Oracle® Solaris 11 安装手册页

文件号码 E25757
2012 年 3 月，E25757-03

ORACLE

120411@25097

目录

前言

本文档提供 Oracle Solaris 11 系统安装工具的手册页。

概述

以下内容包含对手册页各部分及其所引用信息的简要说明：

- 第 1 部分介绍了适用于操作系统的各种命令。
- 第 1M 部分介绍了主要用于系统维护和管理的各种命令。
- 第 5 部分包含其他文档，如字符集表。

下面是手册页的通用格式。每个手册的手册页部分通常遵循该顺序，但只包括需要的标题。例如，如果未报告任何已知问题，则不包括“已知问题”部分。通常，可以使用 `man` 命令来查看有关手册页的更多信息。

名称　本部分提供了记录的命令或函数的名称，后跟其用途的简要说明。

用法概要　本部分显示了命令或函数的语法。如果命令或文件不存在于标准路径中，则显示其全路径名。除非要求使用不同的参数顺序，否则选项和参数均按字母顺序排列，首先是单个字母的参数，接下来是带有参数的选项。

本部分使用以下特殊字符：

[ ]　括号。扩在这些括号中的选项或参数是可选的。如果省略括号，则必须指定参数。

. . .　省略号。可以为前一个参数提供多个值，或者可以多次指定前一个参数，例如 "`filename...`"。

|　分隔符。一次只能指定一个由该字符分隔的参数。

{ }　大括号。括在大括号内的选项和/或参数是相互依赖的，因此必须将大括号中的所有内容视为一个单元。

协议　本部分仅在第 3R 子部分出现，用于指示协议说明文件。

描述　本部分定义了服务的功能和行为。因此，它简明地介绍了命令执行哪些操作。它不讨论“选项”或引用“示例”。在“用法”下介绍了交互式命令、子命令、请求、宏和函数。

选项　本部分列出了各命令选项及每个选项用途的简明摘要。逐个列出各个选项，并以它们在“用法概要”部分显示的顺序排列。在选项下讨论各个选项可能的参数，还提供缺省值（如果适用）。

操作数　本部分列出了命令操作数，并介绍它们对命令操作的影响。

输出　本部分介绍了命令所生成的输出（标准输出、标准错误或输出文件）。

返回值　如果手册页记录返回值的函数，则本部分列出这些值并介绍返回这些值应满足的条件。如果函数只能返回常量值（例如 0 或 –1），则将在标记的段落中列出这些值。否则，会有单个段落介绍每个函数的返回值。声明为 void 的函数不返回值，因此不会在“返回值”中讨论这些函数。

错误　对于故障，大多数函数将指出它们出现故障的原因的错误代码置于全局变量 `errno` 中。本部分按字母顺序列出了函数可以生成的所有错误代码，并介绍了导致每个错误的条件。如果多个条件可以导致同一错误，则在错误代码下以单独的段落介绍每个条件。

用法　本部分列出了需要详细说明的特殊规则、功能和命令。此处列出的子部分用于说明内置功能：

命令
修饰符
变量
表达式
输入语法

示例　本部分提供了用法的示例，或者如何使用命令或函数的示例。尽可能显示包括命令行条目和计算机响应的完整示例。每当给定一个示例，就会出现提示并显示为 `example%`，或者如果用户必须为超级用户，则显示为 `example#`。示例后面跟有说明、变量替换规则或返回值。大部分示例说明了“用法概要”、“说明”、“选项”和“用法”部分的概念。

环境变量　本部分列出了命令或函数影响的所有环境变量，其后附加了关于影响的简要说明。

退出状态　本部分列出了命令返回到调用程序或 shell 中的值以及导致返回这些值的条件。通常，返回零表示成功完成，返回非零值表示各种错误条件。

文件　本部分列出了手册页引用的所有文件名称、相关文件以及命令创建或所需的文件。每个文件名称后面都具有描述性摘要或说明。

属性　本部分通过定义属性类型及其相应的值列出了命令、实用程序和设备驱动程序的特征。有关更多信息，请参见 `attributes(5)` 手册页。

另请参见　本部分列出了对其他手册页、内部文档和外部出版物的引用。

诊断　本部分列出了诊断消息以及导致错误的条件的简要说明。

警告	本部分列出了有关特殊条件的警告，这些条件可能会严重影响您的工作状况。此部分不是诊断列表。
附注	本部分列出了不属于页面任何部分的其他信息。它采用对用户旁白提示的形式，包含用户特别关注的要点。此处不包含关键信息。
已知问题	本部分介绍了已知问题，并尽可能给出解决方法。

参 考 文 档

系统管理命令

引用名

aimanifest – 修改自动化安装程序 (Automated Installer, AI) 使用的 XML 文件

用法概要

```
/usr/bin/aimanifest [-h]

aimanifest add [-r] path value

aimanifest get [-r] path

aimanifest set [-r] path value

aimanifest load [-i] filename

aimanifest validate
```

描述

aimanifest 命令可创建新的 XML 清单，或修改现有的 XML 清单。尽管包含对某个 DTD 定义的有效 !DOCTYPE 引用的任何 XML 文件都可以使用 aimanifest，但是，该命令的目标用途是创建自动化安装程序 (Automated Installer, AI) 使用的派生清单。有关 AI 派生清单的信息，请参见《安装 Oracle Solaris 11 系统》。

可以多次调用 aimanifest 命令来开发清单。AIM_MANIFEST 环境变量用于指定 aimanifest 要修改的清单的位置。必须设置 AIM_MANIFEST。每次结合 load、add 或 set 子命令调用 aimanifest 命令时都会打开、修改和保存 AIM_MANIFEST 文件。

aimanifest 命令可以修改的 AIM_MANIFEST 文件至少必须同时包含下列两个片段：

- 对某个 DTD 的 !DOCTYPE 引用，该引用对于所开发的 XML 清单应该有效。
- 该清单的根元素。

如果使用一个空的 AIM_MANIFEST 文件开始操作（就像 AI 执行派生的清单脚本时那样），那么，第一个 aimanifest 命令必须指定 load 子命令来加载最起码所需的 AIM_MANIFEST 文件。用于修改清单的后续 aimanifest 命令将使用该 DTD 来确定在开发清单中的哪个位置添加元素。

如果既要在 stdout 和 stderr 中显示消息，又要在某个文件中保存错误消息和提示性消息，请将 AIM_LOGFILE 环境变量设置为某个日志文件位置。信息将附加在该日志文件中。该日志文件不被清除。

选项

aimanifest 命令包括以下选项：

-h，--help　　显示用法帮助消息。

aimanifest 命令的 add、get 和 set 子命令包括以下选项：

-r, --return-path　　返回该 aimanifest 命令创建或对其执行操作的 XML 元素的路径。该返回路径是一个节点 ID 链。您可以保存这个返回路径，以便对 aimanifest 执行后续调用时使用。使用 -r 选项返回的路径，比使用 XML 元素和属性值指定路径更加可靠，因为在构建 AI 清单时，这些值可能会更改。有关 - r 选项返回的路径的更多信息，请参见“返回路径”部分。

aimanifest 命令的 load 子命令包括以下选项：

`-i, --incremental` 添加新数据之前不清除 `AIM_MANIFEST` 数据。

子命令 支持以下子命令：

`add [-r | --return-path]` *path value*
向 XML 清单添加新元素。在 *path* 上使用值 *value* 添加新元素。有关 *path* 的更多信息，请参见“操作数”部分。如果 *path* 以某个属性结尾 (`@`*attr*)，则新元素将包含 *attr* 属性，*value* 是该属性的值。

除了检查 *path* 中的父/子关系外，不执行任何验证。

`-r` 选项返回新增节点的路径。有关更多信息，请参见“返回路径”部分。

如果父路径与 `AIM_MANIFEST` 文件中的某个元素匹配，则该路径只能与一个元素匹配。新元素创建为匹配父元素的子级。路径可以指定元素和属性值，以匹配唯一的父元素，如本部分的“示例 2：包含值的路径”中所示。

如果父路径与 `AIM_MANIFEST` 文件中的元素不匹配，则会根据需要创建新元素，并将新的子元素添加到新父级。根据以下规则，将所添加元素的路径与现存元素分割开来：

- 分割发生在路径中用于指定某个值的所有组成部分之后。
- 分割发生在路径中用于指定某个值的所有组成部分之后，且在 DTD 允许多个带相同标记的相关元素的第一个位置处。

使用此 XML 清单结构分析以下示例：

- 清单以单个 A 节点开头。
- A 节点只能包含一个 B 节点子级。
- B 节点可以包含多个 C 节点子级。
- C 节点可以包含多个 D 节点子级。

示例 1：简单路径。AI 清单包含一个 A 节点、一个 B 节点和一个 C 节点：`/A/B/C`. 使用 *path* 值 `/A/B/C/D` 发出了 `add` 子命令。在此情况下，将会创建新的 C 节点，因为 C 节点是路径中可以包含具有相同标记的同级的首批节点。将为新的 C 节点添加新的 D 节点作为子级。生成的清单的结构为 `/A/B/{C,C/D}`。针对 D 的不同值发出相同的命令将会导致生成三个 C 节点：`/A/B/{C,C/D,C/D}`.

示例 2：包含值的路径。AI 清单包含一个 A 节点、一个 B 节点和两个 C 节点。其中只有一个 C 节点使用值 1，因此，清单的结构为 `/A/B/{C,C=1}`。使用 *path* 值 `/A/B/C=1/D` 和 *value* 值 10 发出了 `add` 子命令。在此情况下，将不会添加新的 C 节点，原因是为 C 指定值 1 将会标识一个唯一节点，并且无法在指定某个值的分支位置上或该位置的前面分割路径。该路径的第一个可分割位置是 D。将新增一个值为 10 的 D 节点，作为值为 1 的 C 节点的子级。生成的清单的结构为 `/A/B/{C,C=1/D=10}`。使用值 20 针对 D 发出同一命令会导致生成 `/A/B/{C,C=1/{D=10,D=20}}`。

`get [-r | --return-path]` *path*
检索元素或属性值。对于空的元素或属性值，将显示空字符串 ("")。 *path* 必须匹配某个现有的唯一元素或属性。有关 *path* 的更多信息，请参见“操作数”部分。

`-r` 选项返回所访问节点的路径，作为第二个返回字符串。有关更多信息，请参见“返回路径”部分。

`set [-r | --return-path]` *path value*
更改现有元素或属性的值，或者为现有元素创建新属性。不执行验证。

更改现有元素的值时，*path* 必须匹配某个现有的唯一元素。如果该元素包含标记相同的同级，则使用元素值或属性或者目标元素的子元素来使路径保持唯一。请参见“路径操作数”部分。

设置某个属性的值时，该属性不需要存在，但该属性所属的元素必须存在。

`-r` 选项返回所更改元素的路径。有关更多信息，请参见“返回路径”部分。

`load [-i | --incremental]` *filename*
从文件 *filename* 加载某个 XML 清单或部分 XML 清单。除了检查元素的父/子关系外，不执行任何验证。

如果未指定 `-i` 选项，则会覆盖任何现有的 XML 数据。`AIM_MANIFEST` 文件中的所有数据将替换为 *filename* 文件的内容。*filename* 文件必须包含对某个 DTD 的 `!DOCTYPE` 引用，以便后续的 `aimanifest` 命令能够修改此文件。

如果指定了 `-i` 选项，则添加新数据之前不清除 `AIM_MANIFEST` 数据。而是以递增方式在现有 XML 数据中插入新数据，或者将新数据与现有 XML 数据合并。`AIM_MANIFEST` 中的 `!DOCTYPE` 引用指定的 DTD 用于确定如何以及在何处合并 *filename* 数据。如果缺少 `!DOCTYPE` 引用，将使用位于 `/usr/share/install/ai.dtd` 的 AI 清单 DTD。如果无法使用 DTD 对 *filename* 中的数据进行核对，将返回非零错误状态。

以下事项会影响新数据在 `AIM_MANIFEST` 清单中的插入位置：

- 靠近 `AIM_MANIFEST` 数据路径开头的元素标记与 *filename* 数据路径的匹配程度
- 这些 `AIM_MANIFEST` 数据元素下面允许哪些子元素
- 在哪个位置允许标记相同的同级元素
- 无子级 `AIM_MANIFEST` 数据节点位于哪个位置

在处理 *filename* 数据的每个元素时，如果下列所有情况属实，则一般而言，不会为 `AIM_MANIFEST` 数据中的该元素创建新节点。而是使用新数据替换某个现有节点。

- 两组数据都包含一个标记与位置相同的节点。
- `AIM_MANIFEST` 中的 `!DOCTYPE` 引用指定的 DTD 不允许这两个节点作为标记相同的同级元素共同存在。
- *filename* 数据元素包含子级。

插入 *filename* 中的某个元素后，将在尽量靠近 AIM_MANIFEST 数据根、开始创建新节点的位置完成分割。将在允许带相同标记的同级元素的第一个位置，或者 AIM_MANIFEST 中不存在带相同标记的元素的第一个相应位置，创建分割操作的第一个新节点。

使用此 XML 清单结构分析以下示例：

- 清单以单个 A 节点开头。
- A 节点只能包含一个 B 节点子级。
- B 节点可以包含多个 C 节点子级。
- B 节点只能包含一个 E 节点子级。

示例 1：插入标记相同的元素。如果 AIM_MANIFEST 的内容为 /A/B/C1/D1，*filename* 的内容为 /A/B/C2/D2，则发出 load -i 命令后，AIM_MANIFEST 文件的内容为 /A/B/{C1/D1,C2/D2}。C 节点是第一个可以添加新节点的位置。*filename* 数据中的 C 节点添加在 AIM_MANIFEST 数据中的现有 C 节点之后。如果两个 A 元素使用不同的值，或者两个 B 元素使用不同值，则 *filename* 元素的值将替换 AIM_MANIFEST 元素的值。如果两个 A 元素使用不同的属性，或者两个 B 元素使用不同的属性，则会合并这些属性值。

- 同时在 AIM_MANIFEST 文件和 *filename* 文件中存在的 A 与 B 的属性将使用合并文件中 *filename* 文件的值。
- 在 AIM_MANIFEST 文件或 *filename* 文件中存在（但不在这两个文件中同时存在）的 A 与 B 的属性全部保留在合并文件中。

示例 2：插入标记不同的元素。如果 AIM_MANIFEST 的内容为 /A/B/C/D，*filename* 的内容为 /A/B/E/F，则发出 load -i 命令后，AIM_MANIFEST 文件的内容为 /A/B/{E/F,C/D}。E 节点添加在 DTD 允许这种节点的第一个位置。元素 A 和 B 的值是 *filename* 中的值，A 和 B 的属性已根据上述“示例 1”从 *filename* 合并为 AIM_MANIFEST 。

有时，无法确定正确的合并位置。如果需要跟在待合并节点后面的某个同级尚未添加，则可能会发生此情况。为避免出现此问题，请按照 DTD 强制要求的顺序，将多个节点或子树添加到某个公用父节点。如果无法确定某个节点在其新同级中的适当位置，请将此节点放置在新同级列表的末尾。

validate
根据 !DOCTYPE 语句中引用的 DTD 验证 AIM_MANIFEST 清单。错误将在 stderr 中显示。如果验证失败，则返回非零状态。

操作数

需要使用以下操作数。

Filename 操作数

load 子命令需要使用 *filename* 操作数，该操作数是要加载到 AIM_MANIFEST 清单的完整清单或部分清单的名称。

Value 操作数

add 和 set 子命令需要使用 *value* 操作数。*value* 操作数是 *path* 操作数指定的元素或属性的有效值。

Path 操作数

aimanifest 命令的 add、get 和 set 子命令需要使用 *path* 操作数。该路径定义了由元素和属性组成的 XML 分层结构中的某个节点。

XML 元素分层结构又称为 XML 树。在以下部分 AI 清单中，auto_install 元素是树根，ai_instance 和 software 元素是分枝，或子树的根。

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

在 aimanifest 路径语法中，使用正斜杠字符 (/) 表示树结构中的分枝。在当前示例中，software 元素的路径为 /auto_install/ai_instance/software 。

属性与某个元素绑定。在 aimanifest 路径语法中，使用 @ 符号标识属性名称。software 元素的 type 属性的路径为 /auto_install/ai_instance/software@type 。

aimanifest*path* 操作数必须对应于单个元素。请根据需要包含元素和属性值，以使路径保持唯一。例如，要为以下部分 AI 清单中定义的第二个分片指定大小，可以使用路径 /auto_install/ai_instance/target/disk/slice[@name="4"]/size@val 来指明要为哪个分片指定大小。

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

允许相对路径。以上段落中显示的 slice 路径在指定时可以使用 ai_instance、target、disk 或 slice 开头，因为只有一个 slice 使用了 name 属性值 4。例如，可以使用路径 slice[@name="4"]/size@val。

如果 *path* 中的某个 *value* 包含正斜杠字符，则必须将该值括在单引号或双引号中，例如 /name="pkg:/entire"。

如果 aimanifest 调用发生在 shell 脚本中，包含引号的值可能需要额外的特殊处理。在 shell 脚本中，可能需要使用前导反斜杠字符 (\) 对 aimanifest 路径值中的引号进行转义，使得 shell 不会删除或解释这些引号。检查使用的 shell 规则。以下示例显示了某个 ksh93 脚本中的一个包含正斜杠字符的值 :

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

本手册页中的大多数示例都省略了反斜杠转义字符，因为本手册页假定 `aimanifest` 不是在脚本或特定 shell 中调用的。有关 AI 派生的清单脚本的信息，请参见《安装 Oracle Solaris 11 系统》。

下列格式的分支显示了如何构造某个元素或元素属性的路径。

`/A`
: A 是元素的标记名称，例如 `/auto_install`。此分支规范又称为简单分支。仅包含简单分支的路径称为简单路径。

`/A=`*value*
: A 是元素的标记名称， *value* 是该元素的值，例如 `/name="pkg:/entire"`。

`/A[B/C=`*value*`]`
: A 是元素，B 是作为 A 子级的元素，C 是作为 B 子级的元素，*value* 是 C 元素的值。此路径格式指定了 A 元素，该元素包含值为 *value* 的孙级元素 C。例如，如果您的 AI 清单包含多个 software 部分，则可以使用此格式对用于安装软件包 `pkg:/entire` 的 software 部分进行操作，如以下路径所示：

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

`/A[@Aattr=`*value*`]`
: A 是元素，`Aattr` 是 A 的属性，*value* 是 `Aattr` 属性的值。此路径格式指定了 A 元素，该元素包含值为 *value* 的属性 `Aattr`。例如，如果您的 AI 清单定义了多个分片，则可以使用此格式对 `name` 值为 4 的分片进行操作，例如 `slice[@name="4"]`

`/A[B/C@Cattr=`*value*`]`
: A 是元素，B 是 A 的子级，C 是 B 的子级， `Cattr` 是 C 的属性，*value* 是 `Cattr` 属性的值。此路径格式指定了 A 元素，该元素包含一个孙级元素 C，该孙级元素的 `Cattr` 属性值为 *value*。例如，如果您的 AI 清单包含多个 software 部分，则可以使用此格式对包含某个 publisher 部分（其 name 值为 `solaris`）的 software 部分进行操作，例如，使用路径 `software[source/publisher@name="solaris"]` 。

`/A[1]`
: `/A[1]` 指定清单中某个 A 元素的第一个实例。例如，如果您的 AI 清单包含多个 software 部分，则可以使用此格式对第二个 software 部分进行操作，例如 `/auto_install[1]/ai_instance[1]/software[2]` 。

  这是 `-r` 选项返回的路径格式。请参见“返回路径”部分。

`/A@Aattr`
: 此路径指定了 A 元素的 `Aattr` 属性。此路径没有指定 A 元素，而是指定了 `Aattr` 属性。使用此格式可以设置或获取 `Aattr` 属性。

`/A[B/C=`*value*`]@Aattr`
: 此路径格式指定了 A 元素的 `Aattr` 属性，该元素包含值为 *value* 的孙级元素 C。

`/A[B/C@Cattr=`*value*`]@Aattr`
此路径格式指定了 `A` 元素的 `Aattr` 属性，该元素包含孙级元素 `C`，该孙级元素的 `Cattr` 属性值为 *value*。

`/A/B=`*value*`@Battr`
此路径指定了值为 *value* 的 `B` 元素的 `Battr` 属性。`B` 元素是 `A` 元素的子级。

返回路径

使用 `-r` 选项，`add`、`get` 和 `set` 子命令将返回子命令创建或访问的元素的地址。这个返回的地址采用了节点 ID 链的格式。可以使用这个返回的地址来再次访问相同的元素，即使与该元素关联的值已发生更改。

下列示例表明，使用 `-r` 选项返回的地址，比使用用于指定元素和属性值的路径可能要容易得多。从以下节点树开始：

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

添加一个 `name` 属性值为 `data2`、`name_type` 属性值为 `volid` 的新 `disk` 节点：

```
        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
```

使用一条命令就能轻松添加一个包含单个属性的新 `disk_name` 元素。要添加第二个和第三个属性，必须指定要更改的 `disk_name` 元素。比较用于多次访问同一节点的以下两种方法。

通过使用值来指定路径

本示例中的命令通过使用值来指定路径。请注意，您必须在第一个命令中分配唯一值，以便能够使用该值在后续命令中指定唯一路径。如果值发生变化，该方法可能会产生错误的结果。

```
$ 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
```

通过使用返回路径来指定路径

多次访问同一节点的最可靠方法是将路径保存到新的 `disk_name` 元素，然后使用这个保存的路径进行后续访问。

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

通过 `-r` 选项返回到 `$NewDisk` 的路径以 ID 形式表示节点，并且未赋值：

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

示例

要尝试演示这些示例，需要设置 `AIM_MANIFEST`。

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

`aimanifest` 命令可以修改的 `AIM_MANIFEST` 文件至少必须同时包含下列两个片段：

- 对某个 DTD 的 `!DOCTYPE` 引用，该引用对于所开发的 XML 清单应该有效。
- 该清单的根元素。

以下示例显示了某个 AI 清单最起码所需的 `AIM_MANIFEST` 清单文件：

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

通常，您会在对现有有效 AI 清单执行操作的派生清单脚本中使用 `aimanifest` 命令。要尝试演示这些示例，您可以复制 `/usr/share/auto_install/manifest/default.xml`，然后定义 `AIM_MANIFEST` 以引用该副本。确保该副本可写。

示例 1 设置 auto_reboot 属性

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

示例 2 获取 auto_reboot 值

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

示例 3 通过使用值路径添加发布者

此示例中的软件包系统信息库是位于 `file:///net/host2/export/extras_repo` 的文件系统信息库。发布者为 `extras`。

```
$ 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
```

这些 `aimanifest` 命令将导致生成下列 AI 清单条目。`software` 元素是允许相同标记的同级的 *path* 中的第一个元素，因此，该 XML 代码部分跟在已出现在输出文件中的最后一个 `software` 部分的后面。

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

示例 4 通过使用返回路径添加发布者

该示例与前一示例相同，但使用不同的方法来实现相同的结果。

```
$ 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)
```

示例 5 通过添加清单段来添加发布者

该示例与前一示例相同，但使用第三种方法来实现相同的结果。

创建一个名称为 `extras.xml`、包含以下内容的文件：

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

示例 5 通过添加清单段来添加发布者 （续）

```
      </source>
    </software>
  </ai_instance>
</auto_install>
```

尽管您只需要 software 部分，但您必须同时包含 auto_install 和 ai_instance 元素。必须包含 *path* 操作数中所需的任何内容。如果加载的文件为 auto_install 或 ai_instance 元素指定了属性，则这些属性值将替换现有值，或者被添加。

使用以下命令将此 software 部分添加到 AIM_MANIFEST 清单：

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

示例 6 通过使用值路径添加软件包

该示例通过指定发布者名称作为路径中的值，将一个软件包添加到前一示例生成的 software 元素，该元素包含名称为 extras 的 publisher 元素。该示例还显示了如何使用相对路径规范。

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

该 aimanifest 命令添加了以下 software_data 部分。

```
<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>
```

示例 7 通过使用返回路径添加软件包

该示例与前一示例相同，但使用不同的方法来实现相同的结果。该示例没有指定发布者名称作为路径中的值，而是使用了示例“通过使用返回路径添加发布者”所述的保存在 SW_PATH 中的 software 元素的路径。

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

示例 8 验证清单

验证 AIM_MANIFEST 清单。

```
$ aimanifest validate
```

退出状态

将返回以下退出值：

`0` 该命令已成功处理。

`>0` 出现错误。

文件

`AIM_MANIFEST` 该环境变量值是正在构建的 AI 清单的位置。

`AIM_LOGFILE` 该环境变量值是 `aimanifest` 操作的日志文件的位置。

属性

有关下列属性的说明，请参见 `attributes(5)`：

属性类型	属性值
可用性	`system/install/auto-install/auto-install-common`
接口稳定性	Uncommitted（未确定）

另请参见

`installadm(1M)`、`pkg(1)`

《安装 Oracle Solaris 11 系统》中的第 III 部分, “使用安装服务器安装”

引用名

distro_const – 用于创建 Oracle Solaris 映像和介质的实用程序

用法概要

```
/usr/bin/distro_const

distro_const --help

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

描述

`distro_const` 命令使用户能够通过将指定的清单文件用作映像蓝图来创建映像。

可以创建用于在 x86 系统或 SPARC 系统上安装 Oracle Solaris 操作系统的文本安装程序映像。

或者可以创建相当于包含 Oracle Solaris 操作系统的 LiveCD 映像的 ISO 映像。

或者，您可以创建可用于 SPARC 客户机上 Oracle Solaris OS 的网络安装的 SPARC AI ISO 映像，或者可用于 x86 客户机上 Oracle Solaris OS 的网络安装的 x86 AI ISO 映像。

亦或者，可以创建定制 ISO 映像。

不带选项的基本 `distro_const` 命令在一个步骤中创建完整映像。

命令选项使用户能够在各种检查点暂停和继续映像创建过程，从而使用户能够在每个阶段查看映像的状态并检查已知问题。检查点通过允许您忽略已执行至少一次的冗长步骤来节省构建时间。

注 – 您必须采用超级用户角色或具有超级用户特权来运行 `distro_const` 命令。

使用分发构造器时，只能在 SPARC 系统上创建 SPARC 映像。并且，只能在 x86 系统上创建 x86 映像。此外，系统上的操作系统发行版必须与您构建的映像的发行版相同。

子命令

`distro_const` 命令具有下面列出的子命令和选项。另请参见“示例”部分。

`--help`
显示用法。

build
`distro_const build` *manifest*

需要 "build" 子命令。

将指定的清单文件用作映像的蓝图来创建完整映像。需要清单名称。

`-v distro_const build - v`
指定详细模式。

`-l distro_const build [- l]` *manifest*
列出您可以选择停止或继续构建映像的所有有效检查点。该命令选项查询清单文件以寻找有效检查点。对于其他检查点命令选项，将该命令提供的名称用作有效值。需要 build 子命令。

检查点值取决于清单文件中的条目。

`-p distro_const build [- p` *checkpoint*`] ` *manifest*
构建映像，但在指定的检查点名称处暂停构建映像。使用 -l 选项查找有效名称。可以将 -r 选项和 -p 选项结合使用。需要检查点名称和清单名称。需要 build 子命令。

`-r distro_const build [- r` *checkpoint*`] ` *manifest*
从指定的检查点名称处继续构建映像。指定的名称必须是上一次构建停止执行的检查点，或者是更早的检查点。以后的检查点无效。使用 -l 选项可确定可继续进行的检查点。-p 选项可与 -r 选项结合使用。需要检查点名称和清单名称。需要 build 子命令。

`-h distro_const [- h]`
显示命令用法。

示例

示例 1 使用检查点选项创建映像

1. 检查可用的检查点。本示例中的清单名称为 `dc_livecd.xml`。

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

   显示有效的检查点名称，如本样例输出。

   ```
   Checkpoint           Resumable  Description
   ----------------------------------------------
   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. 开始构建映像，在 ba-init 检查点暂停。

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

3. 从 ba-init 检查点重新启动构建。完成创建映像。

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

示例 2 在一个步骤中创建映像

要运行完整的映像构建过程而不暂停，请使用不带检查点选项的基本 `distro_const` 命令。清单文件名称为 dc_livecd.xml。

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

属性

有关下列属性的说明，请参见 attributes(5)：

属性类型	属性值
可用性	install/distribution-constructor
接口稳定性	Under Development（开发中）

另请参见

dc_manifest(4)

在 OTN 文档库中为当前版本创建定制 Oracle Solaris 11 安装映像。

引用名

installadm – 管理网络上的自动化安装

用法概要

```
/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
```

描述

自动化安装程序 (Automated Installer, AI) 用于通过网络在一个或多个 SPARC 和 x86 系统上自动完成 Oracle Solaris OS 的安装。

通过网络利用 AI 所需的计算机拓扑需要包含一台安装服务器、一台 DHCP 服务器（这可以是与安装服务器相同的系统），以及安装客户机。在安装服务器上，安装服务设置为包含 AI 引导映像（为客户机提供该映像以便它们通过网络引导）、输入规范（AI 清单和派生的清单脚本，将为客户机选择其中一项）和服务管理工具 (Service Management Facility, SMF) 配置文件（将为客户机选择其中的零项或多项）。

AI 引导映像的内容作为软件包 `install-image/solaris-auto-install` 发布，并通过 `create-service` 子命令安装。`create-service` 子命令还能够接受和解压缩 AI ISO 文件以创建 AI 引导映像。

安装服务是使用缺省 AI 清单创建的，但是可以使用 `create-manifest` 子命令将定制的清单或派生的清单脚本（以下称“脚本”）添加到安装服务。有关如何创建清单和派生的清单脚本的信息，请参见《安装 Oracle Solaris 11 系统》。`create-manifest` 子命令还允许指定条件，这些条件用于确定应该为安装客户机选择哪个清单或脚本。可以使用 `set-criteria` 子命令修改已经与清单或脚本关联的条件。

清单可以包括诸如目标设备、分区信息、软件包列表和其他参数之类的信息。脚本包含的命令用于查询正在运行的 AI 客户机系统并基于它找到的信息构建定制清单。使用脚本调用 AI 时，AI 的第一项任务是运行该脚本以生成清单。

在客户机引导时，将启动与客户机的计算机条件匹配的清单或脚本搜索。在找到匹配的清单或脚本时，将根据匹配清单文件中的规范或派生自匹配脚本的清单文件中的规范，在客户机上安装 Oracle Solaris 发行版。每个客户机只能使用一个清单或脚本。

每个服务都有一个缺省清单或脚本。其他清单或脚本的条件与所安装的系统不匹配时，会使用缺省清单或脚本。可以将任何清单或脚本指定为缺省清单或脚本。与缺省清单或脚本关联的任何条件将变为非活动，且在选择清单或脚本期间不予考虑。如果稍后将其他清单或脚本设为缺省清单或脚本，则之前的缺省清单或脚本的条件重新变为活动。没有与其关联的条件的清单或脚本只能用作缺省清单或脚本。将其他清单或脚本指定为缺省清单或脚本时，没有条件的清单或脚本将变为非活动。

系统配置文件是对清单和脚本的补充，因为它们也包含安装规范。特别是，使用配置文件指定配置信息，如用户名、用户口令、时区、主机名和 IP 地址。配置文件可以包含变量，用于从安装服务器环境或从 `create-profile` 子命令中指定的条件获取配置参数。这样，单个配置文件可以在不同的客户机上设置不同的配置参数。请参见“示例”部分。

系统配置文件由 `smf`(5) 处理且符合文档格式 `service_bundle`(4)。有关系统配置文件的更多信息，请参见 `sysconfig`(1M) 和《安装 Oracle Solaris 11 系统》。每个客户机可以使用任意数目的系统配置文件。不能为每个客户机系统多次指定某个特定的 SMF 属性。

如果希望特定的客户机使用某个特定的安装服务，可以使用 `create-client` 子命令将该客户机与该服务关联。也可以使用 `create-client` 修改现有的客户机。

`installadm` 实用程序可以用于完成以下任务：

- 设置安装服务和别名
- 设置安装映像
- 设置或删除客户机
- 添加、更新或删除清单和脚本
- 指定或修改清单或脚本的条件
- 导出清单和脚本
- 添加或删除系统配置文件
- 验证配置文件
- 指定或修改配置文件的条件
- 导出配置文件
- 启用或禁用安装服务
- 列出安装服务
- 列出安装服务的客户机
- 列出安装服务的清单和脚本
- 列出安装服务的配置文件

安装服务器配置属性

`svc:/system/install/server:default` SMF 服务的以下三个属性可用于配置安装服务器。

`all_services/networks`
要允许或禁用的网络的列表，采用 CIDR 格式（例如 192.168.56.0/24），允许还是禁用具体取决于 `all_services/exclude_networks` 属性的设置。

使用该网络列表可指定此安装服务器为哪些客户机提供服务。缺省情况下，AI 安装服务器配置为向该服务器连接的所有网络上的安装客户机提供服务（如果该服务器为多宿主服务器）。

`all_services/exclude_networks`
布尔值。如果为 true，则从此安装服务器提供服务的网络中排除由 `all_services/networks` 属性指定的网络。如果为 false，将包括由 `all_services/networks` 属性指定的网络。

`all_services/port`
指定托管 AI 安装服务 Web 服务器的端口。缺省情况下，在端口 5555 上托管 Web 服务器。

如果要使用与缺省值不同的端口号，则在创建任何安装服务之前定制 `port` 属性。

选项

`installadm` 命令具有以下选项：

`-h`，`--help`　显示用法帮助消息。

子命令

`installadm` 命令具有下面列出的子命令。另请参见下面的“示例”部分。

`help [`*subcommand*`]`
显示 `installadm` 实用程序的语法。

subcommand　仅显示指定子命令的语法。

`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]`
该子命令在指定的 *imagepath* 目录中设置网络引导映像（网络映像），并创建一个安装服务，用于指定如何安装从网络映像引导的客户机。

AI 引导映像内容作为软件包 `install-image/solaris-auto-install` 发布。如果未指定 `-s` 选项，则从系统发布者首选项列表中第一个提供该软件包实例的发布者安装该软件包。`-s` 选项接受 `pkg` 规范作为映像 ISO 文件的完整 FMRI 或位置。生成的网络映像最终位于 *imagepath* 中。该网络映像将用于客户机安装。

请注意以下规范：

- 在安装服务器上创建给定体系结构的第一个安装服务时，会自动创建该服务的别名 `default-i386` 或 `default-sparc`。此缺省服务用于具有该体系结构、且未通过 `create-client` 子命令显式添加到安装服务器的客户机的所有安装。要更改 `default-`*arch* 服务用作别名的服务，请使用 `set-service` 子命令。

  如果将 `default-`*arch* 别名更改为新的安装服务且找到了本地 ISC DHCP 配置，则此缺省别名引导文件将设置为该体系结构的缺省 DHCP 服务器范围的引导文件。
- 如果希望客户机使用与该体系结构的缺省安装服务不同的安装服务，则必须使用 `create-client` 子命令创建特定于客户机的配置。
- 如果使用了 `-i` 选项和 `-c` 选项，且尚未配置 DHCP 服务器，则会配置 ISC DHCP 服务器。

  如果已配置 ISC DHCP 服务器，将更新该 DHCP 服务器。

  即使提供了 `-i` 和 `-c` 参数且配置了 DHCP，在所创建的安装服务和 IP 范围之间也不会存在绑定。传递 `-i` 和 `-c` 时，会设置 IP 范围，根据需要创建新的 DHCP 服务器，并且该 DHCP 服务器将保持打开状态并为要使用的所有安装服务和所有客户机运行。为 DHCP 服务器提供的网络信息与所创建的服务没有特定的关系。
- 如果请求的 IP 范围不在安装服务器与之具有直接连接的子网上且安装服务器是多宿主的，则 `-B` 选项将用于提供引导文件服务器的地址（通常为此系统上的 IP 地址）。仅当在安装服务器上配置了多个 IP 地址且使用了 DHCP 中继时，才需要使用该选项。在其他所有配置中，软件可以自动确定此情况。

`-n|--service` *svcname*
可选：使用此安装服务名称而不是系统生成的服务名称。*svcname* 可以包含字母数字字符、下划线 (_) 和连字符 (-)。*svcname* 的第一个字符不得为连字符。

如果未指定 `-n` 选项，则自动生成服务名称。

`-t|--aliasof` *existing_service*
可选：此新服务是 *existing_service* 的替代名称。

`-a|--arch` *architecture*
可选：选择特定的变体体系结构。合法值为 `i386` 或 `sparc`。如果未指定，则选择与服务器的体系结构相对应的变体。

仅当 `-s` 参数是 `pkg`(5) 软件包时，`-a` 选项才适用。

`-p|--publisher` *prefix=origin*
可选：`pkg`(5) 发布者，格式为 *prefix=origin*（从其安装客户机映像）。

如果未指定 `-p` 选项，则使用系统的发布者首选项列表中第一个提供该软件包实例的发布者。

`-s|--source` *FMRI_or_ISO*
可选：指定网络映像的数据源。它可以是以下任一项：

- `pkg`(5) 软件包的完整 FMRI。
- AI ISO 映像的路径。

如果未指定 `-s`，则使用的 `install-image/solaris-auto-install` 软件包来自以下任一项：

- 使用 `-p` 参数指定的发布者。
- 系统的发布者首选项列表中第一个提供该软件包实例的发布者。

`-b|--boot-args` *boot_property=value*`,...`
可选：仅用于 x86 客户机。在服务映像中特定于服务的 `menu.lst` 文件中设置属性值。使用此选项可设置特定于此服务的引导属性。此选项可以接受多个逗号分隔的 *boot_property=value* 对。

`-i|--ip-start` *dhcp_ip_start*
可选：指定要添加到本地 DHCP 配置的范围中的起始 IP 地址。IP 地址的数量由 `-c` 选项提供。如果本地 ISC DHCP 配置不存在，则启动 ISC DHCP 服务器。

`-c|--ip-count` *count_of_ipaddr*
可选：设置 DHCP 配置中的 IP 地址总数（等于 *count_of_ipaddr* 的值）。第一个 IP 地址是 `-i` 选项提供的 *dhcp_ip_start* 的值。

`-B|--bootfile-server` *server_ipaddr*
可选：用于提供客户机应该从其请求引导文件的引导服务器的 IP 地址。仅当通过其他方法无法确定此 IP 地址时才是必需的。

`-d|--imagepath` *imagepath*
可选：指定创建网络映像的路径。如果未指定，则使用缺省位置 `/export/auto_install/` *svcname*。除非还指定了 `-y`，否则将显示确认提示。

`-y|--noprompt`
可选：禁止任何确认提示，并使用提供的选项和任何缺省值继续进行服务创建（请参见 `-d`）。

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

`-o|--option` *prop=value*
指定要设置的属性和值。

prop=value 可以是：

- `aliasof=`*existing_service*

  使 *svcname* 成为 *existing_service* 的别名。

- `default-manifest=`*manifest_name*

  将已向给定服务注册的特定清单或脚本指定为该服务的缺省清单或脚本。使用以下命令可显示向此服务注册的清单和脚本的列表。

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

svcname
必需：指定要设置其属性的安装服务名称。

`rename-service` *svcname newsvcname*
将安装服务 *svcname* 重命名为 *newsvcname*。*newsvcname* 可以包含字母数字字符、下划线 (_) 和连字符 (-)。*newsvcname* 的第一个字符不得为连字符。

`enable` *svcname*
启用 *svcname* 安装服务。

`disable` *svcname*
禁用 *svcname* 安装服务。

`delete-service [-r|--autoremove ] [-y|--noprompt]` *svcname*
删除安装服务。完成以下操作：

- 删除此安装服务的清单、配置文件、客户机配置文件和 Web 服务器配置。
- 删除用于实例化服务的映像。
- 如果服务为缺省别名且存在本地 ISC DHCP 配置，则从 ISC DHCP 配置中删除与此服务关联的引导文件。

`-r|--autoremove`
如果已指定，则分配给此服务的任何客户机和作为此服务的别名的任何服务也将被删除。

`-y|--noprompt`
禁止任何确认提示并继续进行服务删除。

svcname
必需：指定要删除的安装服务名称。

`list [-n|--service` *svcname*`] [-c|--client] [-m|- -manifest] [-p|--profile]`
列出服务器上已启用的所有安装服务。

`-n|--service` *svcname*
可选：列出有关本地服务器上特定安装服务的信息。

- 如果指定了 `-c` 选项，则列出与安装服务关联的客户机信息。
- 如果指定了 `-m` 选项，则列出与安装服务关联的清单和脚本。
- 如果指定了 `-p` 选项，则列出与安装服务关联的配置文件。

`-c|--client`
可选：列出本地服务器上安装服务的客户机。

`-m|--manifest`
可选：列出本地服务器上与安装服务关联的清单和脚本。

如果未指定 `-n`，则显示每个服务的简化列表。这包括缺省清单和脚本以及具有与之关联的条件的所有非缺省清单和脚本。不列出条件。

如果指定了 -n，则使用包括每个清单的条件的更完整列表格式，显示给定服务的所有清单和脚本。没有关联的条件且未指定为缺省清单的非活动清单会进行相应的标记。与缺省清单关联的条件被标记为非活动。

-p|--profile
可选：列出本地服务器上与安装服务关联的配置文件。

如果未指定 -n，则显示每个服务的简写列表，其中包括配置文件名称。

如果指定了 -n，则显示所请求的服务的配置文件及其条件。

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]
为特定的服务创建清单或脚本，从而使该清单或脚本在网络上可用，该操作独立于创建服务。非缺省清单或脚本在没有关联的条件时才可以使用（可以处于活动状态）。可以在命令行上输入条件 (-c)，也可以通过条件 XML 文件输入条件 (-C)。在清单或脚本不再指定为缺省值之前，暂时忽略指定的任何条件以及 -d 选项。

按以下顺序确定清单的名称：

1. 由 -m 选项指定的 *manifest_name*（如果存在）。
2. ai_instancename 属性的值（如果在清单中存在）。
3. 清单或脚本文件名的基名。

-n|--service *svcname*
必需：指定此清单或脚本要与之关联的安装服务名称。

-f|--file *manifest_or_script_filename*
必需：指定要添加的清单或脚本的路径名称。

-m|--manifest *manifest_name*
可选：指定清单或脚本的 AI 实例名称。将清单的 ai_instance 元素的 name 属性设置为 *manifest_name*。在后续的 installadm 命令和 installadm list 输出中，清单或脚本由 *manifest_name* 引用。

-c|--criteria *criteria=value*|*list*| *range*...
可选：指定要与添加的清单或脚本关联的条件。请参见下面的“条件”部分。发布缺省清单时，会注册条件，但在清单或脚本不再指定为缺省清单或脚本之前使其保持非活动状态。可以多次指定 -c 选项。

-C|--criteria-file *criteriafile*
可选：指定包含要与添加的清单或脚本关联的条件的条件 XML 文件的路径名称。发布缺省清单或脚本时，会注册条件，但在清单或脚本不再指定为缺省清单或脚本之前使其保持非活动状态。

-d|--default
可选：指定此清单或脚本是服务的新缺省清单或脚本。在清单或脚本不再是缺省值之前，将忽略指定的任何条件。

`update-manifest -n|--service` *svcname* `-f|--file` *manifest_or_script_filename* `[-m|--manifest` *manifest_name* `]`
更新与特定的安装服务关联的特定清单或脚本。在对清单或脚本进行更新时，任何条件或缺省状态都保持不变。

按以下顺序确定清单的名称：

1. 由 `-m` 选项指定的 *manifest_name*（如果存在）。
2. `ai_instancename` 属性的值（如果在更改后的清单中存在，且它与现有清单的 `ai_instancename` 值匹配）。
3. 清单或脚本文件名的基名（如果它与现有清单中的 `ai_instancename` 属性值匹配）或由 `installadm list` 提供的名称（如果它与现有脚本的名称匹配）。

供替换的清单或脚本由 *manifest_or_script_filename* 提供。

`-n|--service` *svcname*
必需：指定此清单或脚本要与之关联的安装服务名称。

`-f|--file` *manifest_or_script_filename*
必需：指定供替换的清单或脚本的路径名称。

`-m|--manifest` *manifest_name*
可选：指定供替换的清单或脚本的 AI 实例名称。

`delete-manifest -m|--manifest` *manifest_name* `-n|--service` *svcname*
删除通过特定的安装服务发布的清单或脚本。不能删除缺省清单或脚本。

`-m|--manifest` *manifest_name*
必需：使用 `-n` 选项将清单或脚本的 AI 实例名称指定为 `installadm list` 的输出。

`-n|--service` *svcname*
必需：指定此清单与之关联的安装服务名称。

`create-profile -n|--service` *svcname*`-f|--file`*profile_filename*
`...[-p|--profile`*profile_name* `] [-c|--criteria`*criteria* `=`*value*`|`*list*`|` *range*`...|`
`-C|--criteria-file` *criteriafile*`]`
为特定的安装服务创建配置文件。通过在命令行 (`-c`) 上或者通过条件 XML 文件 (`-C`) 输入条件，可以根据需要使这些条件与配置文件关联。在没有条件的情况下创建的配置文件与服务的所有客户机关联。

按以下顺序确定配置文件的名称：

1. 由 `-p` 选项指定的 *profile_name*（如果存在）。
2. 配置文件名称的基名。

配置文件名称对于 AI 服务必须是唯一的。如果使用了多个 `-f` 选项来创建具有相同条件的多个配置文件，则 `-p` 选项无效，且配置文件的名称派生自其文件名。

`-n|--service` *svcname*
必需：指定要更新的配置文件的安装服务名称。

`-f|--file` *profile_filename*`...`
必需：指定添加配置文件所使用的文件的路径名称。可以指定多个配置文件。

`-p|--profile` *profile_name*
可选：指定要创建的配置文件的名称。仅对单个配置文件的创建有效。

`-c|--criteria` *criteria*=*value*|*list*| *range*`...`
可选：指定要与配置文件关联的条件。请参见下面的"条件"部分。可以指定多个 `-c` 选项。

`-C|--criteria-file` *criteriafile*
可选：指定条件 XML 文件的路径名称，该文件包含与指定配置文件相关联的条件。

`delete-profile -p|--profile` *profile_name*`...` `-n|--service` *svcname*
从 *svcname* 安装服务中删除 *profile_name* 配置文件。

`-p|--profile` *profile_name*`...`
必需：指定要删除的配置文件的名称。可以指定多个 `-p` 选项。

`-n|--service` *svcname*
必需：指定要删除的配置文件的安装服务名称。

`export -n|--service` *svcname* `-m|--manifest` *manifest_name*`...` `-p|--profile` *profile_name*`...` `[-o|--output` *pathname*`]`
显示（导出）属于服务的指定清单/脚本和/或配置文件。必须至少指定一个清单/脚本或配置文件。除非 `-o` 选项重定向到文件或目录，否则显示转到 `stdout`。

`-n|--service` *svcname*
必需：指定与要导出的清单或配置文件关联的安装服务。

`-m|--manifest` *manifest_name*`...`
指定要导出的清单或脚本的 AI 实例名称。可以指定多个 `-m` 选项：

`-p|--profile` *profile_name*`...`
指定要导出的配置文件的名称。可以指定多个 `-p` 选项。

`-o|--output` *pathname*
可选：重定向输出。如果请求多个清单、脚本和/或配置文件，则 *pathname* 必须是目录。如果仅请求一个清单、脚本或配置文件，则 *pathname* 可以是文件。

`validate -n|--service` *svcname*`-P|--profile-file` *profile_filename*`...| -p|--profile` *profile_name*`...`
验证指定的配置文件。`validate` 子命令可以用于验证数据库中的配置文件 (`-p`)，或者对正在开发、尚未进入数据库的配置文件进行验证 (`-P`)。

`-n|--service` *svcname*
必需：指定配置文件与之关联的服务。

`-P|--profile-file` *profile_filename*`...`
指定要验证的外部配置文件。

`-p|--profile` *profile_name*`...`
指定要验证的配置文件的名称。

`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*`...`
更新已发布的清单/脚本和/或配置文件的条件。可以通过命令行或者通过条件 XML 文件指定条件。必须使用互斥选项 `-a`、`-c` 和 `-C` 之一指定条件。

在 `create-manifest` 子命令下对有效条件进行了说明。

`-m|--manifest` *manifest_name*
指定清单或脚本的 AI 实例名称。

`-p|--profile` *profile_name*`...`
指定配置文件的名称。可以指定任意数目的配置文件。

`-n|--service` *svcname*
必需：指定与此清单/脚本或配置文件关联的安装服务名称。

`-c|--criteria` *criteria*`=`*value*`|`*list*`|` *range*`...`
指定用于替换清单/脚本或配置文件的所有现有条件的条件。请参见下面的“条件”部分。

`-C|--criteria-file` *criteriafile*
指定一个条件 XML 文件的路径名称，该文件包含用于替换清单/脚本或配置文件的所有现有条件的条件。

`-a|--append-criteria` *criteria*`=`*value*`|`*list*`|` *range*`...`
指定要附加到清单/脚本或配置文件的现有条件的条件。请参见下面的“条件”部分。如果指定的 *criteria* 已存在，则该条件的 *value*|*list*|*range* 将由指定的 *value*|*list*|*range* 替换。

`create-client [-b|--boot-args` *property*`=`*value*`,...] - e|--macaddr` *macaddr*
`-n |--service` *svcname*
完成指定客户机的可选设置任务，以便提供与 `create-service` 子命令使用的缺省设置不同的定制客户机设置。使用户可以指定客户机的非缺省服务名称和引导参数。也可以用于修改现有的客户机。

如果客户机是 x86 系统且存在本地 ISC DHCP 配置，则将在 ISC DHCP 配置中配置客户机。

`-b|--boot-args` *property*`=`*value*`,...`
可选：仅用于 x86 客户机。在 `/etc/netboot` 中特定于客户机的 `menu.lst` 文件中设置属性值。使用此选项可设置特定于此客户机的引导属性。此选项可以接受多个 *property*=*value* 对。

`-e|--macaddr` *macaddr*
必需：指定客户机的 MAC 地址。

`-n|--service` *svcname*
必需：指定客户机安装的安装服务。

`delete-client` *macaddr*
删除现有客户机的具体服务信息，这些信息是以前使用 `create-client` 子命令设置的。

如果客户机是 x86 系统且存在本地 ISC DHCP 配置，则将在 ISC DHCP 配置中取消配置客户机。

macaddr 必需：指定要删除的客户机的 MAC 地址。

条件

清单、脚本和配置文件可以用于根据某些特征或条件以不同方式配置 AI 客户机。只有一个清单或脚本可以与特定的客户机关联。任意数目的配置文件可以与特定的客户机关联。

条件值由 AI 客户机在启动期间确定。

除非另有说明，否则可以同时为清单/脚本和配置文件指定以下 AI 客户机系统条件。

请参见“示例”部分以了解如何在命令行上指定条件。有关创建条件文件的信息，请参见《安装 Oracle Solaris 11 系统》。

条件	说明
`arch`	每个 `uname -m` 的体系结构。
`cpu`	每个 `uname -p` 的 CPU 类
`hostname`	已指定主机名。只能用于配置文件，而不能用于清单。
`ipv4`	IP 版本 4 网络地址。
`mac`	含有冒号 (:) 分隔符的十六进制 MAC 地址。
`mem`	每个 `prtconf`(1M) 的内存大小 (MB)。
`network`	IP 版本 4 网络编号。
`platform`	每个 `uname -i` 的平台名称。
`zonename`	每个 `zones`(5) 的区域名称。

`ipv4`、`mac`、`mem` 和 `network` 规范可以表示为由连字符 (`-`) 分隔的范围值。要指定对范围的一端没有限制，请使用 `unbounded`。

不能指定为范围的任何条件都可以指定为空格分隔值列表。

示例

示例 1 从 ISO 文件设置新的 x86 安装服务

首次设置安装服务器和 x86 安装服务。命令包括起始 IP 地址和 IP 地址的总计数，以便配置 DHCP 服务器。

```
# 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
```

AI ISO 映像位于 `/export/isos/sol-11-i386.iso`。命令在基于 AI ISO 映像的 `/export/images/soli386` 处设置网络映像和安装服务。此网络映像将用于客户机安装。

起始 IP 地址 172.0.0.10 和十个 IP 地址将添加到本地 ISC DHCP 配置。如果本地 ISC DHCP 配置不存在，则启动 ISC DHCP 服务器。

由于这是创建的第一个 `i386` 服务，因此会自动创建 `default-i386` 服务并将其作为此服务的别名。`default-i386` 别名是可操作的，通过 PXE 引导的客户机将从 `default-i386` 服务引导和安装。

示例 2 从 ISO 文件设置新的 SPARC 安装服务

首次设置 SPARC 安装服务。

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

AI ISO 映像位于 `/export/isos/sol-11-sparc.iso`。命令在基于 AI ISO 映像的 `/export/images/solsparc` 处设置网络映像和安装服务。此网络映像将用于客户机安装。

由于这是创建的第一个 SPARC 服务，因此会自动创建 `default-sparc` 服务并将其作为此服务的别名。`default-sparc` 别名是可操作的，SPARC 客户机将从 `default-sparc` 服务引导和安装。

示例 3 从软件包系统信息库设置 i386 安装服务

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

在 i386 安装服务器上，此命令在缺省映像位置 `/export/auto_install/mysvc` 设置 i386 网络映像和名为 `mysvc` 的安装服务。`-y` 选项提供对缺省位置可接受的确认。由于未指定体系结构，因此创建的服务具有与安装服务器相同的体系结构。此命令假定安装服务器的 `pkg publisher` 列表上的软件包系统信息库包含 `install-image/solaris-auto-install` 软件包。

要指定在此服务器上创建 SPARC 服务，请使用 `-a` 选项。

要指定 `solaris-auto-install` 软件包的源，请使用 `-p` 选项。例如，使用以下命令可将位于 `http://example.company.com:4281` 的 `ai-image` 系统信息库指定为 `solaris-auto-install` 软件包的源：

示例 3 从软件包系统信息库设置 i386 安装服务 （续）

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

示例 4 将客户机与安装服务关联

使用以下样例命令可将客户机与特定的安装服务关联。安装服务必须已存在。

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

在此示例中，该命令为 MAC 地址为 `0:e0:81:5d:bf:e0` 的系统创建了一个特定于客户机的设置。此客户机将使用以前设置的、名为 `my-i386-service` 的安装服务以及该服务的关联网络映像。此命令将在 `/etc/netboot` 中特定于客户机的 `menu.lst` 文件中设置引导属性 `console=ttya`。

示例 5 在不修改缺省服务的情况下添加新安装服务

使用以下样例命令可添加名为 `my-sparc-service` 的新服务、保留现有的服务并使现有的缺省服务保持不变。

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

示例 6 添加新的安装服务并更新缺省服务

使用以下两个样例命令可添加名为 `my-sparc-service` 的新服务、保留现有的服务并使新服务成为 SPARC 客户机的缺省服务。

```
# 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
```

示例 7 将定制的缺省 AI 清单添加到安装服务

使用以下样例命令可将新清单添加到 `service_092910` 安装服务，并使其成为服务的缺省清单。清单数据在 `my_manifest.xml` 中。将来的 `installadm` 命令会将此清单作为 `my_manifest` 进行引用。

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

示例 8 将派生的清单脚本添加到安装服务

使用以下样例命令可将名为 `my_script` 的派生清单脚本添加到名为 `service_092910` 的现有安装服务。脚本的添加方式与清单相同。

示例 8 将派生的清单脚本添加到安装服务 （续）

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

有关如何创建派生的清单脚本的信息，请参见《安装 Oracle Solaris 11 系统》。

示例 9 替换安装服务的缺省 AI 清单

使用以下样例命令可将现有安装服务 `service_092910` 的缺省清单替换为已作为 `my_manifest` 添加到服务的定制清单。通过为 `create-manifest` 子命令指定 `-m my_manifest`，将此清单添加到了服务。

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

示例 10 列出安装服务

使用以下样例命令可列出本地服务器上的安装服务。

```
$ 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
```

示例 11 列出与安装服务关联的客户机

使用以下样例命令可列出本地服务器上特定安装服务的客户机。

```
$ 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
```

示例 12 列出与安装服务关联的清单

使用以下样例命令可列出与本地服务器上的特定安装服务关联的清单和脚本。

```
$ 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
```

示例 12 列出与安装服务关联的清单 （续）

此示例显示以下输出：

- 具有条件的非缺省清单 (manifest2)
- 具有条件的非缺省脚本 (sparc_setup)
- 具有被忽略的条件的缺省清单 (new_default)
- 由于没有条件而被标记为非活动的非缺省清单 (orig_default)

示例 13 列出配置文件

使用以下样例命令可列出本地服务器上的配置文件。

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

示例 14 将没有名称的定制 AI 清单添加到安装服务

使用以下样例命令可将 /export/my_manifest.xml 中的清单添加到 svc1（具有 MAC 地址等于 aa:bb:cc:dd:ee:ff 的条件）。

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

在此示例中，清单不包含名称属性，因此清单名称取自文件名。

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

示例 15 将具有定制名称的定制 AI 清单添加到安装服务

使用以下样例命令可将 /export/my_manifest.xml 中的清单添加到 svc1（具有 IPv4 范围从 10.0.2.100 到 10.0.2.199 的条件）。

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

在此示例中，清单名称取自 -m 选项。

```
$ installadm list -m -n svc1
Manifest       Criteria
--------       --------
```

示例 15 将具有定制名称的定制 AI 清单添加到安装服务 （续）

```
chosen_name    ipv4 = 10.0.2.100 - 10.0.2.199
```

示例 16 添加具有在清单中指定的名称的定制 AI 清单

使用以下样例命令可将 /export/manifest3.xml 中的清单添加到 svc1（具有内存为 2048 MB 或更大且体系结构为 i86pc 的条件）。

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

在此示例中，清单名称取自清单中 ai_instance 元素的 name 属性，如以下部分清单所示：

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

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

示例 17 将系统配置文件添加到安装服务

使用以下样例命令可将 /export/profile4.xml 中的配置文件添加到 svc1（具有为 myhost1、host3 或 host6 中任一主机名的条件）。

```
# 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
```

示例 18 添加所有客户机的系统配置文件

如果未指定条件，则配置文件将由使用指定安装服务的所有客户机使用。在以下示例中，创建的配置文件由使用 svc1 服务的所有客户机使用。

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

示例 19 添加具有替换标记的系统配置文件

配置文件可以使用替换标记，这些标记充当来自用户环境（请参见 environ(4)）或来自 create-profile 子命令的 -c 选项中指定的条件的定制客户机配置信息的占位符。使用替换标记可对任意数目的不同系统重用配置文件。在以下示例中，每个配置文件将与取自 -c 条件选项的 hostname 值一起存储：

```
# 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 文件包含以下行：

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

create-profile 命令进行替换，以便 myhost1_hostname 配置文件包含以下行：

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

使用相同的 hostname.xml 输入文件，myhost2_hostname 配置文件包含以下行：

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

替换标记 {{AI_HOSTNAME}} 已替换为每个 create-profile 调用的不同值，因为使用了 hostname 条件并将其替换到了配置文件中。有关将替换标记与配置文件一起使用的更多信息，请参见《安装 Oracle Solaris 11 系统》。

示例 20 将条件添加到现有清单

使用以下样例命令可将内存为 4096 MB 或更大的条件附加到 svc1 的 manifest2 的条件。

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

示例 21 替换现有清单的条件

使用以下样例命令可将 svc1 的 manifest2 的条件替换为在文件 /tmp/criteria.xml 中指定的条件。

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

有关条件 XML 文件的内容的信息，请参见《安装 Oracle Solaris 11 系统》。

示例 22 验证正在开发的配置文件

使用以下样例命令可验证存储在文件 `myprofdir/myprofile.xml` 和 `herprofdir/herprofile.xml` 中、正在开发的配置文件。

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

示例 23 导出配置文件的内容

使用以下样例命令可导出 `svc1` 服务中的配置文件 `myprofile.xml`。

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

示例 24 替换现有 AI 清单的内容

使用以下样例命令可用文件 `/home/admin/new_spec.xml` 中清单的内容更新服务 `svc2` 中具有清单名称或 AI 实例名称 `spec` 的清单。

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

示例 25 导出并更新现有的 AI 清单

使用以下样例命令可导出服务 `svc2` 中名为 `spec` 的现有清单的数据，然后用修改后的内容更新清单。

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

对 `/home/admin/spec.xml` 进行更改。

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

退出状态

将返回以下退出值：

0 该命令已成功处理。

0 出现错误。

属性

有关下列属性的说明，请参见 `attributes(5)`：

属性类型	属性值
可用性	`install/installadm`
接口稳定性	Uncommitted（未确定）

另请参见

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

《安装 Oracle Solaris 11 系统》中的第 III 部分, “使用安装服务器安装”

《从 Oracle Solaris 10 JumpStart 转换至 Oracle Solaris 11 自动化安装程序》

引用名

js2ai – 转换 JumpStart 规则和配置文件以用于自动化安装程序 (Automated Installer, AI)。

用法概要

```
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
```

描述

`js2ai` 是一个实用程序，用于将 Oracle Solaris 10 JumpStart `rules`、配置文件和 `sysctg` 配置文件转换为与自动化安装程序 (Automated Installer, AI) 兼容的格式。此实用程序“尽力”将那些可以转换的 JumpStart 关键字转换为 AI 上下文。尽管此转换过程并不为 JumpStart 创建完整的一对一对等项，但是它确实可以提供可用作模板的 AI 清单和系统配置文件条目，之后可借助此模板根据从 JumpStart 配置文件收集的信息创建完整的 AI 配置设置。

使用 `js2ai` 可以实现以下目的：

- 处理当前工作目录中的 `rules` 文件和关联的配置文件。
- 处理指定目录中的 `rules` 文件和关联的配置文件。
- 处理特定的配置文件或 `sysidcfg` 文件。
- 将生成的输出文件定向到特定的目录。有关 `js2ai` 输出文件的更多信息，请参见“示例”和“文件”部分。

转换规则关键字

表 1 JumpStart 规则关键字转换

JumpStart 规则关键字	AI 条件关键字
`arch`	`cpu`
`hostaddress`	`ipv4`
`karch`	`arch`
`memsize`	`mem`
`model`	`platform`
`network`	`ipv4`

不受 `js2ai` 支持的 JumpStart 规则关键字：

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

转换配置文件关键字

表 2 JumpStart 配置文件关键字

JumpStart 配置文件关键字	附注
`boot_device`	如果事先未通过 `root_device` 关键字进行设置，`rootdisk` 将设置为指定的设备。
`fdisk`	`disk_name` 的值必须为设备。不支持 `all` 设备。`fdisk` 类型必须为 `solaris`。不支持大小为 0 或 `delete`。 如果 `partitioning` 为 `default` 且尚未设置 `rootdisk`，则遇到的第一个 `fdisk solaris` 分区将用作 `rootdisk`。
`filesys`	当指定的挂载点为 / 或 `swap` 时，支持本地文件系统和镜像文件系统。 不执行大小验证。可能需要调整生成的 AI 清单中指定的大小才能使用此清单实现成功安装。
`install_type`	仅支持值 `initial_install`。
`locale`（**语言环境**）	不执行转换。请确保指定的语言环境在 Oracle Solaris 11 中受支持。
`package`（**软件包**）	尝试将指定的软件包转换为其 Oracle Solaris 11 等效项。不支持指定软件包的位置。软件包查找可能会花费大量的时间。如果配置文件包含的软件包列表很长，则在转换过程中可能需要使用 `--local` 标志。
`partitioning`	支持的类型是 `default` 和 `explicit`。与 JumpStart 不同，在指定 `partitioning default` 时，仅使用 `js2ai` 可识别的磁盘。如果任何关键字中都未指定磁盘，则生成的配置文件将通知 AI 选择要使用的磁盘。
`pool`（**池**）	如果在配置文件中指定了池，则使用指定的设备创建 ZFS 根池。在确定用于 ZFS 根池的设备时，`pool` 关键字将取代所有其他关键字。 不执行对池大小、交换大小或转储大小的验证。可能需要调整生成的 AI 清单中的这些大小才能使用此清单实现成功安装。
`root_device`	`rootdisk` 设置为指定的设备。
`system_type`	仅支持值 `standalone`。
`usedisk`	在转换期间，可以使用指定的设备解析 `any` 或 `rootdisk` 设备。在未镜像 ZFS 根池时，不用于此目的任何指定设备都将添加到该池。

不受 `js2ai` 支持的 JumpStart 配置文件关键字：

```
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
```

如何在配置文件转换期间确定系统的根磁盘

由于在配置文件转换期间 `js2ai` 无法访问配置文件所引用的实际系统，因此 `js2ai` 将在转换期间使用尽可能与 JumpStart 匹配的进程来尝试确定根磁盘。

`js2ai` 工具执行以下步骤以确定用于根磁盘的设备。

阶段	操作
1	如果在配置文件中指定了 `root_device` 关键字，则 `js2ai` 会将 `rootdisk` 设置为分片所在的设备。
2	如果未设置 `rootdisk` 但在配置文件中指定了 `boot_device` 关键字，则 `js2ai` 将 `rootdisk` 设置为引导设备。
3	如果未设置 `rootdisk` 但指定了 `partitioning default`，且遇到了 `solarisfdisk` 条目，则 `js2ai` 将 `rootdisk` 设置为指定的 `disk_name`。
4	如果未设置 `rootdisk` 但在配置文件中指定了 `filesys` *cwtxdysz size* `/` 条目，则 `js2ai` 将 `rootdisk` 设置为该条目中指定的 *cwtxdysz* 磁盘。
5	如果未设置 `rootdisk` 但在配置文件中指定了 `usedisk` *disk_name* 条目，则 `js2ai` 将 `rootdisk` 设置为该条目中指定的 *disk_name* 磁盘。
6	如果未设置 `rootdisk` 但在配置文件中遇到以下规范（其中 *size* 不为 0 或 `delete`，*disk_name* 不为 `all`），则 `rootdisk` 将设置为该 *disk_name*。 `fdisk` *disk_name* `solaris` *size*
7	如果未设置 `rootdisk`，则将设备指定为 `rootdisk` 的任何实例都会生成转换错误。

如何在配置文件转换期间转换 any 设备

`js2ai` 工具执行以下步骤以确定在指定了 `any` 关键字时要使用的设备。

阶段	操作
1	如果指定了 `any` 设备且指定了关键字操作（非镜像池或具有 `/` 挂载点的 `filesys`），则在设置了 `rootdisk` 时 `any` 设备将设置为 `rootdisk`。
2	如果 `any` 设备尚未转换且配置文件中存在 `usedisk` 语句，则 `any` 设备将设置为由 `usedisk` 语句指定的设备。
3	如果 `any` 设备尚未转换且指定了 `any` 设备的操作导致创建 ZFS 根池，则 AI 将选择该设备。在指定镜像池时，这一点并不适用。

如何在配置文件转换期间确定 ZFS 根池

`js2ai` 工具执行以下步骤以确定用于 ZFS 根池的设备。在确定 ZFS 根池后，如果遇到的后续定义与已经确定的 ZFS 根池相冲突，则它们将标记为错误。

阶段	操作
1	如果配置文件指定了 pool 关键字，则 js2ai 将 ZFS 根池设置为由 pool 关键字指定的设备。
2	如果尚未确定 ZFS 根池但配置文件指定了具有 / 挂载点的 filesys，则使用指定设备创建 ZFS 根池。
3	如果尚未确定 ZFS 根池但已处理配置文件中的所有关键字，且设置了 rootdisk，则使用 rootdisk 设备创建 ZFS 根池。
4	如果尚未确定 ZFS 根池且分区类型为 default，则 AI 将选择要用于 ZFS 根池的设备。
5	如果尚未确定 ZFS 根池且在处理期间未出现错误，则 AI 将选择要用于 ZFS 根池的设备。
6	如果 ZFS 根池不是镜像池，且指定的一个或多个 usedisk 设备尚未用于 rootdisk 或 any 设备转换，则这些磁盘将添加到 ZFS 根池。

转换 sysidcfg 关键字

表 3 JumpStart sysidcfg 关键字

sysidcfg 关键字	附注
keyboard	不执行转换。请确保在 Oracle Solaris 11 中支持 sysidcfg 文件中指定的键盘。
name_service	支持的值为 None、DNS、NIS 和 LDAP。NIS+ 名称服务将转换为 NIS。
network_interface	仅支持单个接口。对 PRIMARY 的支持有限。仅处理在 sysidcfg 文件中遇到的第一个接口。
root_password	不需要进行转换。
security_policy	支持的值如下： None
service_profile	支持的值如下： limited_net
system_locale	不执行转换。请确保在 Oracle Solaris 11 中支持 sysidcfg 文件中指定的语言环境。
terminal	不执行转换。请确保在 Oracle Solaris 11 中支持 sysidcfg 文件中指定的终端类型。
timeserver	支持的值如下： localhost
timezone	不需要进行转换。

不受 js2ai 支持的 JumpStart sysidcfg 关键字：

nfs4_domain

选项

js2ai 命令具有以下选项。在“示例”部分中对这些选项的作用进行了说明。

-h， --help
显示用法帮助消息。

`--version`
显示 `js2ai` 实用程序的版本号。

`-d` *jumpstart_dir*, `- -dir` *jumpstart_dir*
指定 `rules` 以及配置文件或 `sysidcfg` 文件的位置。

`-D` *destination_dir*, `- -dest` *destination_dir*
指定输出文件的位置。

`-l`, `--local`
在 JumpStart 配置文件中搜索 `package` 关键字值的映像包管理系统 (Image Packaging System, IPS) 等效项时，会搜索在主机系统上安装的 IPS 软件包而不是 IPS 软件包系统信息库中的软件包。

`-p` *profile_name*, `- -profile` *profile_name*
转换指定的 JumpStart 配置文件并为所处理的配置文件生成清单。在这种情况下，不需要或不生成条件文件。

`-r`, `--rule`
转换规则和关联的配置文件并为所处理的每个配置文件生成清单。

`-s`, `--sysidcfg`
处理 `sysidcfg` 文件并将结果输出到 `sc_profile.xml`。

`-S`, `--skip`
跳过验证。

`-v`，`--verbose`
提供有关在处理期间发生的操作的详细信息。

`-V` *filename*
验证指定的 AI 清单文件或 SMF 系统配置文件。不支持 AI 条件验证。

错误报告

在转换期间出现一个或多个错误时，`js2ai` 工具将生成错误报告。

```
# 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
```

对于 `js2ai` 在其中遇到错误的每个文件，该报告中都包含一个条目。要生成错误报告（即使未出现错误），请指定 `- v` 或 `--verbose`。

该报告将告知您在哪些文件中出现了哪种类型的错误。定义了以下五种错误类型：警告、处理错误、不支持的项、转换错误和验证错误。

警告
不需要更正这些消息中的项。例如，您可能收到一条警告消息，指出未提供诸如主机名或超级用户口令之类的信息，将使用缺省值。

处理错误
这些错误表示出现了阻止 `js2ai` 处理文件或文件中某行的问题。当文件存在语法错误时，通常就会发生处理错误。

不支持的项
这些项是指 `js2ai` 不支持的行。更改与关键字关联的值可能会消除此错误。

转换错误
这些错误表示出现了阻止 `js2ai` 处理行的情况。应该手动更正这些错误，或者从文件中删除有问题的行。

验证错误
这些错误是指根据 AI 所用的架构定义验证生成的清单时出现的错误。必须先更正这些错误，AI 才能使用清单。

`js2ai.log` 文件指出了在哪个行上出现了哪个错误。

```
# 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
```

如果出现清单验证错误，`js2ai.log` 文件将提供一个指向包含验证错误的日志文件的指针，如以下示例所示：

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

转换策略 **规则和配置文件转换的建议策略**

不存在 JumpStart 和 AI 之间的一对一转换。以下步骤提供了执行转换的一般过程。

1. `js2ai` 实用程序尝试标记它遇到的任何错误，但是 `js2ai` 假定所转换的规则、配置文件和 `sysidcfg` 文件是有效的。
2. 将 `rules`、配置文件和 `sysfg` 配置文件的 JumpStart 配置目录复制到已安装 `install/installadm` 软件包的 Oracle Solaris 11 系统。
3. 在步骤 2 中复制到 Oracle Solaris 11 系统的 JumpStart 配置目录中，运行 `js2ai` 转换工具。

   ```
   # js2ai -rS
   ```

此命令将对 rules 文件和 rules 文件所引用的配置文件执行转换操作。将根据 AI 客户机置备清单 /usr/share/auto_install/manifest/default.xml 处理 rules 文件中引用的每个配置文件。此步骤将为 JumpStart rules 文件中指定的每个配置文件创建一个名为 AI_*profile_name* 的目录。AI_*profile_name* 目录包含已转换配置文件的一个或多个 AI 清单，其格式为 *profile_name*${arch}.xml。有关更多信息，请参见“文件”部分。

-S 选项用于跳过验证序列。验证在步骤 5 中执行。

4. 如果输出了消息“成功完成转换”，请跳到步骤 5。否则，请检查 js2ai.log 文件并按以下步骤操作：
   a. 更正任何处理错误。
   b. 从 rules 和配置文件中删除作为不支持的项列出的任何行。
   c. 检查转换错误并更正错误（如果可能）。否则，请删除导致错误的行。
   d. 检查任何警告消息并确保不需要进行更正。
   e. 重复步骤 3，直到不再报告处理错误、不支持的项和转换错误。
5. 不使用 -S 选项重新运行 js2ai。

   ```
   # js2ai -r
   ```

   如果已处理的任何配置文件出现任何验证错误，则必须手动更正生成的 AI 清单。检查 js2ai.log 文件以了解有关故障的详细信息。有关 AI 清单的信息，请参见 AI 文档。
6. 转换与此 JumpStart 配置关联的任何 sysidcfg 文件。

   对于每个 sysidcfg 文件，执行以下命令：

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

   对于所处理的每个 sysidcfg 文件，此步骤将在调用 js2ai 命令的目录中创建一个名为 sc_profile.xml 的 AI 系统配置文件。可使用 -D 选项为 sc_profile.xml 文件指定其他目录。
7. 如果输出了消息“成功完成转换”，请跳到步骤 8。否则，请检查 js2ai.log 文件并按以下步骤操作：
   a. 更正任何处理错误。
   b. 从 sysidcfg 文件中删除作为不支持的项列出的任何行。
   c. 检查转换错误并更正错误（如果可能）。否则，请删除导致错误的行。
   d. 检查任何警告消息并确保不需要进行更正。
   e. 重复步骤 6，直到不再报告处理错误、不支持的项和转换错误。
8. 不使用 -S 选项重新运行 js2ai。

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

   如果处理的任何 sysidcfg 文件出现任何验证错误，则必须手动更正生成的 AI 系统配置文件。检查 js2ai.log 文件以了解有关故障的详细信息。有关系统配置文件的信息，请参见 AI 文档。

9. `js2ai` 转换过程已完成。对生成的条件、AI 清单和系统配置文件执行手动验证。Oracle Solaris 11 安装的磁盘空间要求与 Oracle Solaris 10 安装的磁盘空间要求不同。请确保在 AI 清单中分配的磁盘空间满足 Oracle Solaris 11 的要求。
10. 配置 AI 以使用新生成的文件。将新生成的条件、AI 清单和系统配置文件添加到现有的 AI 安装服务。

    使用带有 `create-manifest` 子命令的 `installadm` 命令添加每个 AI 清单以及用于选择该清单的条件。每个客户机只能使用一个 AI 清单。

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

    使用 `create-profile` 子命令添加每个配置文件以及用于选择该配置文件的条件。每个客户机可以使用一个或多个系统配置文件。

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

    有关配置 AI 安装服务的信息，请参见 AI 文档和 `installadm`(1M) 手册页。

示例

示例 1 处理 JumpStart 配置

以下命令处理当前目录中的 JumpStart 规则和配置文件。输出也放置在此目录中。

```
# js2ai -r
```

示例 2 处理特定的 JumpStart 目录

以下命令将处理指定目录中的 JumpStart 规则和配置文件，并将输出文件放置在同一目录中。

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

有关输出文件的更多信息，请参见示例 4 和“文件”部分。

示例 3 处理特定的 JumpStart 目录和单独的目标目录中的配置文件

以下命令处理 `/export/jumpstart` 目录中的 JumpStart `rules` 和配置文件，并将输出文件放置在 `/export/output` 中。

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

示例 4 指定的规则及其配置文件的示例输入和生成的输出

规则 :

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

配置文件:

示例 4 指定的规则及其配置文件的示例输入和生成的输出 （续）

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

转换命令：

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

输出文件：

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

将创建两个清单，一个用于 SPARC，另一个用于 x86，即使 `rules` 文件将 CPU 类型指定为 SPARC 也是如此。在转换过程中，将彼此独立地处理规则和配置文件。

示例 5 将生成的文件添加到 AI 安装服务

此示例使用在示例 4 中生成的文件将清单和条件添加到现有的服务。

文件:

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

`installadm` 命令：

```
# 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
```

示例 6 处理 sysidcfg 文件

以下命令处理当前目录中的 `sysidcfg` 文件，并将生成的 SMF 系统配置文件作为 `sc_profile.xml` 输出到同一目录中。

```
# js2ai -s
```

退出状态

将返回以下退出值：

0 所有文件都已成功处理。

0 出现错误。

文件

output_directory`/AI_${`*profile_name*`}`
包含已转换为与配置文件关联的新 AI 语法的所有对应文件的目录。

output_directory`/AI_${`*profile_name*`}.${`*arch*`}.xml`
作为转换配置文件的结果而创建的清单文件。 `${`*arch*`}` 可以是以下三个值之一：`sparc`、`x86` 或 `generic`。格式为 `${`*profile_name*`}.generic.xml` 的清单文件可以同时用于安装 x86 和 SPARC 系统。

output_directory/AI_${ *profile_name*}/criteria-*rule_number*.xml
生成的 criteria-*rule_number*.xml 文件与 rules 文件中的规则相对应，*rule_number* 是基于其在 rules 文件中位置的规则编号。然后可以将此条件文件与 installadm 命令的 -C 选项一起使用。

由于多个规则可以指定同一配置文件，因此每个目录中可以存在多个条件文件，但是每个输出目录中应仅存在 ${*profile_name*}.${*arch*}.xml 文件的一个实例。

注 – 如果使用了 -p 选项，则不为所处理的配置文件生成条件文件。仅当与 -r 选项一起使用时才生成条件文件。

output_directory/js2ai.err
此文件包含在处理期间出现的意外情况的栈跟踪。通常不创建此文件。

output_directory/js2ai.log
此文件包含所处理文件以及在处理期间发现的任何错误的日志。

output_directory/sc_profile.xml
此文件是使用 -s 选项转换 sysidcfg 文件时生成的 SMF 系统配置文件。

属性

有关下列属性的说明，请参见 attributes(5)：

属性类型	属性值
可用性	install/js2ai
接口稳定性	Uncommitted（未确定）

另请参见

installadm(1M)、pkg(1)

《从 Oracle Solaris 10 JumpStart 转换至 Oracle Solaris 11 自动化安装程序》

《安装 Oracle Solaris 11 系统》中的第 III 部分, “使用安装服务器安装”

参 考 文 档

文件格式

引用名 ai_manifest – 自动化安装清单文件格式

用法概要 `/usr/share/install/ai.dtd.1`

描述 自动化安装程序 (Automated Installer, AI) 为 Oracle Solaris 提供可定制的无人参与安装机制，并使用基于 XML 的文件格式作为安装参数的说明。此安装参数文件称为 AI 清单。可以通过各种方式（如磁盘布局和要在系统上安装的软件）定制安装。

AI 清单具有以下部分：

- 自动化安装设置。指定在安装期间使用的设置。
- 磁盘布局。指定安装的磁盘布局。
- 软件。指定要安装的软件包。
- 引导配置（仅限 x86）。指定如何配置 GRUB 引导菜单。
- 其他配置。指定要在系统上安装的其他配置组件。

下面将更详细地介绍这些部分。

要创建新的 AI 清单，请使用模板副本或来自相关安装服务映像的缺省清单。例如，如果安装服务映像位于 *imagepath*，则以下文件可用：

imagepath`/auto_install/manifest/default.xml`
此安装服务的原始缺省 AI 清单。

imagepath`/auto_install/manifest/ai_manifest.xml`
包含示例定制的带注释 AI 清单样例。

可以使用 `installadm export` 命令检索安装服务中已存在的任何清单的副本。

AI 清单还用于安装非全局区域（使用 `zoneadm install` 命令）。可以将 AI 清单文件传递给此命令以定制区域安装。在 AI 清单指定的内容中，只有一部分适用于安装非全局区域。下面的各部分中对这些指定内容进行了说明。

服务管理工具 (Service Management Facility, SMF) 配置文件对 AI 清单进行了补充。这些配置文件指定已安装系统的系统配置，如主机名、联网以及超级用户和初始用户的帐户设置。

有关安装服务、AI 清单和配置文件的更多信息，请参见 `installadm(1M)` 手册页和《安装 Oracle Solaris 11 系统》中的第 III 部分, “使用安装服务器安装”。有关配置文件格式的信息，请参见 `smf(5)`。

自动化安装设置 `ai_instance` 元素具有以下属性：

`name` 此清单实例的名称。

`http_proxy` 在安装期间用于访问远程文件的 HTTP 代理。例如，映像包管理系统 (Image Packaging System, IPS) 软件包系统信息库中的软件包即是在安装期间访问的远程文件。`http_proxy` 的值是 HTTP URI，如 `http://myproxy.mycompany.com:8080/`。

在安装非全局区域时此属性不适用；如果提供了此属性，它将被忽略。

`auto_reboot` 指定在安装后是否自动重新引导的标志。`auto_reboot` 的缺省值为 `false`。当 `auto_reboot` 为 `false` 时，安装将等待手动干预以重新引导。

当 `auto_reboot` 为 `true` 时，如果安装成功，则计算机将自动重新引导到新安装的引导环境。

在安装非全局区域时此属性不适用；如果提供了此属性，它将被忽略。

以下示例演示如何使用 `ai_instance` 元素：

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

磁盘布局

AI 支持各种磁盘指定方式，从完全自动化选择安装目标到对磁盘布局的细粒度控制。

`target` 元素指定磁盘布局。未指定 `target` 元素时缺省磁盘布局具有以下特征：

- 一个磁盘全部用于安装 Oracle Solaris OS。此磁盘通常为引导磁盘或第一个磁盘。
- 对于 x86，将会分配 `fdisk` 分区，该分区使用完整磁盘内容。有关 `fdisk` 分区的更多信息，请参见 fdisk(1M) 手册页。
- 将会分配单个分片 0，该分片为磁盘的完整大小（对于 SPARC）或 `fdisk` 分区的完整大小（对于 x86）。
- 将会创建单个根池，该根池使用整个分片 0。
- 如果空间可用，则会在根池中创建交换卷和转储卷。

`target` 元素具有以下结构：

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

```
    </logical>
</target>
```

通过 target 元素的子元素可以指定磁盘和逻辑布局。

在安装非全局区域时，磁盘的指定不适用；如果指定了磁盘，它们将被忽略。

有些磁盘布局元素具有 size 子元素。size 元素具有以下格式：

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

start_sector 值是一个数值，用于指定新分区或分片的所需开始扇区。如果省略了 *start_sector* 属性，则安装程序将搜索第一个足以包含指定 *size* 的位置。

size 的值是具有以下后缀之一的数值：

- s 或 sec：扇区
- b：字节
- k 或 kb：千字节 (2^{10})
- m 或 mb：兆字节 (2^{20})
- g 或 gb：千兆字节 (2^{30})
- t 或 tb：兆兆字节 (2^{40})
- p 或 pb：拍字节 (2^{50})
- e 或 eb：艾字节 (2^{60})
- z 或 zb：泽字节 (2^{70})

此部分的其余内容详细介绍了 disk 和 logical 元素。

安装位置

如果在客户机上未指定安装 Oracle Solaris OS 的位置，则 AI 将选择该客户机的缺省位置。

缺省的安装位置是每个客户机上找到的首个符合大小要求的磁盘。如果磁盘大小大于或等于所需的大小，那么安装程序将选择该磁盘作为安装位置。如果磁盘大小小于所需的大小，安装程序将检查下一磁盘。如果没有找到符合大小要求的磁盘，自动化安装在该客户机上将失败。/system/volatile/install_log 中的安装日志显示了该系统的磁盘选择过程的详细信息。

target 部分的 disk 部分指定安装位置。

在安装非全局区域时，磁盘的指定不适用；如果指定了磁盘，它们将被忽略。

可以使用以下类型的选择条件之一来选择磁盘：

- 第 1 组：确定性条件，如磁盘名称或 IP 地址。使用下面“目标设备名称”中所述的 <disk_name> 子元素，或者使用下面“ISCSI 目标设备”中所述的 <iscsi> 子元素。
- 第 2 组：非确定性条件，如磁盘大小或供应商。使用下面“目标设备属性”中所述的 <<disk_prop> 子元素。

- 第 3 组：关键字条件，如 boot_disk 关键字。使用下面“目标设备关键字”中所述的 <disk_keyword> 子元素。

只能从以上三组之一指定条件。如果使用第 2 组选择条件，则可以指定多个条件。例如，可以同时指定大小和供应商。如果使用第 1 组选择条件，则只能指定那些条件之一。

目标设备名称

使用 disk_name 元素可指定非 iSCSI 设备的目标设备名称。disk_name 元素具有以下属性：

name　name 属性指定目标设备的名称。

name_type　name_type 属性指定目标设备名称的类型。name_type 属性可以具有以下值之一：

ctd：控制器目标磁盘名称
这是 CTD 名称（如 c0t0d0）或 MPXIO 名称（如 c0t2000002037CD9F72d0）。运行 format(1M) 命令时，通常可看到此类型的名称。

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

如果省略 name_type 属性，则这是缺省的目标设备名称类型。

volid：卷标识符
这是卷标识符，可以通过 format (1M) 命令进行设置。

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

devpath：设备路径
这是相对于 /devices 目录的设备路径。

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

devid：设备标识符
这是设备标识符，可在带有 -iEn 选项的 iostat(1M) 命令的输出的 "Device Id" 中找到。

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

receptacle：插口标识符
这是 CRO（Chassis, Receptacle, Occupant，机箱、插口、插卡）配置中的插口值，可在带有 -o cR 选项的 croinfo(1M) 命令的输出中找到。

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

ISCSI 目标设备

使用 iscsi 元素可将 iSCSI 磁盘指定为安装目标。iscsi 元素具有以下属性：

source　　source 属性指定 iSCSI 配置数据的源。source 属性可以具有以下值：

manifest
此值是指此 AI 清单。如果没有为 source 属性指定值，则这是缺省值。

省略 source 属性或者 source 属性的值为 manifest 时，必须指定 target_lun 和 target_ip 属性。

dhcp
此值是指通过在 DHCP rootpath 参数中指定 iSCSI 信息获取该信息时使用 DHCP。

当 source 属性的值为 dhcp 时，不要指定任何其他 iscsi 属性。

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

target_name　　target_name 属性指定 iSCSI 目标的 iSCSI 限定名 (iSCSI Qualified Name, IQN) 或扩展唯一标识符 (Extended Unique Identifier, EUI)，如以下示例所示：

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

如果未提供 target_name 属性，则 AI 在 sendtargets 模式下使用 iscsiadm(1M)。

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

如果提供了 target_name 属性，则 AI 使用静态搜索。

```
<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　　如果 iSCSI 目标提供了多个 LUN，则通过为 target_lun 指定一个整数值来指定要使用哪个 LUN。LUN 编号从 0 开始。要指定第一个 LUN，请将 target_lun 值指定为 0。

如果仅提供了一个 LUN，则可以省略此属性。

target_port　　如果未指定，则使用 target_port 的缺省值 3260（iSCSI 标准端口）。通过此属性可以指定替代端口号。

target_ip　　此属性的值是服务器的 IP 地址。

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

目标设备属性

使用 disk_prop 元素可指定目标设备的属性。可以指定多个属性。AI 将尝试基于提供的条件来查找最佳匹配。

使用 disk_prop 元素的属性可指定目标属性。disk_prop 元素具有以下属性：

dev_type：设备类型
目标磁盘的类型。可能的值包括 SCSI、ATA 和 USB。此值不区分大小写。

dev_vendor：设备供应商
由 format(1M) 命令的 inquiry 菜单选项显示的供应商。

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

dev_chassis：设备机箱
CRO（Chassis, Receptacle, Occupant，机箱、插口、插卡）配置中的机箱值，可在带有 -o cA 选项的 croinfo(1M) 命令的输出中找到。

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

dev_size：设备大小
磁盘的最小大小。其值为带有大小单位的数值。

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

disk_prop 元素允许同时指定多个属性，以便进一步约束磁盘搜索。以下示例将磁盘选择限制为大小至少为 100 GB 的 Hitachi 驱动器。

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

目标设备关键字

可以使用 disk_keyword 元素将系统的引导磁盘指定为目标磁盘。

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

key 属性支持的唯一值是 boot_disk。

整个磁盘、分区和分片

对磁盘进行布局的最简单方法是，通过将 whole_disk 属性设置为 true 将整个磁盘用于安装。

对于更复杂的磁盘布局，可以指定分区（仅限 x86 系统）和分片。

disk 元素具有以下属性：

whole_disk　此属性的缺省值为 false。当 whole_disk 为 false 时，必须定义分区或分片。除非通过为分区或分片的 action 属性指定 delete 值来删除任何现有的分区或分片，否则将保留它们。

当 whole_disk 为 true 时，除非通过为分区或分片的 action 属性指定 preserve 值来保留任何现有的任何分区或分片，否则将删除它们。

以下示例指定将整个磁盘用于安装：

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

in_zpool　　in_zpool 属性将此磁盘链接到在 AI 清单的 logical 部分中定义的 ZFS 池。in_zpool 属性的值必须与对应 zpool 元素的 name 属性的值匹配。

如果在此处指定了 in_zpool 属性，则不要为任何从属分区或分片指定 in_zpool。

in_vdev　　in_vdev 属性将此磁盘链接到在 AI 清单的 logical 部分中定义的虚拟设备。in_vdev 属性的值必须与对应 vdev 元素的 name 属性的值匹配。

如果在此处指定了 in_vdev 属性，则不要为任何从属分区或分片指定 in_vdev。

分区

只能在安装到 x86 系统时指定分区。如果为 SPARC 系统指定了分区，则安装会失败。partition 元素具有以下属性：

name　　name 属性是 fdisk 分区编号。值 1、2、3 和 4 为主要分区。如果主要分区之一为扩展分区，则可以为逻辑分区指定值 5 到 32。

除非指定的 action 为 use_existing_solaris2，否则 name 属性是必需的。

action　　action 属性可以具有以下值：

create
　这是分区的缺省操作。create 操作通知安装程序创建具有指定名称的分区。如果已存在同名分区，则首先删除该现有分区。

delete
　delete 操作通知安装程序删除指定的分区。如果指定的分区不存在，则跳过 delete 操作，并输出一条警告消息。

preserve
　preserve 操作通知安装程序使指定的分区保持不变。如果在同一磁盘上的其他位置安装了其他操作系统，则通常使用此操作。

use_existing_solaris2
　use_existing_solaris2 操作通知安装程序使用现有的 Solaris2 分区。安装程序将搜索现有的 Solaris2 分区。

　指定了 use_existing_solaris2 时，会忽略 name 和 part_type 属性。

part_type　　part_type 是 fdisk 分区类型。缺省值为 191，这是 Solaris2 分区的分区类型。有关可能的分区类型的更多信息，请参见 fdisk(1M) 命令。

in_zpool　　in_zpool 属性将此分区链接到在 AI 清单的 logical 部分中定义的 ZFS 池。in_zpool 属性的值必须与对应 zpool 元素的 name 属性的值匹配。

如果指定了 in_zpool 属性，则不要为关联的 disk 元素或任何从属的 slice 元素指定 in_zpool。

in_vdev　in_vdev 属性将此分区链接到在 AI 清单的 logical 部分中定义的虚拟设备。in_vdev 属性的值必须与对应 vdev 元素的 name 属性的值匹配。

如果指定了 in_vdev 属性，则不要为关联的 disk 元素或任何从属的 slice 元素指定 in_vdev。

分区可以具有 size 子元素以指定分区的大小。有关如何使用 size 元素的详细信息，请参见“磁盘布局”部分的开头。

以下示例使用缺省属性值创建一个 10 GB Solaris2 分区：

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

如果未指定大小，则使用父元素的大小。

preserve、delete 和 use_existing_solaris2 操作不需要指定 size。

分片

对于 x86 系统，分片必须包含在分区定义内。

slice 元素具有以下属性：

name　name 属性是分片编号。其值可以为 0 到 7。

action　action 属性可以具有以下值：

- create　这是分片的缺省操作。create 操作通知安装程序创建具有指定名称的分片。如果已存在同名分片，则首先删除该现有分片。
- delete　delete 操作通知安装程序删除指定的分片。如果指定的分片不存在，则跳过 delete 操作，并输出一条警告消息。
- preserve　preserve 操作通知安装程序使指定的分片保持不变。当存在来自以前安装的数据时，通常使用此操作。

is_swap　此属性的缺省值为 false。当 is_swap 为 false 时，安装程序将在根池中创建一个交换卷。

当 is_swap 为 true 时，指定的分片将用作交换设备。当 is_swap 为 true 时，不要使用 in_zpool 或 in_vdev 属性。

`force` 此属性的缺省值为 `false`。

当 `force` 为 `true` 时，安装程序将忽略可能已在使用中的任何现有分片（例如，在现有的 ZFS 存储池中使用的分片），并继续对指定的分片执行指定的操作。

`in_zpool` `in_zpool` 属性将此分片链接到在 AI 清单的 `logical` 部分中定义的 ZFS 池。`in_zpool` 属性的值必须与对应 `zpool` 元素的 `name` 属性的值匹配。

如果指定了 `in_zpool` 属性，则不要为关联的 `partition` 或 `disk` 元素指定 `in_zpool`。

`in_vdev` `in_vdev` 属性将此分片链接到在 AI 清单的 `logical` 部分中定义的虚拟设备。`in_vdev` 属性的值必须与对应 `vdev` 元素的 `name` 属性的值匹配。

如果指定了 `in_vdev` 属性，则不要为关联的 `partition` 或 `disk` 元素指定 `in_vdev`。

分片可以具有 `size` 子元素以指定分片的大小。有关如何使用 `size` 元素的详细信息，请参见“磁盘布局”部分的开头。如果未指定大小，则使用父元素的大小。

以下示例为 SPARC 系统创建一个 20 GB 分片（使用缺省属性值）和一个 4 GB 交换分片：

```
<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>
```

以下示例是用于 x86 系统的相同示例：

```
<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>
```

交换和转储

通过将 slice 元素的 is_swap 属性设置为 true，可以显式定义交换分片，如上面的“分片”所示。

通过将 zvol 元素的 use 属性设置为 swap 或 dump，可以将池中的卷显式定义为交换卷或转储卷，如下面的“ZFS 卷”所示。

缺省情况下，如果空间可用，则自动创建交换卷和转储卷。

在内存较低的系统上，交换分片可能比交换卷更可取，因为卷会导致少量内存开销。

如果要显式指定交换或转储但不希望自动创建交换卷或转储卷，请将 logical 元素的以下属性设置为 true：

noswap　此属性的缺省值为 false。当 noswap 为 false 时，如果空间允许，则安装程序将在根池中自动创建交换卷。

当 noswap 为 true 时，不会自动创建交换卷。

nodump　此属性的缺省值为 false。当 nodump 为 false 时，如果空间允许，则安装程序将在根池中自动创建转储卷。

当 nodump 为 true 时，不会自动创建转储卷。

ZFS 存储池

使用 target 部分的 logical 部分可指定任意数目的 ZFS 存储池。

通过使用 logical 元素的 zpool 子元素，可以定义多个池。只有其中的一个池可以是根池。如果定义多个根池，则安装会失败。

如果 zpool 元素定义了一个根池，而在 AI 清单中未指定目标磁盘、分区或分片，则安装程序将按照上面的“安装位置”中所述选择一个目标。此选择将自动指定给根池。

如果在 AI 清单中指定了目标磁盘、分区或分片，则 zpool 必须与其中至少一个磁盘、分区或分片关联。要建立此关联，请使用 disk 元素、partition 元素或 slice 元素的 in_zpool 属性。

zpool 元素具有以下属性：

name　这是新池的名称。此值必须是可以传递给 zpool create 命令的名称。

此名称应该用作 disk 、 partition 或 slice 元素的 in_zpool 属性的值，以将该磁盘、分区或分片定义为 zpool 中的成员设备。

action　action 属性可以具有以下值：

create
　这是 zpool 的缺省操作。create 操作通知安装程序创建具有指定名称的池。

delete
　delete 操作通知安装程序删除指定的池。

preserve
: preserve 操作通知安装程序使指定的池保持不变。只能为非根池指定此操作。

  在以下情况下，action 属性的值必须为 preserve：

  - 任何从属 filesystem 的 action 属性值为 preserve。
  - 任何从属 zvol 的 action 属性值为 preserve。
  - 任何从属 zvol 的 action 属性值为 use_existing。

use_existing
: use_existing 操作通知安装程序安装到现有的根池。将保留任何现有的卷或文件系统（数据集）。

is_root
: 此属性的缺省值为 false。当 is_root 为 false 时，将定义一个数据池。

  当 is_root 为 true 时，将在指定的池中创建新引导环境。

mountpoint
: mountpoint 属性指定池的顶层文件系统的挂载点。缺省挂载点为 */poolname*。挂载点必须是绝对路径。

要对新池设置 ZFS 属性，请使用 pool_options 元素。同样，要对自动创建的 ZFS 数据集设置 ZFS 属性，请使用 dataset_options 元素。pool_options 和 dataset_options 元素都具有 option 子元素。每个 option 元素都具有 name 属性和 value 属性。使用这些名称/值对设置的属性所受的限制与 zpool(1M) 命令强制实施的限制相同。以下示例显示如何设置这些属性：

```
<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>
```

可以为池定义任意数目的虚拟设备冗余组（vdev 元素）、ZFS 数据集（filesystem 元素）或 ZFS 卷（zvol 元素）。可以为池指定引导环境（be 元素）。以下部分介绍 vdev、filesystem、zvol 和 be 元素。

虚拟设备冗余组

使用 vdev 元素可定义 zpool 的大小或结构。可以指定多个 vdev 元素，每个元素具有不同的冗余类型。

如果 zpool 包含多个 vdev 元素，则在使用 in_zpool 属性定义的任何 disk、partition 或 slice 元素上必须使用 in_vdev 属性。

如果 vdev 名称在整个 AI 清单中是唯一的，则可以在磁盘、分区或分片上省略 in_zpool 属性。

如果 zpool 仅包含一个 vdev 元素，则可以在 disk、partition 或 slice 上省略 in_vdev 属性。

vdev 元素具有以下属性：

name　这是新 vdev 的名称。

此名称应该用作 disk、partition 或 slice 元素的 in_vdev 属性的值，以将该磁盘、分区或分片定义为 vdev 中的成员设备。

redundancy　redundancy 属性可以具有以下值：

mirror
　这是缺省值。如果 redundancy 为 mirror 或者未指定，则将包含的所有设备视为彼此的镜像。

raidz、raidz1、raidz2、raidz3
　组中具有这些值之一的设备用于定义 RAIDZ 分组。

spare
　如果发生故障，则将此组中的设备视为热备件。

cache
　此组中的设备为池提供缓存。

log、logmirror
　此组中的设备用于日志记录。如果指定了 logmirror，则设备为镜像。

none
　当 redundancy 为 none 时，不定义冗余。如果此组中包括多个设备，则条带化这些设备。

可以将根池定义为仅具有以下配置之一：

- redundancy 类型为 none，包括一个设备。此配置不支持使用多个设备。
- redundancy 类型为 mirror，包括多个设备。

要将设备添加到 vdev，请使用 disk、partition 或 slice 元素的 in_zpool 和 in_vdev 属性。以下示例指定一个名为 rpool 的根池，在两个磁盘上进行镜像：

```
<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>
```

如果 in_zpool 或 in_vdev 属性引用的池或虚拟设备是明确的，则可以省略其中一个属性。

文件系统（数据集）

使用 filesystem 元素可定义 ZFS 池内的 ZFS 文件系统或数据集。

filesystem 元素具有以下属性：

name　这是新 filesystem（相对于 zpool）的名称。例如，如果在名为 rpool 的 zpool 中将 filesystem 命名为 export，则 ZFS 数据集名称为 rpool/export。

如果 filesystem 的 in_be 属性设置为 true，则此名称相对于引导环境的根数据集。

action　action 属性可以具有以下值：

- create　这是 filesystem 的缺省操作。create 操作通知安装程序创建具有指定名称的文件系统。
- delete　delete 操作通知安装程序删除指定的文件系统。
- preserve　preserve 操作通知安装程序使指定的文件系统保持不变。如果为 filesystem 指定了 preserve，则应该为关联的 zpool 指定 preserve。

mountpoint　mountpoint 属性指定新文件系统的挂载点。如果未指定挂载点，则文件系统从其父级继承挂载点。

in_be　此属性的缺省值为 false。当 in_be 为 false 时，将在所有引导环境之间共享新的数据集。

当 in_be 为 true 时，将在每个引导环境中创建此新数据集的单独副本。如果 in_be 为 true，则 name 属性值相对于引导环境的根数据集。

使用 options 子元素可对 filesystem 设置 ZFS 数据集属性。可以设置任何可编辑的 ZFS 文件系统属性。对 filesystem 使用 options 元素与对 zpool 使用 dataset_options 元素类似，如以下示例所示：

```
<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 会继承对父 filesystem 设置的任何属性，除非以不同方式显式设置该属性。这是 ZFS 文件系统的缺省行为。

ZFS 卷

使用 zvol 元素可定义 ZFS 池内的 ZFS 卷。zvol 通常用于交换或转储设备，但是它可以具有其他用途。

zvol 元素具有以下属性：

name　这是新 ZFS 卷的名称。

action　action 属性可以具有以下值：

- create　这是 zvol 的缺省操作。create 操作通知安装程序创建具有指定名称的 ZFS 卷。
- delete　delete 操作通知安装程序删除指定的卷。
- preserve　preserve 操作通知安装程序使指定的 zvol 保持不变。如果为 zvol 指定了 preserve，则应该为关联的 zpool 指定 preserve。
- use_existing　如果为交换或转储设备指定了此值，则会重用现有的卷。如果为 zvol 指定了 use_existing，则应该为关联的 zpool 指定 preserve。

use　use 属性可以具有以下值：

- none　这是缺省值。当 use 为 none 时，在安装期间将创建 zvol 但不使用它。
- swap　当 use 为 swap 时，将创建 zvol 并将其用作交换设备。在安装期间 zvol 也用作交换设备。
- dump　当 use 为 dump 时，将创建 zvol 并将其用作转储设备。在安装期间 zvol 也用作转储设备。

使用 size 子元素可指定 zvol 的大小。有关如何使用 size 元素的详细信息，请参见“磁盘布局”部分的开头。

使用 options 子元素可对 zvol 设置 ZFS 卷选项。对 zvol 使用 options 元素与对 zpool 使用 dataset_options 元素类似，如以下示例所示：

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

引导环境

使用 be 元素可指定在安装期间创建引导环境的方式。

be 元素具有一个属性：

name　这是由安装程序创建的新引导环境的名称。如果未指定 be 元素，则此引导环境的缺省名称是 solaris。

安装程序利用由引导环境子系统提供的自动命名功能。在安装到现有的目标区域中时（例如，在安装区域时），具有由 be 元素 name 属性指定的名称的引导环境可能已存在。如果指定的引导环境名称已存在，则此名称将用作生成新名称的基础。例如，如果未指定 be，且名为 solaris 的引导环境已存在，则将新的引导环境命名为 solaris-*n*，其中 *n* 是构成尚不存在的引导环境名称的、按计数顺序的第一个整数。

引导环境将被创建为 ZFS 数据集，并且可以对其设置 ZFS 属性。使用 options 子元素可对引导环境设置 ZFS 属性，如以下示例所示：

```
<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>
```

软件

software 元素指定要安装的软件。software 部分指定以下信息：

- 软件源的类型
- 源的位置
- 要安装或卸载的软件包的名称
- 要安装的可选软件组件
- 映像属性

- 访问 IPS 系统信息库所需的 SSL 密钥和证书

`software` 元素具有以下属性：

`name` 这是 `software` 实例的名称。此名称在此 AI 清单中的所有 `software` 实例之间必须唯一。

`type` 这是软件源的类型。

`type` 属性可以具有以下值之一。如果未指定 `type`，则缺省值为 `IPS`。

- `IPS`：IPS 软件包系统信息库
- `P5I`：IPS 软件包文件
- `SVR4`：SVR4 软件包
- `CPIO`：`cpio` 归档文件

`software` 元素具有以下结构：

```
<!-- 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 安装

如果未指定 type 属性，则缺省安装类型为 IPS。

对于 IPS 类型的安装，只能指定单个 source 元素。

使用 source 元素可指定要用于安装软件包的发布者。可以指定多个发布者。每个发布者必须具有至少一个源。每个发布者可以具有多个源和镜像。

在 AI 清单中定义发布者的顺序是在发布者中搜索要安装的 IPS 软件包的顺序以及在已安装系统中设置发布者的顺序。

安装非全局区域时，系统信息库由该区域使用。在 AI 清单中指定的任何发布者将按它们在 AI 清单中出现的顺序添加，在系统信息库提供的发布者之后。有关系统信息库的更多信息，请参见 pkg(1) 和 pkg.sysrepo(1m)。

以下示例指定多个发布者，其中一个发布者具有镜像以及源：

```
<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>
```

使用 software_data 元素可指定要安装或卸载的软件包。action 属性可以具有以下两个值之一：

`install` 安装在 name 子元素中指定的 IPS 软件包。如果未指定 action 属性，这将是缺省值。至少一个 software_data 元素必须具有安装操作。

`uninstall` 删除在 name 子元素中指定的 IPS 软件包。

IPS 安装不支持 action 属性的其他值。

对于其中的每个操作，可以在 name 元素中指定一个或多个软件包，如以下示例所示：

```
<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 安装

`.p5i` 文件是描述 IPS 发布者、软件包，还可能描述镜像的文件。

要指定将处理的一个或多个 `.p5i` 文件，请将这些文件作为 `publisher` 元素中的源提供，如以下示例所示：

```
<software type="P5I">
    <source>
        <publisher>
            <origin name="/somewhere/image1.p5i"/>
            <origin name="/somewhere/image2.p5i"/>
        </publisher>
    </source>
</software>
```

如果此 AI 清单也没有 IPS 类型的软件部分，请确保 `.p5i` 文件指定了源。

P5I 安装不支持指定要安装的软件包。因此，在 `P5I` 类型的 `software` 元素中不支持 `software_data` 元素。

SVR4 安装

对于 SVR4 传输，必须使用文件目录路径或 FILE URI 指定一个包含 SVR4 软件包子目录的目录或 SVR4 软件包数据流文件。也可以使用 HTTP URI 指定 SVR4 软件包数据流文件。

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

`software_data` 元素用于指定要执行的操作。`action` 属性可以具有以下两个值之一：

`install` 将文件从源复制到新的引导环境。如果未指定 `action` 属性，这将是缺省值。至少一个 `software_data` 元素必须具有安装操作。

`uninstall` 从新的引导环境中删除文件。

SVR4 安装不支持 `action` 属性的其他值。

对于其中的每个操作，可以在 `name` 元素中指定一个或多个软件包，如以下示例所示：

```
<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 安装

对于 CPIO 传输，必须指定源目录。在安装期间目标目录被设置为新引导环境的挂载点。

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

`software_data` 元素用于指定要执行的操作。`action` 属性可以具有以下值之一：

`install` 将文件从源复制到新的引导环境。如果未指定 `action` 属性，这将是缺省值。至少一个 `software_data` 元素必须具有安装操作。

使用 `name` 元素可指定要复制的文件或目录。在 `name` 元素中指定的路径相对于源。

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

`uninstall` 从新的引导环境中删除文件。

使用 `name` 元素可指定要删除的文件或目录。在 `name` 元素中指定的路径相对于目标。

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

可选软件组件和映像属性

使用 `destination` 元素和 `image` 子元素可指定以下信息：

- 要安装的可选软件组件
- 映像属性
- SSL 密钥和证书

`destination` 部分仅适用于 IPS 和 P5I 安装类型。`destination` 元素只能具有一个 `image` 子元素。

SSL 密钥和证书

使用 image 元素的属性可指定使用客户机 SSL 验证的发布者所需的 SSL 密钥和证书。此处指定的密钥和证书适用于在此 AI 清单中定义的第一个发布者。

ssl_key　　此属性映射到以下 pkg 命令：

pkg set-publisher -k *ssl_key*

ssl_key 属性的值为 *ssl_key*。有关 pkg set-publisher 命令的更多信息，请参见 pkg(1) 手册页。

ssl_cert　　此属性映射到以下 pkg 命令：

pkg set-publisher -c *ssl_cert*

ssl_cert 属性的值为 *ssl_cert*。

可选软件组件

使用 image 元素的 facet 子元素可指定要安装的可选软件组件。侧面不是单独的软件包，而是任何给定软件包（如语言环境）、文档和开发文件（如包含调试信息的文件）的可选组件。例如，可通过指定仅要安装一种或两种语言来节省空间。有关 IPS 侧面的更多信息，请参见 pkg(1) 手册页。

facet 元素具有一个布尔型 set 属性和一个为 IPS 侧面名称的值。

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

以下示例指定仅应安装软件包的德语和英语侧面。该示例首先指定不应安装任何语言环境，然后指定应该安装德语和英语语言环境。

```
<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>
```

映像属性

使用 image 元素的 property 子元素，可以为此安装创建的新映像指定 IPS 映像属性。

property 元素具有一个布尔型 val 属性和一个为属性名称的值。

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

有关可以设置哪些属性的信息，请参见 pkg(1) 手册页的“映像属性”部分。

引导配置（仅限 x86）

可以使用 AI 清单修改在已安装的系统上配置 GRUB 引导菜单的方式。

此部分不适用于区域安装，在安装非全局区域时将被忽略。

使用 `boot_mods` 元素和 `boot_entry` 子元素可修改 GRUB 引导菜单。

`boot_mods` 元素具有以下属性：

`title` `title` 属性的值是由此 `boot_mods` 元素的 `boot_entry` 子元素指定的引导条目的基本标题。此属性值将覆盖从 `/etc/release` 的第一行或安装介质自动生成的名称。

`timeout` `timeout` 属性的值是选择此 `boot_mods` 元素的缺省 `boot_entry` 之前等待的秒数。

在 SPARC 系统上只能设置 `title` 属性。对于 SPARC 系统，将忽略此部分中的所有其他设置。

使用 `boot_entry` 子元素可将一个或多个菜单项添加到引导菜单。这些菜单项是对安装程序自动生成的任何菜单项的补充。

`boot_entry` 元素具有以下属性：

`default_entry` 如果将此布尔值设置为 `true`，则此菜单项是引导时选定的缺省选项。此属性的缺省值为 `false`。

如果多个 `boot_entry` 元素已将 `default_entry` 设置为 `true`，则最后一个此类条目是引导时选定的缺省选项。

`insert_at` 可以将此属性设置为以下两个值之一：

- `end` 将条目放置在生成的引导菜单的结尾。这是缺省放置。
- `start` 将条目放置在生成的引导菜单的开头。

然后由以下子元素定义 `boot_entry` 菜单项：

`title_suffix` 此元素是必需的。此元素定义要添加到在 `boot_mods` 元素中指定的标题结尾的文本。

`kernel_args` 此元素是可选的。此元素是由引导装载程序传递到内核的一串值。

以下示例指定一个名为 "Boot Testing Default Boot Entry"（引导测试缺省引导项）的引导菜单项，该项是菜单上的最后一项，在 20 秒后将自动选定该项：

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

其他配置

configuration 元素支持非全局区域配置。在安装全局区域系统时，会使用在 AI 清单中指定的区域配置在安装全局区域后将非全局区域安装到系统上。

configuration 元素具有以下属性：

type　要安装的配置的类型。AI 支持的唯一类型为 zone。

name　为配置提供的名称。此名称在 AI 清单中的所有配置元素之间必须唯一。对于 zone 类型的配置，此名称还用作区域的 zonename。

source　AI 从中下载此配置元素的配置文件的位置。值可以为指定的 HTTP 或 FILE URI。对于 zone 类型的配置，此值应该指向从 zonecfg export 命令生成的区域配置文件。

指定以下内容可在安装客户机上安装 zone1：

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

有关配置和安装区域的更多信息，请参见《安装 Oracle Solaris 11 系统》中的第 12 章 “安装和配置区域”。

文件

/usr/share/auto_install/manifest/default.xml
　指定缺省的系统安装，不进行定制。在系统上提供的此 AI 清单仅供参考。要创建新的 AI 清单，请从相关的安装服务映像使用此文件的副本。有关从安装服务复制此文件的信息，请参见“说明”部分。

/usr/share/auto_install/manifest/zone_default.xml
　缺省区域安装，不进行定制。此文件由 zoneadm install 命令用作缺省清单以安装非全局区域。

/usr/share/auto_install/manifest/ai_manifest.xml
　模板 AI 清单，已注释掉详细信息。此文件提供可以执行的一些定制的示例。在系统上提供的此文件仅供参考。要创建新的 AI 清单，请从相关的安装服务映像使用此文件的副本。有关从安装服务复制此文件的信息，请参见“说明”部分。

属性

有关下列属性的说明，请参见 attributes(5)：

属性类型	属性值
可用性	system/install/auto-install/auto-install-common
接口稳定性	Uncommitted（未确定）

另请参见

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)

《安装 Oracle Solaris 11 系统》中的第 III 部分, “使用安装服务器安装”

引用名 dc_manifest – 定制分发构造器的清单文件

用法概要 **下列清单文件可用于构建各种 Oracle Solaris 映像。这些清单文件包含在分发构造器软件包中。**

要构建 x86 Oracle Solaris LiveCD 映像：
/usr/share/distro_const/dc_livecd.xml

要构建 x86 automated installation 映像：
/usr/share/distro_const/dc_ai_x86.xml

要构建 SPARC automated installation 映像：
/usr/share/distro_const/dc_ai_sparc.xml

要构建 x86 text installation 映像：
/usr/share/distro_const/dc_text_x86.xml

要构建 SPARC text installation 映像：
/usr/share/distro_const/dc_text_sparc.xml

描述 分发构造器 (distribution constructor, DC) 可用于构建 Oracle Solaris 安装映像。

DC XML 清单文件用作分发构造器的输入。这些清单定义分发构造器构建的映像。按照上述列表，可以使用不同的清单构建不同种类的映像。

使用 distro_const 命令构建映像，在命令中引用清单文件。

如果要定制映像规范，请复制清单文件，定制该副本，然后在构建映像时将该副本用作 distro_const 命令的输入。

至少，您需要编辑清单中的目标元素，以指定可构造映像的构建区域的位置。并且，需要编辑软件名称元素来指定发布者和系统信息库的位置（其中包含构建映像所需的软件包）。

清单部分 清单包括下列主要元素。

注 – 下列提供的缺省元素和属性因使用的清单而异。

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

该元素为计划构建的映像提供缺省名称 Oracle_Solaris_Text_X86。您可以使用该名称，或为您的映像提供唯一的名称。

如果您要构建一系列映像，并保留增量式映像，可以将时间戳变量更改为 "true"，然后时间戳将自动附加到每个映像的名称。

如果需要指定 HTTP 代理，取消注释包含代理变量的 distro 名称元素，然后输入代理位置。例如，

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

<boot_mods>

该元素指定要应用于映像的引导菜单修改。

在下列示例中，一个标题为 “myentry” 的专门的引导菜单将应用于映像。超时属性指定自动激活缺省引导条目之前的时间。

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

可以通过为每个新条目添加新的 boot_entry 元素来添加各个引导菜单条目。各条目被按顺序添加到引导菜单，该顺序基于每个引导条目的 insert_at 属性值 "start" 或 "end"。

注 – 在现有 "with magnifier" 条目之前添加新条目。

请参见下列单个 boot_entry 元素的示例。

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

因为标题子元素未包含在本示例中，因此使用缺省标题。缺省标题为首行 `/etc/release`。

title_suffix 是必需的子元素，它是将附加到条目标题的文本字符串。可选的 kernel_args 子元素将内核参数传递到引导装载程序。

boot_entry 元素的可选属性包括 :

default_entry	将该属性设置为 "true" 可使该引导条目成为缺省值。如果多个条目被设置为 "true"，则如此定义的最后一个条目将覆盖前面的条目。
insert_at	将值设置为 "start" 或 "end" 以指示相对于其他引导条目的插入点。

<target>

该元素定义将用于该构建的 ZFS 构建数据集。该数据集是将创建映像的区域。您必须输入有效的数据集位置。

请参见以下示例。

```
<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">

此部分指定分发构造器可以下载软件包并用于构建映像的位置。

映像包管理系统 (Image Packaging System, IPS) 发布者提供一个或多个软件包系统信息库的软件包。

在本部分的源元素中，编辑发布者名称和源名称元素来指定要使用的发布者以及软件包系统信息库所处的位置。可以列出多个发布者。当分发构造器尝试定位要安装的软件包时，按在此处列出的顺序搜索发布者。

如果需要指定发布者的镜像，请取消注释并编辑镜像名称元素。

请参见以下示例。

```
<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 – 该元素还包括一个目标标记，该标记指定在构建映像期间将使用的数据挂载点。建议不要更改目标属性。

<software_data action="install">

带有 install 属性的该 software_data 元素根据您使用的清单，列出为了构建特定类型的映像而需安装的一组软件包。例如，dc_livecd.xml清单列出构建 LiveCD 映像所需的软件包。

每个名称标记列出一个软件包名称或包含许多软件包的软件包组的名称。

```
<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>
```

如果您有要添加到映像的软件包，通过为每个软件包添加名称标记来附加软件包名称。

缺省情况下，将安装指定系统信息库中可用的最新软件包版本。如果需要其他版本，使用以下格式将版本号附加到 'entire' 引用：

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

使用以下命令检查可用的版本。

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

注 – 请勿删除 'entire' 条目。'entire' 是用于管理多个软件包的集合。

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

带有 uninstall 属性的 software_data 元素可用于卸载单个软件包或卸载软件包组定义。

在下列示例中，'server_install' 是包含许多单个软件包的软件包组的名称。

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

可以卸载软件包组。卸载软件包组实际只卸载了组定义。不会卸载之前作为该组的一部分安装的单个软件包。但是，可以在不卸载软件包组的情况下卸载这些单独的软件包。保留软件包组可用于正在进行的引用。

还可以使用名称标记来卸载单个软件包。在卸载部分的末尾附加将卸载的其他软件包。

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

当使用映像（使用分发构造器创建）安装系统后，该元素将影响该系统。

在源元素中，使用发布者名称和可选镜像名称标记来指定已安装系统可访问将下载并安装的其他软件包的位置。请参见以下示例。

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

<software name="ba-init">

该元素为构建的映像列出在引导归档文件中将安装或卸载的文件和目录。相关信息请参见清单文件中的注释。

注意 – 修改引导归档文件内容会导致系统不可引导。

<execution stop_on_error="true">

清单中的执行元素列出在映像构建过程中执行的一系列检查点。按此部分中的排列顺序执行检查点。构建缺省安装映像所需的缺省检查点包含在每个清单中。

每个检查点名称标记都包括用于指定检查点脚本所在位置的 mod 路径属性。

使用 `distro_const`(1M) 命令选项可控制在特定的检查点停止和重新启动构建过程。

某些检查点标记包括带有提供的缺省值的参数。有关详细信息，请参见清单注释。

如果创建将在构建映像期间使用的定制脚本，必须添加指向该脚本位置的检查点名称标记。

有关如何添加新的检查点名称标记以指向定制脚本的信息，请参见以下示例。

用户创建定制脚本 `/tmp/myscript.sh` 以在构建过程中运行（在缺省 transfer-ips-checkpoint 之后）。

为了指向新的脚本，在 transfer-ips-checkpoint 名称之后将下列标记添加到清单。

```
<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>
```

其中 'arg1' 和 'arg2' 是脚本使用的可选参数。

'{PKG_IMAGE_PATH}' 或 '{BOOT_ARCHIVE}' 的值在执行过程中由 `distro_const` 分别替换为 `<ZFS Dataset>/build_data/pkg_image` 和 `<ZFS Dataset>/build_data/boot_archive`。

注 – 可以在 DC 清单中指定多个定制脚本检查点。每个检查点必须具有唯一的名称。

<configuration name="pre-pkg-img-mod" type="sysconf" source="/etc/svc/profile/generic_limited_net.xml">

清单中的配置名称元素列出在映像构建过程中应用到介质的 SMF 服务配置文件。这些 SMF 服务指定将在引导的介质上运行或不运行的服务。按在该元素中指定的顺序应用配置文件。

此元素很少会被修改。

属性

有关下列属性的说明，请参见 attributes(5)：

属性类型	属性值
可用性	install/distribution-constructor package
接口稳定性	Under Development（开发中）

另请参见

`distro_const`(1M)、`pkg`(1)•

在 OTN 文档库中为当前版本创建定制 Oracle Solaris 11 安装映像。