Customizing Automated Installations With Manifests and Profiles

Part No: E89348
November 2020

ORACLE

Customizing Automated Installations With Manifests and Profiles

Part No: E89348

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Référence: E89348

Restrictions de licence/Avis d'exclusion de responsabilité en cas de dommage indirect et/ou consécutif

Ce logiciel et la documentation qui l'accompagne sont protégés par les lois sur la propriété intellectuelle. Ils sont concédés sous licence et soumis à des restrictions d'utilisation et de divulgation. Sauf stipulation expresse de votre contrat de licence ou de la loi, vous ne pouvez pas copier, reproduire, traduire, diffuser, modifier, accorder de licence, transmettre, distribuer, exposer, exécuter, publier ou afficher le logiciel, même partiellement, sous quelque forme et par quelque procédé que ce soit. Par ailleurs, il est interdit de procéder à toute ingénierie inverse du logiciel, de le désassembler ou de le décompiler, excepté à des fins d'interopérabilité avec des logiciels tiers ou tel que prescrit par la loi.

Exonération de garantie

Les informations fournies dans ce document sont susceptibles de modification sans préavis. Par ailleurs, Oracle Corporation ne garantit pas qu'elles soient exemptes d'erreurs et vous invite, le cas échéant, à lui en faire part par écrit.

Avis sur la limitation des droits

Si ce logiciel, ou la documentation qui l'accompagne, est livré sous licence au Gouvernement des Etats-Unis, ou à quiconque qui aurait souscrit la licence de ce logiciel pour le compte du Gouvernement des Etats-Unis, la notice suivante s'applique :

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software" or "commercial computer software documentation" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

Avis sur les applications dangereuses

Ce logiciel ou matériel a été développé pour un usage général dans le cadre d'applications de gestion des informations. Ce logiciel ou matériel n'est pas conçu ni n'est destiné à être utilisé dans des applications à risque, notamment dans des applications pouvant causer un risque de dommages corporels. Si vous utilisez ce logiciel ou matériel dans le cadre d'applications dangereuses, il est de votre responsabilité de prendre toutes les mesures de secours, de sauvegarde, de redondance et autres mesures nécessaires à son utilisation dans des conditions optimales de sécurité. Oracle Corporation et ses affiliés déclinent toute responsabilité quant aux dommages causés par l'utilisation de ce logiciel ou matériel pour des applications dangereuses.

Marques

Oracle et Java sont des marques déposées d'Oracle Corporation et/ou de ses affiliés. Tout autre nom mentionné peut correspondre à des marques appartenant à d'autres propriétaires qu'Oracle.

Intel et Intel Inside sont des marques ou des marques déposées d'Intel Corporation. Toutes les marques SPARC sont utilisées sous licence et sont des marques ou des marques déposées de SPARC International, Inc. AMD, Epyc, et le logo AMD sont des marques ou des marques déposées d'Advanced Micro Devices. UNIX est une marque déposée de The Open Group.

Avis d'exclusion de responsabilité concernant les services, produits et contenu tiers

Ce logiciel ou matériel et la documentation qui l'accompagne peuvent fournir des informations ou des liens donnant accès à des contenus, des produits et des services émanant de tiers. Oracle Corporation et ses affiliés déclinent toute responsabilité ou garantie expresse quant aux contenus, produits ou services émanant de tiers, sauf mention contraire stipulée dans un contrat entre vous et Oracle. En aucun cas, Oracle Corporation et ses affiliés ne sauraient être tenus pour responsables des pertes subies, des coûts occasionnés ou des dommages causés par l'accès à des contenus, produits ou services tiers, ou à leur utilisation, sauf mention contraire stipulée dans un contrat entre vous et Oracle.

Date de publication et mention de la version préliminaire de Disponibilité Générale ("Pre-GA")

Version préliminaire de Disponibilité Générale ("Pre-GA") : 15.01.2020

Avis sur la version préliminaire de Disponibilité Générale ("Pre-GA") de la documentation

Si ce document est fourni dans la Version préliminaire de Disponibilité Générale ("Pre-GA") à caractère public ou privé :

Cette documentation est fournie dans la Version préliminaire de Disponibilité Générale ("Pre-GA") et uniquement à des fins de démonstration et d'usage à titre préliminaire de la version finale. Celle-ci n'est pas toujours spécifique du matériel informatique sur lequel vous utilisez ce logiciel. Oracle Corporation et ses affiliés déclinent expressément toute responsabilité ou garantie expresse quant au contenu de cette documentation. Oracle Corporation et ses affiliés ne sauraient en aucun cas être tenus pour responsables des pertes subies, des coûts occasionnés ou des dommages causés par l'utilisation de cette documentation.

Mention sur les informations confidentielles Oracle

INFORMATIONS CONFIDENTIELLES ORACLE. Destinées uniquement à un usage autorisé. Ne pas distribuer à des tiers.

Avis sur la reconnaissance du revenu

Si ce document est fourni dans la Version préliminaire de Disponibilité Générale ("Pre-GA") à caractère privé :

Les informations contenues dans ce document sont fournies à titre informatif uniquement et doivent être prises en compte en votre qualité de membre du customer advisory board ou conformément à votre contrat d'essai de Version préliminaire de Disponibilité Générale ("Pre-GA") uniquement. Ce document ne constitue en aucun cas un engagement à fournir des composants, du code ou des fonctionnalités et ne doit pas être retenu comme base d'une quelconque décision d'achat. Le développement, la commercialisation et la mise à disposition des fonctions ou fonctionnalités décrites restent à la seule discrétion d'Oracle.

Accessibilité de la documentation

Pour plus d'informations sur l'engagement d'Oracle pour l'accessibilité de la documentation, visitez le site Web Oracle Accessibility Program, à l'adresse : http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Accès aux services de support Oracle

Les clients Oracle qui ont souscrit un contrat de support ont accès au support électronique via My Oracle Support. Pour plus d'informations, visitez le site http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info ou le site http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs si vous êtes malentendant.

Contents

Using This Documentation

- **Overview** – Describes how to troubleshoot and maintain the server
- **Audience** – System administrators
- **Required knowledge** – Advanced experience in installing the operating system on multiple clients

Product Documentation Library

Documentation and resources for this product and related products are available at http://www.oracle.com/pls/topic/lookup?ctx=E37838-01.

Feedback

Provide feedback about this documentation at http://www.oracle.com/goto/docfeedback.

♦ ♦ ♦ CHAPTER 1

Overview of Customized Installations

Customized installations are a way of installing the OS that addresses different installation requirements. They are suited for an environment consisting of multiple system models and platforms with their own particular hardware configurations.

This guide describes how to customize automated installation through the use of AI manifests and configuration profiles. Use this guide as a companion to *Automatically Installing Oracle Solaris 11.4 Systems*.

This chapter covers the following topics:

- "About AI Manifests"
- "About System Configuration Profiles"
- "About Creating Client Systems"
- "Exporting an AI Manifest or a System Configuration Profile"
- "Using Rights Profiles to Install Oracle Solaris"

For an example of a highly customized automated installation that uses specially configured manifests and profiles, see `https://blogs.oracle.com/solaris/customize-network-configuration-with-solaris-automated-installs-v2`.

About AI Manifests

An AI installation runs by using a manifest. The manifest contains instructions to apply to clients. To customize an automated install, you adjust the instructions accordingly.

When you create an install service for the first time, a default manifest is automatically created for that service. The file is called *imagepath*/*service-name*`/auto_install/default.xml`. Unless you specify otherwise when creating the service, *imagepath*, by default, is `/export/auto_install`.

You can customize manifests in one of two ways:

- Modify the contents of the `default.xml`.

  Use this method for simple changes to the installation behavior. For example, to make clients automatically reboot after installation, you would add the `auto_reboot` attribute to the file:

  ```
  <ai-instance name="default" auto_reboot="true">
  ```

- Create new manifests with their own definitions.

  The `default.xml` manifest is intended to serve all the clients. You should not use this file for customized installations, but instead create other manifests. You can create as many manifests as you want. Then you define criteria that would match clients with specific manifests.

As a safe approach, create a new manifest out of an existing one, such as `default.xml`. For more details, see Chapter 2, “Working With AI Manifests”.

About System Configuration Profiles

System configuration profiles are System Management Facility (SMF) XML files. They contain defined system properties that specific SMF services apply to the client when the client is booted after installation.

You manually create profiles. An AI client can use multiple profiles that are associated with the install service. For these cases, you would also need to specify client criteria to ensure that the correct clients use the appropriate profiles during the installation. If you do not define client criteria, then the profile is applied to all AI clients.

Profiles are not required at installation. If no profile exists, then you would manually perform post-install configurations through the System Configuration Interactive (SCI) tool.

About Creating Client Systems

In a data center with multiple systems, creating client and service associations ensures that the correct install service, manifest, and profile are used for a specific client. When you create a client, you must know the client system's MAC address and the install service to which the system is assigned.

For instructions, see “Registering Clients With Install Services” in *Automatically Installing Oracle Solaris 11.4 Systems*.

Exporting an AI Manifest or a System Configuration Profile

The `installadm export` command copies the contents of AI manifests or system configuration profiles of an install service to a file or directory.

Use the `-o` *pathname* option to specify where the manifest and profile contents are exported. If you omit this option, contents are generated in `stdout`. If you are exporting multiple files, then *pathname* must be a directory.

The specified manifest can be the name of an XML AI manifest or a derived manifests script.

Use the `installadm export` command for the following tasks:

- Check the specifications in a manifest or a profile.
- Modify an existing manifest or profile.
- Use an existing manifest or profile as a base for creating a new manifest or profile.

Using Rights Profiles to Install Oracle Solaris

Oracle Solaris implements role-based access control (RBAC) to control system access. To perform specific tasks and run privileged commands on the system, you must have the profiles that provide you the authorization.

The following list shows some of the profiles that need to be assigned to you to install Oracle Solaris.

- Install Client Management enables you to install Oracle Solaris on client systems.
- Install Manifest Management enables you to create or configure manifests to customize the installation.
- Install Profile Management enables you to create and configure system configuration profiles to customize the installation.

Some profiles are supersets of a combination of profiles. For example, the Install Service Management profile contains the three profiles in the previous list.

The list of required profiles expands if you perform additional tasks that might be indirectly connected to your current one, such as network configuration or zone configuration.

An administrator that has the `solaris.delegate.*` authorization can assign the necessary profiles to users to enable them to perform administrative tasks in Oracle Solaris.

For example, an administrator assigns the Install Service Management rights profile to user `jdoe`. Before `jdoe` executes a privileged installation command, `jdoe` must be in a profile shell. The shell can be created by issuing the `pfbash` command. Or, `jdoe` can combine `pfexec` with every privileged command that is issued, such as `pfexec installadm`.

As an alternative, instead of assigning profiles directly to users, a system administrator can create a role that would contain a combination of required profiles to perform a range of tasks.

Suppose that a role `installadmin` is created with the profiles for installation as well as for zone creation and configuration. User `jdoe` can issue the `su` command to assume that role. All roles automatically get `pfbash` as the default shell.

For more information about rights profiles, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

♦ ♦ ♦ CHAPTER 2

Working With AI Manifests

This chapter describes how to use AI manifests to determine the manner of installing Oracle Solaris. It covers the following topics:

Creating and Customizing an AI Manifest

The following are the general steps to customize an automated installation:

1. Create the AI manifest with specific configuration definitions.
2. Associate the manifest with an install service and specify criteria that determine which clients use that manifest.

   For more information about install services, see "Working With Install Services" in *Automatically Installing Oracle Solaris 11.4 Systems*
3. Create customized configuration profiles as needed and define client criteria.
4. To start the installation, boot the client from the network.

   See "Starting Automatic Installation on Clients" in *Automatically Installing Oracle Solaris 11.4 Systems*.

To create and edit manifests, three tools are available.

- Interactive editor
- AI manifest wizard
- Direct editing of the XML file

Use either the interactive editor or the AI manifest wizard if you are not knowledgeable about XML. These tools guide you through each parameter to ensure that all the necessary parameters are configured.

For samples of customized manifests, see "Examples of AI Manifests" on page 26.

Note - A related method to customize installations is the use of derived manifests as described in Chapter 4, "Using a Script to Customize an Installation".

Using the Interactive Editor

The `installadm` command automatically runs in interactive mode when you create a new manifest, whether based on an existing manifest or not.

▼ How to Use the AI Interactive Tool

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Install Oracle Solaris" on page 15.

1. **Launch the interactive tool with one of the following methods:**

   - **Create a new manifest.**
     For example:

     ```
     aiserver$ installadm create-manifest -n solaris11_4-i386 -m mem1
     ```

   - **Create a new manifest based on an existing one.**
     For example:

     ```
     aiserver$ installadm create-manifest -e -n solaris11_4-i386 -m mem1 \
        -f /var/tmp/x86manifest.xml
     ```

     - `-e` activates the command's interactive mode.
     - `-n` identifies the install service that will use the new manifest.
     - `-m` identifies the manifest instance.
     - `-d` specifies the full path and XML file of the manifest.

   The interactive mode is indicated by the `installadm:`*manifest*> prompt.

2. **Modify or add parameter definitions to the file.**

Choose one of the following methods:

- **Type `walk` to go through each parameter.**

  ```
  installadm:mem1> walk
  ```

  Configure the first parameter or press Enter to skip to subsequent parameters.

- **Use the Interactive Editor commands.**

  Use editor commands to go directly to the parameters you want to configure or to add definitions. For a list of these commands, see "Interactive Editor Commands" or type `help` at the interactive prompt.

3. **Type exit to quit the interactive mode.**

   An exit menu is displayed.

4. **Type the menu item number of your choice.**

Interactive Editor Commands

Use the following interactive editor commands to configure the manifest:

add	Adds an object or property. The command prompts you for objects and properties. See the description of `walk` in this list.
cancel	Discards any changes made on the current level and navigates up one level.
commit	At the top level, validates changes, saves the manifest and continues editing.
delete	Deletes an object or property.
end	Validates changes made on the current level and if no errors occur navigates up one level.
exit	Prompts to save the changes and exit, to exit without saving changes, or to continue editing.
info	Displays all objects and properties up to one level down.
move	Changes the order of multiple objects, such as software publishers.

select	Selects an object and navigates to that level.
set	Sets the value of a property.
walk	Prompts for each settable property and any settable subobjects or subproperties for the select object.

Examples of the Use of the Interactive Edit Mode

In the following examples, all user input is in bold. See also examples in the `installadm`(8) man page.

EXAMPLE 1 Using the `walk` Interactive Command

This example shows an extract of the screen when you use the `walk` subcommand while customizing a manifest.

```
aiserver$ installadm create-manifest -n x86-service -m mymanifest
Type help to see list of subcommands.
installadm:mymanifest> walk
   *** To terminate walk, use Ctrl-D. ***
   http:proxy [not specified]:
   auto-reboot [false]: true
    Ctrl-D is pressed
installadm:mymanifest> exit
1. Save manifest and exit.
2. Exit without saving committed changes.
3. Continue editing.
Please select choice: 1
```

EXAMPLE 2 Adding a Publisher

This example uses interactive subcommands add a second publisher called `firstoot` to the manifest.

```
aiserver$ installadm create-manifest -n default-sparc -m test
Type help to see list of subcommands.
installadm:test> select software
installadm:test:software> add -w publisher
   *** To terminate walk, use Ctrl-D ***
   name [<not specified>]: firstboot
   key [<not specified>]:
   cert [<not specified>]:
   ca-cert [<not specified>]:
```

```
   origin [<not specified>]: file:///net/host1/export/firstbootrepo
    Ctrl-D is pressed
installadm:test:software:publisher> info
   name: firstboot
   key: <not specified>
   cert: <not specified>
   ca-cert: <not specified>
   origin: file:///net/host1/export/firstbootrepo
   mirror: <not specified>
   cmd-options: <not specified>
installadm:test:software:publisher> end
installadm:test:software> exit
1. Save manifest and exit
2. Exit without saving uncommitted changes
3. Continue editing
Please select choice: 1
```

EXAMPLE 3 Changing the Order of an Object

In this example, the order of the publishers is switched so that `firstboot` precedes `solaris`.

```
aiserver$ installadm update-manifest -n default-sparc -m test
Type help to see list of subcommands.
installadm:test> select software
installadm:test:software> move publisher 2 1
installadm:test:software> info
   type: IPS
   name: <not specified>
     ...
  publisher[1]:
      name: firstboot
      key: <not specified>
      cert: <not specified>
      ca-cert: <not specified>
      origin: http://example.com/solaris/mybuild
      mirror: <not specified>
      cmd-options: <not specified>
   publisher[2]:
      name: solaris
      key: <not specified>
      cert: <not specified>
      ca-cert: <not specified>
      origin: http://pkg.oracle.com/solaris/release
      mirror: <not specified>
      cmd-options: <not specified>
   pkg-list:
      action: install
      name: pkg:/entire@0.5.11-0.175.3
```

```
      name: pkg:/group/system/solaris-large-server
      reject: <not specified>
installadm:test:software> exit
1. Save manifest and exit
2. Exit without saving uncommitted changes
3. Continue editing
Please select choice: 1
```

Using the AI Manifest Wizard

The AI manifest wizard is a browser user interface (BUI) web application. It displays screens in sequence where you configure different sections of the manifest.

▼ How to Create an AI Manifest Using the AI Manifest Wizard

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Install Oracle Solaris" on page 15.

1. **Connect to the AI server remotely with X11 forwarding enabled.**

   ```
   $ ssh -X ai-server
   ```

   As an alternative, open a desktop session directly on the AI server.

2. **(Optional) Enable the wizard web UI.**

   ```
   aiserver$ installadm set-server -u -z
   ```

   The `-z` option causes manifests to be stored locally on the AI server in `/var/ai/wizard-manifest`.

3. **Launch the AI Manifest Wizard either through the browser or through the command line.**

   - **On the browser, type the following address:**

     ```
     http://ai-server.domain:55555
     ```

   - **On the terminal window, type the following command:**

     ```
     aiserver$ /usr/bin/ai-wizard -p 5555
     ```

   Either command opens the wizard user interface.

4. **Configure the parameters on each screen as needed.**

   After selecting whether to work on a specific service's manifest or to create a new one, you proceed to screens where you provide values to the following items:

   - Manifest name and target
   - Root pool and data pools
   - Disk allocations for the pools
   - Repositories
   - Software packages to install
   - Zone definitions

Directly Working on an XML Manifest File

If you are familiar with XML, you can create a file and configure the manifest directly with any text editor. This method enables you to navigate freely throughout the file to configure only those parameters you want to customize.

▼ How to Create and Directly Edit an XML Manifest

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Install Oracle Solaris" on page 15.

1. **Create a new manifest XML file by using one of two methods:**

   - **Create a copy of an existing manifest file.**

     For example:

     ```
     aiserver$ cp /usr/share/auto-install/default.xml /var/tmp/mem1.xml
     ```

   - **Export an existing manifest.**

     a. **List existing manifests.**

     ```
     aiserver$ installadm list -m -n solaris11_4-i386
     Service Name           Manifest Name  Type     Status    Criteria
     ------------           -------------  -------  ------    --------
     solaris11_4-i386       orig_default   derived  default   none
                            x86manifest    xml      inactive  none
     ```

     b. **Export the selected manifest to a new file.**

```
aiserver$ installadm export -o /var/tmp/mem1.xml -m x86manifest -n solaris11_4-
i386
```

c. **(Optional) Verify the results.**

```
aiserver$ ls /var/tmp
...
mem1.xml
...
```

2. **Open and modify the file copy.**

   If you are working on a copy of a manifest, then the valid XML tags are already in the file for you to configure. Otherwise, see the `ai_manifest(5)` man page for instructions on creating or editing XML manifest files.

3. **Create the new manifest's instance name for the install service.**

   Optionally, specify criteria to identify which AI clients should use that manifest. For example:

```
aiserver$ installadm create-manifest -n solaris11_4-i386 -f ./var/tmp/mem1.xml \
   -m mem1 -c mem="2048-unbounded"
```

   The command assigns `mem1` to `solaris11_4-i386`. Clients with memory size 2048 MB or greater will use `mem1`.

   See Chapter 5, “Specifying Criteria for AI Manifests and System Configuration Profiles” for more information about specifying client criteria.

4. **(Optional) List the current manifests.**

```
aiserver$ installadm list -m -n solaris11_4-i386
Service Name           Manifest Name  Type     Status    Criteria
------------           -------------  -------  ------    --------
default-i386           orig_default   derived  inactive  none
solaris11_4-i386       x86manifest    xml      default   none
                       mem1           xml      inactive mem = 2048 MB - unbounded
                       orig_default   derived  inactive  none
```

Associating AI Manifests With Install Services

You associate an AI manifest with a service in one of two ways:

- While creating a manifest.

Each time you create a manifest, you must specify the install service for that manifest. Thus the command creates both the manifest and the association.

For example:

```
$ installadm create-manifest -n solaris11_4-sparc -m mem-manifest \
   [-c criteria]
```

- While configuring an install service.

  This method sets an *existing* manifest to be the service's default manifest.

```
$ installadm create-service | set-service -M mem-manifest -n install-sparc \
   [-c criteria]
```

To remove the association between a manifest and a service, you simply delete the manifest:

```
$ installadm delete-manifest -m mem-manifest -n install-sparc
```

Other AI Manifest Management Tasks

This section describes additional procedures to maintain AI manifests.

Updating an AI Manifest

The `installadm update-manifest` command replaces the contents of a manifest or script with contents from another file or script. The criteria, default status, and manifest name are not changed as a result of the update. The command also validates the XML manifest files while updating.

The manifest must already exist in the specified service. Use the `installadm list -m` command to confirm.

This example updates the content of the `sparc-ent` manifest in the `solaris11_4-sparc` service with the content of `./mymanifests/manifest-new-sparc-ent.xml`.

```
$ installadm update-manifest -n solaris11_4-sparc \
-f ./mymanifests/manifest-new-sparc-ent.xml -m sparc-ent
```

Validating an AI Manifest

The `installadm validate -m` command validates a service's AI manifests for syntactic correctness to ensure they have not been corrupted. If used with the `-M` *manifest-pathname* option, the validation includes manifests that do not have assigned services.

Note - The `create-manifest` subcommand automatically validates AI manifests before adding them to the install service.

Validated manifests are displayed as `stdout` output. Errors are listed as `stderr` output.

Examples of AI Manifests

The examples in this section show the XML elements that the finished AI manifest must have to achieve the stated result. These manifests can be created either by editing the XML directly or by using a derived manifests script.

Caution - The `entire` package constrains system package versions to the same build, so it must be included in each manifest. Proper system updates and correct package selection depend on the presence of this package. Do not remove the installation of this package from your AI manifest, and do not uninstall this package after installation. Removing this package will result in an unsupported system.

Specifying an iSCSI Target Device

In this example, the target for the installation is an iSCSI device. The `whole_disk` attribute of the `disk` element is set to `true`, which is typical for iSCSI disks. See the `ai_manifest(5)` man page for descriptions of the `target_name`, `target_lun`, and `target_ip` attributes.

```
<auto_install>
  <ai_instance name="default">
    <target>
      <disk whole_disk="true">
        <iscsi target_name="iqn.1986-03.com.sun:02:1234567890abcdef"
```

```
            target_lun="1" target_ip="129.158.144.200"/>
      </disk>
      <logical>
        <zpool name="rpool" is_root="true">
          <filesystem name="export" mountpoint="/export"/>
          <filesystem name="export/home"/>
          <be name="solaris"/>
        </zpool>
      </logical>
    </target>
    <software type="IPS">
      <source>
        <publisher name="solaris">
          <origin name="http://pkg.oracle.com/solaris/release"/>
        </publisher>
      </source>
      <software_data action="install">
        <name>pkg:/entire@0.5.11-0.175.2</name>
        <name>pkg:/group/system/solaris-large-server</name>
      </software_data>
    </software>
  </ai_instance>
</auto_install>
```

Specifying a Root Pool and Boot Pool in an AI Manifest

The following sample AI manifest defines an iSCSI device as the root pool, and configures the boot pool to be named `mybpool`. The default name for a boot pool is `bpool`. Because no disk devices are included with the boot pool description, the dedicated eUSB disks will be used by default.

```
<auto_install>
  <ai_instance name="default">
    <target>
      <disk in_zpool="rpool" whole_disk="true">
        <iscsi target_port="326" target_ip="6.6.6.53"\
            target_lun="1">
      </disk>
      <logical>
        <zpool name="rpool" is_root="true">
          <filesystem name="export" mountpoint="/export"/>
          <filesystem name="export/home"/>
          <be name="solaris"/>
```

```
        </zpool>
        <zpool name="mybpool" is_boot="true">
      </logical>
    </target>
    <software type="IPS">
      <source>
        <publisher name="solaris">
          <origin name="http://pkg.oracle.com/solaris/release"/>
        </publisher>
      </source>
      <software_data action="install">
        <name>pkg:/entire@0.5.11-0.175.2</name>
        <name>pkg:/group/system/solaris-large-server</name>
      </software_data>
    </software>
  </ai_instance>
</auto_install>
```

To create a mirrored configuration, specify multiple disks for the root pool. In this case, all root disks are automatically partitioned the same way, with a partition from each disk used to create a mirrored boot pool.

Specifying a RAID Configuration

This example specifies a RAID configuration using the two disks `c0t0d0` and `c0t1d0`. This manifest is similar to the manifest for a mirrored configuration as shown in “Specifying a Mirrored Configuration If at Least Two Disks of a Specified Size Are Present” on page 54. One difference between the two manifests is that the value of the `redundancy` attribute is `raidz` instead of `mirror`. See the `zpool(8)` man page for information about redundancy types. Another difference is that the ZFS pool is not named `rpool`, because `rpool` implies the root pool. By default, the value of the `is_root` attribute of the `zpool` element is `false`, so that assignment could be omitted in this example. Because no root pool is specified, do not configure an initial user for this installation.

```
<auto_install>
  <ai_instance name="default">
    <target>
      <disk in_vdev="raid_vdev" in_zpool="raidpool" whole_disk="true">
        <disk_name name="c0t0d0" name_type="ctd"/>
      </disk>
      <disk in_vdev="raid_vdev" in_zpool="raidpool" whole_disk="true">
        <disk_name name="c0t1d0" name_type="ctd"/>
      </disk>
```

```
      <logical>
        <zpool name="raidpool" is_root="false">
          <vdev name="raid_vdev" redundancy="raidz"/>
        </zpool>
      </logical>
    </target>
    <software type="IPS">
      <source>
        <publisher name="solaris">
          <origin name="http://pkg.oracle.com/solaris/release"/>
        </publisher>
      </source>
      <software_data action="install">
        <name>pkg:/entire@0.5.11-0.175.2</name>
        <name>pkg:/group/system/solaris-large-server</name>
      </software_data>
    </software>
  </ai_instance>
</auto_install>
```

Installing an SVR4 Package

This example demonstrates how to install a SVR4 package. SVR4 packages must be named in a `software` element of type `SVR4`. The value of the name attribute of the origin of the publisher is a directory that contains SVR4 package subdirectories or a SVR4 package datastream file. This origin name for SVR4 package subdirectories can be a full file directory path or a file URI. This origin name for a SVR4 package datastream file can be a full file directory path, a file URI, or an HTTP URI.

Do not install packages that require user input as part of their installation.

```
<auto_install>
  <ai_instance name="default">
    <target>
      <logical>
        <zpool name="rpool" is_root="true">
          <filesystem name="export" mountpoint="/export"/>
          <filesystem name="export/home"/>
          <be name="solaris"/>
        </zpool>
      </logical>
    </target>
    <software type="IPS">
      <source>
```

```
        <publisher name="solaris">
          <origin name="http://pkg.oracle.com/solaris/release"/>
        </publisher>
      </source>
      <software_data action="install">
        <name>pkg:/entire@0.5.11-0.175.2</name>
        <name>pkg:/group/system/solaris-large-server</name>
      </software_data>
    </software>
    <software type="SVR4">
      <source>
        <publisher>
          <origin name="/net/host2/usr/dist"/>
        </publisher>
      </source>
      <software_data>
        <name>SUNWpackage</name>
      </software_data>
    </software>
  </ai_instance>
</auto_install>
```

Installing Multiple SVR4 Packages

To install multiple SVR4 packages, you will need to specify the software tag for each package as shown below.

```
<software type="SVR4">
  <source>
    <publisher>
      <origin name="/net/192.0.2.2/svr4/app1.pkg"/>
    </publisher>
  </source>
  <software_data>
    <name>application1</name>
  </software_data>
</software>
<software type="SVR4">
  <source>
    <publisher>
      <origin name="/net/192.0.2.2/svr4/app2.pkg"/>
    </publisher>
  </source>
  <software_data>
    <name>application2</name>
```

```
  </software_data>
</software>
```

Reusing Existing Disk Slices or Partitions

This sample shows how to specify to use existing disk slices for a SPARC client. For disk slices, the dimensions (start_sector and size) from an existing slice are reused. The configuration process will not search the slices to see if there is already a version of Solaris installed.

```
    <disk>
      <disk_name name="c1t0d0" name_type="ctd"/>
      <slice name="0" action="use_existing" force="true" in_zpool="rpool">
    </disk>
```

The following example shows how to specify for an x86 client that an existing partition on a disk should be reused during the AI process. For partitions, the existing dimensions for the named slice should be reused. In this case which partition is reused is automatically determined during the configuration process.

```
    <partition action="use_existing_solaris2">
        <slice action="use_existing" name="0" force="true"/>
    </partition>
```

Accessing a Unified Archive Using SSL Client Authentication

To access a unified archive using SSL, include a `credentials` element, to provide key, certificate, and CA certificate information in the AI manifest. In this example, the unified archive is called `sslarchive.ua`.

```
<software type="ARCHIVE">
   <source>
       <file uri="https://example.com/sslarchive.ua"/>
```

```
            <credentials>
                <key src="file://root/key.pem"/>
                <cert src="file://root/cert.pem"/>
                <cacert src="file://root/cacert.pem"/>
            </credentials>
        </file>
    </source>
</software>
```

See "Overview of Securing Automated Installations" in *Automatically Installing Oracle Solaris 11.4 Systems* for information about setting up SSL on your AI server.

Accessing a Unified Archive Using Http Authentication Tokens

To use an HTTP authentication token when accessing a unified archive, add the token to a `credentials` element. Also, add a line defining the unified archive.

```
<software type="ARCHIVE">
    <source>
        <file uri="http://example.com/httparchive.ua"/>
            <credentials>
                <http_auth_token>my-specifically-granted-auth-token</http_auth_token>
            </credentials>
        </file>
    </source>
</software>
```

Accessing a Secure IPS Repository

In order to access a secure IPS repository, you must specify the key and certificate for the repository.

```
    <publisher name="solaris">
        <origin name="https://pkg.oracle.com/solaris/support/"/>
        <credentials>
            <key src="/var/pkg/ssl/Oracle_Solaris_11_Support.key.pem" />
            <cert src="/var/pkg/ssl/Oracle_Solaris_11_Support.certificate.pem" />
        </credentials>
    </publisher>
```

Changing the Locale When Installing the `solaris-minimal-server` Package

If you install `solaris-minimal-server` group, only the C/POSIX locale is installed on the system. If you use any of the UTF-8 locales, you need to install the `system/locale` package and customize IPS `facet.locale` attribute setting as needed. If you do not set `facet.locale.*` to false all of the available UTF-8 locales will be installed. For example, if your system's default locale is set to `en_US.UTF-8` in the system configuration profile, you should add these lines to the manifest to add the `en_US` locale:

```
    <software type="IPS">
      <destination>
        <image>
          <!-- Unset all locales -->
          <facet set="false">facet.locale.*</facet>
          <!-- Specify locales to install -->
          <facet set="true">facet.locale.en</facet>
          <facet set="true">facet.locale.en_US</facet>
        </image>
      </destination>
...
      <software_data action="install">
         <name>pkg:/entire@5.12-5.12.0</name>
         <name>pkg:/group/system/solaris-minimal-server</name>
         <name>pkg:/system/locale</name>
      </software_data>
    </software>
```

Not Installing the Man Page Package with the `solaris-minimal-server` Package

Even though the `solaris-minimal-server` group does not include the `man` command, by default it includes man pages. To prevent the installation of the man pages, you need to set the `doc.man` facet to `false` as shown:

```
    <software type="IPS">
      <destination>
        <image>
```

```
        <!-- Specify that man pages should not be installed -->
        <facet set="false">doc.man</facet>
      </image>
    </destination>
...
    <software_data action="install">
       <name>pkg:/entire@5.12-5.12.0</name>
       <name>pkg:/group/system/solaris-minimal-server</name>
       <name>pkg:/system/locale</name>
    </software_data>
  </software>
```

Adding a Specific Package when Installing a Package Group

Even though the `solaris-minimal-server` group does not include the `man` command, by default it includes man pages. To add the `man` command to the system during installation, you need to include the `/text/doctools` package as show below:

```
  <software type="IPS">
...
    <software_data action="install">
       <name>pkg:/entire@5.12-5.12.0</name>
       <name>pkg:/group/system/solaris-minimal-server</name>
       <name>pkg:/text/doctools</name>
    </software_data>
  </software>
```

♦ ♦ ♦ CHAPTER 3

Working With System Configuration Profiles

This chapter describes how to use profiles to specify information needed to configure the AI client after installation. It covers the following topics:

- "Creating Configuration Profiles"
- "Associating System Configuration Profiles With Install Services"
- "Other System Configuration Profile Tasks"
- "Using System Configuration Profile Templates"
- "Specifying Configuration in a System Configuration Profile"

Creating Configuration Profiles

System configuration profiles are XML files that enable you to automate client configurations at first boot after an installation is completed.

You can assign any number of configuration profiles to a client. However, make sure that the client does not use a set of profiles with overlapping property definitions. Otherwise, even if the property setting is the same in those multiple profiles, the behavior of the SMF service being configured remains undefined.

You create profiles by using one of the following tools:

- System Configuration Interactive (SCI) tool
- Any XML editor
- Derived manifests script

  Scripts are discussed in Chapter 4, "Using a Script to Customize an Installation".

Whichever tool you use, the best practice is to use templates so that you do not have to start from an empty profile. See "Templates for System Configuration Profiles" on page 88. On

these files, you can either provide actual value settings, or use variables for a more dynamic process. See “Using System Configuration Profile Templates” on page 40.

Using the SCI Tool

This tool creates an XML profile out of responses that you provide interactively. To launch the tool, issue the following command:

```
# sysconfig create-profile [-o directory][other-options]
```

For other options you can use, see the `sysconfig(8)` man page.

The command opens the SCI tool which prompts you for configuration information to be stored in `sc_profile.xml`. For an overview of the SCI tool's screens, see Appendix A, “System Configuration Profiles”.

By default, `sc_profile.xml` is created in the `/system/volatile/profile/` directory. To store the file in a different existing directory, specify the `-o` *directory* option. The new profile overwrites any profile existing in that location.

Using an XML Editor

Use any XML editor to create an XML file to contain the property specifications. If you start from an empty XML file, include the following required declarations at the top of the file:

```
<!DOCTYPE service_bundle SYSTEM "/usr/share/lib/xml/dtd/service_bundle.dtd.1">
<service_bundle type="profile" name="sysconfig">
```

Refer to “Specifying Configuration in a System Configuration Profile” on page 43 for guidance on how to set SMF property definitions properly in the file.

Using a Derived Manifests Script

In this method, you provide the specifications in the manifest script itself. This method assumes you are familiar with langage scripts and their commands. For an example of how to add a configuration profile in the script, see “Adding a System Configuration Profile” on page 59.

Extracting Configuration Information for KMIP Clients

In Oracle Solaris 11.4, KMIP client configuration is stored in the Service Management Facility (SMF). You can use the `kmipcfg` command to create sn SMF configuration profile based on existing KMIP configuration. You provide the parameters either interactively or directly on the command line.

The SMF configuration profile can be created regardless of whether the system is currently configured as a KMIP client. After it is created, the profile can be used in AI installations to apply its configuration to target clients.

This method assumes that a working KMIP server group already exists. For procedures, see "Creating and Configuring a KMIP Server Group" in *Managing Encryption and Certificates in Oracle Solaris 11.4*.

The procedure consists of the following steps:

1. Extract the configuration into an SMF configuration profile.
2. Assign the profile to an AI service to be used for installation.

   The profile can also be used with the `sysconfig` command for configuring or unconfiguring the Oracle Solaris instance.

To create an SMF system configuration profile on a KMIP client based on an existing server-group, use the following command syntax:

`$` **`kmipcfg extract -p`** *filename* **`[-s] [-t all|pkcs11|libkmip]`** *server-group*

`-p` *filename*	Profile where the extracted configuration information is stored. The filename must include the `.xml` extension. If you do not use this option, the configuration is printed to standard output.
`-s`	Extracts sensitive information such as encoded certificates.
`-t` all | pkcs11 | libkmip	Type of information that is extracted: ■ `pcks11` creates the profile for the `/system/pkcsll.kmip` service instance. ■ `libkmip` creates the profile for the `/system/kmip/client:default` service instance. ■ `all`, the default setting, creates the profile for both services.
server-group	Source of the configuration information

For additional options, see the kmipcfg(8) man page.

Suppose that you have a KMIP server group cluster-1 as follows:

```
$ kmipcfg list
Server group: cluster1
State: enabled
Hosts:  server1.example.com:5696
        server2.example.com:5696
        server3.example.com:5696
Connection timeout: 5
Cache object time to live: 300
Encoding: TTLV
Failover limit: 3
Client keystore: /var/user/testuser/kmip/cluster1
Client PKCS#12 bundle: cluster1_cred.p12
Secondary authentication type: none
```

To store cluster-1's configuration in an SMF profile, issue the following command:

```
$ kmipcfg extract -p /var/tmp/kmip-client.xml cluster-1
```

To use the profile in an AI installation, you would associate kmip-client.xml with an install service. For example, assuming an existing sparc-install install service, you would issue the following command:

```
$ installadm create-profile -n sparc-install -f /var/tmp/kmip-client.xml -p kmip-prof
```

The next section describes associating install services and profiles more fully.

Associating System Configuration Profiles With Install Services

Like manifests, system configuration profiles must be associated with install services in order to be applied during installation.

When you create a profile instance, you are required to name a service for that instance. Thus, the command creates both the profile and its association with the service.

For example:

```
$ installadm create-profile -n solaris11_4-sparc -f ./profile-sparc-ent.xml \
-p sparc-ent
```

-n Specifies the name of the service.

`-f`	Identifies the the XML file on which the profile instance is based.
`-p`	Specifies the name of the profile instance. If you do not use this option, then the profile instance name is based on the base file name that is specified by the `-f` option, thus `profile-sparc-ent`.

The profile configuration is applied to all clients of the service unless you specify client criteria. See "Defining Criteria for Manifests and Profiles" on page 64.

For additional options when creating profiles, see the `installadm`(8) man page.

To remove the association between a profile and a service, simply delete the profile.

```
$ installadm delete-profile -p sparc-ent -n solaris11_4-sparc
```

Other System Configuration Profile Tasks

This section describes other procedures to maintain system configuration profiles.

Updating a System Configuration Profile

The `installadm update-profile` command replaces service's profile with the contents of another file. The update does not change any defined criteria in the first profile.

To ensure that the correct profile is updated, you should specify the profile instance name. Otherwise, the profile instance name is extracted from the base name of the actual file.

The following command updates the content of the `sparc-ent` profile in the `solaris11_4-sparc` service with the content of `./myprofiles/profile-new-sparc-ent.xml`.

```
$ installadm update-profile -n solaris11_4-sparc \
-f ./myprofiles/profile-new-sparc-ent.xml -p sparc-ent
```

Validating a System Configuration Profile

The `installadm validate -p` command validates system configuration profiles of an install service for syntactic correctness to ensure they have not been corrupted. The command ensures

that the profile configurations can be implemented. Use the `installadm list` command to display a list of existing profiles.

To be properly validated, associate profiles with a specific install service. Validated profiles are displayed as `stdout` output. Errors are listed as `stderr` output.

Using System Configuration Profile Templates

Instead of creating multiple profiles for each client, use system configuration profile templates that offer more flexibility. These templates contain variables for property settings. During installation, the variables are replaced with actual values that are applied to the client. Variables enable you to reuse the profile templates for various install operations. For a list of available templates, see "Templates for System Configuration Profiles" on page 88.

To use system configuration profile templates, follow these steps:

1. Create a profile template where property values are set to variables.

   Oracle Solaris has pre-defined templates on which you can base your own templates.
2. Associate the profile with an install service.
3. Run the installation.

Variables for System Configuration Profiles

The following table lists valid variables to be used in a system configuration profile template.

Variable	Description
`AI_ARCH`	Kernel architecture from `uname -m`
`AI_CPU`	Processor type from `uname -p`
`AI_HOSTNAME`	AI client DNS name
`AI_IPV4`	IP version 4 network address
`AI_IPV4_PREFIXLEN`	Prefix length of the IPv4 network address
`AI_MAC`	Hexadecimal MAC address with colon (:) separators
`AI_MEM`	Memory size in megabytes returned by `prtconf`
`AI_NETLINK_DEVICE`	Name of network interface physical device
`AI_NETLINK_VANITY`	Default vanity name of network interface

Variable	Description
AI_NETWORK	IP version 4 network identifier
AI_ROUTER	IP version 4 network address of the AI client's default router
AI_ZONENAME	AI client zone name

Configuring Kerberos

A system configuration profile that includes Kerberos configuration information for an AI client should be created by the `kclient` command. Although the profile can be viewed, editing the file by hand is not suggested. For more information, see "How to Configure Kerberos Clients Using AI" in *Automatically Installing Oracle Solaris 11.4 Systems*.

Using a Configuration Template: An Example

In this example, the `/export/hostandIP.xml` is copied from the `/usr/share/auto_install/sc_profiles/static_network.xml` template and customized with values shown in bold:

- `nodename`, `group name`, and `ipv4-address` have variable values.
- `prefixlen` and `gateway` are assigned actual values.

  These values will be applied to all AI clients that use the profile.

```
<?xml version='1.0'?>
<!DOCTYPE service_bundle SYSTEM "/usr/share/lib/xml/dtd/service_bundle.dtd.1">
<service_bundle type="profile" name="sysconfig">
  <service name="system/identity" version="1" type="service">
    <instance name="node" enabled="true">
      <property_group name="config" type="application">
        propval name="nodename value="{{AI_HOSTNAME}}"/>
      </property_group>
    </instance>
  </service>
  <service name="network/ip-interface-management" version="1" type="service">
    <instance name="default" enabled="true">
      <property_group name="interfaces" type="application">
        <!-- interface configuration -->
        <property_group name="{{AI_NETLINK_VANITY}}" type="interface-ip">
          <property name="address-family" type="astring">
            <astring_list>
              <value_node value="ipv4"/>
              <value_node value="ipv6"/>
            </astring_list>
```

```
          </property>

          <!-- IPv4 static address -->
          <property_group name="v4" type="address-static">
            <propval name="ipv4-address" type="astring" value="{{AI_IPV4}}"/>
            <propval name="prefixlen" type="count" value="24"/>
            <propval name="up" type="astring" value="yes"/>
          </property_group>

          <!-- IPv6 addrconf address -->
          <property_group name="v6" type="address-addrconf">
            <propval name="interface-id" type="astring" value="::"/>
            <propval name="prefixlen" type="count" value="0"/>
            <propval name="stateful" type="astring" value="yes"/>
            <propval name="stateless" type="astring" value="yes"/>
          </property_group>

          <!-- default static route through interface -->
          <property_group name="route-1" type="static-route">
            <propval name="destination" type="astring" value="default"/>
            <propval name="family" type="astring" value="inet"/>
            <propval name="gateway" type="astring" value="203.0.113.1"/>
          </property_group>
        </property_group>
      </property_group>
    </instance>
  </service>
</service_bundle>
```

After saving the template file, you associate the profile with an install service.

```
$ installadm create-profile -n solaris11_4-i386 -f /export/hostandIP.xml
```

Suppose that you want to implement the `hostandIP`'s configuration to the following client:

- Node name: `server1`
- IP address: `203.0.113.2`
- Datalink name: `net0`
- Install service to use: `solaris11_4-i386`

During installation, the client's configuration is read and actual values replace the variables in the profile. In the case of `server1`, the following variable replacements occur in `hostandIP.xml`:

- Value of `nodename` becomes `server1` to replace `AI_HOSTNAME`.
- Value of `name` for the IP interface type becomes `net0` to replace `AI_NETLINK_VANITY`.
- Value of `ipv4-address` is 203.0.113.2 to replace `AI_IPV4`.

Specifying Configuration in a System Configuration Profile

System configuration information is specified through SMF properties. Accordingly, system configuration profiles are structured so that SMF services can read and apply the settings in the profiles. If you specify a service or property that does not apply, that specification is ignored. Make sure that you do not specify any particular property more than one time.

Within a configuration profile, configuration information is divided into sections for each service. Each service section lists one or more property groups. In turn, property groups have one or more specific properties that have value settings. For example, the following section refers to the configuration of the `default` and `host` properties of the `config` property group of the `name-service/switch` service.

```
<service version="1" name="system/name-service/switch">
   <property_group name="config">
      <propval name="default" value="files"/>
      <propval name="host" value="files dns mdns"/>
   </property_group>
   </instance enabled="true" name="default">
</service>
```

The following SMF command displays information about SMF properties:

`svccfg -s` *FMRI* `describe [-v] [-t] [`*propertygroup*`/`*property*`]`

- The `-v` option gives all information available, including descriptions for current settings, constraints, and other possible setting choices.
- The `-t` option shows only the template data for the selection (see the `smf_template(7)` man page), and does not display the current settings for property groups and properties.

See *Property Inspection and Modification Subcommands* in the `svccfg(8)` man page.

In the previous example, for information about the `config` property group in the `name-service/switch` SMF service, you would type:

```
$ svccfg -s name-service/switch describe -v config
config                      application
    name: config
    type: application
    required: true
    target: this
    description: Name service switch configuration data as described in nsswitch.conf
(5).
config/default              astring             files
    type: astring
```

```
    required: true
    Default configuration database entry.
    visibility: readwrite
    minimum number of values: 1
    maximum number of values: 1
  value: files
config/host               astring             "files dns"
    type: astring
    required: false
    Override configuration of host database lookups.  (both IPv4 and IPv6 hosts)
    visibility: readwrite
    Minimium number of values: 1
    Maximum number of values: 1
  value: files dns
config/value_authorization  astring             solaris.smf.value.name-service.switch
```

The -v option indicates if the property setting is required or not. Thus, the option helps you to determine the properties that you must set when editing configuration profiles.

♦ ♦ ♦ CHAPTER 4

Using a Script to Customize an Installation

Instead of working directly with manifests, use derived manifests instead. This chapter describes derived manifests and their advantages. It also explains how to use them for automated install operations.

The chapter covers the following topics:

About Derived Manifests

A derived manifest is an AI manifest that is controlled by a script. Accordingly, the script is called a *derived manifests script*.

The script uses a base AI manifest from which it creates modified manifests when the installation is run. During AI, the script discovers client attributes. Based on those attributes, the script revises the manifest with corresponding definitions to be implemented on the client.

Derived manifests offer the following benefits:

- Efficiency

  With derived manifests, you do not have to maintain multiple manifests. Instead you will be working potentially with fewer scripts that control multiple manifests.
- Flexibility

  Scripts create and modify manifests dynamically. For a group of system clients with relatively little differences among themselves, the script can easily adjust the manifests accordingly. Thus, installation can be almost identical for these clients.
- Control of system configuration files

  Scripts can also be configured to create configuration profiles that perform post-installation configuration.

Working With Derived Manifests

Using derived manifests for installation follows this sequence of steps:

1. Select a manifest to serve as a base file.
2. Create the script.

   The script would contain special instructions that modify the base manifest before the manifest is applied to a specific client installation. The special instructions consist of `aimanifest` subcommand statements.
3. Assign the script to an install service.

AI executes the script at installation time to produce an instance of an AI manifest. AI also syntactically validates the resulting manifest.

After a successful installation, the derived manifest is copied to `/var/log/install/derived/manifest.xml` on the AI client, and the script is copied to `/var/log/install/derived/manifest_script`.

Preparing the Base Manifest

The first component when working with derived manifests is the base manifest.

▼ How to Create a Base Manifest

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Install Oracle Solaris" on page 15.

1. Create the base manifest through one of two ways:

- **Create an empty XML file.**

  At the top, the blank base manifest must contain the following defined elements:

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

- **Create a copy of an existing manifest.**

  For example:

  ```
  $ cp /usr/share/auto_install/default.xml /var/tmp/aimsample.xml
  ```

2. **Set the variables that identify the base manifest and the location of the script's logs.**
   For example, depending on your shell, you might type the following commands:

```
$ export AIM_MANIFEST=/tmp/aimsample.xml
$ export AIM_LOGFILE=/var/log
```

Next Steps Next you would create the script that would modify `aimsample.xml` during installation according to client attributes.

Creating a Script

To create scripts, use any scripting language that is supported by the image, such as `ksh93` and `python`.

A helpful approach is to use a `ksh` script template that is provided in the AI server.

```
aiserver$ cp /usr/share/auto_install/manifest/default.ksh.tmpl /var/tmp/ai-script
```

To configure manifest parameters, you would add `aimanifest` subcommands statements to new script.

The `aimanifest` command must be available on the system where you run the script. If the command is not available, install the `auto-install-common` package. For more details, see the `aimanifest`(8) man page.

Finally, you would associate the script with an install service and optionally establish criteria for its use.

```
aiserver$ installadm create-manifest -n solaris11_4-i386 -f /var/tmp/ai-script \
-m x86manifest
```

Note - You can also use the script to create a configuration profile to apply during installation. For an example, see “Adding a System Configuration Profile” on page 59.

`aimanifest` Subcommands

The `aimanifest` command supports the following subcommands.

`aimanifest load` *file*

Loads an existing XML manifest file. That file's contents would replace the contents of `AIM_MANIFEST`.

The first `aimanifest` command statement on the script must be to load the `AIM_MANIFEST` file, whether a fully developed manifest or an empty file containing only the bare minimum required elements. Subsequent `aimanifest` commands that modify the manifest use the DTD to determine where to add elements in the developing manifest.

`aimanifest add` *path*

Adds new elements to the base manifest. The variable *path* refers to a node in an XML hierarchy of elements and attributes.

`aimanifest set` *path*

Changes the value of an element or attribute in the manifest.

`aimanifest get` *path*

Retrieves the value of each element or attribute which matches *path*. If an element or attribute has no defined value, an empty string (`""`) is displayed

The following example returns the action of the `software_data` element that contains the package name `pkg:/entire`.

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

`aimanifest delete` *path*

Deletes all nodes or attributes which match *path*. If a path specifies nodes, delete the subtree rooted at each node which matches path. If a path matches attributes, delete all attributes which match path; do not delete any nodes.

A derived manifest should also contain tests that would stop the install operation when errors are encountered. The template script already contains this test.

Retrieving Client Attributes

In the derived manifests script, the `aimanifest` subcommands work with defined environment variables that represent client attributes. These variables are listed in the following table:

TABLE 1 AI Client Attribute Environment Variables

Environment Variable Name	Description
`SI_ARCH`	The architecture of the AI client to be installed. Equivalent to the output of `uname -p`.
`SI_CONFIG_PROFILE_DIR`	The directory where user supplied system configuration profiles may be stored and used by the install service.

Environment Variable Name	Description
SI_CPU	The ISA or processor type of the AI client to be installed. Equivalent of the output of `uname -p`.
SI_DISKNAME_#	A flat set of variables representing the names of the disks found on the AI client. There will exist SI_NUMDISKS number of SI_DISKNAME_# variables, where the # is replaced by an integer starting at 1, up to SI_NUMDISKS. This set of variables correlates with the set of variables described by SI_DISKSIZE_#.
SI_DISKSIZE_#	A flat set of variables representing the disk sizes of the disks found on the AI client. There will exist SI_NUMDISKS number of SI_DISKSIZE_#variables, where the # is replaced by an integer starting at 1, up to SI_NUMDISKS. This set of variables correlates with the set of variables described by SI_DISKNAME_#. The sizes are integer numbers of megabytes.
SI_HOSTADDRESS	The IP address of the AI client as set in the install environment.
SI_HOSTNAME	The host name of the AI client as set in the install environment.
SI_INSTALL_PROFILE_DIR	The directory where user supplied system configuration profiles for the install environment may be stored and used by the install service.
SI_INSTALL_SERVICE	The name of the install service used to obtain the manifest script. This environment variable has a value only for network boots, not for media boots.
SI_KARCH	The kernel architecture of the AI client. Equivalent to the output of `uname -m`.
SI_MEMSIZE	The amount of physical memory on the AI client. The size is an integer number of megabytes.
SI_NATISA	The native instruction set architecture of the AI client. Equivalent to the output of `isainfo -n`.
SI_NETWORK	The network number of the AI client. The network number is (IP_ADDR & *netmask*).
SI_NUMDISKS	The number of disks on the AI client.
SI_PLATFORM (or SI_MODEL)	The platform of the AI client. Equivalent to the output of `uname -i` for x86 systems and `prtconf -b` for SPARC systems.
SI_SYSPKG	The release of the Oracle Solaris incorporation package on the AI client (currently named `entire`). If the AI client's entire package is `pkg://solaris/entire@0.5.11,5.11-0.175.0.0.0.2.0:20111020T143822Z`, the value of SI_SYSPKG would be `pkg:/entire@0.5.11-0.175.0`. For an update release or SRU, if the AI client's entire pkg is `pkg://solaris/entire@0.5.11,5.11-0.175.1.19.0.6.0:20140508T221351Z`, the value of SI_SYSPKG would be `pkg:/entire@0.5.11-0.175.1`.

For examples of derived manifests scripts, see "Examples of Derived Manifests Scripts" on page 50.

How to Test the Script in an Install Environment

This procedure describes how to test the derived manifests script on one of the intended systems without running the full installation process.

Before You Begin Make sure to set environment variables in the test environment with values that represent the systems that will be installed using the script. Use the sample file `/usr/share/auto_install/derived_manifest_test_env.sh` as a template. Change the values as applicable.

Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Install Oracle Solaris" on page 15.

1. **Boot an AI image on a system.**

   The boot menu appears.

2. **From the boot menu, select `Text Installer and command line`, then select `Shell`.**

3. **Copy your script from the AI server.**

   Use `wget` or `sftp` to copy your script from the AI server.

4. **Debug the script.**

   Use one of the following methods to debug the script:

   - **Run the script manually.**

     Type each individual `aimanifest` command statement one by one and see the results.

   - **Run AI in a test mode.**

     Use the following command to run AI in a test mode:

     ```
     $ auto-install -m script -i
     ```

5. **Inspect the AI log file `/system/volatile/install_log`.**

   The following line indicates that the script validates correctly:

   ```
   Derived Manifest Module: XML validation completed successfully
   ```

6. **If you made corrections to the script, copy the script back to the AI server.**

Examples of Derived Manifests Scripts

This section contains examples of derived manifests scripts. The scripts determine client attributes and use that information to customize the AI manifest. These examples do not necessarily include all the information required to produce a valid AI manifest.

Specifying Disk Partitioning Based on Disk Size

This script customizes the base manifest to use only half of the target disk on an Oracle Solaris `fdisk` partition if the size of the disk is greater than 1 TB. Try setting `SI_DISKSIZE_1` to less than 1 TB and then greater than 1 TB for different runs of this script. Also set `SI_NUMDISKS` and `SI_DISKNAME_1` before you run the script. Note that this script is only for use with x86 clients because the specified partitioning only applies to x86 clients.

```
#!/bin/ksh93

SCRIPT_SUCCESS=0
SCRIPT_FAILURE=1

function handler
{
    exit $SCRIPT_FAILURE
}

trap handler ERR

/usr/bin/aimanifest load /usr/share/auto_install/manifest/default.xml

# Check that there is only one disk on the system.
if [[ $SI_NUMDISKS -gt "1" ]] ; then
    print -u2 "System has too many disks for this script."
    exit $SCRIPT_FAILURE
fi

/usr/bin/aimanifest add \
    /auto_install/ai_instance/target/disk/disk_name@name $SI_DISKNAME_1

if [[ $SI_DISKSIZE_1 -gt "1048576" ]] ; then
    typeset -i PARTN_SIZE=$SI_DISKSIZE_1/2

    # Default action is to create.
    /usr/bin/aimanifest add \
        /auto_install/ai_instance/target/disk[disk_name@name=\"$SI_DISKNAME_1\"]/
partition@name 1
    /usr/bin/aimanifest add \
        /auto_install/ai_instance/target/disk/partition[@name=1]/size@val \
        ${PARTN_SIZE}mb
else
    /usr/bin/aimanifest add \
        /auto_install/ai_instance/target/disk[disk_name@name=\"$SI_DISKNAME_1\"]/
partition@action \
        use_existing_solaris2
fi
```

```
exit $SCRIPT_SUCCESS
```

For AI clients where the value of `SI_DISKSIZE_1` is less than or equal to 1048576, the following elements are added to `$AIM_MANIFEST`:

```
<target>
  <disk>
    <disk_name name="/dev/dsk/c0t0d0s0"/>
    <partition action="use_existing_solaris2"/>
  </disk>
  <!-- <logical> section -->
</target>
```

For AI clients where the value of `SI_DISKSIZE_1` is greater than 1048576, elements similar to the following are added to `$AIM_MANIFEST`, depending on the value of `SI_DISKSIZE_1`:

```
<target>
  <disk>
    <disk_name name="/dev/dsk/c0t0d0s0"/>
    <partition name="1">
      <size val="524288mb"/>
    </partition>
  </disk>
  <!-- <logical> section -->
</target>
```

The `disk_name` is specified in the command to add the partition to avoid creating a separate disk specification for the partition. The script in this example specifies that the partition is on the `$SI_DISKNAME_1` disk, not on a different disk. If the appropriate lines in this example are replaced by the following lines, you do not get the result you intend:

```
    /usr/bin/aimanifest add \
        /auto_install/ai_instance/target/disk/partition@name 1
    /usr/bin/aimanifest add \
        /auto_install/ai_instance/target/disk/partition[@name=1]/size@val \
        ${PARTN_SIZE}mb
else
    /usr/bin/aimanifest add \
        /auto_install/ai_instance/target/disk/partition@action \
        use_existing_solaris2
```

Instead of the output shown above, this script would give you the following incorrect output:

```
<target>
  <disk>
    <disk_name name="c0t0d0s0"/>
  </disk>
  <disk>
    <partition name="1">
```

```
      <size val="524288mb"/>
    </partition>
  </disk>
</target>
```

Specifying the Root Pool Layout Based on the Existence of Additional Disks

This script customizes the AI manifest to configure a mirror of the root pool if a second disk exists, and configure a three-way mirror if a third disk exists. Set `SI_NUMDISKS` and `SI_DISKNAME_1` before you run the script. Set `SI_DISKNAME_2`, `SI_DISKNAME_3`, and any others as necessary, depending on the value you set for `SI_NUMDISKS`. These environment variables will be set and available to derived manifests scripts during AI installations.

This example demonstrates using the `aimanifest` return path (`-r` option). See the `aimanifest(8)` man page for more information about the return path.

```
#!/bin/ksh93

SCRIPT_SUCCESS=0
SCRIPT_FAILURE=1

function handler
{
    exit $SCRIPT_FAILURE
}

trap handler ERR

/usr/bin/aimanifest load /usr/share/auto_install/manifest/default.xml

# Use the default if there is only one disk.
if [[ $SI_NUMDISKS -ge 2 ]] ; then
    typeset -i disk_num

    # Turn on mirroring. Assumes a root zpool is already set up.
    vdev=$(/usr/bin/aimanifest add -r \
        target/logical/zpool[@name=rpool]/vdev@name mirror_vdev)
    /usr/bin/aimanifest set ${vdev}@redundancy mirror

    for ((disk_num = 1; disk_num <= $SI_NUMDISKS; disk_num++)) ; do
        eval curr_disk="$"SI_DISKNAME_${disk_num}
        disk=$(/usr/bin/aimanifest add -r target/disk@in_vdev mirror_vdev)
        /usr/bin/aimanifest set ${disk}@in_zpool rpool
        /usr/bin/aimanifest set ${disk}@whole_disk true
```

```
        disk_name=$(/usr/bin/aimanifest add -r \
            ${disk}/disk_name@name $curr_disk)
        /usr/bin/aimanifest set ${disk_name}@name_type ctd
    done
fi
exit $SCRIPT_SUCCESS
```

For a system with two disks named `c0t0d0` and `c0t1d0`, the output of this example is the following XML element:

```
<target>
  <disk in_vdev="mirror_vdev" in_zpool="rpool" whole_disk="true">
    <disk_name name="c0t0d0" name_type="ctd"/>
  </disk>
  <disk in_vdev="mirror_vdev" in_zpool="rpool" whole_disk="true">
    <disk_name name="c0t1d0" name_type="ctd"/>
  </disk>
  <logical>
    <zpool name="rpool" is_root="true">
      <vdev name="mirror_vdev" redundancy="mirror"/>
      <filesystem name="export" mountpoint="/export"/>
      <filesystem name="export/home"/>
      <be name="solaris"/>
    </zpool>
  </logical>
</target>
```

Specifying a Mirrored Configuration If at Least Two Disks of a Specified Size Are Present

This script customizes the AI manifest to specify a mirrored configuration if the system has at least two 200 GB disks. Use the first two disks found that are at least 200 GB. Set `SI_NUMDISKS`, `SI_DISKNAME_1`, and `SI_DISKSIZE_1` in your test environment before you run the script. Also set `SI_DISKNAME_2`, `SI_DISKSIZE_2`, and any others as necessary, depending on the value you set for `SI_NUMDISKS`. These environment variables will be set and available to derived manifests scripts during AI installations.

The example shows how to modify a node when more than one node with the same path is present. The shell implementation uses the return path (`-r`) option of `aimanifest` to return the path to a specific node, and uses that path to make additional modifications to the same node. The Python implementation demonstrates the use of subpathing (using `[]` inside a node path) to make additional modifications to the same node.

```
#!/bin/ksh93
```

```
SCRIPT_SUCCESS=0
SCRIPT_FAILURE=1

function handler
{
    exit $SCRIPT_FAILURE
}

trap handler ERR

# Find the disks first.
typeset found_1
typeset found_2
typeset -i disk_num

for ((disk_num = 1; disk_num <= $SI_NUMDISKS; disk_num++)) ; do
    eval curr_disk="$"SI_DISKNAME_${disk_num}
    eval curr_disk_size="$"SI_DISKSIZE_${disk_num}
    if [[ $curr_disk_size -ge "204800" ]] ; then
        if [ -z $found_1 ] ; then
            found_1=$curr_disk
        else
            found_2=$curr_disk
            break
        fi
    fi
done

# Now, install them into the manifest.
# Let the installer take the default action if two large disks are not found.

/usr/bin/aimanifest load /usr/share/auto_install/manifest/default.xml

if [[ -n $found_2 ]] ; then
    # Turn on mirroring.
    vdev=$(/usr/bin/aimanifest add -r \
        /auto_install/ai_instance/target/logical/zpool/vdev@redundancy mirror)
    /usr/bin/aimanifest set ${vdev}@name mirror_vdev

    disk=$(/usr/bin/aimanifest add -r \
        /auto_install/ai_instance/target/disk@in_vdev mirror_vdev)
    disk_name=$(/usr/bin/aimanifest add -r ${disk}/disk_name@name $found_1)
    /usr/bin/aimanifest set ${disk_name}@name_type ctd

    disk=$(/usr/bin/aimanifest add -r \
        /auto_install/ai_instance/target/disk@in_vdev mirror_vdev)
    disk_name=$(/usr/bin/aimanifest add -r ${disk}/disk_name@name $found_2)
    /usr/bin/aimanifest set ${disk_name}@name_type ctd
```

```
fi

exit $SCRIPT_SUCCESS
```

The following script is a Python version of the preceding Korn shell version.

```
#!/usr/bin/python2.6

import os
import sys

from subprocess import check_call, CalledProcessError

SCRIPT_SUCCESS = 0
SCRIPT_FAILURE = 1

def main():

    # Find the disks first.
    found_1 = ""
    found_2 = ""

    si_numdisks = int(os.environ["SI_NUMDISKS"])
    for disk_num in range(1, si_numdisks + 1):
        curr_disk_var = "SI_DISKNAME_" + str(disk_num)
        curr_disk = os.environ[curr_disk_var]
        curr_disk_size_var = "SI_DISKSIZE_" + str(disk_num)
        curr_disk_size = os.environ[curr_disk_size_var]
        if curr_disk_size >= "204800":
            if not len(found_1):
                found_1 = curr_disk
            else:
                found_2 = curr_disk
                break

    # Now, write the disk specifications into the manifest.
    # Let the installer take the default action if two large disks are not found.

    try:
        check_call(["/usr/bin/aimanifest", "load",
            "/usr/share/auto_install/manifest/default.xml"])
    except CalledProcessError as err:
        sys.exit(err.returncode)

    if len(found_2):
        try:
            check_call(["/usr/bin/aimanifest", "add",
               "target/logical/zpool[@name=rpool]/vdev@redundancy", "mirror"])
            check_call(["/usr/bin/aimanifest", "set",
```

```
               "target/logical/zpool/vdev[@redundancy='mirror']@name", "mirror_vdev"])

            check_call(["/usr/bin/aimanifest", "add",
                "target/disk/disk_name@name", found_1])
            check_call(["/usr/bin/aimanifest", "set",
                "target/disk/disk_name[@name='" + found_1 + "']" + "@name_type", "ctd"])
            check_call(["/usr/bin/aimanifest", "set",
                "target/disk[disk_name@name='" + found_1 + "']" + "@in_vdev",
 "mirror_vdev"])

            check_call(["/usr/bin/aimanifest", "add",
                "target/disk/disk_name@name", found_2])
            check_call(["/usr/bin/aimanifest", "set",
                "target/disk/disk_name[@name='" + found_2 + "']" + "@name_type", "ctd"])
            check_call(["/usr/bin/aimanifest", "set",
                "target/disk[disk_name@name='" + found_2 + "']" + "@in_vdev",
 "mirror_vdev"])
        except CalledProcessError as err:
            sys.exit(err.returncode)

    sys.exit(SCRIPT_SUCCESS)

if __name__ == "__main__":
    main()
```

Specifying Packages to Install Based on IP Address

This script customizes the AI manifest to install one package if the IP address of the AI client is in a specified range, and install a different package if the IP address of the AI client is in a different range. Set `SI_HOSTADDRESS` in your test environment before you run the script. This environment variable will be set and available to derived manifests scripts during AI installations.

```
#!/bin/ksh93

SCRIPT_SUCCESS=0
SCRIPT_FAILURE=1

function handler
{
    exit $SCRIPT_FAILURE
}

trap handler ERR
```

```
/usr/bin/aimanifest load /usr/share/auto_install/manifest/default.xml

# First determine which range the host IP address of the client is in.
echo $SI_HOSTADDRESS | sed 's/\./ /g' | read a b c d

# Assume all systems are on the same class A and B subnets.

# If the system is on class C subnet = 100, then install the /pkg100 package.
# If the system is on class C subnet = 101, then install the /pkg101 package.
# Otherwise, do not install any other additional package.

if ((c == 100)) ; then
    /usr/bin/aimanifest add \
    software/software_data[@action='install']/name pkg:/pkg100
fi
if ((c == 101)) ; then
    /usr/bin/aimanifest add \
    software/software_data[@action='install']/name pkg:/pkg101
fi

exit $SCRIPT_SUCCESS
```

Specifying that the Target Disk Must Be At Least a Certain Size

This script customizes the AI manifest to install only on a disk that is at least 50 GB. Ignore smaller disks. Set `SI_NUMDISKS`, `SI_DISKNAME_1`, and `SI_DISKSIZE_1` in your test environment before you run the script. Also set `SI_DISKNAME_2`, `SI_DISKSIZE_2`, and any others as necessary, depending on the value you set for `SI_NUMDISKS`. These environment variables will be set and available to derived manifests scripts during AI installations.

```
#!/bin/ksh93

SCRIPT_SUCCESS=0
SCRIPT_FAILURE=1

function handler
{
    exit $SCRIPT_FAILURE
}

trap handler ERR

/usr/bin/aimanifest load /usr/share/auto_install/manifest/default.xml
```

```
typeset found
typeset -i disk_num
for ((disk_num = 1; disk_num <= $SI_NUMDISKS; disk_num++)) ; do
    eval curr_disk="$"SI_DISKNAME_${disk_num}
    eval curr_disk_size="$"SI_DISKSIZE_${disk_num}
    if [[ $curr_disk_size -ge "512000" ]] ; then
        found=$curr_disk
        /usr/bin/aimanifest add \
            /auto_install/ai_instance/target/disk/disk_name@name $found
        break
    fi
done

if [[ -z $found ]] ; then
    exit $SCRIPT_FAILURE
fi

exit $SCRIPT_SUCCESS
```

Adding a System Configuration Profile

Sometimes a system configuration change is needed for each AI client. Rather than having to create an individual system configuration profile on the AI server for each AI client, you could configure a derived manifests script to create the profile for you. The profile must be stored in `/system/volatile/profile` in order for the install service to be able to use it. In this script the settings for the local default router are used when the AI client is reconfigured.

```
ROUTER-CONFIG=/system/volatile/profile/router-config.xml
ROUTER=`netstat -rn | grep "^default" | awk '{print $2}'`

cat<<EOF>${ROUTER-CONFIG}
<!DOCTYPE service_bundle SYSTEM "/usr/share/lib/xml/dtd/service_bundle.dtd.1">
 <service_bundle type="profile" name="router">
   <service name="network/install" version="1" type="service">
     <instance name="default" enabled="true">
       <property_group name="install_ipv4_interface" type="application">
         <propval name="default_route" type="net_address_v4" value="${ROUTER}"/>
       </property_group>
     </instance>
   </service>
 </service_bundle>
EOF
```

Script With Incorrect Manifest Specifications

This script example contains errors.

```
#!/bin/ksh93

SCRIPT_SUCCESS=0
SCRIPT_FAILURE=1

function handler
{
    exit $SCRIPT_FAILURE
}

trap handler ERR

/usr/bin/aimanifest load /usr/share/auto_install/manifest/default.xml

/usr/bin/aimanifest set \
    software[@type="IPS"]/software_data/name pkg:/driver/pcmcia
/usr/bin/aimanifest set \
    software/software_data[@name=pkg:/driver/pcmcia]@action uninstall

return $SCRIPT_SUCCESS
```

This example has three problems with writing to `$AIM_MANIFEST`.

- The `set` subcommand of `aimanifest` can change the value of an existing element or attribute or create a new attribute. The `set` subcommand cannot create a new element. The first `set` subcommand attempts to modify an existing package name in the manifest instead of creating a new package name. If more than one package name already exists in the manifest, an ambiguity error results because the package to be modified cannot be determined. The first `set` subcommand in this example should have been an `add` subcommand.
- In the second `set` subcommand in this example, an element `name` with value `pkg:/driver/pcmcia` is specified with a preceding @ sign. Although attribute values are specified with a preceding @ sign, element values are not.
- The value `pkg:/driver/pcmcia` should be enclosed in quotation marks. Values with slashes or other special characters must be quoted.

The following lines should replace the two `set` lines in this example:

```
/usr/bin/aimanifest add \
    software[@type="IPS"]/software_data@action uninstall
/usr/bin/aimanifest add \
    software/software_data[@action=uninstall]/name pkg:/driver/pcmcia
```

These two `add` subcommands add the following lines to the end of the `software` section of the manifest that is being written:

```
<software_data action="uninstall">
  <name>pkg:/driver/pcmcia</name>
</software_data>
```

♦ ♦ ♦ CHAPTER 5

Specifying Criteria for AI Manifests and System Configuration Profiles

Defining criteria is important in customized installations. This chapter describes how the install service uses criteria so that the correct AI manifest and configuration files are implemented on the correct AI clients. The chapter covers the following topics:

About Manifest Selection During Installation

When you create an install service for the first time, a default manifest (`default.xml`) is automatically created and associated with that service. All clients of the same architecture for which the service was created will use the default manifest during installation.

With multiple and customized manifests, you would need to define criteria to associate manifests with the correct clients. Clients can use only one manifest in one installation instance. Thus, the criteria ensure that the correct clients use a specific manifest and that non-matching systems are excluded.

Keywords that define criteria form a hierarchy. If a client matches criteria defined for multiple manifests, the manifest with the higher priority keyword will be used. The `installadm` tool verifies that criteria of the same type do not overlap.

Suppose that for a SPARC service, two manifests are assigned. For `ManifestA`, SPARC M8 servers constitute a platform criterion. `ManifestB` specifies an address for a MAC criterion, which has a higher priority. A SPARC M8 client whose MAC address matches the MAC criterion will use `ManifestB`, even though the client also matches the platform criterion of `ManifestA`.

For the list of keywords and their priorities, see Table 2, “Criteria Keywords and Criteria Hierarchy,” on page 65.

About Configuration Profile Selection During Installation

No rules determine which profile is used during installation. An install service can have multiple associated profiles. If a client matches the criteria defined for several of a service's profiles, then the configuration in all of those profiles are applied to the client.

For example, two profiles separately have host name and memory size criteria. If a client matches both criteria, then the install service uses both profiles on the client.

Defining Criteria for Manifests and Profiles

The following commands enable you to define criteria for manifests and profiles:

- `installadm create-manifest`
- `installadm create-profile`
- `installadm set-criteria`

Some criteria keywords accept both individual values or a range of values. To specify no limit to a range, use `unbounded` as the end limit. For example, a memory definition of `4096-unbounded` means 4 MB or greater.

To specify criteria, you use one of two options with the commands:

- `-c` *criteria*

  With this option, you specify the criteria definition in the command line. For example:

  ```
  $ installadm create-manifest -c mac="aa:bb:cc:dd:ee:ff" other-options
  $ installadm create-profile -c mem="2048-unbounded" -c zonename="zone1 zone2" \
      other-options
  ```

- `-C` *criteria-file*

  With this option, all criteria definitions are in an XML file. In the command, you simply refer to the file. For example:

  ```
  $ installadm set-criteria -C /var/tmp/mycriteria.xml other-options
  ```

Use any XML editor to create a file. To specify a value or a series of value for a criterion, use the `<value>` tag. For a range of values, use the `<range>` tag.

The following is an example of a criteria file's contents:

```
<?xml version="1.0" encoding="UTF-8"?>
<ai_criteria_manifest>
   <ai_criteria name="mem">
      <range>2048
      unbounded</range>
   </ai_criteria>
   <ai_criteria name="platform">
      <value>SUNW,SPARC Enterprise</value>
   </ai_criteria>
   <ai_criteria name="cpu">
      <value>sparc</value>
   </ai_criteria>
   <ai_criteria name="hostname">
      <value>host1 host3 host5</value>
   </ai_criteria>
</ai_criteria_manifest>
```

The following table lists the keywords for defining criteria in an XML file.

TABLE 2 Criteria Keywords and Criteria Hierarchy

Priority	Criteria Keyword	Description	Command Line and XML File Examples

1	`mac`	Hexadecimal MAC address with colon (:) separators, or range of MAC addresses	**CLI:**
			`-c mac=0:14:4F:20:53:94[-0:14:4F:20:53:A0]`
			XML:
			`<ai_criteria name="mac">`
			`  <value>0:14:4F:20:53:97</value>`
			`</ai_criteria>`
			or
			`<ai_criteria name="mac">`
			`  <range>`
			`  0:14:4F:20:53:94`
			`  0:14:4F:20:53:A0`
			`  </range>`
			`</ai_criteria>`

2	`ipv4`	IP version 4 network address, or range of IP addresses	**CLI:** `-c ipv4="192.0.2.5[-192.0.2.10]"`

Priority	Criteria Keyword	Description	Command Line and XML File Examples

			XML:
			`<ai_criteria name="ipv4">`
			`  <value>192.0.2.5</value>`
			`</ai_criteria>`
			or
			`<ai_criteria name="ipv4">`
			`  <range>`
			`  192.0.2.5`
			`  192.0.2.10`
			`  </range>`
			`</ai_criteria>`

3	`platform`	Platform name returned by `uname -i` for x86 systems and `prtconf -b` for SPARC systems
		Values include:
		`i86pc`
		`SUNW,SPARC-Enterprise` for M4000 and M5000 servers
		`ORCL, SPARC-T4-2` for T4 servers
			CLI:
			`-c platform="SUNW,SPARC-Enterprise"`
			XML:
			`<ai_criteria name="platform">`
			`  <value>SUNW,SPARC-Enterprise</value>`
			`</ai_criteria>`

4	`arch`	Architecture returned by `uname -m`
		Values: `i86pc`, `sun4u`, or `sun4v`
			CLI:
			`-c arch="i86pc"`
			XML:
			`<ai_criteria name="arch">`
			`  <value>i86pc</value>`
			`</ai_criteria>`

5	`cpu`	CPU class returned by `uname -p`
		Values: `i386` or `sparc`
			CLI:
			`-c cpu="sparc"`
			XML:
			`<ai_criteria name="cpu">`
			`  <value>sparc</value>`
			`</ai_criteria>`

6	`network`	IP version 4 network number, or a range of network numbers	**CLI:**
			`-c network="`*10.0.0.0*`[-`*11.0.0.0*`]"`
			XML:
			`<ai_criteria name="network">`
			`  <value>`*10.0.0.0*`</value>`
			`</ai_criteria>`
			or
			`<ai_criteria name="network">`

Priority	Criteria Keyword	Description	Command Line and XML File Examples
			<range> *10.0.0.0* *11.0.0.0* </range> </ai_criteria>
7	mem	Memory size in megabytes returned by prtconf, or a range of memory size The unbounded keyword indicates no upper limit in a range.	**CLI:** -c mem="4096[-unbounded]" **XML:** <ai_criteria name="mem"> <value>4096</value> </ai_criteria> or <ai_criteria name="mem"> <range> 2048 unbounded </range> </ai_criteria>
8	zonename	Name or list of names of zones as shown by zoneadm list.	**CLI:** -c zonename="zoneA[zoneB zoneC]" **XML:** <ai_criteria name="zonename"> <value>zoneA[zoneB zoneC]</ value> </ai_criteria>
9	hostname	Client host name or list of client host names.	**CLI:** -c hostname="host1 [host2 host6 ...]" **XML:** <ai_criteria name="hostname"> <value>host1 [host host6 …]</ value> </ai_criteria>

♦ ♦ ♦ CHAPTER 6

Running a Custom Script During First Boot

This chapter describes how you can create a script that can perform additional actions that cannot be completed by the AI manifest or a system configuration file. It covers the following topics:

Automatically Creating a First-boot Service and Package

This section covers some of the options when running the `svc-create-first-boot` command to create a first-boot service and package. For these examples, all changes are handled by command line options. No manual editing is needed.

EXAMPLE 4 Creating a First-boot Service and Package in Command Line Mode

In the following example, a service is created using the `first-boot-script.sh` script. Any executable script can be used. The archive will be stored in `myp5p.p5p`.

```
$ /usr/sbin/svc-create-first-boot -s first-boot-script.sh -d myp5p.p5p
```

EXAMPLE 5 Creating a First-boot Service and Package in Command Line Mode With Customizations

In the following example, a service is created using the `first-boot-script.sh` script. The network milestone is added as a dependency. The archive will be stored in `myp5p.p5p`.

```
$ /usr/sbin/svc-create-first-boot -s first-boot-script.sh -o service-
dependency=milestone/network -d myp5p.p5p
```

EXAMPLE 6 Shell Script to Use in a First-boot Service

The contents of the script that is included in the first-boot service can be any regular executable script. For instance, to configure two interfaces, you could use a script like the following:

```
!/bin/ksh
# Create and configure addresses on two IP interfaces
/usr/sbin/ipadm create-ip net0
/usr/sbin/ipadm create-ip net1
/usr/bin/ipadm create-addr -a 10.153.125.222/24 net0
/usr/bin/ipadm create-addr -a 10.169.254.182.77/24 net1
```

Interactively Creating a First-boot Service Package

You can interactively create an IPS package containing an SMF service to run a first-boot script, by running the `svc-create-boot-script` command without any arguments. You will need to provide the following information:

- Path to the first-boot script.
- FMRI of the first-boot service - The default is svc:/site/first-boot-svc.
- Select if the dependencies need to be customized - if you select No, the multiuser milestone is selected. If you select Yes, the following questions are presented;
  - Does your first boot service configure the network or require access to the network? - If you select Yes, the network milestone is added as a dependency.
  - Does your first boot service need to lookup non-local hosts or user names? - If you select Yes, the `name-services` milestone is added as a dependency.
  - Does your first boot service need to access any files outside of the root file system? - If you select Yes, the `multi-user` milestone is added as a dependency.
  - Enter any additional dependency.
- Select if your first-boot script should be rerun on cloned instances.
- Enter the method script timeout in seconds - the default is `60` seconds.
- Enter the package FMRI - the default FMRI is `first-boot-svc`.
- Enter the publisher name - the default name is `firstboot`.
- Enter the URI to an existing repository or the path to the generated p5p archive.

Manually Creating a Script to Run at First Boot

To know what source you can use for your script, you need to know what tools are installed on the AI client at first boot. The `solaris-large-server` group package is installed by default which includes tools that become available at first boot, such as Python, `bash`, and `ksh`. For a complete list of packages that are included in the group package, use the `pkg contents` command as described in "Listing All Installable Packages in a Group Package" in *Updating Systems and Adding Software in Oracle Solaris 11.4*. If you want to use a source for your script that is not available in the `solaris-large-server` package, identify the package you need and specify it in the AI manifest. For information about how to find the names of other packages that you might want to install, see *Updating Systems and Adding Software in Oracle Solaris 11.4*.

- Use only one first-boot script to avoid having different commands in different scripts that collide with one another.
- Do not reboot in the first-boot script.

Any first-boot script must perform the following foundational actions:

- Load `/lib/svc/share/smf_include.sh` in order to use definitions such as SMF method exit codes.
- Test whether the script already ran in a prior boot and if true, exit the process.

  The test is performed by the following line in the script:

  ```
  completed=`svcprop -p config/completed site/first-boot-script-svc:default`
  ```

  The test consists of two steps:

  - Obtain the current value of the `config/completed` property of the `site/first-boot-script-svc` service.
  - Assign that value to the local `completed` variable in the script.

  If the property's value is `true`, then an exit code is sent to end the script's process and disable the script. The exit signal is performed by the following line:

  ```
  smf_method_exit $SMF_EXIT_TEMP_DISABLE script_completed "Configuration completed"
  ```
- Save a copy of the boot environment (BE) that was just created by the AI installation.

  The copy serves in recovery. If the first-boot script introduces problems to the system, simply reboot the system to the saved BE. Add the following line to the first-boot script.

  ```
  bename=`beadm list -Hd|nawk -F ';' '$3 ~ /R/ {print $1}'`
  ```

```
beadm create ${bename}.orig
echo "Original boot environment saved as ${bename}.orig"
```

- After the script completes, set the value of the `completed` property to `true`, refresh the service with the new property value, exit the `start` method, and temporarily disable the service.

  Copy the previous test and exit lines and add these to the end of the script.

EXAMPLE 7 Template First-Boot Script

By default, `sh` is `ksh93`.

```
#!/bin/sh

# Load SMF shell support definitions
. /lib/svc/share/smf_include.sh

# If nothing to do, exit with temporary disable
completed=`(svcprop -p config/completed site/first-boot-script-svc:default`
[ "${completed}" = "true" ] && \
    smf_method_exit $SMF_EXIT_TEMP_DISABLE completed "Configuration completed"

# Obtain the active BE name from beadm: The active BE on reboot has an R in
# the third column of 'beadm list' output. Its name is in column one.
bename=`beadm list -Hd|nawk -F ';' '$3 ~ /R/ {print $1}'`
beadm create ${bename}.orig
echo "Original boot environment saved as ${bename}.orig"

# Place your one-time configuration tasks here

# Record that this script's work is done
svccfg -s site/first-boot-script-svc:default setprop config/completed = true
svcadm refresh site/first-boot-script-svc:default

smf_method_exit $SMF_EXIT_TEMP_DISABLE method_completed "Configuration completed"
```

In the middle of the template, marked by the heading `Place your one time configuration tasks here`, you can add additional shell commands to perform other tasks. For example, to configure multiple IP interfaces, add the following lines:

```
# Create and configure addresses on two IP interfaces
/usr/sbin/ipadm create-ip net0
/usr/sbin/ipadm create-ip net1
/usr/sbin/ipadm create-addr -a 203.0.113.5/27 net0
/usr/sbin/ipadm create-addr -a 203.0.113.35/27 net1

# Add a default route with net0 as the gateway
```

```
/usr/sbin/route add default 203.0.113.1 -ifp net0
```

You can also add `useradd` commands to create multiple users on the system.

Tip - Use the `-n` option to check for syntax errors in your script:

```
$ ksh -n first-boot-script.sh
```

Manually Creating a First-boot Service Package

The following procedures show how to manually create the pieces that are used in a first-boot service package:

1. "Using the Manifest Creation Tool to Create an SMF Manifest File Including a First-boot Script" on page 73 - Once you have created the AI service, you can also do the following:
   - "How to Ensure A First-Boot Script is Only Run One Time" on page 76
   - "Customizing the Generated Manifest Created by `svcbundle`" on page 77
2. "How to Create and Publish the IPS Package for a First-Boot Script" on page 79
3. Create an IPS package that contains the service manifest and the script.
4. Add the package to an IPS package repository.
5. Install that package during the AI installation by specifying that package in the AI manifest.

Using the Manifest Creation Tool to Create an SMF Manifest File Including a First-boot Script

Create an SMF manifest file that defines a service that executes a script.

- The `start` method of the service executes the first-boot script.
- This example specifies the `multi-user` dependency to make sure that the first-boot script executes late in the startup sequence after first boot. Depending on what your first-boot script does, you might not need such a dependency. If you do not specify such a dependency, your script might run before the system is adequately configured.

Tip - Evaluate your script's dependencies and construct the service to run the script after its dependencies are satisfied.

- The `completed` property is defined with a value of `false`.

You can use the `svcbundle` command to generate a valid service manifest. In the following example, notice that by default, a manifest generated by the `svcbundle` command specifies a transient service and specifies the `multi-user` dependency.

EXAMPLE 8 Generated SMF Service Manifest

In the following command, the name of the script shown in "Manually Creating a Script to Run at First Boot" on page 71 is specified as the value of `start-method`. The name of the script is specified as `/opt/site/first-boot-script.sh` because the package created in "How to Create and Publish the IPS Package for a First-Boot Script" on page 79 installs the `first-boot-script.sh` script into `/opt/site/first-boot-script.sh`.

In the following command, the `completed` property is specified by a colon-separated list of property group name, property name, property type, and initial property value.

```
$ svcbundle -s service-name=site/first-boot-script-svc \
-s start-method=/opt/site/first-boot-script.sh \
-s instance-property=config:completed:boolean:false \
> first-boot-script-svc-manifest.xml
```

In the generated service manifest shown below, the first-boot script, `/opt/site/first-boot-script.sh`, is the value of the `exec` attribute of the `start` method. The `completed` property is specified in the `instance` element that defines the default instance of this service, `first-boot-script-svc:default`.

```
<?xml version="1.0" ?>
<!DOCTYPE service_bundle
  SYSTEM '/usr/share/lib/xml/dtd/service_bundle.dtd.1'>
<!--
    Manifest created by svcbundle (2014-Jan-14 16:39:30-0700)
-->
<service_bundle type="manifest" name="site/first-boot-script-svc">
    <service version="1" type="service" name="site/first-boot-script-svc">
        <!--
            The following dependency keeps us from starting until the
            multi-user milestone is reached.
        -->
        <dependency restart_on="none" type="service"
            name="multi_user_dependency" grouping="require_all">
            <service_fmri value="svc:/milestone/multi-user"/>
        </dependency>
        <exec_method timeout_seconds="60" type="method" name="start"
            exec="/opt/site/first-boot-script.sh"/>
        <!--
```

```
                The exec attribute below can be changed to a command that SMF
                should execute to stop the service.  See smf_method(7) for more
                details.
            -->
            <exec_method timeout_seconds="60" type="method" name="stop"
                exec=":true"/>
            <!--
                The exec attribute below can be changed to a command that SMF
                should execute when the service is refreshed.  Services are
                typically refreshed when their properties are changed in the
                SMF repository.  See smf_method(7) for more details.  It is
                common to retain the value of :true which means that SMF will
                take no action when the service is refreshed.  Alternatively,
                you may wish to provide a method to reread the SMF repository
                and act on any configuration changes.
            -->
            <exec_method timeout_seconds="60" type="method" name="refresh"
                exec=":true"/>
            <property_group type="framework" name="startd">
                <propval type="astring" name="duration" value="transient"/>
            </property_group>
            <instance enabled="true" name="default">
                <property_group type="application" name="config">
                    <propval type="boolean" name="completed" value="false"/>
                </property_group>
            </instance>
            <template>
                <common_name>
                    <loctext xml:lang="C">
                        <!--
                            Replace this comment with a short name for the
                            service.
                        -->
                    </loctext>
                </common_name>
                <description>
                    <loctext xml:lang="C">
                        <!--
                            Replace this comment with a brief description of
                            the service
                        -->
                    </loctext>
                </description>
            </template>
        </service>
    </service_bundle>
```

How to Ensure A First-Boot Script is Only Run One Time

The following procedure shows how to ensure that the script runs only at the first boot of the newly installed system, and that the script runs only one time.

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Install Oracle Solaris" on page 15.

1. **Create a service to run the script.**

   These easiest way to create this simple service is to use the `svcbundle` command as shown in "Using the Manifest Creation Tool to Create an SMF Manifest File Including a First-boot Script" on page 73.

2. **Set a script completion flag before the script runs.**

   Define a Boolean completion property in the service manifest, and set its value to `false`. See the `completed` property in the manifest in Example 8, "Generated SMF Service Manifest," on page 74.

3. **Set the script completion flag at the end of the script.**

   Use the `svccfg` command to set the `completed` property to `true` at the end of the script. Use the `svcadm` command to refresh the service with the new property value. See the end of the sample script in Example 7, "Template First-Boot Script," on page 72.

4. **Disable the service if the script completed.**

   In the service manifest, the default service instance is created and enabled. The service is disabled in the script. When you exit your first-boot script, use the `SMF_EXIT_TEMP_DISABLE` exit code to exit the `start` method of the service and temporarily disable the service. The service is disabled, and the `stop` method of the service does not run.

   Temporarily disabling the service is preferable to permanently disabling the service so that the service can be more easily re-enabled. In some situations, the script (and therefore the service) must be re-run to update configuration work that was done, such as zone cloning or migration. If the service is permanently disabled, the `svcadm enable` command must be run to re-enable the service.

   Temporarily disabling the service is also preferable to leaving the service online. A service that is online might appear to be doing work on every reboot. In this example, the name of the service is `site/first-boot-script-svc`. After the AI client is booted, you can see the service is in the `disabled` state:

```
$ svcs first-boot-script-svc
```

```
STATE          STIME    FMRI
disabled        8:24:16 svc:/site/first-boot-script-svc:default
```

Customizing the Generated Manifest Created by `svcbundle`

The service manifest generated with the `svcbundle` command might meet your needs with no modification necessary. The following example shows a modification of the service manifest.

If you modify a service manifest, use the `svccfg validate` command to ensure the manifest is still valid.

EXAMPLE 9 Customized Service Manifest: Increase the Time Allowed for the Script to Run

In the following copy of the generated service manifest, the default `exec_method` timeout of 60 seconds has been increased for the `start` method. Make sure the `start` method has adequate time to run the first-boot script.

```
<?xml version="1.0" ?>
<!DOCTYPE service_bundle
  SYSTEM '/usr/share/lib/xml/dtd/service_bundle.dtd.1'>
<!--
    Manifest created by svcbundle (2014-Jan-14 16:39:30-0700)
-->
<service_bundle type="manifest" name="site/first-boot-script-svc">
    <service version="1" type="service" name="site/first-boot-script-svc">
        <!--
            The following dependency keeps us from starting until the
            multi-user milestone is reached.
        -->
        <dependency restart_on="none" type="service"
            name="multi_user_dependency" grouping="require_all">
            <service_fmri value="svc:/milestone/multi-user"/>
        </dependency>
        <!--
            Make sure the start method has adequate time to run the script.
        -->
        <exec_method timeout_seconds="360" type="method" name="start"
            exec="/opt/site/first-boot-script.sh"/>
        <!--
            The exec attribute below can be changed to a command that SMF
            should execute to stop the service.  See smf_method(7) for more
            details.
```

```
        -->
        <exec_method timeout_seconds="60" type="method" name="stop"
            exec=":true"/>
        <!--
            The exec attribute below can be changed to a command that SMF
            should execute when the service is refreshed.  Services are
            typically refreshed when their properties are changed in the
            SMF repository.  See smf_method(7) for more details.  It is
            common to retain the value of :true which means that SMF will
            take no action when the service is refreshed.  Alternatively,
            you may wish to provide a method to reread the SMF repository
            and act on any configuration changes.
        -->
        <exec_method timeout_seconds="60" type="method" name="refresh"
            exec=":true"/>
        <property_group type="framework" name="startd">
            <propval type="astring" name="duration" value="transient"/>
        </property_group>
        <instance enabled="true" name="default">
            <property_group type="application" name="config">
                <propval type="boolean" name="completed" value="false"/>
            </property_group>
        </instance>
        <template>
            <common_name>
                <loctext xml:lang="C">
                    <!--
                        Replace this comment with a short name for the
                        service.
                    -->
                </loctext>
            </common_name>
            <description>
                <loctext xml:lang="C">
                    <!--
                        Replace this comment with a brief description of
                        the service
                    -->
                </loctext>
            </description>
        </template>
    </service>
</service_bundle>

$ svccfg validate first-boot-script-svc-manifest.xml
```

EXAMPLE 10 Customized Service Manifest: Ensure the Script Runs After Non-Global Zones Are Installed

In the following service manifest excerpt, the dependency on `svc:/milestone/multi-user` is changed to a dependency on `svc:/system/zones-install` to ensure that the first-boot script runs after all non-global zones are installed.

```
<!--
    The following dependency keeps us from starting until all
    non-global zones are installed.
-->
<dependency restart_on="none" type="service"
    name="ngz_dependency" grouping="require_all">
    <service_fmri value="svc:/system/zones-install"/>
</dependency>
```

How to Create and Publish the IPS Package for a First-Boot Script

Create an IPS package that contains:

- The service manifest file.
- The first-boot script.
- Any files needed by the script that cannot be provided from another location such as the AI server.

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Install Oracle Solaris" on page 15.

1. Create the directory hierarchy.

In this example, the service manifest is installed into `/lib/svc/manifest/site`, and the first-boot script is installed into `/opt/site`.

```
$ mkdir -p proto/lib/svc/manifest/site
$ mkdir -p proto/opt/site
$ cp first-boot-script-svc-manifest.xml proto/lib/svc/manifest/site
$ cp first-boot-script.sh proto/opt/site
```

2. Create the package manifest.

Create the following file named `first-boot-script.p5m`.

```
set name=pkg.fmri value=first-boot-script@1.0,5.11-0
set name=pkg.summary value="AI first-boot script"
set name=pkg.description value="Script that runs at first boot after AI installation"
set name=info.classification value=\
    "org.opensolaris.category.2008:System/Administration and Configuration"
file lib/svc/manifest/site/first-boot-script-svc-manifest.xml \
    path=lib/svc/manifest/site/first-boot-script-svc-manifest.xml owner=root \
    group=sys mode=0444
dir  path=opt/site owner=root group=sys mode=0755
file opt/site/first-boot-script.sh path=opt/site/first-boot-script.sh \
    owner=root group=sys mode=0555
```

Depending on what your first-boot script does, you might need to specify dependencies. If you modify this manifest, verify the new manifest is correct. You can ignore warnings. See Chapter 2, “Packaging Software With IPS” in *Packaging and Delivering Software With the Image Packaging System in Oracle Solaris 11.4* for information about how to create a package, including information about the `pkgdepend`, `pkgmogrify`, and `pkglint` commands.

3. Create the repository for the package.

This example creates the repository in the local directory, with `firstboot` as the publisher.

Note - Create the repository in a directory that is accessible by the AI clients at installation time.

```
$ pkgrepo create firstbootrepo
$ pkgrepo -s firstbootrepo add-publisher firstboot
```

4. Publish the package.

```
$ pkgsend publish -d ./proto -s ./firstbootrepo first-boot-script.p5m
pkg://firstboot/first-boot-script@1.0,5.11-0:20140114T022508Z
PUBLISHED
```

AI clients can install the package from the `firstbootrepo` repository. The `firstboot` publisher with `firstbootrepo` origin is defined in the AI manifest as shown in the next section.

5. Verify that the package is available.

List the package to verify that the package is available.

```
$ pkg list -g ./firstbootrepo first-boot-script
NAME (PUBLISHER)                 VERSION    IFO
first-boot-script (firstboot)    1.0-0      ---
```

6. (Optional) Test installation of the package.

The -n option indicates not to install the package.

```
$ pkg set-publisher -g ./firstbootrepo firstboot
$ pkg publisher
PUBLISHER  TYPE    STATUS P LOCATION
solaris    origin  online F http://pkg.oracle.com/solaris/release/
firstboot  origin  online F file:///home/user1/firstboot/firstbootrepo/
$ pkg list -af first-boot-script
NAME (PUBLISHER)                 VERSION    IFO
first-boot-script (firstboot)    1.0-0      ---
$ pkg install -nv first-boot-script
           Packages to install:        1
     Estimated space available: 50.68 GB
Estimated space to be consumed: 64.66 MB
       Create boot environment:       No
Create backup boot environment:       No
          Rebuild boot archive:       No

Changed packages:
firstboot
  first-boot-script
    None -> 1.0,5.11-0:20140114T022508Z
Planning linked: 0/2 done; 1 working: zone:z2
Linked image 'zone:z2' output:
|      Estimated space available: 50.68 GB
| Estimated space to be consumed: 62.07 MB
|           Rebuild boot archive:       No
`
Planning linked: 1/2 done; 1 working: zone:z1
Linked image 'zone:z1' output:
|      Estimated space available: 50.67 GB
| Estimated space to be consumed: 62.07 MB
|           Rebuild boot archive:       No
```

Next Steps See *Creating Package Repositories in Oracle Solaris 11.4* for instructions to make the new repository accessible to AI clients through either NFS sharing or HTTP.

▼ How to Install the First-Boot Script IPS Package

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Install Oracle Solaris" on page 15.

1. Add the package to the AI manifest.

Add the package to the software installation section of the AI manifest. Either customize an AI manifest XML file or write a derived manifests script to add these elements. See Chapter 2, "Working With AI Manifests" for information about customizing an AI manifest.

Use the `installadm export` command to retrieve the content of one or more existing AI manifests. The following example shows the XML elements you need to add.

```
<software type="IPS">
  <source>
    <publisher name="solaris">
      <origin name="http://pkg.oracle.com/solaris/release"/>
    </publisher>
    <publisher name="firstboot">
      <origin name="file:///net/host1/export/firstbootrepo"/>
    </publisher>
  </source>
  <software_data action="install">
    <name>pkg:/first-boot-script</name>
  </software_data>
</software>
```

Make sure the origin is a URI the AI clients can access during AI installation. Use `zfs set sharenfs` to export the repository so that AI clients can access the local repository.

2. **Update the modified AI manifest in the AI install service.**

   Use the `installadm update-manifest` command to replace the AI manifest content with the content that includes the first-boot script package. Any criteria or default status remain with the manifest or script following the update.

3. **Network boot the AI client.**

   Network boot the AI client to use AI to install the Oracle Solaris OS and your custom `first-boot-script` package. When the AI client is booted after installation, the service executes the first-boot script.

▼ How to Update a First-Boot Script or Service

If you change the script or the service manifest, use this procedure to install the update.

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Install Oracle Solaris" on page 15.

1. **Copy the updated files to your prototype directory.**

```
$ cp first-boot-script-svc-manifest.xml proto/lib/svc/manifest/site
$ cp first-boot-script.sh proto/opt/site
```

2. **Increment the package version.**

   In the package manifest, change the value of the `pkg.fmri` attribute to the following, for example:

   ```
   first-boot-script@1.0,5.11-0.1
   ```

3. **Publish the new version.**

   Publish the new version of the package to the repository.

   ```
   $ pkgsend publish -d ./proto -s ./firstbootrepo first-boot-script.p5m
   pkg://firstboot/first-boot-script@1.0,5.11-0.1:2013123T231948Z
   PUBLISHED
   ```

4. **Update the package.**

   Use the `pkg list -af` command to make sure you can access the new version. You might need to use the `pkg refresh firstboot` command to update the package list. Use the `pkg update` command to update the package.

5. **Reboot the test system.**

6. **After booting, verify installed packages.**

   Security best practices recommend that you run the following command to help ensure that packaged file system objects have not been changed insecurely. For more information, see "Verifying Packages and Fixing Verification Errors" in *Updating Systems and Adding Software in Oracle Solaris 11.4*.

   ```
   $ pkg verify -v
   ```

Testing the First-Boot Service

To test the service before you test an AI installation, you can simply install the package on a test system and reboot that test system.

```
$ pkg install first-boot-script
           Packages to install:  1
       Create boot environment: No
Create backup boot environment: No
```

```
DOWNLOAD                                  PKGS       FILES    XFER (MB)   SPEED
Completed                                  1/1         2/2      0.0/0.0    0B/s

PHASE                                        ITEMS
Installing new actions                         7/7
Updating package state database               Done
Updating image state                          Done
Creating fast lookup database                 Done
Reading search index                          Done
$ pkg list first-boot-script
NAME (PUBLISHER)                              VERSION      IFO
first-boot-script (firstboot)                 1.0-0        i--
$ pkg info first-boot-script
          Name: first-boot-script
       Summary: AI first-boot script
   Description: Script that runs at first boot after AI installation
      Category: System/Administration and Configuration
         State: Installed
     Publisher: firstboot
       Version: 1.0
 Build Release: 5.11
        Branch: 0
Packaging Date: Dec 23, 2013 02:50:31 PM
          Size: 3.89 kB
          FMRI: pkg://firstboot/first-boot-script@1.0,5.11-0:20131223T145031Z
```

Reboot the test system. If the script created a new boot environment as shown above, be sure to boot into that new boot environment.

Check that the script is in the `/opt/site` directory and the effects of the script are correct.

Check the state of the service. If the script finished and exited correctly, the service should be in the disabled state.

```
$ svcs first-boot-script-svc
STATE          STIME    FMRI
disabled        8:24:16 svc:/site/first-boot-script-svc:default
```

Use one of the following commands to check the value of the `completed` property:

```
$ svcprop first-boot-script-svc:default
config/completed boolean true
$ svcprop -p config/completed first-boot-script-svc:default
true
```

If you want to review the service log file, use the following command to find the location of the log file:

```
$ svcs -x first-boot-script-svc
```

```
svc:/site/first-boot-script-svc:default (?)
 State: disabled since Dec 23, 2013 08:24:16 AM PDT
Reason: Temporarily disabled by service method: "Configuration completed."
   See: https://support.oracle.com/msg/SMF-8000-1S
   See: /var/svc/log/site-first-boot-script-svc:default.log
Impact: This service is not running.
```

The log file contains the following information:

```
[ Jul 23 08:22:57 Enabled. ]
[ Jul 23 08:24:14 Executing start method ("/opt/site/first-boot-script.sh"). ]
[ Jul 23 08:24:16 Method "start" exited with status 101. ]
[ Jul 23 08:24:16 "start" method requested temporary disable: "Configuration
 completed" ]
[ Jul 23 08:24:16 Rereading configuration. ]
```

♦ ♦ ♦ APPENDIX A

System Configuration Profiles

The first part of this appendix lists the different screens of the System Configuration Interactive tool. The second part shows the contents of the system configuration files that are provided by Oracle Solaris.

Screens of the System Configuration Interactive Tool

This section describes the different screens of the SCI tool to create a system configuration profile with the following command:

```
$ sysconfig create-profile -o directory
```

- `System Identity` – Prompts you for the system name.
- `Network Configuration` – Prompts for the type of network configuration if a network interface is detected.

  If you configure the network, you are further prompted whether you use DHCP or a static address. Further prompts appear depending on your selection.
- `Time Zone: Regions` – Prompts for the region to apply to the system.
- `Time Zone: Locations` – Prompts for the location in the selected region to apply to the system.
- `Time Zone` – Prompts for the time zone of the selected location to apply to the system.
- `Locale: Language` – Prompts for the language to use. The default is English.
- `Locale: Territory` – Prompts for the territory of the selected language.
- `Keyboard` – Prompts for the keyboard language to use.
- `Users` – Prompts for user account information.

  The first field prompts for the root password which is required. The fields for the user account information are optional.
- `System Configuration Summary` – Displays the settings you specified in previous screens.

When you apply the settings, these are saved in `sc_profile.xml` in the directory you specified with the `-o` option.

Templates for System Configuration Profiles

These files are in `/usr/share/auto_install/sc_profiles` on any Oracle Solaris system. Each of the files also contains instructions on settings that you must manually specify to make the files applicable during installation.

Caution - The contents displayed in this appendix are extracts to show only the important property settings. They do not constitute valid XML files.

The following profiles are covered:

- "`sc_sample.xml`"
- "`custom_network.xml`"
- "`enable_sci.xml`"
- "`ib_network.xml`"
- "`ipmp_network.xml`"
- "`static_network.xml`"
- "`vnic_network.xml`"
- "`dns.xml`"
- "`unconfig.xml`"

`sc_sample.xml`

The profile creates a user account, root password, keyboard mapping and timezone settings, and an IPv4 DHCP configuration.

Note - Other templates contain the same configuration settings as found in `sc_sample.xml`, but would have additional parameters as well. This appendix does not show the duplicated `sc_sample` settings in those templates.

```
<service_bundle type="profile" name="system configuration">
    <service name="system/config-user" version="1">
      <instance name="default" enabled="true">
        <property_group name="user_account">
          <propval name="login" value="jack"/>
          <propval name="password" value="9Nd/cwBcNWFZg"/>
```

```
      <propval name="description" value="default_user"/>
      <propval name="shell" value="/usr/bin/bash"/>
      <propval name="gid" value="10"/>
      <propval name="uid" value="101"/>
      <propval name="type" value="normal"/>
      <propval name="roles" value="root"/>
      <propval name="profiles" value="System Administrator"/>
    </property_group>
    <property_group name="root_account">
        <propval name="password"
           value="$5$dnRfcZse$Hx4aBQ161Uvn9ZxJFKMdRiy8tCf4gMT2s2rtkFba2y4"/>
        <propval name="type" value="role"/>
    </property_group>
  </instance>
</service>

<service version="1" name="system/identity">
  <instance enabled="true" name="node">
    <property_group name="config">
       <propval name="nodename" value="solaris"/>
    </property_group>
  </instance>
</service>

<service name="system/console-login" version="1">
  <instance name="default" enabled="true">
    <property_group name="ttymon">
      <propval name="terminal_type" value="sun"/>
    </property_group>
  </instance>
</service>

<service name="system/keymap" version="1">
  <instance name="default" enabled="true">
    <property_group name="keymap">
      <propval name="layout" value="US-English"/>
    </property_group>
  </instance>
</service>

<service name="system/timezone" version="1">
  <instance name="default" enabled="true">
    <property_group name="timezone">
      <propval name="localtime" value="UTC"/>
    </property_group>
  </instance>
</service>
```

```
    <service name="system/environment" version="1">
      <instance name="init" enabled="true">
        <property_group name="environment">
          <propval name="LANG" value="en_US.UTF-8"/>
        </property_group>
      </instance>
    </service>

    <service name="network/ip-interface-management" version="1" type="service">
      <instance name="default" enabled="true">
 <property_group name="interfaces" type="application">

   <!-- interface configuration -->
   <property_group name="{{AI_NETLINK_VANITY}}" type="interface-ip">
            <property name="address-family" type="astring">
              <astring_list>
  <value_node value="ipv4"/>
  <value_node value="ipv6"/>
              </astring_list>
            </property>

     <!-- IPv4 DHCP address -->
            <property_group name="v4" type="address-dhcp"/>
   </property_group>

 </property_group>
      </instance>
    </service>
</service_bundle>
```

custom_network.xml

The profile customizes data link names for network configuration.

```
<service_bundle type="profile" name="system configuration">
    <service name="network/datalink-management" type="service" version="1">
      <instance name="default" enabled="true">
 <property_group name="datalinks" type="application">

   <!--
       customize datalink names for ixgbe0/1 to "corporate0"/"lab0", set
       lab0 mtu to 9000
   -->
   <property_group name="corporate0" type="datalink-phys">
            <propval name='devname' type='astring' value='ixgbe0'/>
   </property_group>
```

```
   <property_group name="lab0" type="datalink-phys">
            <propval name='devname' type='astring' value='ixgbe1'/>
            <propval name='mtu' type='count' value='9000'/>
   </property_group>

 </property_group>
      </instance>
    </service>
</service_bundle>
```

enable_sci.xml

The profile launches the SCI tool after first boot for an interactive configuration.

```
<service_bundle type="profile" name="enable_SCI_tool">
    <service name="milestone/config" version="1" type="service">
        <instance name="default" enabled="true">
            <property_group name="sysconfig" type="sysconfig">
                <propval name="configure" value="true" type="boolean"/>
                <propval name="interactive_config" value="true" type="boolean"/>
                <propval name="config_groups" value="system" type="astring"/>
            </property_group>
        </instance>
    </service>
</service_bundle>
```

ib_network.xml

The profile creates an InfiniBand network configuration.

```
<service_bundle type="profile" name="system configuration">
  <service name='network/datalink-management' type='service' version='0'>
    <instance name='default' enabled='true'>
      <property_group name='datalinks' type='application'>
        <property_group name='part1' type='datalink-part'>
          <propval name='linkover' type='astring' value='net1'/>
          <propval name='media' type='astring' value='Infiniband'/>
          <propval name='pkey' type='astring' value='0xFFFF'/>
        </property_group>
      </property_group>
    </instance>
  </service>
  <service name='network/ip-interface-management' type='service' version='0'>
      <instance name='default' enabled='true'>
        <property_group name='interfaces' type='application'>
```

```
            <property_group name='part1' type='interface-ip'>
              <property_group name='v4' type='address-static'>
                <propval name='ipv4-address' type='astring' value='x.x.x.x'/>
                <propval name='prefixlen' type='count' value='x'/>
                <propval name='up' type='astring' value='yes'/>
              </property_group>
              <propval name='address-family' type='astring' value='ipv4'/>
            </property_group>
          </property_group>
          <property_group name='static-routes' type='application'>
            <property_group name='route-1' type='static-route'>
              <propval name='destination' type='astring' value='default'/>
              <propval name='family' type='astring' value='inet'/>
              <propval name='gateway' type='astring' value='x.x.x.x'/>
              <propval name='isgateway' type='boolean' value='true'/>
            </property_group>
          </property_group>
        </instance>
    </service>
    <service name="network/dns/client" version="1">
        <property_group name="config">
            <property name="nameserver">
                <net_address_list>
                    <value_node value="x.x.x.x"/>
                </net_address_list>
            </property>
            <property name="search">
                <astring_list>
                    <value_node value="example.com"/>
                </astring_list>
            </property>
        </property_group>
        <instance name="default" enabled="true"/>
    </service>

    <service version="1" name="system/name-service/switch">
        <property_group name="config">
            <propval name="default" value="files"/>
            <propval name="host" value="files dns mdns"/>
        </property_group>
        <instance enabled="true" name="default"/>
    </service>

    <service version="1" name="system/name-service/cache">
        <instance enabled="true" name="default"/>
    </service>
</service_bundle>
```

ipmp_network.xml

The profile creates an IPMP configuration over 2 IP interfaces.

```
<service_bundle type="profile" name="system configuration">

    <service name="network/ip-interface-management" version="1" type="service">
      <instance name="default" enabled="true">
 <property_group name="interfaces" type="application">

   <!-- net1 interface configuration -->
          <property_group name="net1" type="interface-ip">
            <property name="address-family" type="astring">
              <astring_list>
  <value_node value="ipv4"/>
  <value_node value="ipv6"/>
              </astring_list>
            </property>
            <propval name="ipmp-interface" type="astring" value="ipmp1"/>
          </property_group>

   <!-- net2 standby interface configuration -->
   <property_group name="net2" type="interface-ip">
            <property name="address-family" type="astring">
              <astring_list>
  <value_node value="ipv4"/>
  <value_node value="ipv6"/>
              </astring_list>
            </property>
            <propval name="ipmp-interface" type="astring" value="ipmp1"/>
            <property_group name="ip" type="interface-protocol-ip">
              <propval name="standby" type="astring" value="on"/>
            </property_group>
   </property_group>

   <!-- IPMP interface configuration -->
   <property_group name="ipmp1" type="interface-ipmp">
            <property name="address-family" type="astring">
              <astring_list>
  <value_node value="ipv4"/>
  <value_node value="ipv6"/>
              </astring_list>
            </property>
            <property name="under-interfaces" type="astring">
              <astring_list>
  <value_node value="net1"/>
  <value_node value="net2"/>
```

```
            </astring_list>
          </property>

    <!-- IPv4 static address -->
          <property_group name="data1" type="address-static">
            <propval name="ipv4-address" type="astring" value="2.3.4.5"/>
      <propval name="prefixlen" type="count" value="24"/>
            <propval name="up" type="astring" value="yes"/>
          </property_group>
  </property_group>

 </property_group>
     </instance>
   </service>
</service_bundle>
```

static_network.xml

The profile configures the network with a static IPv4 address, a static route, and an `addrconf` address.

```
<service_bundle type="profile" name="system configuration">
  <service name="network/ip-interface-management" version="1" type="service">
    <instance name="default" enabled="true">
      <property_group name="interfaces" type="application">

 <!-- interface configuration -->
 <property_group name="{{AI_NETLINK_VANITY}}" type="interface-ip">
         <property name="address-family" type="astring">
           <astring_list>
      <value_node value="ipv4"/>
      <value_node value="ipv6"/>
           </astring_list>
         </property>

  <!-- IPv4 static address -->
  <property_group name="v4" type="address-static">
           <propval name="ipv4-address" type="astring" value="{{AI_IPV4}}"/>
           <propval name="prefixlen" type="count" value="{{AI_IPV4_PREFIXLEN}}"/>
           <propval name="up" type="astring" value="yes"/>
  </property_group>

  <!-- IPv6 addrconf address -->
         <property_group name="v6" type="address-addrconf">
           <propval name="interface-id" type="astring" value="::"/>
           <propval name="prefixlen" type="count" value="0"/>
```

```
            <propval name="stateful" type="astring" value="yes"/>
            <propval name="stateless" type="astring" value="yes"/>
          </property_group>

    <!-- default static route through interface -->
          <property_group name="route-1" type="static-route">
            <propval name="destination" type="astring" value="default"/>
            <propval name="family" type="astring" value="inet"/>
            <propval name="gateway" type="astring" value="{{AI_ROUTER}}"/>
          </property_group>
 </property_group>

      </property_group>
    </instance>
  </service>

    <service name="network/dns/client" version="1">
        <property_group name="config">
            <property name="nameserver">
                <net_address_list>
                    <value_node value="x.x.x.x"/>
                </net_address_list>
            </property>
            <property name="search">
                <astring_list>
                    <value_node value="example.com"/>
                </astring_list>
            </property>
        </property_group>
        <instance name="default" enabled="true"/>
    </service>

    <service version="1" name="system/name-service/switch">
        <property_group name="config">
            <propval name="default" value="files"/>
            <propval name="host" value="files dns mdns"/>
        </property_group>
        <instance enabled="true" name="default"/>
    </service>

    <service version="1" name="system/name-service/cache">
        <instance enabled="true" name="default"/>
    </service>
</service_bundle>
```

vnic_network.xml

The profile creates a virtual network configuration with a VNIC, link aggregation, and flow settings.

```
<service_bundle type="profile" name="system configuration">
    <service name="network/datalink-management" version="1" type="service">
      <instance name="default" enabled="true">
 <property_group name="datalinks" type="application">

   <!-- vnic configured over net0 -->
          <property_group name="vnic0" type="datalink-vnic">
            <propval name="linkover" type="astring" value="net0"/>
            <propval name="mac-address-type" type="astring" value="random"/>
            <propval name="media" type="astring" value="Ethernet"/>
            <propval name="vid" type="astring" value="0"/>
          </property_group>

   <!-- aggregation of net1 and net2 -->
          <property_group name="aggr1" type="datalink-aggr">
            <propval name="aggr-mode" type="astring" value="trunk"/>
            <propval name="force" type="boolean" value="false"/>
            <propval name="key" type="count" value="0"/>
            <propval name="lacp-mode" type="astring" value="off"/>
            <propval name="lacp-timer" type="astring" value="short"/>
            <propval name="media" type="astring" value="Ethernet"/>
            <propval name="num-ports" type="count" value="2"/>
            <propval name="policy" type="astring" value="L4"/>
            <property name="ports" type="astring">
              <astring_list>
  <value_node value="net1"/>
  <value_node value="net2"/>
              </astring_list>
            </property>
          </property_group>
 </property_group>

 <property_group name="flows" type="application">

   <!-- https flow over vnic0 for port 443 -->
          <property_group name="https1" type="flow">
            <propval name="linkover" type="astring" value="vnic0"/>
            <propval name="local-port" type="count" value="443"/>
            <propval name="max-bw" type="astring" value="500M"/>
            <propval name="transport" type="astring" value="tcp"/>
          </property_group>
```

```
</property_group>
     </instance>
   </service>

   <service name="network/ip-interface-management" version="1"
type="service">
     <instance name="default" enabled="true">
<property_group name="interfaces" type="application">

  <!-- configuration for vnic0 -->
  <property_group name="vnic0" type="interface-ip">
           <property name="address-family" type="astring">
             <astring_list>
 <value_node value="ipv4"/>
 <value_node value="ipv6"/>
             </astring_list>
           </property>

    <!-- IPv4 DHCP address -->
           <property_group name="v4" type="address-dhcp"/>

    <!-- IPv6 addrconf address -->
           <property_group name="v6" type="address-addrconf">
             <propval name="interface-id" type="astring" value="::"/>
             <propval name="prefixlen" type="count" value="0"/>
             <propval name="stateful" type="astring" value="yes"/>
             <propval name="stateless" type="astring" value="yes"/>
           </property_group>
  </property_group>

</property_group>
     </instance>
   </service>
</service_bundle>
```

dns.xml

The profile creates a network/dns/client configuration.

```
<service_bundle type="profile" name="dns configuration">

    <service name="network/dns/client" version="1">
        <property_group name="config">
            <property name="nameserver">
                <net_address_list>
                    <value_node value="x.x.x.x"/>
```

```
                    <value_node value="x.x.x.x"/>
                </net_address_list>
            </property>
            <property name="search">
                <astring_list>
                    <value_node value="example.com"/>
                </astring_list>
            </property>
        </property_group>
        <instance name="default" enabled="true"/>
    </service>

    <service version="1" name="system/name-service/switch">
        <instance enabled="true" name="default"/>
        <property_group name="config">
            <propval name="default" value="files"/>
            <propval name="host" value="files dns mdns"/>
        </property_group>
    </service>

    <service version="1" name="system/name-service/cache">
        <instance enabled="true" name="default"/>
    </service>
</service_bundle>
```

unconfig.xml

The activates an unconfigure scenario.

```
<service_bundle type="profile" name="unconfig_profile">
    <service name="milestone/unconfig" version="1" type="service">
        <property_group name="sysconfig" type="application">
            <propval name="shutdown" type="boolean" value="false"/>
            <propval name="destructive_unconfig" type="boolean" value="false"/>
            <propval name="unconfig_groups" type="astring" value="system "/>
            <propval name="unconfigure" type="boolean" value="true"/>
        </property_group>
    </service>
</service_bundle>
```

♦ ♦ ♦ APPENDIX B

Network Properties

This appendix provides an overview of the properties and property groups that can be used in a system configuration profile to define a network. The appendix covers the following topics:

- "Datalink Service Instance"
- "InfiniBand Host Channel Adapter Properties"
- "IP Interface Service Instance"

Note - For detailed information about the properties, refer to the `dladm`(8) and `ipadm`(8) man pages.

Datalink Service Instance

Use the `network/datalink-management:default` service instance to configure datalink networks, flows and WLANs. You can configure each network type, using the property groups and properties described in this section.

Datalink Properties

The following section covers all of the properties and property groups that are part of the `datalinks` property group in the `network/datalink-management:default` service. These values are defined within per-datalink property groups nested within the `datalinks` property group. The property group name is the name of the datalink, for example `net0`, and the property group type is the datalink class.

Properties for the following datalink classes are included:

- "Shared Datalink Properties" on page 100
- "Aggregation Datalink Properties" on page 101

Shared Datalink Properties

The following table provides a description of properties that are common to most datalink classes. These properties are defined in per-datalink property groups nested within the `datalinks` property group. The property group name is the name of the datalink, for example `net0`, and the property group type is the datalink class, for example `datalink-phys`.

TABLE 3 Properties Shared By Many Datalink Property Groups

Property	Type	Description
`autopush`	`astring`	STREAM modules to push on stream for link.
`allowed-dhcp-cids`	`astring`	Outbound packet allowed DHCP client ids for non-global zones.
`allowed-ips`	`astring`	Outbound packet allowed source IP addresses for non-global zones.
`bridge`	`astring`	Bridge that link is member of.
`bw-share`	`astring`	Minimum percentage share of bandwidth.
`cos`	`count`	802.1p priority associated with the link. Range is `0` to `7`.
`cpus`	`astring`	Bind packet processing to specific CPUs for link.
`forward`	`boolean`	Forwarding for a VLAN. Set the value to `true` to enable; `false` to disable.
`mtu`	`count`	Maximum client send data unit (SDU) supported by the device. Valid range is `68-65536`.
`mac-address`	`astring`	Primary MAC address for the datalink.
`max-bw`	`count`	Full duplex bandwidth. The bandwidth is specified as an integer with one of the scale suffixes (K, M, or G for Kbps, Mbps, and Gbps). The default is no bandwidth limit.
`pool`	`astring`	Bind packet processing to a specified processor pool.
`priority`	`astring`	Relative priority of the link. Set the value to `low`, `medium` or `high`.

Property	Type	Description
protection	astring	Link protection. The value can be dhcp-nospoof for DHCP client ID (DUID for DHCPv6) and hardware address anti-spoof, ip-nospoof for IP address anti-spoof; mac-nospoof for MAC address anti-spoof; and restricted to restrict outgoing packet types to just IPv4, IPv6, and ARP.
rx-fanout	astring	Number of receive-side fan out threads.
rx-rings	astring	Number of receive rings for a MAC client.
tx-rings	astring	Number of transmit rings for a MAC client.

Aggregation Datalink Properties

This datalink class is for Link Aggregation as either Datalink Multipathing (dlmp) or IEEE 802.3ad trunk. For more information see Chapter 2, "Configuring High Availability by Using Link Aggregations" in *Managing Network Datalinks in Oracle Solaris 11.4*. In addition to the class specific properties listed below, any system configuration profile including the datalink-aggr property group type may also include the shared datalink properties. See "Shared Datalink Properties" on page 100.

A sample system configuration profile using the datalink-aggr property group type can be seen in /usr/share/auto_install/sc_profiles/vnic_network.xml.

TABLE 4 Aggregation Datalink Properties

Property	Type	Description
aggr-mode	astring	IEEE 802.3ad complaint link aggregation mode.
default-tag	astring	Default VLAN ID. The range for the ID is from 0 to 4094.
force	boolean	Force creation even if underlying MAC does not support link update notification. Set the value to true to enable; false to disable.
gvrp-timeout	count	Wait between VID announcment broadcasts in milliseconds. Also see vlan-announce.
key	count	Aggregation key.
lacp-mode	astring	Link aggregation control protocol (LACP) activation mode.
lacp-timer	astring	LACP timer.
learn-decay	count	Decay rate for source changes limited by learn-limit.
learn-limit	count	Number of MAC sources to be learned over a bridge.
num-ports	count	Number of aggregate ports.
ofport	count	OpenFlow port assigned to the datalink.
openvswitch	astring	Switching on the physical datalink is managed by Open vSwitch (OVS). The value can be on or off.
policy	astring	Aggregation policies.

Property	Type	Description
poll	astring	Poll mode. The value can be auto to select the default poll setting; on to enable polling; or off to disable polling.
ports	astring	Port names.
probe-ip	astring	Source IP addresses for ICMP probing.
probe-fdt	astring	Fault detection time.
probe-vlan-id	count	VLAN ID for ICMP and transitive probing.
pvlan-tag-mode	astring	Determines how outbound packets are tagged. Allowed values are primary to use the primary VID or secondary to use the secondary VID.
stp	boolean	Spanning Tree Protocol (STP) on a bridge. Set the value to true to enable, or false to disable.
stp-cost	count	STP and RSTP cost.
stp-edge	count	Bridge edge port detection. Set the value to 1 to enable, or 0 to disable.
stp-mcheck	boolean	RSTP Force BPDU migration check. Set the value to true to enable, or false to disable.
stp-p2p	astring	Bridge point-to-point operation mode. If the value is true, port mode is forced to use point-to-point. If the value is false, port mode is forced to use normal multipoint mode. If the value is auto, point-to-point connections are automatically discovered.
stp-priority	count	STP and RSTP port priority value. The range can be between 0 and 255.
tag	astring	Tag associated with link.
tag-mode	astring	802.1Q VLAN tag control. If the value is normal, the service will add a VLAN tag if the outgoing packet belongs to a VLAN or if the user \selected priority tagging. If the value is vlanonly then the service will add a VLAN tag when the outgoing packet belongs to a VLAN.
tph	astring	Transaction processing hints (TPH) that allowed I/O devices to populate data. Set the value to on to turn TPH on, off to turn TPH off, or auto for the OS to decide whether to enable TPH on link.
virtual-switching	astring	Inter-VM communication policy. Set the value to auto to automatically configure inter-VM communication with LLDP, locale to select that inter-VM communication to be switched locally; and remote to configure switches as pass-through.
vlan-announce	astring	Automatic VLAN ID announcement control. The value is off by default. This is enabled by default. The value can be off, or gvrp to select announcements sent using GVRP protocol, as defined in 802.1D. Also see gvrp-timeout.

Bridge Datalink Properties

The bridge datalink class is for a bridge datalink instance. The datalink-bridge property group type only uses shared datalink properties. See “Shared Datalink Properties” on page 100.

Packet Capture Datalink Properties

The `cap` datalink class is for a packet capture datalink instance. The `datalink-cap` property group type only uses shared datalink properties. See "Shared Datalink Properties" on page 100.

Ethernet Over InfiniBand Datalink Properties

In addition to the class specific properties listed below, any system configuration profile including the `datalink-eoib` property group type may also include the shared datalink properties. See "Shared Datalink Properties" on page 100.

TABLE 5 Properties for EOIB

Property	Type	Description
`default-tag`	`astring`	Default VLAN ID. The range for the ID is from `0` to `4094`.
`gvrp-timeout`	`count`	Wait between VID announcement broadcasts in milliseconds. Also see `vlan-announce`.
`gwname`	`astring`	InfiniBand gateway name.
`gwport`	`astring`	InfiniBand gateway port.
`ibport`	`astring`	InfiniBand port.
`learn-decay`	`count`	Decay rate for source changes limited by `learn-limit`.
`learn-limit`	`count`	Number of MAC sources to be learned over a bridge.
`speed`	`count`	Link speed of physical Ethernet over InfiniBand datalink.
`stp`	`boolean`	Spanning Tree Protocol (STP) on a bridge. Set the value to `true` to enable, `false` to disable.
`stp-cost`	`count`	STP and RSTP cost.
`stp-edge`	`count`	Bridge edge port detection. Set the value to `1` to enable, `0` to disable.
`stp-mcheck`	`boolean`	RSTP force BPDU migration check. Set the value to `true` to enable, `false` to disable.
`stp-p2p`	`astring`	Bridge point-to-point operation mode. If the value is `true`, port mode is forced to use point-to-point. If the value is `false`, port mode is forced to use normal multipoint mode. If the value is `auto`, point-to-point connections are automatically discovered.
`stp-priority`	`count`	STP and RSTP port priority value. The range can be between `0` and `255`.
`tag-mode`	`astring`	802.1Q VLAN tag control. If the value is `normal`, the service will add a VLAN tag if the outgoing packet belongs to a VLAN or if the user selected priority tagging. If the value is `vlanonly` then the service will add a VLAN tag when the outgoing packet belongs to a VLAN.

Property	Type	Description
vlan-announce	astring	Automatic VLAN ID announcement control. The value is off by default. The value can be off, or gvrp to select announcements sent using GVRP protocol, as defined in 802.1D. Also see gvrp-timeout.

Ethernet Stub Datalink Properties

In addition to the class specific properties listed below, any system configuration profile including the datalink-etherstub property group type may also include the shared datalink properties. See "Shared Datalink Properties" on page 100.

TABLE 6 Ethernet Stub Datalink Properties

Property	Type	Description
default-tag	count	Default VLAN ID. The range for the ID is from 0 to 4094.
learn-decay	count	Decay rate for source changes limited by learn-limit. See also learn-limit.
learn-limit	count	Number of MAC sources to be learned over a bridge.
mac-address-len	count	MAC address length.
mac-address-prefix-len	count	MAC address prefix length.
mac-address-slot	count	Factory MAC address slot.
mac-address-type	astring	MAC address type. You can set the following values: factory, fixed, random, and unknown. Also, a value of auto selects to automatically obtain the MAC address; primary uses the address of the primary MAC client; and vrid uses the VRRP VID to calculate the MAC address. See the vrid property.
ofport	count	OpenFlow port assigned to the datalink.
openvswitch	astring	Switching on the physical datalink is managed by Open vSwitch (OVS). The value can be on or off.
stp	boolean	Spanning Tree Protocol (STP) on a bridge. Set the value to true to enable, false to disable.
stp-cost	count	STP and RSTP cost. Allowed range 0 to 65535.
stp-edge	count	Bridge edge port detection. Set the value to 1 to enable, 0 to disable.
stp-mcheck	boolean	RSTP force BPDU migration check. Set the value to true to enable, false to disable.
stp-p2p	astring	Bridge point-to-point operation mode. If the value is true, port mode is forced to use point-to-point. If the value is false, port mode is forced to use normal multipoint mode. If the value is auto, point-to-point connections are automatically discovered.
stp-priority	count	STP and RSTP port priority. Allowed range 0 to 255.
vraf	astring	VRRP address family. Values are inet for IPv4 VRRP address family and inet6 for IPv6 VRRP address family.

Property	Type	Description
vrid	count	VRRP VLAN identifier.

IP Tunnel Datalink Properties

In addition to the class specific properties listed below, any system configuration profile including the datalink-iptun property group type may also include the shared datalink properties. See “Shared Datalink Properties” on page 100.

TABLE 7 IP Tunnel Datalink Properties

Property	Type	Description
encap-limit	count	IPv6 encapsulation limit.
hop-limit	count	IPv4 TTL or IPv6 hop limit.
local	astring	IP tunnel local address.
remote	astring	IP tunnel remote address.
type	astring	IP tunnel type. The value can be set to ipv4, ipv6 or 6to4.

InfiniBand Datalink Properties

In addition to the class specific properties listed below, any system configuration profile including the datalink-part property group type may also include the shared datalink properties. See “Shared Datalink Properties” on page 100.

A sample system configuration profile using the datalink-part property group type can be seen in /usr/share/auto_install/sc_profiles/ib_network.xml.

TABLE 8 InfiniBand Datalink Properties

Property	Type	Description
broadcast-group	astring	Broadcast group state. The value can be set to absent, joined, unknown, or unsuccessful.
force	boolean	Force creation even if underlying MAC does not support link update notification. Set the value to true to enable, false to disable.
link-mode	astring	Link transport service type. Value can be set to cm for connected mode or ud for unreliable datagram mode.
linkover	astring	Parent link.
pkey	astring	InfiniBand partition key.

Physical Datalink Properties

In addition to the class specific properties listed below, any system configuration profile including the `datalink-phys` property group type may also include the shared datalink properties. See “Shared Datalink Properties” on page 100.

A sample system configuration profile using the `datalink-phys` property group type can be seen in `/usr/share/auto_install/sc_profiles/custom_network.xml`.

TABLE 9 Physical Datalink Properties

Property	Type	Description
`authentication`	`astring`	IEEE 802.1x complaint link authentication mode. Value is off by default.
`authentication-state`	`astring`	Datalink authentication state for IEEE 802.1X. Value can be `succeeded`, `failed`, `in-progress`, or `off`.
`auto-connect`	`astring`	Autoconnect to known WLANs.
`auto-negotiation`	`boolean`	Advertise autonegotiation capability. Set the value to `true` to enable, `false` to disable.
`default-tag`	`count`	Default VLAN ID. The range for the ID is from `0` to `4094`.
`devname`	`astring`	Name of the physical device.
`devname`	`astring`	Device name associated with link.
`ets-bw-local`	`count`	ETS bandwidth configured on transmit side for a link.
`ets-bw-remote-advice`	`count`	ETS bandwidth value to recommend to peer.
`extaddr`	`count`	Anet ID associated with kernel zone implicit link.
`flow-control`	`astring`	Advertised flow-control modes.
`gvrp-timeout`	`count`	Wait between VID announcement broadcasts in milliseconds. Also see `vlan-announce`.
`iov`	`astring`	Single root I/O virtualization (SR-IOV) mode. Allowed values are `on`, `off`, and `auto` to apply the default IOV setting.
`learn-decay`	`count`	Decay rate for source changes limited by learn-limit. See also `learn-limit`.
`learn-limit`	`count`	Number of MAC sources to be learned over a bridge.
`loc`	`astring`	Datalink location (for example, MB).
`lro`	`astring`	Large-receive offload disposition. Allowed values are `on`, `off`, or `auto` to apply the default LRO setting.
`mtu`	`count`	Maximum client send data unit (SDU) supported by the device. Valid range is `68` to `65536`.
`pfcmap`	`astring`	8-bit mask where each bit shows if priority-based flow control (PFC) is enabled for corresponding priority.
`power-mode`	`astring`	WiFi power management mode.
`radio`	`astring`	WiFi radio mode.

Property	Type	Description
ring-group	astring	Hardware ring group type. This read-only value can be either exclusive or shared.
speed	count	Link speed of physical Ethernet over InfiniBand datalink.
speed-duplex	astring	Speed/duplex values, for example, 1g-f is 1 GigaBit full-duplex.
stp	boolean	Spanning Tree Protocol (STP) on a bridge. Set the value to true to enable, false to disable.
stp-cost	count	STP and RSTP cost. Allowed range 0 to 65535.
stp-edge	count	Bridge edge port detection. Set the value to 1 to enable, 0 to disable.
stp-mcheck	boolean	RSTP force BPDU migration check. Set the value to true to enable, false to disable.
stp-p2p	astring	Point-to-point operation mode. If the value is true, port mode is forced to use point-to-point. If the value is false, port mode is forced to use normal multipoint mode. If the value is auto, point-to-point connections are automatically discovered.
stp-priority	count	STP and RSTP port priority value. Allowed range is 0 to 255.
tag	astring	Tag associated with link.
tag-mode	astring	802.1Q VLAN tag control. If the value is normal, the service will add a VLAN tag if the outgoing packet belongs to a VLAN or if the user selected priority tagging. If the value is vlanonly then the service will add a VLAN tag when the outgoing packet belongs to a VLAN.
tph	astring	Transaction processing hints (TPH) that allowed I/O devices to populate data. Allowed values are on, off, and auto for the OS to decide whether to enable TPH on the link.
vlan-announce	astring	Automatic VLAN ID announcement control. The value is off by default. The value can be off, or gvrp to select announcements sent using GVRP protocol, as defined in 802.1D. Also see gvrp-timeout.
vsi-manager-id	astring	IP address of VSI manager.
vsi-manager-id-encoding	astring	Encoding associated with the physical link vsi-manager-id. The value can be set to none if vsi-typeid and vsi-ver are not automatically generated over this link for VNICs that do not have their vsi-mgrid explicitly set. The oracle_v1 value selects Oracle VSI Manager.

Virtual Ethernet Point-to-point Datalinks Properties

In addition to the class specific properties listed below, any system configuration profile including the datalink-veth property group type may also include the shared datalink properties. See “Shared Datalink Properties” on page 100.

TABLE 10 Virtual Ethernet Point-to-point Datalinks Properties

Property	Type	Description
`default-tag`	`count`	Default VLAN ID. The value range for the ID is from `0` to `4094`.
`gvrp-timeout`	`count`	Wait between VID announcement broadcasts in milliseconds. Also see `vlan-announce`.
`learn-decay`	`count`	Decay rate for source changes limited by learn-limit. See also `learn-limit`.
`learn-limit`	`count`	Number of MAC sources to be learned over a bridge.
`ofport`	`count`	OpenFlow port assigned to the datalink.
`openvswitch`	`astring`	Switching managed by Open vSwitch (OVS). The value can be `on` or `off`
`stp`	`boolean`	Spanning Tree Protocol (STP). Set the value to `true` to enable, `false` to disable.
`stp-cost`	`count`	STP and RSTP cost. The value range is `0` to `65535`.
`stp-edge`	`count`	Bridge edge port detection. Set the value to `1` to enable, `0` to disable.
`stp-mcheck`	`boolean`	RSTP force BPDU migration check. Set the value to `true` to enable, `false` to disable.
`stp-p2p`	`astring`	Point-to-point operation mode. If the value is `true`, port mode is forced to use point-to-point. If the value is `false`, port mode is forced to use normal multipoint mode. If the value is `auto`, point-to-point connections are automatically discovered.
`stp-priority`	`count`	STP and RSTP port priority. The allowed value range is `0` to `255`.
`vethpeer`	`astring`	Veth peer.
`vlan-announce`	`astring`	Automatic VLAN ID announcement control. The value is `off` by default. The value can be `off`, or `gvrp` to select announcements sent using GVRP protocol, as defined in 802.1D. Also see `gvrp-timeout`.

Virtual LAN Link Properties

In addition to the class specific properties listed below, any system configuration profile including the `datalink-vlan` property group type may also include the shared datalink properties. See “Shared Datalink Properties” on page 100.

TABLE 11 Virtual LAN Link Properties

Property	Type	Description
`iov`	`astring`	Single root I/O virtualization (SR-IOV) mode. Allowed values are `on`, `off`, and `auto` to apply the default IOV setting.
`linkover`	`astring`	Parent link.
`mac-address-len`	`count`	MAC address length.
`mac-address-prefix-len`	`count`	MAC address prefix length.
`mac-address-slot`	`count`	Factory MAC address slot.

Property	Type	Description
mac-address-type	astring	MAC address type. You can set the following values: factory, fixed, random, and unknown. Also, a value of auto selects to automatically obtain the MAC address; primary uses the address of the primary MAC client; and vrid uses the VRRP VID to calculate the MAC address. See the vrid property.
ofport	count	OpenFlow port assigned to the datalink.
pkey	astring	InfiniBand partition key.
poll	astring	Poll mode. The value can be auto to select the default poll setting; on to enable polling; or off to disable polling.
ring-group	astring	Hardware ring group type. The value can be set at VNIC creation time. The value can be auto which specifies that the OS should decide whether exclusive or shared is used on a particular physical link; exclusive which specifies that VNIC creation should fail if an exclusive ring-group is not available; or shared which specifies that dedicated resources are not allocated. The default value is shared.
vid	astring	VLAN identifier.
vraf	astring	VRRP address family. Values are inet for IPv4 VRRP address family and inet6 for IPv6 VRRP address family.
vrid	count	VRRP VLAN identifier.

Virtual NIC Datalink Properties

In addition to the class specific properties listed below, any system configuration profile including the datalink-vnic property group type may also include the shared datalink properties. See “Shared Datalink Properties” on page 100.

A sample system configuration profile using the datalink-vnic property group type can be seen in /usr/share/auto_install/sc_profiles/vnic_network.xml.

TABLE 12 Virtual NIC Datalink Properties

Property	Type	Description
broadcast-group	astring	Broadcast group state. The value can be set to absent, joined, unknown, or unsuccessful.
ets-bw-local	count	ETS bandwidth configured on transmit side for a link.
ets-bw-remote-advice	count	ETS bandwidth value to recommend to peer.
iov	astring	Single root I/O virtualization (SR-IOV) mode. Allowed values are on, off, and auto to apply the default IOV setting.
lro	astring	Large-receive offload disposition. Allowed values are on, off, or auto to apply the default LRO setting.
ring-group	astring	Hardware ring group type. The value change be set toexclusive, shared or auto for the OS to decide on the ring-group type. .
vsi-manager-id	astring	IP address of VSI manager.

Virtual eXtensible LAN Datalink Properties

In addition to the class specific properties listed below, any system configuration profile including the `datalink-vxlan` property group type may also include the shared datalink properties. See "Shared Datalink Properties" on page 100.

TABLE 13 Virtual eXtensible LAN Datalink Properties

Property	Type	Description
`default-tag`	`count`	Default VLAN ID. The range for the ID is from `0` to `4094`.
`ip-interface`	`astring`	IP Interface for the VXLAN link.
`ip-version`	`astring`	IP version of the address to be automatically selected if the interface property is set. Set the value to `v4` or `v6`.
`local-ip`	`astring`	IPv4 address, IPv6 address or a hostname of the VXLAN link.
`multicast-group`	`astring`	Multicast group associated with the VXLAN link.
`ofport`	`count`	OpenFlow port assigned to the datalink.
`openvswitch`	`astring`	Switching managed by Open vSwitch (OVS). The value can be `on` or `off`.
`vni`	`count`	The VXLAN segment number that to which the VXLAN link belongs.

Flow Datalink Properties

The following tables lists all of the properties that are part of the `flows` property group in the `network/datalink-management:default` service.

TABLE 14 Flows Datalink Properties

Property	Type	Description
`arp-op`	`astring`	ARP operation. The value can be set to `request` or `response`.
`arp-sender`	`astring`	Hardware address of the sender.
`arp-sip`	`astring`	IP address of the ARP sender.
`arp-target`	`astring`	Hardware target address.
`arp-tip`	`astring`	IP address of the ARP target.
`destination-mac-address`	`astring`	Destination MAC address.
`direction`	`astring`	Flow direction. Set the value to `in` for inbound, `out` for outbound, or `bi` for bidirectional.
`dscp`	`count`	Differentiated service code point.
`dsfield`	`count`	Differentiated services value for flow.
`dsfield-mask`	`count`	Differentiated services mask value for flow.

Property	Type	Description
hw-flow	astring	Hardware flow. Allowed values are on to enable flow offload, off to disable flow offload and auto to apply the default flow offload setting.
icmp-code	count	ICMP packet code.
icmp-type	count	ICMP packet type.
linkover	astring	Parent link.
local-ip	astring	Local flow IP address.
local-port	astring	Flow local port.
max-bw	count	Maximum flow bandwidth. The bandwidth is specified as an integer with one of the scale suffixes (K, M, or G for Kbps, Mbps, and Gbps). The default is no bandwidth limit.
nd-sll	astring	Hardware address of source in IPv6 neighbor discovery (ND).
nd-tll	astring	Hardware address of target in IPv6 neighbor discovery (ND).
nd-target	astring	IP address of target in IPv6 neighbor discovery (ND).
ofaction	astring	OpenFlow action list.
priority	astring	Relative priority of the link. Set the value to low, medium or high.
rank	count	Flow rank. The allowed range is from 1 to 65535.
remote-ip	astring	Remote flow IP address
remote-port	astring	Flow remote port.
sap	astring	Ethertype.
source-mac-address	astring	Source MAC address.
source-port	count	OpenFlow source port.
tcp-flags	astring	TCP flags.
transport	astring	Flow transport. The value can be set to tcp, udp, sctp, icmp, or icmpv6.
ttl	astring	Time to live from the IP header.
tun-dsfield	count	Tunnel differentiated services value for flow.
tun-flags	astring	Tunnel flags.
tun-id	astring	Tunnel ID.
tun-local-ip	astring	IP address for tunnel local flow.
tun-remote-ip	astring	IP address for tunnel remote flow.
tun-ttl	astring	Tunnel time to live from the IP header.
vlan-tci	astring	VLAN header.

Known WLAN Properties

Information about visited WLANs is automatically stored in property groups in the known-wlans parent property group, but this information can also be specified at install time in a

system configuration profile. The following tables lists all of the properties that can be included in the `network/datalink-management:default` service.

TABLE 15 Known WLAN Properties

Property	Type	Description
`bssids`	`astring`	Basic server set ID (BSSID) for the WiFi network.
`key`	`astring`	Secure object name to be associated with this known WLAN.
`priority`	`count`	Relative priority of the known WLAN. Set the value to `0` for the highest priority, and `1` for a lower priority, and so forth.
`security-mode`	`astring`	Encryption mode for the Wifi network for the known WLAN. Value can be set to `none`, `wep`, or `wpa`.

InfiniBand Host Channel Adapter Properties

The following section covers all of the properties and property groups that are part of the `network/ib/ib-management` service. An example of a system configuration file with these properties is given in the `ibmgmtd(8)` man page. These properties are all set at the instance level. You can use these properties to configure HCAs and VHCAs. Properties for the following classes are included:

- "InfiniBand HCA and VHCA Properties" on page 112
- "InfiniBand HCA and VHCA Partition Key Property" on page 113

Note - Unlike Ethernet interfaces that are managed at the port level, InfiniBand interfaces are managed at the card level.

InfiniBand HCA and VHCA Properties

Any system configuration profile including the `ibadm-type-hca` property group type may include the following properties.

TABLE 16 InfiniBand HCA Properties

Property	Type	Description
`iov`	`astring`	Single root I/O virtualization (SR-IOV) mode. Allowed values are `on` and `off`.

Property	Type	Description
max-vhcas	count	Maximum number of VHCAs.
zone	astring	Zone the HCA or VHCA is assigned to.

InfiniBand HCA and VHCA Partition Key Property

Any system configuration profile including the `ibadm-type-port` property group type should include the following `pkeys` property. The data type of this property is `astring`. This property should be used to define the InfiniBand partition keys.

IP Interface Service Instance

Use the `network/ip-interface-management:default` service instance to configure IP, loopback and IPMP interfaces. You can configure properties at the global level or for each individual interface. The following section describes all of the property groups and properties that you can use.

Global IP Interface Properties

The following section covers all of the properties that are part of the `interface-ip` property group in the `network/ip-interface-management:default` service. These are global properties set for all interfaces on the system.

TABLE 17 Global IP Interface Properties

Property	Type	Description
arp	astring	Use ARP for all interfaces. Set the value to `on` to enable, `off` to disable. Allowed with the IPv4 protocol.
exchange-routes	astring	Exchange routing information for all interfaces. Set the value to `on` to enable, `off` to disable. Allowed with the IPv4 and IPv6 protocols.
forwarding	astring	IP forwarding for all interfaces. Set the value to `on` to enable, `off` to disable. Allowed with the IPv4 and IPv6 protocols.
fwifgroup	astring	Firewall policy group for all interfaces. Allowed with IP protocols.
group	astring	IPMP interface group name. Allowed with IP protocols.
metric	count	Routing metric for all interfaces. Allowed with the IPv4 and IPv6 protocols.

Property	Type	Description
`mtu`	`count`	Maximum transmission unit in bytes for all interfaces. The value range is `68` to `65536`. Allowed with the IPv4 and IPv6 protocols.
`standby`	`astring`	Standby interface for an IPMP group. This is not applicable to non-IPMP interfaces. Set the value to `on` to enable, `off` to disable.
`usesrc`	`astring`	Interface for source address selection. Allowed with the IPv4 and IPv6 protocols.

Interface Properties for a Specific IP Interface

The following section covers all of the properties and property groups that are part of the `interface-ip` property group within the `interfaces` in the `network/ip-interface-management:default` service. These properties define an individual interface rather than global settings for all interfaces on the system. For global properties see "Global IP Interface Properties" on page 113 and "Global IP Protocol Properties" on page 121.

Properties for the following datalink classes are included:

- "Per Interface IP Protocol Properties" on page 114
- "Per Interface IPv4 Protocol Properties" on page 115
- "Per Interface IPv6 Protocol Properties" on page 115
- "Per Interface Static Address Properties" on page 116
- "Per Interface IPv6 Interface Properties" on page 116
- "Per Interface DHCP Interface Properties" on page 117
- "Per Interface VRRP Interface Properties" on page 118
- "Per Interface Static Route Properties" on page 118

Per Interface IP Protocol Properties

Any system configuration profile including the `interface-protocol-ip` property group type may include the properties listed below.

TABLE 18 Per Interface IP Protocol Properties

Property	Type	Description
`allow-xprobe`	`astring`	Transitive probe based failure detection per the IPMP group interface. This is not applicable to non-IPMP interfaces. Set the value to `true` to enable, `false` to disable, or `inherit` to use the `svc:/network/ipmp/config/transitive-probing` value. *suggest changing to on and off*

Property	Type	Description
fwifgroup	astring	Firewall policy group.
group	astring	IPMP interface group name. It is read-only for other interface classes.
standby	astring	Standby interface for an IPMP group. This is not applicable to non-IPMP interfaces. Set the value to on to enable, off to disable.

Per Interface IPv4 Protocol Properties

Any system configuration profile including the interface-protocol-ipv4 property group type may include the properties listed below.

TABLE 19 Per Interface IPv4 Protocol Properties

Property	Type	Description
arp	astring	Use ARP. Set the value to on to enable, off to disable.
exchange-routes	astring	Exchange routing information. Set the value to on to enable, off to disable.
forwarding	astring	IP forwarding. Set the value to on to enable, off to disable.
metric	count	Routing metric.
mtu	count	Maximum transmission unit in bytes. The value range is 68 to 65536.
usesrc	astring	Interface for source address selection.

Per Interface IPv6 Protocol Properties

Any system configuration profile including the interface-protocol-ipv6 property group type may include the properties listed below.

TABLE 20 Per Interface IPv6 Protocol Properties

Property	Type	Description
exchange-routes	astring	Exchange routing information. Set the value to on to enable, off to disable.
forwarding	astring	IP forwarding. Set the value to on to enable, off to disable.
metric	count	Routing metric.
mtu	count	Maximum transmission unit in bytes. The value range is 68 to 65536.
nud	astring	Neighbor unreachability detection. Set the value to on to enable, off to disable.
usesrc	astring	Interface for source address selection.

Per Interface Static Address Properties

Any system configuration profile including the `address-static` property group type may include the properties listed below.

TABLE 21 Per Interface Static Address Properties

Property	Type	Description
`deprecated`	`astring`	No non-deprecated addresses. Set the value to `on` to enable, `off` to disable.
`ipv4-address`	`astring`	IPv4 local or source address.
`ipv4-remote-address`	`astring`	IPv4 remote or destination address.
`ipv6-address`	`astring`	IPv6 local or source address.
`ipv6-remote-address`	`astring`	IPv6 remote or destination address.
`prefixlen`	`count`	Prefix length. The value range is `0` to `128`.
`private`	`count`	No address advertising by routing daemons. Set the value to `on` to enable, `off` to disable.
`transmit`	`astring`	Transmit packets with this address. Set the value to `on` to enable, `off` to disable.
`up`	`count`	Address is up. Set the value to `yes` to enable, `no` to disable.
`zone`	`count`	Zone address.

Per Interface IPv6 Interface Properties

Any system configuration profile including the `address-addrconf` property group type may include the properties listed below.

TABLE 22 IPv6 Interface Properties

Property	Type	Description
`client-id`	`astring`	DHCP client identifier. The ID should be specified as 01<*hex-string*> where the hex string usually corresponds to the MAC address.
`deprecated`	`astring`	No non-deprecated addresses. Set the value to `on` to enable, `off` to disable.
`interface-id`	`astring`	IPv6 autoconfiguration interface ID.
`offer-wait`	`count`	Wait, in seconds, between checking for valid OFFER after sending a DISCOVER. The value range is `1` to `20`.
`param-ignore-list`	`astring`	Specifies a list of comma-separated integer values of DHCP option numbers from `/etc/dhcp/inittab` that DHCP client will ignore. The value range is `1` to `64`. The default is that no DHCP options are ignored.

Property	Type	Description
param-request-list	astring	Specifies a list of comma-separated integer values of DHCP option numbers for which the DHCP client would like values, or symbolic Site or Option option names. The default value is 7,12,23,24,27,29. The range is the set of possible DHCP option numbers from /etc/dhcp/inittab. Symbolic option names are resolved through /etc/dhcp/inittab. The value range is 1 to 64.
remote-interface-id	astring	Destination IPv6 autoconfiguration interface ID.
stateful	astring	Stateful IPv6 autoconfiguration. Set the value to on to enable, off to disable.
stateless	astring	Stateless IPv6 autoconfiguration. Set the value to on to enable, off to disable.

Per Interface DHCP Interface Properties

Any system configuration profile including the address-dhcp property group type may include the properties listed below.

TABLE 23 Per Interface DHCP Interface Properties

Property	Type	Description
client-id	astring	DHCP client identifier. The ID should be specified as 01<*hex-string*> where the hex string usually corresponds to the MAC address.
deprecated	astring	No non-deprecated addresses. Set the value to on to enable, off to disable.
dhcp-wait	count	DHCP timeout value
offer-wait	count	Wait, in seconds, between checking for valid OFFER after sending a DISCOVER. The value range is 1 to 20.
param-ignore-list	astring	Specifies a list of comma-separated integer values of DHCP option numbers from /etc/dhcp/inittab that DHCP client will ignore. The value range is 1 to 64. The default is that no DHCP options are ignored.
param-request-list	astring	Specifies a list of comma-separated integer values of DHCP option numbers for which the DHCP client would like values, or symbolic Site or Option option names. The default value is 7,12,23,24,27,29. The range is the set of possible DHCP option numbers from /etc/dhcp/inittab. Symbolic option names are resolved through /etc/dhcp/inittab. The value range is 1 to 64.
prefixlen	count	Prefix length. The value range is 0 to 128.
primary-interface	boolean	DHCP primary interface. Set the value to true to enable, false to disable.
private	count	No address advertising by routing daemons. Set the value to on to enable, off to disable.
transmit	astring	Transmit packets with this address. Set the value to on to enable, off to disable.
up	count	Address is up. Set the value to yes to enable, no to disable.

Property	Type	Description
verified-lease-only	astring	Verify the DHCP client's lease with the DHCP server. Allowed values are yes to verify the lease and no for no verification.
zone	count	Zone address.

Per Interface VRRP Interface Properties

Any system configuration profile including the address-vrrp property group type may include the properties listed below.

TABLE 24 Per Interface VRRP Interface Properties

Property	Type	Description
deprecated	astring	No non-deprecated addresses. Set the value to on to enable, off to disable.
ipv4-address	astring	IPv4 local or source address.
ipv6-address	astring	IPv6 local or source address.
prefixlen	count	Prefix length. The value range is 0 to 128.
private	count	No address advertising by routing daemons. Set the value to on to enable, off to disable.
transmit	astring	Transmit packets with this address. Set the value to on to enable, off to disable.
up	count	Address is up. Set the value to yes to enable, no to disable.
vrrp-router-name	astring	VRRP router name.
zone	count	Zone address.

Per Interface Static Route Properties

Any system configuration profile including the static-route property group type may include the properties listed below.

TABLE 25 Per Interface Static Route Properties

Property	Type	Description
blackhole	boolean	Silently discard packages. Set the value to true to enable, false to disable.
destination	astring	Destination address.
destination-type	astring	Destination address type. The value can be set as host, net or any.

Property	Type	Description
export	count	Lifetime for an entry.
family	astring	Address family. The value can be set as link, inet or inet6.
gateway	astring	Gateway address.
hopcount	count	Maximum hop count. The range can be between 1 and 255.
indirect	boolean	Add routes when gateway not on-link. Set the value to true to enable, false to disable.
isgateway	astring	Route is not directly reachable.
locks	astring	Lock specific metric values. The value can be expire, hopcount, mtu, recvpipe, sendpipe, ssthresh, rtt and rttvar.
mtu	count	Maximum transmission unit in bytes. The value range is 68 to 65536.
multirt	boolean	Create redundant route. Set the value to true to enable, false to disable.
netmask	astring	Netmask.
private	astring	No address advertising for the interface. Set the value to on to prevent advertising, off to allow advertising.
proto1	boolean	Protocol specific routing flag 1. Set the value to true to enable, false to disable.
proto2	boolean	Protocol specific routing flag 2. Set the value to true to enable, false to disable.
recvpipe	count	Receive pipe size in bytes.
reject	astring	ICMP unreachable when matched.
rtt	count	Round-trip time in microseconds.
rttvar	count	Round-trip time variance in microseconds.
secattr	astring	Security attributes of a route.
sendpipe	count	Send pipe size in bytes.
setsrc	astring	Assign default source address
ssthresh	count	Slow start threshold.

Interface Properties for a Loopback Interface

The following section covers all of the properties and property groups that are part of the interface-loopback property group in the network/ip-interface-management:default service.

TABLE 26 Loopback Interface Properties

Property	Type	Description
fwifgroup	astring	Firewall policy group for the loopback interface. Allowed with IP protocols.

Property	Type	Description
mtu	count	Maximum transmission unit in bytes for the loopback interface. The value range is 68 to 65536. Allowed with the IPv4 and IPv6 protocols.

Interface Properties for IPMP Interfaces

The following section covers all of the properties and property groups that are part of the interface-ipmp property group in the network/ip-interface-management:default service.

In addition, any IP interface that is part of the IPMP group must include a line in the interface-ip property group description that associates the interface with the named IPMP group. For instance, if the IPMP property group is called ipmp1, then the following line must be included in each interface-ip property group that is part of that IPMP group:

```
<propval name="ipmp-interface" type="astring" value="ipmp1"/>
```

You can see an example of a system configuration profile using the interface-ipmp property group at /usr/share/auto_install/sc_profiles/ipmp_network.xml.

TABLE 27 IPMP Interface Properties

Property	Type	Description
allow-xprobe	astring	Transitive probe based failure detection per the IPMP group interface. This is not applicable to non-IPMP interfaces. Set the value to true to enable, false to disable, or inherit to use the svc:/network/ipmp/config/transitive-probing value.
arp	astring	Use ARP for all IPMP interfaces. Set the value to on to enable, off to disable. Allowed with the IPv4 protocol.
exchange-routes	astring	Exchange routing information for all IPMP interfaces. Set the value to on to enable, off to disable. Allowed with the IPv4 and IPv6 protocols.
forwarding	astring	IP forwarding for all IPMP interfaces. Set the value to on to enable, off to disable. Allowed with the IPv4 and IPv6 protocols.
fwifgroup	astring	Firewall policy group for all IPMP interfaces. Allowed with IP protocols.
group	astring	IPMP interface group name. Allowed with IP protocols.
metric	count	Routing metric for all IPMP interfaces. Allowed with the IPv4 and IPv6 protocols.
mtu	count	Maximum transmission unit in bytes for all IPMP interfaces. The value range is 68 to 65536. Allowed with the IPv4 and IPv6 protocols.
nud	astring	Neighbor unreachability detection for all IPMP interfaces. Set the value to on to enable, off to disable.
standby	astring	Standby interface for an IPMP group. This is not applicable to non-IPMP interfaces. Set the value to on to enable, off to disable.

Property	Type	Description
usesrc	astring	Interface for source address selection. Allowed with the IPv4 and IPv6 protocols.

Global IP Protocol Properties

The following section covers all of the properties and property groups that are part of the protocols property group in the network/ip-interface-management:default service. These properties define global settings for all interfaces on the system rather than for an individual interface. For properties for individual interfaces see "Interface Properties for a Specific IP Interface" on page 114.

Properties for the following IP classes are included:

DHCPv4 Protocol Properties

Any system configuration profile including the protocol-dhcpv4 property group type may include the properties listed below. These properties are defined in property groups within the protocols property group.

TABLE 28 DHCPv4 Protocol Properties

Property	Type	Description
client_id	astring	Value that should be used to uniquely identify the DHCP client to the DHCP server.
offer-wait	count	Wait, in seconds, between checking for valid OFFER after sending a DISCOVER. The value range is 1 to 20.

Property	Type	Description
param-ignore-list	astring	Specifies a list of comma-separated integer values of options that DHCP client will ignore.
param-request-list	astring	Specifies a list of comma-separated integer values of options for which the DHCP client would like values, or symbolic Site or Option option names. Symbolic option names are resolved through /etc/dhcp/inittab.
verified-lease-only	astring	Verify the DHCP client's lease with the DHCP server. Allowed values are yes to verify the lease and no for no verification.

DHCPv6 Protocol Properties

Any system configuration profile including the protocol-dhcpv6 property group type may include the properties listed below.

TABLE 29 DHCPv6 Protocol Properties

Property	Type	Description
client_id	astring	Value that should be used to uniquely identify the DHCP client to the DHCP server.
offer-wait	count	Wait, in seconds, between checking for valid OFFER after sending a DISCOVER. The value range is 1 to 20.
param-ignore-list	astring	Specifies a list of comma-separated integer values of options that DHCP client will ignore.
param-request-list	astring	Specifies a list of comma-separated integer values of options for which the DHCP client would like values, or symbolic Site or Option option names. Symbolic option names are resolved through /etc/dhcp/inittab.
verified-lease-only	astring	Verify the DHCP client's lease with the DHCP server. Allowed values are yes to verify the lease and no for no verification.

ICMP Protocol Properties

Any system configuration profile including the protocol-icmp property group type may include the properties listed below.

TABLE 30 Properties in the protocol-icmp Property Group

Property	Type	Description
max-buf	count	Maximum size of the send or receive socket buffer for the ICMP protocol. This value limits the maximum value of recv-buf and send-buf. The value can be set to 2048 to 1,073,741,824.

Property	Type	Description
recv-buf	count	Maximum size of the receive socket buffer for the ICMP protocol. This value must be less than max-buf. The value can be set to 2048 to max-buf.
send-buf	count	Maximum size of the send socket buffer for the ICMP protocol. This value must be less than max-buf. The value can be set to 2048 to max-buf.

IP Protocol Properties

Any system configuration profile including the protocol-ip property group type may include the properties listed below.

TABLE 31 IP Protocol Properties

Property	Type	Description
arp-publish-count	count	Number of gratuitous ARP messages to send. Gratuitous ARP messages announce the physical addresses of the configured IP interfaces during the boot process. The value range is from 1 to 20. The default value is 5.
arp-publish-interval	count	Time interval between gratuitous ARP messages in milliseconds. The value range is 1000 to 20,000. The default value is 2000.
icmp-accept-clear	astring	IPsec policy bypass for inbound cleartext ICMP messages. If the value is on (the default), allow certain cleartext ICMP messages to bypass IPsec policy. For ICMP echo requests (ping messages), protect the response like the request. If the value is off, treat ICMP messages like other IP traffic. The default value is on.
igmp-accept-clear	astring	IPsec policy bypass for inbound cleartext IGMP messages. If the value is on, allow inbound cleartext IGMP messages to bypass IPsec policy. If the value is off, treat IGMP messages like other IP traffic. The default value is on.
ndp-unsolicit-count	count	Number of NDP advertisement messages to send to the announce local IPv6 addresses. The value range is 1 to 20. The default value is 3.
ndp-unsolicit-interval	count	Time interval between NDP advertisement messages sent to the announce local IPv6 addresses in milliseconds. The value range is 1000 to 20,000.
persock-require-priv	astring	Privileges for socket. If the value is equal to on (the default), require the PRIV_SYS_IP_CONFIG privilege in order to set the algorithms in the per-socket policy different from the system-wide policy. If the value is equal to off, allow an unprivileged user to change the algorithms, but not to bypass policy altogether.
pim-accept-clear	required	IPsec policy bypass for inbound cleartext PIM messages. If the value is on, allow inbound cleartext PIM messages to bypass IPsec policy. If the value is off, treat IGMP messages like other IP traffic. The default value is on..
verify-bind	astring	Verify bind succeeds. Set the value to on to enable, off to disable. If the value is off, set up binding and listening on a socket before the IP interface is up. The default value is on.

IPv4 Protocol Properties

Any system configuration profile including the `protocol-ipv4` property group type may include the properties listed below.

TABLE 32 IPv4 Protocol Properties

Property	Type	Description
`forwarding`	`astring`	IP forwarding. Set the value to `on` to enable, `off` to disable.
`hostmodel`	`astring`	Multihomed send and receive behaviour as in RFC1122. The value can be set to `weak` or `strong`. Setting the value to `src-priority` selects a weak hostmodel with IP source address configured on outgoing interface preferred. Setting the value to `custom` selects custom send/receive settings defined in kernel.
`send-redirects`	`astring`	Send ICMPv4 redirect messages. Set the value to `on` to enable, `off` to disable.
`ttl`	`count`	IPv4 TTL (time-to-live) value. Used to prevent a system from reaching other systems more than *N* hops away. The value range is from `1` to `255`.

IPv6 Protocol Properties

Any system configuration profile including the `protocol-ipv6` property group type may include the properties listed below.

TABLE 33 IPv6 Protocol Properties

Property	Type	Description
`forwarding`	`astring`	Enable IP forwarding. Set the value to `on` to enable, `off` to disable.
`hoplimit`	`count`	IPv6 hoplimit value. Used to prevent a system from reaching other systems more than *N* hops away. The value range is from `1` to `255`.
`hostmodel`	`astring`	Multihomed send and receive behaviour as in RFC1122. The value can be set to `weak` or `strong`. Setting the value to `src-priority` selects a weak hostmodel with IP source address configured on outgoing interface preferred. Setting the value to `custom` selects custom send/receive settings defined in kernel.
`send-redirects`	`astring`	Send ICMPv6 redirect messages. Set the value to `on` to enable, `off` to disable.

SCTP Protocol Properties

Any system configuration profile including the `protocol-sctp` property group type may include the properties listed below.

TABLE 34 Properties in the `protocol-sctp` Property Group

Property	Type	Description
`extra-priv-ports`	`count`	Defines additional privileged ports outside of the 1-1023 range using the SCTP protocol. The value can be set to `1024` to `65535`.
`largest-anon-port`	`count`	Upper bound on ephemeral or short-lived ports created when establishing out-bound network connects. The value can be set to `32,768` to `65,535`.
`max-buf`	`count`	Maximum size of the send or receive socket buffer for the SCTP protocol. This value limits the maximum value of `recv-buf` and `send-buf`.The value can be set to `2048` to `1,073,741,824`.
`recv-buf`	`count`	Maximum size of the receive socket buffer for the SCTP protocol. This value must be less than `max-buf`. The value can be set to `2048` to `max-buf`.
`send-buf`	`count`	Maximum size of the send socket buffer for the SCTP protocol. This value must be less than `max-buf`. The value can be set to `2048` to `max-buf`.
`smallest-anon-port`	`count`	Lower bound on ephemeral or short-lived ports created when establishing out-bound network connects. Should be less than or equal to `largest-anon-port`. The value can be set to `2048` to `largest-anon-port`.
`smallest-nonpriv-port`	`count`	Smallest port number for a non-privileged port using the SCTP protocol. The value range is `1024` to `32768`.
`reuseport-lbalg`	`astring`	Reuseport load balancing. The value can be set to `rr` for round robin, `src-dest-ip` for source and destination IP address hashing, `src-dest-ip-ports` for source and destination IP address and port hashing, `src-ip` for source IP address hashing, or `src-ip-port` for source IP address and port hashing.

TCP Protocol Properties

Any system configuration profile including the `protocol-tcp` property group type may include the properties listed below.

TABLE 35 Properties in the `protocol-tcp` Property Group

Property	Type	Description
`cwnd-max`	`count`	Maximum congestion window. The value can be from `128` to `1,073,741,824`.
`ecn`	`astring`	Explicit Congestion Notification as in RFC 3168. The value can be `never`, `passive` or `active`. The default value is `active`. For more information,

Property	Type	Description
		see "_deferred_ack_interval Parameter" in *Oracle Solaris 11.4 Tunable Parameters Reference Manual*
extra-priv-ports	count	Additional privileged ports outside of the 1-1023 range using the TCP protocol. The value can be set to 1024 to 65535.
largest-anon-port	count	Upper bound on ephemeral or short-lived ports created when establishing out-bound network connects. The value can be set to 32,768 to 65,535.
max-buf	count	Maximum size of the send or receive socket buffer for the TCP protocol. This value limits the maximum value of recv-buf and send-buf. The value can be set to 2048 to 1,073,741,824.
recv-buf	count	Maximum size of the receive socket buffer for the TCP protocol. The value can be set to 2048 to max-buf.
sack	astring	Selective acknowledgement (SACK) as defined by RFC 2018. Values can be never, passive or active. The default value is active.
send-buf	count	Maximum size of the send socket buffer for the TCP protocol. The value can be set to 2048 to max-buf.
smallest-anon-port	count	Lower bound on ephemeral or short-lived ports created when establishing out-bound network connects. Should be less than or equal to largest-anon-port. The value can be set to 1024 to largest-anon-port.
smallest-nonpriv-port	count	Smallest port number for a non-privileged port using the TCP protocol. The value can be set to 1024 to 32768.
reuseport-lbalg	astring	Reuseport load balancing. The value can be set to rr for round robin, src-dest-ip for source and destination IP address hashing, src-dest-ip-ports for source and destination IP address and port hashing, src-ip for source IP address hashing, or src-ip-port for source IP address and port hashing.

UDP Protocol Properties

Any system configuration profile including the protocol-udp property group type may include the properties listed below.

TABLE 36 UDP Protocol Properties

Property	Type	Description
extra-priv-ports	count	Additional privileged ports. The value range is 1024 to 65535.
largest-anon-port	count	Upper bound on ephemeral or short-lived ports created when establishing out-bound network connects. The value can be set to 32,768 to 65,535.
max-buf	count	Maximum size of the send or receive socket buffer for the UDP protocol. This value limits the maximum value of recv-buf and send-buf. The value can be set to 2048 to 1,073,741,824.
recv-buf	count	Maximum size of the receive socket buffer for the UDP protocol. This value must be less than max-buf. The value can be set to 2048 to max-buf.

Property	Type	Description
send-buf	count	Maximum size of the send socket buffer for the UDP protocol. This value must be less than max-buf. The value can be set to 2048 to max-buf.
smallest-anon-port	count	Lower bound on ephemeral or short-lived ports created when establishing out-bound network connects. Should be less than or equal to largest-anon-port. The value can be set to 1024 to largest-anon-port.
smallest-nonpriv-port	count	Smallest port number for a non-privileged port using the UDP protocol. The value range is 1024 to 32768.
reuseport-lbalg	astring	Reuseport load balancing. The value can be set to rr for round robin, src-dest-ip for source and destination IP address hashing, src-dest-ip-ports for source and destination IP address and port hashing, src-ip for source IP address hashing, or src-ip-port for source IP address and port hashing.

Global Static Route Properties

You can configure global static routes that are used by default for a system by including a static-routes property group description within the ip-interface-management:default service instance. Static routes for a particular interface should be included in the interface-ip property property group. For more information see:

The properties that can be included in a static-routes property group are listed below:

TABLE 37 Global Static Route Properties

Property	Type	Description
blackhole	boolean	Silently discard packages. Set the value to true to enable, false to disable.
destination	astring	Destination address.
destination-type	astring	Destination address type. The value can be set as host, net or any.
export	count	Lifetime for an entry.
family	astring	Address family. The value can be set as link, inet or inet6.
gateway	astring	Gateway address.
hopcount	count	Maximum hop count. The range can be between 1 and 255.
indirect	boolean	Add routes when gateway not on-link. Set the value to true to enable, false to disable.
isgateway	astring	Route is not directly reachable.
locks	astring	Lock specific metric values. The value can be expire, hopcount, mtu, recvpipe, sendpipe, ssthresh, rtt and rttvar.
mtu	count	Maximum transmission unit in bytes. The value range is 68 to 65536.
multirt	boolean	Create redundant route. Set the value to true to enable, false to disable.
netmask	astring	Netmask.

Property	Type	Description
private	astring	No advertising of the status of the route. Set the value to on to prevent advertising, off to allow advertising.
proto1	boolean	Protocol specific routing flag 1. Set the value to true to enable, false to disable.
proto2	boolean	Protocol specific routing flag 2. Set the value to true to enable, false to disable.
recvpipe	count	Receive pipe size in bytes.
reject	astring	ICMP unreachable when matched.
rtt	count	Round-trip time in microseconds.
rttvar	count	Round-trip time variance in microseconds.
secattr	astring	Security attributes of a route.
sendpipe	count	Send pipe size in bytes.
setsrc	astring	Assign default source address
ssthresh	count	Slow start threshold.

Index

D

E

F

K

L

M

N

O

P

R

S

T

U

V

W

Z