ちます。

  create
  : これは、filesystem のデフォルトのアクションです。create アクションは、指定した名前でファイルシステムを作成するようにインストーラに指示します。

  delete
  : delete アクションは、名前付きのファイルシステムを削除するようにインストーラに指示します。

  preserve
  : preserve アクションは、名前付きファイルシステムを変更しないでおくようにインストーラに指示します。filesystem に preserve を指定する場合、関連付けられた zpool に preserve を指定する必要があります。

mountpoint
: mountpoint 属性は、新しいファイルシステムのマウントポイントを指定します。マウントポイントを指定しない場合、ファイルシステムはその親からマウントポイントを継承します。

in_be
: この属性のデフォルト値は false です。in_be が false の場合、新しいデータセットがすべてのブート環境で共有されます。

in_be が true の場合、各ブート環境内にこの新しいデータセットの個別のコピーが作成されます。in_be が true の場合、name 属性の値はブート環境のルートデータセットに相対的になります。

filesystem に ZFS データセットプロパティーを設定するには、options サブ要素を使用します。編集可能な任意の ZFS ファイルシステムプロパティーを設定できます。filesystem の options 要素の使い方は、下の例に示すように、zpool の dataset_options 要素の使い方に似ています。

```
<logical>
    <zpool name="rpool" is_root="true">
        <filesystem name="export">
            <options>
                <option name="compression" value="off"/>
                <option name="dedup" value="on"/>
            <options>
        </filesystem>
    </zpool>
</logical>
```

子 filesystem は親 filesystem に設定されているプロパティーが明らかに異なって設定されていないかぎり、そのプロパティーを継承します。これは ZFS ファイルシステムのデフォルトの動作です。

ZFS ボリューム

ZFS プール内に ZFS ボリュームを定義するには、zvol 要素を使用します。zvol は一般に、スワップデバイスまたはダンプデバイスに使用しますが、ほかの用途もあります。

zvol 要素には次の属性があります。

name　これは、新しい ZFS ボリュームの名前です。

action　action 属性は、次のいずれかの値を持ちます。

- create　これは、zvol のデフォルトのアクションです。create アクションは、指定した名前で ZFS ボリュームを作成するようにインストーラに指示します。
- delete　delete アクションは、名前付きのボリュームを削除するようにインストーラに指示します。
- preserve　preserve アクションは、名前付き zvol を変更しないでおくようにインストーラに指示します。zvol に preserve を指定する場合、関連付けられた zpool に preserve を指定する必要があります。

use_existing　この値をスワップデバイスまたはダンプデバイスに指定した場合、既存のボリュームが再利用されます。zvol に use_existing を指定する場合、関連付けられた zpool に preserve を指定する必要があります。

use　use 属性は、次のいずれかの値を持ちます。

none　これがデフォルト値です。use が none の場合、zvol が作成されますが、インストール時には使用されません。

swap　use が swap の場合、zvol が作成され、スワップデバイスとして使用されます。zvol はインストール時にもスワップデバイスとして使用されます。

dump　use が dump の場合、zvol が作成され、ダンプデバイスとして使用されます。zvol はインストール時にもダンプデバイスとして使用されます。

zvol のサイズを指定するには、size サブ要素を使用します。size 要素の使用方法の詳細については、「ディスクの配置」セクションの初めを参照してください。

zvol に ZFS ボリュームオプションを設定するには、options サブ要素を使用します。zvol の options 要素の使い方は、下の例に示すように、zpool の dataset_options 要素の使い方に似ています。

```
<logical>
    <zpool name="rpool" is_root="true">
        <zvol name="swap">
            <options>
                <option name="compression" value="off"/>
            <options>
        </zvol>
    </zpool>
</logical>
```

ブート環境

インストール時のブート環境の作成方法を指定するには、be 要素を使用します。

be 要素には、1 つの属性があります。

name　これは、インストーラによって作成される新しいブート環境の名前です。be 要素を指定しない場合、このブート環境のデフォルトの名前は、solaris になります。

インストーラはブート環境サブシステムによって提供されている自動名前付け機能を利用します。既存のターゲット領域にインストールする場合 (たとえば、ゾーンのインストール時)、be 要素の name 属性によって指定された名前のブート環境がすでに存在する可能性があります。指定したブート環境名がすでに存在する場合、こ

の名前は新しい名前を生成するためのベースとして使われます。たとえば、be を指定せず、solaris という名前のブート環境がすでに存在する場合、新しいブート環境には、solaris-*n* という名前が付けられます。ここで、*n* は、すでに存在していないブート環境名を形成するためのカウント順の先頭の整数です。

ブート環境は、ZFS データセットとして作成され、ZFS プロパティーを設定できます。ブート環境に ZFS プロパティーを設定するには、次の例に示すように、options サブ要素を使用します。

```
<logical>
    <zpool name="rpool" is_root="true">
        <be name="installed_be">
            <options>
                <option name="compression" value="on"/>
                <option name="dedup" value="on"/>
            <options>
        </be>
    </zpool>
</logical>
```

ソフトウェア

software 要素はインストールするソフトウェアを指定します。software セクションは、次の情報を指定します。

- ソフトウェアソースのタイプ
- ソースの場所
- インストールまたはアンインストールするソフトウェアパッケージの名前
- インストールするオプションのソフトウェアのコンポーネント
- イメージプロパティー
- IPS リポジトリにアクセスするために必要な SSL キーと証明書

software 要素には次の属性があります。

name　これは software インスタンスの名前です。この名前はこの AI マニフェストのすべての software インスタンスの中で一意である必要があります。

type　これは、ソフトウェアソースのタイプです。

type 属性は、次のいずれかの値を持ちます。type を指定しない場合のデフォルト値は IPS です。

- IPS: IPS パッケージリポジトリ
- P5I: IPS パッケージファイル
- SVR4: SVR4 パッケージ
- CPIO: cpio アーカイブ

software 要素には次の属性があります。

```
<!-- one or more software elements -->
<software>
```

```
    <!-- zero or one destination element
         This element is only used when type is IPS or P5I.
    -->
    <destination>
        <!-- image properties and
             optional software components
        -->
    </destination>
    <!-- one or more source elements
         IPS type: only one source element
    -->
    <source>
        <!-- one or more publisher or dir elements
             IPS, P5I, and SVR4 types:
                 one or more publisher/origin elements
             CPIO types: one or more dir elements
        -->
    </source>
    <!-- zero or more software_data elements
         At least one software_data element must have an
             action of install.
         P5I type: zero software_data elements
    -->
    <software_data>
        <!-- one or more name elements -->
    </software_data>
</software>
```

IPS インストール

type 属性を指定しない場合のデフォルトのインストールタイプは IPS です。

タイプ IPS のインストールの場合、単一の source 要素のみを指定できます。

source 要素を使用して、パッケージのインストールに使用するパブリッシャーを指定します。複数のパブリッシャーを指定できます。各パブリッシャーには少なくとも 1 つの起点が必要です。各パブリッシャーには複数の起点とミラーを指定できます。

AI マニフェストにパブリッシャーが定義されている順番で、インストールする IPS パッケージのパブリッシャーが検索され、インストールされたシステムにパブリッシャーが設定されます。

非大域ゾーンをインストールする場合、ゾーンによってシステムリポジトリが使用されます。AI マニフェストに指定されたパブリッシャーは、システムリポジトリによって提供されたパブリッシャーの後に、AI マニフェスト内の順番で追加されます。システムリポジトリの詳細については、pkg(1) および pkg.sysrepo(1m) を参照してください。

次の例では、複数のパブリッシャーを指定しており、そのうちの 1 つにミラーと起点があります。

```
<software type="IPS">
  <source>
    <publisher name="solaris">
      <origin name="http://pkg.oracle.com/solaris/release"/>
      <mirror name="http://localpkg.mycompany.com/solaris"/>
    </publisher>
    <publisher name="internal-software">
      <origin name="http://internalsoft.mycompany.com/"/>
    </publisher>
  </source>
</software>
```

software_data 要素を使用して、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指定します。action 属性は、次の 2 つのうちいずれかの値を持ちます。

install　　name サブ要素に指定されている IPS パッケージをインストールします。action 属性が指定されていない場合は、これがデフォルトになります。少なくとも 1 つの software_data 要素に、インストールのアクションが必要です。

uninstall　　name サブ要素に指定されている IPS パッケージを削除します。

action 属性のその他の値は、IPS インストールでサポートされていません。

次の例に示すように、これらのアクションごとに、name 要素に 1 つ以上のパッケージを指定できます。

```
<software_data> <!-- defaults to install action -->
    <name>pkg:/entire</name>
    <name>pkg:/group/system/solaris-large-server</name>
</software_data>
<software_data action="uninstall">
    <name>pkg:/unwanted/pkg</name>
</software_data>
```

P5I インストール　　.p5i ファイルは、IPS パブリッシャー、パッケージ、ミラーなどを記述するファイルです。

1 つ以上の .p5i ファイルを処理するように指定するには、次の例に示すように、publisher 要素に起点としてファイルを指定します。

```
<software type="P5I">
    <source>
        <publisher>
            <origin name="/somewhere/image1.p5i"/>
```

```
            <origin name="/somewhere/image2.p5i"/>
        </publisher>
    </source>
</software>
```

この AI マニフェストに、IPS タイプソフトウェアセクションもない場合、`.p5i` ファイルに起点が指定されていることを確認してください。

インストールするパッケージの指定は、P5I インストールでサポートされていません。そのため、`software_data` 要素はタイプ `P5I` の `software` 要素でサポートされません。

SVR4 インストール

SVR4 転送の場合、SVR4 パッケージサブディレクトリまたは SVR4 パッケージデータストリームファイルを格納するディレクトリを、ファイルディレクトリパスまたは FILE URI を使用して指定する必要があります。SVR4 パッケージデータストリームファイルは HTTP URI を使用して指定することもできます。

```
<software type="SVR4">
    <source>
        <publisher>
            <origin name="/somedir"/>
        </publisher>
    </source>
</software>
```

`software_data` 要素は、実行されるアクションを指定するために使用します。`action` 属性は、次の 2 つのうちいずれかの値を持ちます。

`install` ソースからファイルを新しいブート環境にコピーします。`action` 属性が指定されていない場合は、これがデフォルトになります。少なくとも 1 つの `software_data` 要素に、インストールのアクションが必要です。

`uninstall` 新しいブート環境からファイルを削除します。

`action` 属性のその他の値は、SVR4 インストールではサポートされていません。

次の例に示すように、これらのアクションごとに、`name` 要素に 1 つ以上のパッケージを指定できます。

```
<software type="SVR4">
    <source>
        <publisher>
            <origin name="/somedir"/>
        </publisher>
    </source>
    <software_data> <!-- defaults to install action -->
        <name>ORGpackage1</name>
        <name>ORGpackage2</name>
```

```
    </software_data>
    <software_data action="uninstall">
        <name>ORGpackage8</name>
    </software_data>
</software>
```

CPIO インストール

CPIO 転送の場合、ソースディレクトリを指定する必要があります。転送先ディレクトリは、インストール時の新しいブート環境のマウントポイントに設定されます。

```
<software type="CPIO">
    <source>
        <dir path="/somedir"/>
    </source>
</software>
```

software_data 要素は、実行されるアクションを指定するために使用します。action 属性は、次のいずれかの値を持ちます。

install

ソースからファイルを新しいブート環境にコピーします。action 属性が指定されていない場合は、これがデフォルトになります。少なくとも 1 つの software_data 要素に、インストールのアクションが必要です。

name 要素を使用して、コピー元のファイルまたはディレクトリを指定します。name 要素に指定されるパスは、ソースに相対的になります。

```
<software_data>
  <!-- defaults to install action -->
  <name>path/relative/to/source</name>
  <name>another/path/relative/to/source</name>
</software_data>
```

uninstall

新しいブート環境からファイルを削除します。

name 要素を使用して、削除されるファイルまたはディレクトリを指定します。name 要素に指定されるパスは、宛先に相対的になります。

```
<software_data action="uninstall">
    <name>path/relative/to/destination</name>
</software_data>
```

オプションのソフトウェアコンポーネントとイメージプロパティー

destination 要素と image サブ要素を使用して、次の情報を指定します。

- インストールするオプションのソフトウェアのコンポーネント
- イメージプロパティー
- SSL キーおよび証明書

destination セクションは、IPS および P5I インストールタイプにのみ適用されます。destination 要素には image サブ要素を 1 つだけ指定できます。

SSL キーおよび証明書

image 要素の属性を使用して、クライアント SSL 認証を使用するパブリッシャーに必要な SSL キーおよび証明書を指定します。ここに指定したキーと証明書は、AI マニフェストに最初に定義されているパブリッシャーに適用されます。

ssl_key　この属性は、次の pkg コマンドにマップします。

```
pkg set-publisher -k ssl_key
```

ssl_key 属性の値は *ssl_key* です。pkg set-publisher コマンドの詳細については、pkg(1) マニュアルページを参照してください。

ssl_cert　この属性は、次の pkg コマンドにマップします。

```
pkg set-publisher -c ssl_cert
```

ssl_cert 属性の値は *ssl_cert* です。

オプションのソフトウェアコンポーネント

インストールするオプションのソフトウェアコンポーネントを指定するには、image 要素の facet サブ要素を使用します。ファセットは、個別のソフトウェアパッケージではありませんが、ロケール、ドキュメント、デバッグ情報を含むファイルなどの開発ファイルなど、特定のソフトウェアパッケージのオプションのコンポーネントです。たとえば、1 つか 2 つの言語のみをインストールするように指定することで、領域を節約できます。IPS ファセットの詳細については、pkg(1) マニュアルページを参照してください。

facet 要素には、ブール型の set 属性と IPS ファセットの名前の値があります。

```
<facet set="true|false">facet_name</facet>
```

次の例では、パッケージのドイツ語と英語のファセットのみをインストールするように指定しています。例では、まずロケールをインストールしないように指定し、次に、ドイツ語と英語のロケールをインストールするように指定しています。

```
<destination>
    <image>
        <!-- de-select all locales -->
        <facet set="false">facet.locale.*</facet>
        <!-- specify specific locales to install -->
        <!-- install German and English only -->
        <facet set="true">facet.locale.de</facet>
        <facet set="true">facet.locale.de_DE</facet>
        <facet set="true">facet.locale.en</facet>
        <facet set="true">facet.locale.en_US</facet>
```

```
    </image>
</destination>
```

イメージプロパティー

このインストールで作成する新しいイメージの IPS イメージプロパティーを指定するには、`image` 要素の `property` サブ要素を使用します。

`property` 要素には、ブール型の `val` 属性とプロパティーの名前の値があります。

```
<property val="true|false">property_name</property>
```

設定できるプロパティーについては、pkg(1) マニュアルページの「イメージプロパティー」セクションを参照してください。

ブート構成 (x86 のみ)

AI マニフェストを使用して、インストール先のシステムの GRUB ブートメニューの構成方法を変更できます。

このセクションは、ゾーンのインストールには適用されず、非大域ゾーンのインストール時に無視されます。

GRUB ブートメニューを変更するには、`boot_mods` 要素と `boot_entry` サブ要素を使用します。

`boot_mods` 要素には次の属性があります。

`title` `title` 属性の値は、この `boot_mods` 要素の `boot_entry` サブ要素で指定されたブートエントリのベースタイトルです。この属性値は、`/etc/release` の先頭行またはインストールメディアから自動的に生成される名前を上書きします。

`timeout` `timeout` 属性の値は、この `boot_mods` 要素のデフォルトの `boot_entry` が選択されるまで待つ秒数です。

SPARC システムでは、`title` 属性のみを設定できます。SPARC システムでは、このセクションのその他のすべての設定が無視されます。

ブートメニューに 1 つ以上のメニュー項目を追加するには、`boot_entry` サブ要素を使用します。これらのメニュー項目は、インストーラによって自動的に生成されるメニュー項目に追加されます。

`boot_entry` 要素には次の属性があります。

`default_entry` ブール値が `true` に設定されている場合、このメニュー項目がブート時にデフォルトで選択されるオプションになります。この属性のデフォルト値は `false` です。

複数の boot_entry 要素の default_entry が true に設定されている場合、それらの最後のエントリがブート時にデフォルトで選択されるオプションになります。

insert_at　この属性には、次の 2 つのうちいずれかの値を設定できます。

- end　生成されるブートメニューの末尾にエントリを配置します。これがデフォルトの配置です。
- start　生成されるブートメニューの先頭にエントリを配置します。

boot_entry メニュー項目は、次のサブ要素によって定義されます。

title_suffix　この要素は必須です。この要素は、boot_mods 要素に指定されたタイトルの末尾に追加されるテキストを定義します。

kernel_args　この要素は省略可能です。この要素は、ブートローダーによってカーネルに渡される一連の値です。

次の例では、メニューの最後のエントリで、20 秒後に自動的に選択される「Boot Testing Default Boot Entry」というブートメニューエントリを指定しています。

```
<boot_mods title="Boot Testing" timeout="20">
    <boot_entry default_entry="true">
        <title_suffix>Default Boot Entry</title_suffix>
    </boot_entry>
</boot_mods>
```

その他の構成

configuration 要素は、非大域ゾーンの構成をサポートします。大域ゾーンシステムをインストールする場合、大域ゾーンのインストール後に、AI マニフェストに指定されたゾーン構成を使用して、非大域ゾーンがシステムにインストールされます。

configuration 要素には次の属性があります。

type　インストールする構成のタイプ。AI によってサポートされるタイプは zone のみです。

name　構成に指定される名前。この名前は、AI マニフェストのすべての構成要素で一意である必要があります。タイプ zone の構成の場合、この名前は、ゾーンの zonename としても使われます。

source　AI がこの構成要素の構成ファイルをダウンロードする場所。値は HTTP または FILE URI の指定です。タイプ zone の構成の場合、この値は、zonecfg export コマンドから生成されるゾーン構成ファイルを示している必要があります。

次の指定では、zone1 をインストールクライアントにインストールします。

```
<configuration type="zone" name="zone1"
    source="http://myserver.com/configs/zone1/config"/>
```

ゾーンの構成とインストールの詳細については、『Oracle Solaris 11 システムのインストール』の第 12 章「ゾーンのインストールと構成」を参照してください。

ファイル

`/usr/share/auto_install/manifest/default.xml`
: カスタマイズなしのデフォルトのシステムインストールの指定。この AI マニフェストは、参考用にのみシステムに提供されています。新しい AI マニフェストを作成するには、関連インストールサービスイメージからこのファイルのコピーを使用します。インストールサービスからこのファイルをコピーする方法については、「説明」セクションを参照してください。

`/usr/share/auto_install/manifest/zone_default.xml`
: カスタマイズなしのデフォルトのゾーンインストール。このファイルは、`zoneadm install` コマンドによって、非大域ゾーンをインストールするためのデフォルトのマニフェストとして使われます。

`/usr/share/auto_install/manifest/ai_manifest.xml`
: 詳細がコメントアウトされているテンプレート AI マニフェスト。このファイルは、実行可能ないくつかのカスタマイズの例を提供しています。このファイルは、参考用にのみシステムに提供されています。新しい AI マニフェストを作成するには、関連インストールサービスイメージからこのファイルのコピーを使用します。インストールサービスからこのファイルをコピーする方法については、「説明」セクションを参照してください。

属性

属性についての詳細は、マニュアルページの `attributes(5)` を参照してください。

属性タイプ	属性値
使用条件	`system/install/auto-install/auto-install-common`
インタフェースの安定性	不確実

関連項目

`installadm(1M)`, `beadm(1M)`, `pkg(1)`, `grub(5)`, `prtconf(1M)`, `format(1M)`, `zfs(1M)`, `zpool(1M)`, `pkg.sysrepo(1m)`, `smf(5)`, `zoneadm(1M)`, `zonecfg(1M)`

『Oracle Solaris 11 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」

名前

dc_manifest – ディストリビューションコンストラクタのマニフェストファイルのカスタマイズ

形式

次のマニフェストファイルを使用して、さまざまな Oracle Solaris イメージを構築できます。これらのマニフェストは distribution-constructor パッケージに含まれます。

x86 Oracle Solaris LiveCD イメージを構築するには:
/usr/share/distro_const/dc_livecd.xml

x86 自動インストールイメージを構築するには:
/usr/share/distro_const/dc_ai_x86.xml

SPARC 自動インストールイメージを構築するには:
/usr/share/distro_const/dc_ai_sparc.xml

x86 テキストインストールイメージを構築するには:
/usr/share/distro_const/dc_text_x86.xml

SPARC テキストインストールイメージを構築するには:
/usr/share/distro_const/dc_text_sparc.xml

機能説明

ディストリビューションコンストラクタ (DC) を使用して、Oracle Solaris インストールイメージを構築できます。

DC XML マニフェストファイルは、ディストリビューションコンストラクタへの入力として使用されます。これらのマニフェストは、ディストリビューションコンストラクタが構築するイメージを定義します。上記のリストごとに、異なるマニフェストを使用してさまざまな種類のイメージを構築できます。

イメージを構築するには、distro_const コマンドを使用します。これにより、コマンド内のマニフェストファイルが参照されます。

イメージの仕様をカスタマイズする場合は、マニフェストファイルをコピーし、そのコピーをカスタマイズし、イメージを構築するときにそのコピーを distro_const コマンドの入力として使用します。

少なくとも、マニフェスト内でターゲット要素を編集し、イメージを作成できる構築領域の場所を指定する必要があります。また、ソフトウェア名要素を編集し、イメージの構築に必要なパッケージを含むパブリッシャーとリポジトリの場所を指定する必要があります。

マニフェストセクション

マニフェストには、次のプライマリ要素が含まれます。

注 – 次に示されているデフォルトの要素と属性は、使用するマニフェストによって異なります。

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

この要素は、構築する予定のイメージに対してデフォルトの名前 Oracle_Solaris_Text_X86 を指定します。この名前を使用するか、またはイメージに対して一意の名前を指定できます。

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配布名要素のコメントを解除して、プロキシの場所を入力します。たとえば、次のように指定します。

```
<distro name="Oracle_Solaris_Text_SPARC" add_timestamp="false"
http_proxy="http://example.com">
```

<boot_mods>

この要素は、イメージに適用されるブートメニュー変更を指定します。

次の例では、「myentry」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="myentry" timeout="5">
```

新しいエントリごとに boot_entry 要素を追加することで、ブートメニューエントリを個別に追加できます。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 – 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>
```

この例ではタイトルサブ要素が含まれていないため、デフォルトが使用されます。デフォルトのタイトルは、`/etc/release` の先頭行です。

title_suffix は必須のサブ要素で、エントリのタイトルに追加されるテキスト文字列です。オプションの kernel_args サブ要素は、カーネル引数をブートローダーに渡します。

boot_entry 要素のオプションの属性には次のものがあります。

default_entry　この属性を「true」に設定すると、このブートエントリがデフォルトになります。複数のエントリが「true」に設定されている場合、最後のエントリとして定義されたエントリによってそれ以前のエントリは無効にされます。

insert_at　値を「start」または「end」に設定して、その他のブートエントリに対する挿入ポイントを示します。

<target>

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入力する必要があります。

次の例を参照してください。

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

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

このセクションでは、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場所を指定します。

Image Packaging System (IPS) のパブリッシャーは、1 つまたは複数のパッケージリポジトリでパッケージを提供します。

このセクションのソース要素で、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。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。

次の例を参照してください。

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

```
    <origin name="http://example3.com/dev"></origin>
  </publisher>
</source>
```

注 – この要素には、イメージの構築中に使用されるデータマウントポイントを指定する宛先タグも含まれます。宛先属性の変更は推奨されません。

<software_data action="install">

インストール属性を持つこの 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 CD 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" type="IPS">
  <name>pkg:/entire</name>
  <name>pkg:/server_install</name>
  <name>pkg:/system/install/text-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「entire」参照にバージョン番号を追加します。

```
<name>pkg:/entire@0.5.11-0.build#</name>
```

利用できるバージョンを確認するには、次のコマンドを使用します。

```
# pkg list -af entire
```

注 –「entire」エントリは削除しないでください。「entire」は、複数のパッケージを管理するために使用される incorporation です。

<software_data action="uninstall" type="IPS">

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。

次の例で、「server_install」は、多数の個別パッケージを含むグループパッケージの名前です。

```
<software_data action="uninstall" type="IPS">
  <name>pkg:/server_install</name>
</software_data>
```

グループパッケージはアンインストールできます。グループパッケージをアンインストールしても、実際にアンインストールされるのはグループ定義だけです。そのグループの一部として以前にインストールされた個々のパッケージはアンインストールされません。ただし、グループパッケージをアンインストールせずに、それらの個々のパッケージをアンインストールできます。グループパッケージを保持すると、参照を継続する場合に役立つ可能性があります。

また、名前タグを使用して個々のパッケージをアンインストールすることもできます。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。

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

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るまで、この要素がそのシステムに影響を与えることはありません。

ソース要素で、パブリッシャー名とオプションのミラー名のタグを使用して、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します。次の例を参照してください。

```
<source>
  <publisher name="solaris">
    <origin name="http://pkg.oracle.com/solaris/release/"/>
  </publisher>
</source>
```

<software name="ba-init">

この要素は、構築されるイメージのブートアーカイブに、インストールまたはアンインストールされるファイルとディレクトリを一覧表示します。詳細は、マニフェストファイルのコメントを参照してください。

注意 – ブートアーカイブの内容を変更すると、システムがブートできなくなる可能性があります。

<execution stop_on_error="true">

マニフェストの実行要素は、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。

各チェックポイント名タグには、チェックポイントスクリプトが存在する場所を指定する mod-path 属性が含まれています。

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(1M) コマンドオプションを使用します。

一部の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。詳細は、マニフェストのコメントを参照してください。

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し示すチェックポイント名タグを追加する必要があります。

カスタムスクリプトを指し示す新しいチェックポイント名タグを追加する方法については、次の例を参照してください。

あるユーザーが、デフォルトの transfer-ips-checkpoint の後に構築処理で実行するカスタムスクリプト /tmp/myscript.sh を作成します。

この新しいスクリプトを指し示すには、transfer-ips-checkpoint 名のあとで次のタグをマニフェストに追加して、新しいスクリプトを指し示します。

```
<checkpoint name="custom-script"
 desc="my new script"
 mod_path="solaris_install/distro_const/checkpoints/custom_script"
 checkpoint_class="CustomScript">
   <args>/tmp/myscript.sh arg1 arg2/{PKG_IMAGE_PATH}</args>
</checkpoint>
```

ここで、「arg1」と「arg2」は、スクリプトが取るオプションの引数です。

'{PKG_IMAGE_PATH}' または '{BOOT_ARCHIVE}' の値は、distro_const ユーティリティーによって、実行中にそれぞれ <ZFS Dataset>/build_data/pkg_image または <ZFS Dataset>/build_data/boot_archive に置き換えられます。

注 – DC マニフェスト内では、複数のカスタムスクリプトチェックポイントを指定できます。各チェックポイントの名前は一意である必要があります。

<configuration name="pre-pkg-img-mod" type="sysconf" source="/etc/svc/profile/generic_limited_net.xml">

マニフェスト内の構成名要素は、イメージ作成処理中にメディアに適用される SMF サービスプロファイルを一覧表示します。これらの SMF サービスは、ブートされたメディア上で実行されるサービスと実行されないサービスを指定します。プロファイルは、この要素内で指定された順序で適用されます。

この要素を変更することはほとんどありません。

属性

次の属性については、attributes(5) のマニュアルページを参照してください。

属性タイプ	属性値
使用条件	install/distribution-constructor package
インタフェースの安定性	開発中

関連項目

`distro_const`(1M), `pkg`(1)

現在のリリースの OTN ドキュメントライブラリの『カスタム Oracle Solaris 11 インストールイメージの作成』。