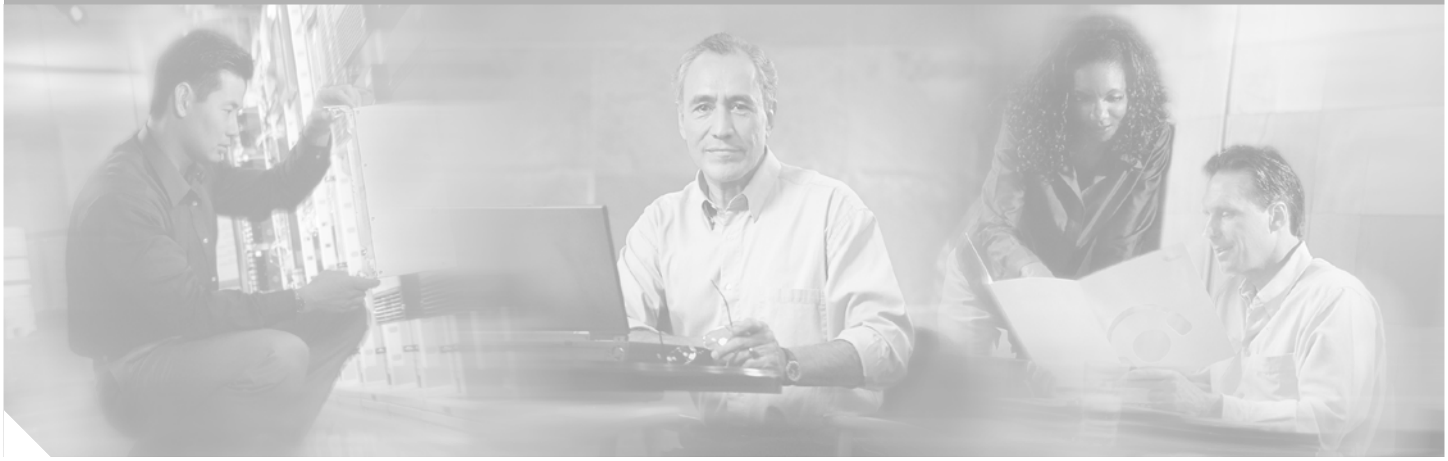


# Cisco Broadband Policy Manager Installation and Configuration Guide

Software Release 1.6

**Corporate Headquarters**
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 526-4100

Customer Order Number:
Text Part Number: OL-7773-01

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS were ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

*Cisco Broadband Policy Manager Software Installation Guide*

# Contents

# Preface

## Overview

This document discusses installing and configuring various Broadband Policy Managers (BPMs) and also using them in clusters to provide resiliency. Before you begin, you should ensure that you have approved platforms. See Chapter 2, *Platforms*, to determine if you have the correct platform.

You must perform the following tasks, described later in this document, to establish the required BPMs, provide resiliency, and perform the appropriate functions:

- *BPDS and BPS Installation*
- *Software Installation*
- *Additional Software Installation*

This chapter describes these topics:

- *Scope*
- *Audience*
- *Documentation Set*

## Scope

This guide provides instructions for installing software necessary for BPM operation and for configuring BPM systems.

## Audience

This guide is for the network or service administrator who installs BPM software. Before you begin, you should have the appropriate technical expertise. This document is not tutorial in nature.

## Documentation Set

The documentation for your Broadband Policy Manager (BPM) system includes the following documents:

- Cisco Broadband IP Service Module User Guide
- Cisco Broadband Policy Design Studio User Guide
- Cisco Broadband Policy Manager Installation and Configuration Guide
- Cisco Broadband Policy Manager Operations Guide
- Cisco Broadband Policy Manager Release Notes
- Cisco Capacity Admission Control Manager User Guide

### Cisco Broadband IP Service Module User Guide

This document discusses the Broadband IP Service Module for session management and network adaptation. It discusses its architecture, components, access methods, and functions.

### Cisco Broadband Policy Design Studio User Guide

This guide provides instructions for installing the Broadband Policy Design Studio (BPDS). It discusses how to use the BPDS to create, deploy, and manage network services and topologies.

### Cisco Broadband Policy Manager Installation and Configuration Guide

This guide describes how to install the software for the BPM. It describes how to install and configure the Solaris operating system for use by the BPM. It also includes procedures to install and configure the BPM software and the procedures to install and log into the BPDS.

### Cisco Broadband Policy Manager Operations Guide

This guide describes the use of the BPDS to obtain information, conduct day-to-day operations, perform maintenance tasks, and troubleshoot problems with the BPM system. These tasks include use of the Log Messages addendum, the Application Log Messages addendum, and the Statistics addendum.

### Cisco Broadband Policy Manager Release Notes

This document describes new features, known limitations, and other important information about the BPM system.

### Cisco Capacity Admission Control Manager User Guide

This document discusses the architecture and components for the Capacity Admission Control Manager product.

## Organization

This guide contains the following chapters:

### Chapter 1 - *Introduction*

This chapter describes the system architecture.

### Chapter 2 - *Platforms*

This chapter describes the hardware platforms supported.

### Chapter 3 - *BPDS and BPS Installation*

This chapter describes the installation and configuration of the Broadband Policy Design Studio (BPDS) software.

### Chapter 4 - *Software Installation*

This chapter describes the installation and configuration of the BPM software.

### Chapter 5 - *Additional Software Installation*

This chapter describes the installation and configuration of additional software.

### Appendix A - *Appendix A - Glossary*

This chapter contains a glossary of terms used in this document.

### Appendix B - *Appendix B - Creating and Provisioning R-Realms*

This chapter describes the process of creating and provisioning R-Realms.

# 1

# Introduction

## Overview

This document provides information for installing and configuring the Broadband Policy Manager (BPM) and related products.

The Admission Control Manager (ACM) provides Capacity Admission Control (CAC) services over broadband access networks. Admission control monitors, controls, and enforces the use of network resources and services with policy-based management. The criteria for policy-based management include identifying users and applications, or identifying traffic based on how, when, and where it enters the network.

The Session Manager (SM) product tracks user sessions connecting to the broadband access network. The SM product offers the ability to add per-subscriber session management storage capability to a policy control solution. It also supplies information for mapping subscribers to physical network devices and ports and provides valuable information to the topology information model. It deploys alongside the NM and CAC products to support them.

The Network Manager (NM) product provides facilities for controlling and querying elements in the network. The NM product offers a variety of interfaces, including Netconf (CLI), SNMP, and RADIUS COA. It deploys alongside the SM and CAC products to support them.

This chapter describes these topics:

- *Basic BPM Concepts*

## Basic BPM Concepts

At the highest level, the Broadband Policy Manager (BPM) consists of a visual development environment, a deployment manager, and execution environments. These elements cooperate to provide a product-line architecture for service-oriented systems. This environment enables and simplifies the repeated production of many related solutions for real-time network policy management.

In the BPM product-line architecture, dataflow programming is the primary compositional element underlying all solutions. Dataflow programming is a well-established programming methodology that promotes the explicit description of data movement and transformation in program execution. Using the development environment component, users develop dataflow programs interactively with a drag-and-drop visual programming tool, the Broadband Policy Design Studio (BPDS) graphical user interface (GUI). These dataflow programs describe the movement of data between operators, which are black-box programming elements exposed by software agents. These agents generally encapsulate specific implementations of abstractions like protocols, network devices, data sources, or logic capabilities. The generated dataflow programming artifacts combine into collections of services, referred to as applications, which provide complete solutions.

The deployment manager publishes these applications into the execution environment. The execution environment is a distributed domain of networked processing nodes. A node is a computer or some other device on a network. Every node has a unique network address. A link is a line or channel over which data is transmitted.

Each node runs a highly concurrent graph-traversal engine, coupled with a fast data switching fabric. Once published to an execution environment on a node, application services are available for execution and monitoring. The execution environment also provides resiliency and failover capabilities to published applications.

This section discusses the following BPM elements:

- *Flows*
- *Services*
- *Rules*
- *Policy Functions*
- *Agents*

The administrator uses the BPS BPDS to deploy and manage these BPM components.

## Flows

In the BPM system, a *flow* (Figure 1) represents movement of data or control and is a basic element of a *service*. The BPM system contains prebuilt flows, services, rules, and policy functions for your immediate use.

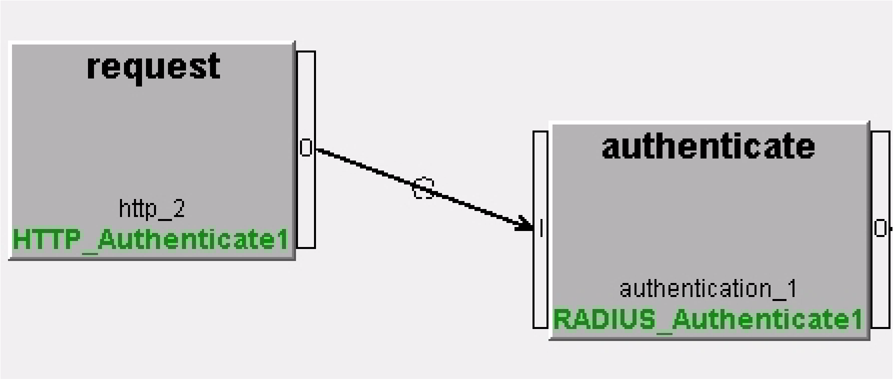


**Figure 1. A Simple Flow**

Figure 1 shows a flow composed of operators connected by routes.

## Services

A simple *service* might contain one flow; however, a service can contain multiple flows (Figure 2).

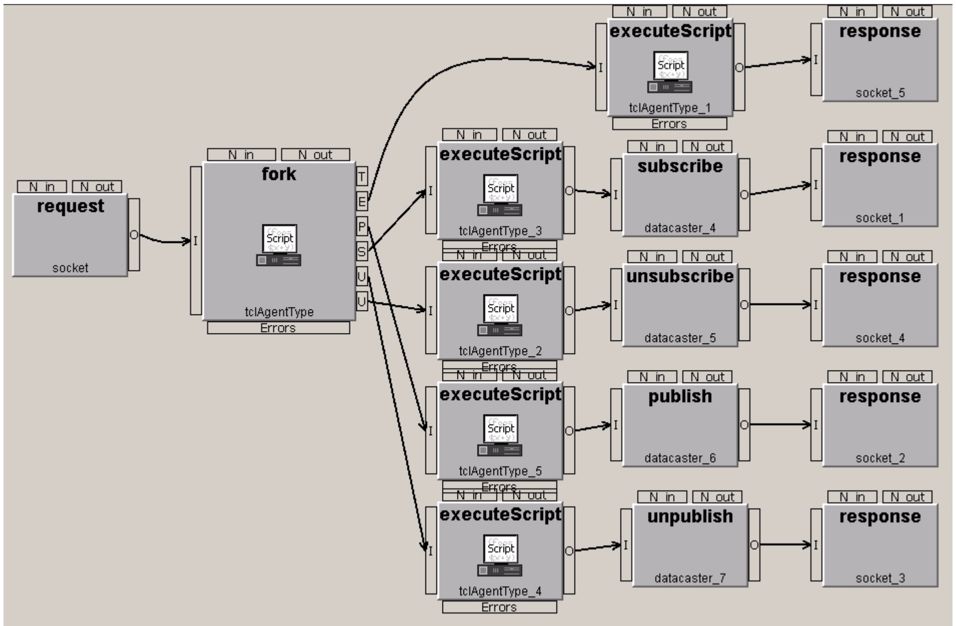


**Figure 2. A Sample Service**

Once a service is created, it can become part of larger, more complex, services. Cisco uses flows and services to develop customized *applications*. The service development environment simplifies the design process, enabling service designers to create complex services and policies by visually interconnecting reusable service assets, in the desired sequence for a particular offering (such as video on demand).

## Rules

A *rule* is a script to perform tasks, such as data manipulation or database queries. One rule can invoke another rule. When encapsulated in a *policy function*, you can include rules in flows just like any other operator.

## Policy Functions

A policy function is a TCL script that can be used as an operator in a flow, to provide a reusable unit of functionality.  For example, a policy function can be used to process or manipulate data.

## Agents

An *agent* is responsible for protocol level interaction with a particular device type or service. For example, a RADIUS agent contains software that allows the interaction with a specific RADIUS device by indicating the device type, IP address, and port number. Using this configuration, flows perform various accounting operations. Agents are invoked with a set of parameters, which they translate into the native protocol of the device, and perform a remote service before returning the result of the operation to the flow.

# 2

# Platforms

## Overview

This section describes the platform requirements for building a basic Broadband Policy Manager (BPM) system and the minimum Solaris operating system configuration required. In addition, this section discusses file system partitions, clusters, and packages, and lists those systems that were tested with this release.

This chapter describes these topics:

- *Tested Systems*
- *Solaris Operating System*

## Tested Systems

This section provides a list of systems that were tested with this release of the BPM. This information is provided for reference only and may change as newer releases or patches become available.

If you require a configuration with a specific model or part number, go to http://www.sun.com or contact your Sun Microsystems, Inc. sales representative.

### Sun Fire V20z

The following describes the Sun Fire V20z tested:

- Sun Fire V20z Server
- 2 CPUs
- 4 GB of memory
- 4 Ethernet ports for clustered systems and 2 for unclustered
- Optional: external SCSI controller (for use with Sun StorEdge Array)

### Sun Fire V40z

The following describes the Sun Fire V40z tested:

- Sun Fire V40z Server
- 4 CPUs
- 8 GB of memory

- 4 Ethernet ports for clustered systems and 2 for unclustered
- Optional: external SCSI controller (for use with StorEdge array)
- Optional: internal RAID controller
- Optional: additional internal disk

## Sun Fire x4100

The following describes the Sun Fire x4100 tested:

- Sun Fire x4100 Server
- 4 dual-core CPUs
- 8 GB of memory
- 4 Ethernet ports

## Sun StorEdge 3310 SCSI Array

We have tested the Sun StorEdge 3310 SCSI Array for external RAID storage. The following describes the Sun StorEdge 3310 SCSI Array tested:

- Sun StorEdge 3310 SCSI Array
- 12 73 GB disks
- Optional: 12 73 GB 15k RPM disks (upgrade to standard 10k RPM)
- RAID controller
- SCSI cables (including 1 jumper and 2 system cables)

# Solaris Operating System

This section describes the Solaris operating system information.

## Release Information

The BPM is built to run on Solaris x86 10 6/06 or newer with the packages included in the standard End-User installation and the latest Recommended Patch Cluster available from SunSolve. Additional packages installed on the tested systems include: SUNWast, SUNWless, SUNWman, SUNWtcsh, and SUNWbtool.

## Solaris Configuration

This section describes the configuration of the Solaris operating system.

### File System Partition

The base installation consists of four file systems on an internal disk drive of at least 36 GB with at least the minimum partition sizes listed below.

**Table 1. File System Partition Information**

| Partition | Size | Detail |
|---|---|---|
| / | as necessary | UFS with logging |
| swap | equal to or grater than two times physical memory | UFS with logging |
| /var | 8 GB minimum | UFS with logging |
| /opt | 16 GB minimum | UFS with logging |

# 3

# BPDS and BPS Installation

## Overview

The Broadband Policy Design Studio (BPDS) is a graphical user interface to the Broadband Policy Manager (BPM). You can use the BPDS to create services, obtain BPM information, conduct day-to-day operations, perform necessary maintenance tasks, and troubleshoot problems. The BPDS includes the following components:

- Network Administration (or Network Admin) - This feature provides BPM information from a network point of view, including agents and services.
- Service Design - This feature allows you to create new services and to view the structure of existing services.

The Broadband Policy Studio (BPS) is a graphical user interface to the BPM, similar to the BPDS. The BPS does not have the Service Design feature. You can use the BPS to obtain Broadband Policy Manager information, conduct day-to-day operations, perform necessary maintenance tasks, and troubleshoot problems. This chapter discusses installing and logging into the BPDS and BPS. The information includes:

- *BPDS or BPS Installation Procedure*
- *BPDS or BPS Login*

# BPDS or BPS Installation Procedure

Depending on the product you are using, you may install either the BPDS or the BPS software. The instructions for installing each product are nearly identical.

Follow the steps in this section to install the BPDS or BPS.

## Procedure: Installing the BPDS or BPS

1. Insert the BPDS or BPS Installation CD into a drive on your computer. The BPDS or BPS Client Installer starts automatically and presents the initial **Cisco** screen (Figure 3).

   **Figure 3. Cisco Systems Screen**

   © 2005 Cisco Systems, Inc
   Cisco, Cisco Systems and Cisco Systems logo are registered trademarks
   of Cisco Systems, Inc. and/or its affiliates in the U.S. and certain other countries

   The system displays the **Introduction** screen.

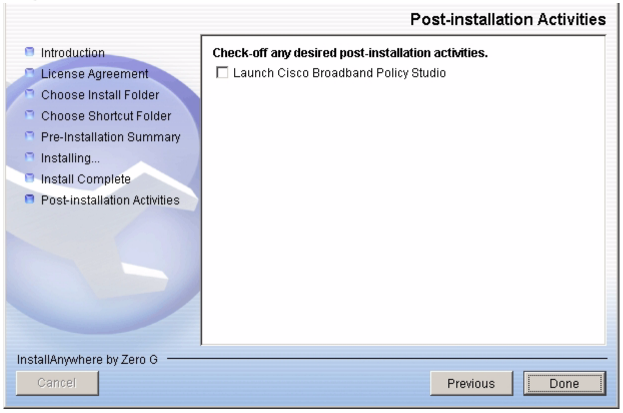
2. Click **Next** to continue. The system displays the license agreement.
3. Click **Next** to continue. The system displays the **Choose Install Folder** Screen.
4. Accept the default location selection and click **Next** to continue. The system displays the **Choose Shortcut Folder** screen.
5. Accept the default location selection and click **Next** to continue. The system displays the **Pre-Installation Summary** screen.
6. Review the installation details and click **Install** to continue.

   The system displays the **Installing Broadband Policy Studio** or **Installing Cisco Broadband Policy Design Studio** screen.

7. When the system displays the **Install Complete** screen, click **Next**.

8. The system presents you with options to launch the BPDS or BPS (Figure 4). Check the option to Launch Studio and click **Done.**

**Figure 4. BPDS or BPS Post-installation Activities Screen**

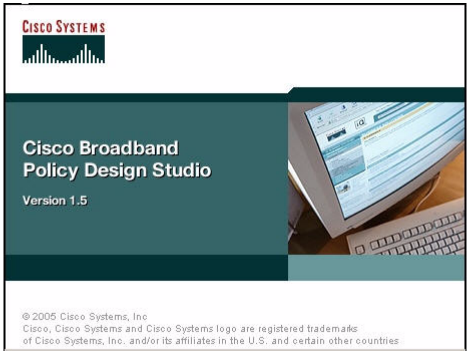


Check-off any desired post-installation activities.

□ Launch Cisco Broadband Policy Studio

9. The system launches the BPDS or BPS. The system displays the **Welcome** screen (Figure 5).

**Figure 5. BPDS Welcome Screen**



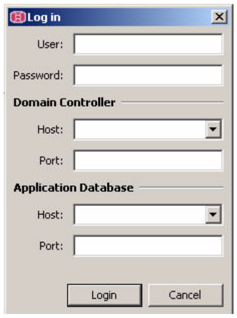
Cisco Broadband Policy Design Studio

Version 1.5

© 2005 Cisco Systems, Inc
Cisco, Cisco Systems and Cisco Systems logo are registered trademaks of Cisco Systems, Inc. and/or its affiliates in the U.S. and certain other countries

The system displays the **Log in** screen (Figure 6).

**Figure 6. BPDS or BPS Log In Screen**



# BPDS or BPS Login

Depending on the product you are using, you may log in to either the BPDS or the BPS software. The instructions for logging in to each product are nearly identical.

When you see the **Log in** screen (Figure 6), you can log in to the BPDS or BPS. Refer to Table 2 to determine the appropriate authentication details for your system.

**Table 2. Log in Screen Information**

| Item | Detail | Default |
|---|---|---|
| User | Name that identifies the user. | administrator |
| Password | Character string that verifies the user name. | administrator |
| Domain Controller Host | Name or IP address of the host system where the domain controller is running. The Domain Controller is a standalone system responsible for domain management, including application deployment, configuration, and health for all systems in the domain. Only one Domain Controller exists per domain. | <local_machine> |
| Domain Controller Port | Port number of the TCP port through which the domain controller communicates with the network. | 10000 |
| Application Database Host | Name or IP address of the host system where the application database is running. The application database stores information used by the application. Different applications require different data. | <local_machine> |
| Application Database Port | Port number of the TCP port through which the application database communicates with the network. | 10005 |

## Procedure: Logging in to the BPDS or BPS System

Use this procedure to log in to the BPDS or BPS system.

1. Enter your user name in the **User** text box on the Authentication Screen.
2. Enter your password in the **Password** text box.
3. Enter the domain controller host name or IP address in the **Host** text box. Use the host name or IP address that you assigned to the domain controller.

4. Enter the domain controller port number in the **Port** text box. Use the port number through which you want your domain controller to communicate with the network. This should be the number you assigned during installation (base install port).

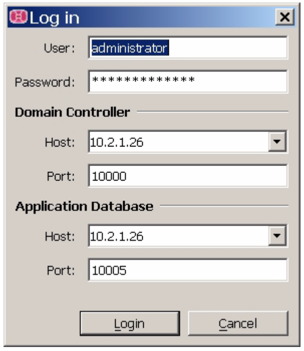
**Note:** *Ensure that the default port number is open and available.*

5. Enter the application database host name or IP address in the **Host** text box. Use the host name or IP address that you assigned to the application database.

6. Enter the application database port number in the **Port** text box. Use the port number through which you want your application database to communicate with the network. This should be the number you assigned during installation (base install port + 5).

7. The display is similar to Figure 7. Click **Login.**

Figure 7. BPDS or BPS - Completed Log In Screen



Log in

User: administrator

Password: *************

Domain Controller

Host: 10.2.1.26

Port: 10000

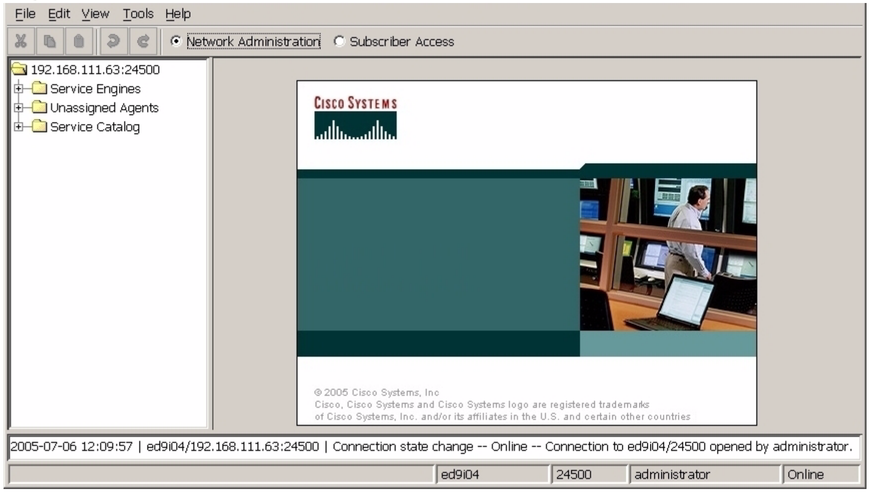
Application Database

Host: 10.2.1.26

Port: 10005

The system displays the initial BPDS or BPS screen.

**Figure 8. Initial BPDS or BPS Screen**



© 2005 Cisco Systems, Inc
Cisco, Cisco Systems and Cisco Systems logo are registered trademarks
of Cisco Systems, Inc. and/or its affiliates in the U.S. and certain other countries

```
2005-07-06 12:09:57 | ed9i04/192.168.111.63:24500 | Connection state change -- Online -- Connection to ed9i04/24500 opened by administrator.
```

| ed9i04 | 24500 | administrator | Online |
|---|---|---|---|

**Note:** *Your actual screen may look different from this sample screen.*

# 4

# Software Installation

## Overview

This chapter describes the process you must follow to successfully install this product.

This chapter includes the following topics:

- *Realms*
- *Before Installation*
- *Installing ACM, SM, and NM*

An installation consists of a Domain Controller system, a resilient pair of Topology Database Servers, one or more Directors, and one or more Resource Controllers (optionally in resilient pairs). Install these components in the following order:

- Install the Domain Controller
- Install the resilient pair of Topology Database Servers
- Install the Resource Controllers
- Install the Directors (ACM only)

## Realms

Topology, resources, active sessions, and active contexts exist in an information *realm*. The realm improves performance by restricting lookups and updates against smaller data sets, providing less lock contention and faster search times. It also allows a Resource Controller to comprehend the realms for which it is responsible. If a request involves a realm that the Resource Controller does not own, it can ignore the request (if that is its configured behavior). When a new Resource Controller is introduced, BRAS responsibility migrates from one Resource Controller to a new Resource Controller. The realm concept allows the information model to consider this a block movement of a realm. The migration affects only the realm that is moving. When scaling the BPM to accommodate more hardware and repartitioning the realms, only the realm/BRAS being moved is unavailable to quality of service (QoS) requests. Calls originating or terminating in other realms remain uninterrupted.

A BRAS defines a realm of self-contained information. The mapping of Resource Controllers to BRASs allows a single Resource Controller to handle multiple BRASs and their access. The state maintenance of various components is specified at the granularity of the Resource Controller. Thus, a single Topology Store Function (TSF) element handles more than one BRAS.

## Domain Realm

The Topology Database Server houses the domain realm, which maintains application-level information about the topology. The nodes in the topology are Director and Resource Controller systems. The Topology Database Server uses the Domain Realm to understand the system topology. Links represent connectivity between cluster pairs. This allows the Topology Database Server to act as the central contact point for the application portal and as the main owner on application-level configuration and management. Resources represent interfaces on the component systems, the health of each system, cluster information, and the configuration of each system.

## Director Realm

The Director realm stores information about what Resource Controller is responsible for a given device (such as a BRAS), and what IP address pools a given BRAS handles. The Topology Database Server maintains the realm, and distributes it to each Director when there are updates. Nodes in this realm represent Resource Realms. A Director uses this information to forward an incoming request to the correct Resource Controller.

## Resource Realm

A resource is any device or other item that can be used, such as a printer, disk drive, or memory. Resources, topologies, active sessions, and active contexts exist in a realm. A Resource Realm is a realm distributed to a Resource Controller. The Resource Controller Realm defines the topology of a given device (for example, a BRAS). It represents the ports, VPs, VCs, and assigned CPE devices for that BRAS. The realm improves performance by restricting lookups and updates against smaller data sets.

## Network Realm

The Network Realm stores specific network adaptation information, (such as the devices active on a particular Resource Controller), profiles, devices, and handlers. The Network Realm is centrally provisioned on the Topology Database Server, and distributed to all Resource Controllers.

## Session Realm

A Session Realm, unique to a Resource Controller, improves performance by restricting lookups and updates against smaller data sets.

# Before Installation

Review this section before you install your software.

## About Software Packages

The software is packaged in discrete packages. Table 3 details the packages.

**Table 3. Software Packages**

| Filename | Description |
|---|---|
| backend-<version>.sh | Backend software. Includes apache plug-in. |

**Table 3. Software Packages**

| | |
|---|---|
| QosServer.tzz | ACM product package containing Resource Controller application, Director application, and Topology Database Server application. |
| smnm_tds.tzz | SM/NM product package for Topology Database Server. |
| smnm_rc.tzz | SM/NM product package for Resource Controller. |
| smnm_base_pif_daf.tzz | SM/NM product package containing the base PIFs and DAFs used by the SM and NM products. |

The `QosServer.tzz` package includes these templates:

- `rc-template`
- `spdf-template`
- `tds-template`
- `tds-gansett-template`
- `pdir-template`

## About Backend Installer

The backend installer always installs all service engine elements, and prompts you to determine if the system should also run the domain controller.

### Using the –dc Installer Option

When run interactively, the installer prompts you to run the domain controller. However, when you run the installer with either the `-T` or `-t` option, the domain controller is *disabled* by default. To enable the domain controller, you must use the `-dc` option. The following example installs a service engine and domain controller:

```
backend<version>.sh -T -d <install_dir>/tazz -dc
```

In contrast, the following command installs the same elements, yet it disables the domain controller. It does not start the database.

```
backend<version>.sh -T -d <install_dir>/tazz
```

### Specifying Number of Connections

While the default of 20 domain database connections and engine database connections is reasonable, you should specify 48 as the number of application database connections. This number includes connections for the policy database, backend processes, command-line interface, and other database connections. To do this, use the `-ca 48` option. This overrides the defaults.

### Installing Apache

The `backend` script always installs apache. The `-apache` flag specifies that apache starts whenever `start_tazz` runs and stops whenever `stop_tazz` runs.

# Installing ACM, SM, and NM

This installation procedure includes steps for installing the ACM product and the optional SM and NM products.

The Admission Control Manager (ACM) provides Capacity Admission Control (CAC) services over broadband access networks. Admission control monitors, controls, and enforces the use of network resources and services with policy-based management. The criteria for policy-based management include identifying users and applications, or identifying traffic based on how, when, and where it enters the network.

The Session Manager (SM) product tracks user sessions connecting to the broadband access network. The SM product offers the ability to add per-subscriber session management storage capability to a policy control solution. It also supplies information for mapping subscribers to physical network devices and ports and provides valuable information to the topology information model. It deploys alongside the NM and CAC products to support them.

The Network Manager (NM) product provides facilities for controlling and querying elements in the network. The NM product offers a variety of interfaces, including Netconf (CLI), SNMP, and RADIUS COA. It deploys alongside the SM and CAC products to support them.

## Creating Group and User

On each system, create group and user by following these steps:

1. Create the tazzadmin group by entering this command on the Domain Controller:

```
groupadd tazzadmin
```

2. Create the tazzadmin user by entering this command on the Domain Controller:

```
useradd -s /usr/bin/tcsh -d /home/tazzadmin -g tazzadmin -m
   tazzadmin
```

## Installing Backends

Install the backend software on each of the BPM system types:

- Single Domain Controller BPM
- Topology Database Server BPM
- Director BPM (ACM only)
- One or more Resource Controller BPMs

You can install the Domain Controller software on the Director system (ACM only). Do not install the Domain Controller on a Resource Controller. The minimum number of computers needed for an installation is three: one for the Domain Controller/Director, one for the Topology Database Server, and one for a Resource Controller.

To install backends, follow this procedure:

**Note:** *After installing each backend, start the BPM by entering this command as user tazzadmin:*

```
<install_dir>/bin/start_tazz
```

*where <install_dir> is the installation directory. If you did not use the* `-apache` *flag when running* `backend`*, use the* `-apache` *flag now to start apache.*

1. Enter this command as user tazzadmin to install the Domain Controller backend on the Domain Controller:

```
backend<version>.sh -T -d <install_dir>/tazz -dc
```

2. For each Director (ACM only), install a Director backend by following these steps:

   a. To install the Director with a Domain Controller, enter this command as user tazzadmin on the Director:

   ```
   backend<version>.sh -T -d <install_dir>/tazz -dc -ca 48 -apache
   ```

   b. To install the Director without a Domain Controller, enter this command as user tazzadmin on the Director:

   ```
   backend<version>.sh -T -d <install_dir>/tazz -ca 48 -apache
   ```

3. Install a Resource Controller backend. If you have a clustered Resource Controller, install the backend on both the Resource Controller active and standby. This also adds a service engine to the Resource Controller active and one to the standby.

   Enter this command as user tazzadmin on the Resource Controller:

   ```
   backend<version>.sh -T -d <install_dir>/tazz -ca 48 -apache
   ```

4. Install the Topology Database Server backend. If you have a clustered Topology Database Server, install the backend on both the Topology Database Server active and standby. This step also adds a service engine to the Topology Database Server active and one to the standby.

   Enter this command as user tazzadmin on the Topology Database Server:

   ```
   backend<version>.sh -T -d <install_dir>/tazz -ca 48 -apache
   ```

**Note:** *After installing each backend, start the BPM by entering this command as user tazzadmin:*

```
<install_dir>/bin/start_tazz
```

*where <install_dir> is the installation directory. If you did not use the* `-apache` *flag when running* `backend`*, use the* `-apache` *flag now to start apache.*

## Installing Agents

Install agents on all systems by following these steps:

1. Use the BPS (see Chapter 4, *BPDS and BPS Installation*) to connect to the system.

2. Install this software package:

   - `schemas.tzz`

**Note:** *To install a software package, choose* **Tools --> TAZZ Packages --> Install***, then navigate to the package directory.*

3. After schemas are installed, use the BPS to install this software package:

   - `qostypes.tzz`

4. Use the BPS to install these agent software packages:
   - `controller.tzz`
   - `ctf.tzz`
   - `error.tzz`
   - `event.tzz`
   - `log.tzz`
   - `netconf.tzz`
   - `poolmap.tzz`
   - `radius.tzz`
   - `scheduler.tzz`
   - `snmpalarm.tzz`
   - `soap.tzz`
   - `socket.tzz`
   - `ssf.tzz`
   - `statisticsmap.tzz`
   - `stringops.tzz`
   - `super.tzz`
   - `tcl.tzz`
   - `tsfc.tzz`
   - `vista.tzz`

## Installing Application Packages

Install application packages on all systems by following these steps:

1. Use the BPS to install this software package:
   - `qosServer.tzz`
2. If installing SM or NM, install these software packages:
   - `smnm_tds.tzz`
   - `smnm_rc.tzz`
   - `smnm_base_pif_daf.tzz`

## Creating Topology Database Server Service Engine

Add Topology Database Server service engine to all systems by following these steps:

1. Using the BPS, on the Network Administration window, right-click the Service Engines folder and choose **Add Service Engine** from the drop-down menu. The New Service Engine dialog appears.

2. Enter the information for the active service engine, including a name, such as `TDS-A`, and the target system's hostname and port.

**Note:** *The port offset is +10 from the base port you specified in the backend install for that target system.*

3. Click **Next** and **OK**.
4. Perform Step 1 through Step 3 for the standby Topology Database Server, including a name, such as `TDS-B`.

**Note:** *The port offset is +10 from the base port you specified in the backend install for that target system.*

## Copying Topology Database Server Service Engine Template

Copy Topology Database Server service engine template, by performing these steps:

1. Use the BPS to log in to the Topology Database Server.
2. Right-click the `tds-template` service engine and choose **Copy Configuration** from the drop-down menu. The Copy Engine dialog appears.
3. Select the active member of the Topology Database Server cluster, and click **Next**. This copies the agents and services from the template system to the active Topology Database Server system.

## Copying SM/NM Topology Database Server Service Engine Template (optional)

If installing SM or NM, copy SM/NM Topology Database Server service engine template, by performing these steps:

1. Use the BPS to log in to the Topology Database Server.
2. Right-click the `smnm-tds-template` service engine and choose **Copy Configuration** from the drop-down menu. The Copy Engine dialog appears.
3. Select the active member of the Topology Database Server cluster, and click **Next**. This copies the agents and services from the template system to the active Topology Database Server system.

## Editing the Topology Database Server Event Agent

On the Topology Database Server active system, change the IP address of the `tds_event` agent to the IP address of the Domain Controller host, instead of the default value.

## Deploying Agents and Services on Topology Database Server

Using the BPS, right-click the active Topology Database Server service engine and choose **Deploy All** from the drop-down menu.

## Running Setup on the Topology Database Server

Run setup on the Topology Database Server by following these steps:

1. On the active Topology Database Server, invoke the tash by entering this command:

   ```
   tash
   ```

2. Run this command:

   ```
   setupTds
   ```

3. Exit the tash by running this command:

   ```
   exit
   ```

## Clustering the Topology Database Server Service Engines

Use the BPS to cluster the active and standby Topology Database Server service engines. Do **not** check **Hot Standby** on the Select Standby Engine dialog. Replicate both the service engine and policy databases. On the active and standby Topology Database Server, remove the `/etc/hostname.<primary_interface>` and `/etc/hostname.<secondary_interface>` files.

## Creating the Topology Information File

Creating topology information files is complex and time-consuming. Please consult *Appendix B - Creating and Provisioning R-Realms* for detailed information.

## Adding Resource Controller Systems to the Domain

Add Resource Controller systems into the domain. For each active Resource Controller, follow these steps:

1. Add a service engine for the Resource Controller, following this procedure:

   a. Using the BPS, on the Network Administration window, right-click the Service Engines folder and choose **Add Service Engine** from the drop-down menu. The New Service Engine dialog appears.

   b. Enter the information for the active service engine, including name, and the target system's hostname and port.

   **Note:** *The port offset is +10 from the base port you specified in the backend install for that target system.*

   c. Click **Next** and **OK**.

2. If the Resource Controller is a cluster member, then follow this procedure for the standby Resource Controller:

   a. Using the BPS, on the Network Administration window, right-click the Service Engines folder and choose **Add Service Engine** from the drop-down menu. The New Service Engine dialog appears.

b. Enter the information for the standby service engine, including name, and the target system's hostname and port.

**Note:** *The port offset is +10 from the base port you specified in the backend install for that target system.*

c. Click **Next** and **OK**.

d. Cluster the active and standby Resource Controller service engines. Check **Hot Standby** on the Select Standby Engine dialog.

3. Right-click the service engine named `aracf-template` and choose **Copy Configuration** from the drop-down menu. The Copy Engine dialog appears.

4. Select the active (or standalone) Resource Controller, and click **Next**. This copies the agents and services from the template system to the active Resource Controller.

5. If installing SM or NM, right-click the service engine named `smnm-rc-template` and choose **Copy Configuration** from the drop-down menu. The Copy Engine dialog appears. Select the active (or standalone) Resource Controller, and click **Next**. This copies the agents and services from the template system to the active Resource Controller.

6. If installing SM or NM, right-click the service engine named `smnm-base-pif-daf-template` and choose **Copy Configuration** from the drop-down menu. The Copy Engine dialog appears. Select the active (or standalone) Resource Controller, and click **Next**. This copies the agents and services from the template system to the active Resource Controller.

7. Right-click the active cluster member and choose **Deploy All** from the drop-down menu.

8. After adding each Resource Controller, verify that it has been successfully added. For example, after adding the first Resource Controller, run commands on the Topology Database Server similar to the following. Note that after adding the second Resource Controller, the first command should return two entries, and so forth. Note also that the output of the `SN domain` command will show an entry for each system added already.

```
% SN domain
     id                       | active
    --------------------------+--------
    192.168.111.52:10000 | t
% SRoN domain 192.168.111.52:10000
     node                | rtype          | rid
    ---------------------+---------------+---------------------
    192.168.111.52:10000 | configuration | 192.168.111.52:10000
    192.168.111.52:10000 | health        | 192.168.111.52:10000
    192.168.111.52:10000 | interface     | 192.168.111.52
    192.168.111.52:10000 | role          | aracf
    192.168.111.52:10000 | configuration | aracf
```

For a Resource Controller, the `role` should be `aracf` and the `configuration` should be `aracf`.

## Adding Director Systems to the Domain (ACM Only)

If installing an ACM, add Director systems into the domain. For each active Director, follow these steps:

1. Add a service engine for the Director, following this procedure:
   - **a.** Using the BPS, on the Network Administration window, right-click the Service Engines folder and choose **Add Service Engine** from the drop-down menu. The New Service Engine dialog appears.
   - **b.** Enter the information for the service engine, including name, and the target system's hostname and port.

     **Note:** *The port offset is +10 from the base port you specified in the backend install for that target system.*

   - **c.** Click **Next** and **OK**.
2. Right-click the service engine named `spdf-template` and choose **Copy Configuration** from the drop-down menu. The Copy Engine dialog appears.
3. Select the Director, and click **Next**. This copies the agents and services from the template system to the Director.
4. Right-click the Director and choose **Deploy All** from the drop-down menu.
5. After adding each Director, verify that it has been successfully added. For example, after adding the first Director, run commands on the Topology Database Server similar to the following. Note that after adding the second Director, the first command should return two entries, and so forth. Note also that the output of the `SN domain` command will show an entry for each system added already.

```
% SN domain
     id                       | active
    --------------------------+--------
    192.168.111.52:10000 | t
% SRoN domain 192.168.111.52:10000
     node                 | rtype          | rid
    ---------------------+---------------+---------------------
    192.168.111.52:10000 | configuration | 192.168.111.52:10000
    192.168.111.52:10000 | health        | 192.168.111.52:10000
    192.168.111.52:10000 | interface     | 192.168.111.52
    192.168.111.52:10000 | role          | spdf
    192.168.111.52:10000 | configuration | spdf
    192.168.111.52:10000 | realm         | App1
```

For a Director, the `role` should be `spdf` and the `configuration` should be `spdf`.

6. As you add each Director, make sure the topology on that Director looks accurate by running the following `tash` command on the Director. If the output shows an `aracf` resource and a `pool` resource, the system has its initial topology correct.

```
% SR App1
        Resource Type: aracf
        id | health | host | port | qos | saf | fof | _health | _host | _port | _qos | _saf | _fof
        ---+--------+------+------+-----+-----+-----+---------+-------+-------+------+------+-----
        -
        (0 rows)
        Resource Type: pool
        id | ip | mask | vpn | data
        ---+----+------+-----+------
        (0 rows)
```

## Creating the Resource Realms and IP Pools

After you know what Resource Realms to provision on your system, set them up using the following appropriate `tash` commands. The `AddResourceRealm` and `AssignResourceRealm` operations may take a long time to complete, and therefore should be passed in `30` (seconds) as a timeout value.

- Add the Resource Realms that are defined in the `.tim` files into the system, using `tash AddResourceRealm` commands.
- For the ACM only, add appropriate IP pools to the Resource Realms, using `tash AddPool` commands.
- Assign each Resource Realm to a Resource Controller, using `tash AssignResourceRealm` commands

For example, the following commands add four Resource Realms, then (for ACM only) add one IP pool to each one, then assign two of those Resource Realms to an Resource Controller, and the other two to a different Resource Controller. In this example, `app1` is the name of the Director realm and 30 is a sample timeout value.

```
tash
AddResourceRealm r192_9_21_1 30
AddResourceRealm r192_9_21_2 30
AddResourceRealm r192_9_21_3 30
AddResourceRealm r192_9_21_4 30
AddPool app1 192.170.1.1 24 r192_9_21_1
AddPool app1 192.170.5.1 24 r192_9_21_2
AddPool app1 192.154.10.1 24 r192_9_21_3
AddPool app1 192.154.12.1 24 r192_9_21_4
AssignResourceRealm r192_9_21_1 192.168.111.82:10000 30
AssignResourceRealm r192_9_21_2 192.168.111.82:10000 30
AssignResourceRealm r192_9_21_3 192.168.111.92:10000 30
AssignResourceRealm r192_9_21_4 192.168.111.92:10000 30
```

## Provisioning the Resource Realms

Follow this procedure:

1. On the Topology Database Server active system, invoke the tash by entering this command (ignore any warnings about the Tcl agent not running):

   ```
   tash
   ```

2. Run this command:

   ```
   Provision <topo_file>
   ```

where `<topo_file>` is a topology file (see *Creating the Topology Information File*).

3. Exit the tash by running this command:

   ```
   exit
   ```

# Appendix A - Glossary

This appendix contains abbreviations, acronyms, terms, and their definitions.

**Table A-1. Terms and Definitions.**

| Term | Definition |
|---|---|
| **A** | |
| Accounting Log Function | ALF. The Accounting Log Function records entrance parameters, internal decisions, and exit responses. |
| ACF | Admission Control Function. The ACF provides the core logic for performing admission control. It is programmed with a set of policies that define admission control behavior. |
| Action | An action is an operational category for changing, or inquiring about, a network element. |
| Active BPM | In a pair of BPMs, the active BPM processes requests. A standby BPM constantly monitors the health of the active BPM. If the active BPM is not viable, the standby BPM becomes the active BPM. |
| Admission Control Function | ACF. The ACF provides the core logic for performing admission control. It is programmed with a set of policies that define admission control behavior. |
| Agent | An internal BPM component that interacts with a device. The designer creates the agent and configures it to interact with a specific device by indicating the device type, IP address, and port number. The designer then assigns the agent to perform service functions. |
| Agent Configuration | Agent information that comprises a specific agent type instance. For example, a RADIUS agent configuration contains appropriate IP address, port, and shared secret values for a RADIUS agent type. |
| Agent Function | The service designer uses the BPDS to drag and drop an agent function into a flow in the BPDS. An agent, interacting with a device, performs the actual operation. |
| Agent Instance | A running instance of an agent type. |
| Agent Package | Software that allows agents to interact with a particular device type. For example, a RADIUS agent package contains software that allows the creation of agents that interact with specific RADIUS devices. |
| Agent Type | The agent type describes a particular type of agent that you can load onto the system. You select the agent type when you create the agent instance. |
| AI | Application Interface. The underlying frameworks use Application Interfaces to notify the Application of network events. The Cisco framework provides these interfaces. |
| Alarm Notification Function | ANF. The Alarm Notification Function issues SNMP traps to alert external systems of aberrant behavior in the BPM. |
| ALF | Accounting Log Function. The Accounting Log Function records entrance parameters, internal decisions, and exit responses. |

Table A-1. Terms and Definitions.

| Term | Definition |
|---|---|
| ANF | Alarm Notification Function. The Alarm Notification Function (ANF) issues SNMP traps to alert external systems of aberrant behavior in the BPM. |
| API | Application Program Interface. An API is a set of routines, protocols, and tools for building software applications. An API makes it easier to develop a program by providing the required building blocks. A programmer puts the blocks together. |
| Application | A service that maps business models and operational procedures directly into IP services, executable by their customers, for example, video on demand or automatic backup. See also Service. |
| Application Interface | AI. The underlying frameworks use Application Interfaces to notify the Application of network events. The Cisco framework provides these interfaces. |
| Application Program Interface | API. An API is a set of routines, protocols, and tools for building software applications. An API makes it easier to develop a program by providing the required building blocks. A programmer puts the blocks together. |
| Application Service Provider | ASP. An ASP is a business that provides computer-based services to customers over a network. |
| ASP | Application Service Provider. An ASP is a business that provides computer-based services to customers over a network. |
| Asynchronous Transfer Mode | ATM. Asynchronous Transfer Mode is a network technology based on transferring data in cells or packets of a fixed size. |
| ATM | Asynchronous Transfer Mode. ATM is a network technology based on transferring data in cells or packets of a fixed size. |
| Attribute | An attribute is a datum about a network session or a device session. Attributes contain a name and value and a distinguishing namespace.<br>In the BPDS Object Manager tool, a simple type with a default value. An object can have several attributes. |
| **B** | |
| Backend | Software that runs on the BPM. It comprises the controller, engine, agent host, activation daemon, and scheduler processes; synonymous with BPM. |
| BGP | Border Gateway Protocol. An exterior gateway routing protocol that enables groups of routers to share routing information to establish efficient, loop-free routes. BGP is commonly used within and between ISPs. |
| Border Gateway Protocol | BGP. An exterior gateway routing protocol that enables groups of routers to share routing information to establish efficient, loop-free routes. BGP is commonly used within and between ISPs. |
| BPDS | Broadband Policy Design Studio. The BPDS is a graphical user interface to the BPM. The BPDS includes a service design feature. |
| BPM | Broadband Policy Manager. The BPM is a product suite used by service providers to create and deploy advanced services on broadband networks. A BPM system can be configured as a Director, Domain Controller, Resource Controller, or Topology Database Server. |
| BPS | Broadband Policy Studio. The BPS is a graphical user interface, similar to the BPDS. The BPS does not include the service design feature. |

Table A-1. Terms and Definitions.

| Term | Definition |
|---|---|
| BRAS | Broadband Remote Access Server. A BRAS device routes traffic to and from the digital subscriber line access multiplexers on an ISP network. |
| Broadband Policy Design Studio | BPDS. The BPDS is a graphical user interface, similar to the BPS. The BPDS includes a service design feature. |
| Broadband Policy Manager | BPM. The BPM is a product suite used by service providers to create and deploy advanced services on broadband networks. |
| Broadband Policy Studio | BPS. The BPS is a graphical user interface, similar to the BPDS. The BPS does not include the service design feature. |
| Broadband Remote Access Server | BRAS. A BRAS device routes traffic to and from the digital subscriber line access multiplexers (DSLAM) on an ISP network. |
| **C** | |
| CAC | Capacity Admission Control. CAC monitors, controls, and enforces the use of network resources and services with policy-based management over broadband access and MPLS core networks. |
| Capacity Admission Control | CAC. CAC monitors, controls, and enforces the use of network resources and services with policy-based management over broadband access and MPLS core networks. |
| Cisco Network Registrar | CNR. The CNR is a full-featured DNS/DHCP system that provides scalable naming and addressing services for service provider and enterprise networks. |
| Class of Service | CoS. This is a traffic prioritization scheme that enables more predictable traffic delivery, based on application requirements. |
| Classless Inter-Domain Routing | CIDR. This IP addressing scheme addresses the size of routing tables and makes more IP addresses available within organizations. CIDR is also called supernetting. |
| CIDR | Classless Inter-Domain Routing. This IP addressing scheme addresses the size of routing tables and makes more IP addresses available within organizations. CIDR is also called supernetting. |
| Client | This is a generic term that denotes the BPM BPDS application. |
| CoS | Class of Service. This is a traffic prioritization scheme that enables more predictable traffic delivery, based on application requirements. |
| CPE | customer premises equipment. This is communications equipment that resides on the customer premises. It is owned or leased by the customer. |
| CLI | command line interface. This is a user interface common to computers. The user enters a command. The computer acts on the command. |
| Cluster | A pair of cooperating and redundant BPMs. |
| CNR | Cisco Network Registrar. The CNR is a full-featured DNS/DHCP system that provides scalable naming and addressing services for service provider and enterprise networks. |
| Command Line Interface | CLI. This is a user interface common to computers. The user enters a command. The computer acts on the command. |
| Component | An object comprising data and code. A component provides a well-specified set of publicly available services. All devices, services, and applications on a network are components. |

Table A-1. Terms and Definitions.

| Term | Definition |
|---|---|
| Configuration | Information necessary to construct an instance of a type (agent, service). |
| Controller | A software element that runs on the BPM and controls various elements of the backend. Usually only one controller exists per backend; therefore, from the BPDS perspective, the controller is the backend. |
| Customer Premises Equipment | CPE. This is communications equipment that resides on the customer premises. It is owned or leased by the customer. |
| **D** | |
| DAF | Device Adapter Function. A DAF translates between the protocol and device type-specific events of at the PIF layer and the abstract application events at the Application layer.  A DAF can be assigned to multiple device types and multiple DAFs can be assigned to one device. |
| Deep Packet Inspection Protocol | DPI. This is network packet filtering that examines packet *data*, searching for nonprotocol compliance or predefined criteria, to decide if the packet can pass. This is in contrast to shallow packet inspection (called packet inspection), which checks only the packet *header*. |
| Device | Any piece of software or hardware connected to a network. RADIUS servers, routers, billing systems, accounting systems, and video servers are devices. An agent communicates with a device. |
| Device Access | A device access is data about accessing a device instance. Most devices require authentication before any device action can occur. The device access contains this authentication data and other related data. Each device instance has one device access per management protocol. |
| Device Action | A device action is the implementation of an action for a given device type. That is, it is the actual set of instructions necessary to change the functioning of the device instance. |
| Device Adapter Function | DAF.A DAF translates between the protocol and device type-specific events of at the PIF layer and the abstract application events at the Application layer.  A DAF can be assigned to multiple device types and multiple DAFs can be assigned to one device. |
| Device Adapter Function Flow | A Flow that handles a protocol event for a specific device type. |
| Device Handler Dispatch Service | DHDS. DHDS provides routing services for PIFs and Applications requesting invocation of DAF operations. |
| Device Instance | A device instance is a device type in use in the network. For example, a Cisco 10K device at IP address 128.148.176.10. Device instances are grouped according to roles. |
| Device Rule | A device rule is a provisioned list of steps that apply a policy to a device. A device rule consists of a set of instructions that the BPM sends to the device to apply the given policy. Device rules can retrieve information from connected devices. Preconfigured device rules are useful for configuring a new BPM system. See also Device Type and Policy Rule. |

Table A-1. Terms and Definitions.

| Term | Definition |
|---|---|
| Device Session | A device session contains data about a device instance used by a network session. For example, information about the bras would be encoded in a device session. |
| Device Type | A device type is a vendor's network element hardware. Device types are grouped according to roles and are based on device attributes, such as vendor, model, hardware version, and software version. See also Device Rule. |
| DHDS | Device Handler Dispatch Service. DHDS provides routing services for PIFs and applications requesting invocation of DAF operations. |
| Digital Subscriber Line | DSL. DSL technologies use sophisticated modulation schemes to pack data onto copper wires. |
| Digital Subscriber Line Access Multiplexer | DSLAM. This mechanism links customer DSL connections to a single high-speed ATM line. |
| Director | A Director is one or more stateless installations that takes requests and routes them to appropriate Resource Controllers, to handle the specific incoming requests. |
| Director Realm | The Director Realm stores information required by Director systems, including information about network devices (such as BRAS devices). The information specifies the Resource Controller responsible for each device and the IP address pools each device handles. A Director uses this information to forward an incoming request to the correct Resource Controller. The Topology Database Server maintains the Director Realm, and the server distributes its updates to each Director when updates occur. |
| Domain | One or more cooperating Broadband Policy Managers (BPMs) managed by a single domain repository. |
| Domain Controller | The Domain Controller is a standalone system responsible for domain management, including application deployment, configuration, and health for all systems in the domain. Only one Domain Controller exists per domain. |
| Domain Data | Data maintained about the elements in a domain; for example, controller host and port configuration, database host and port information, agent and service configuration and deployment information. |
| Domain Realm | The Domain Realm maintains application level information about the physical network topology. The nodes in the topology represent Director and Resource Controller systems. The Topology Database Server uses the Domain Realm to understand the system topology. Links represent connectivity between cluster pairs. Resources represent interfaces on the component systems, system health, cluster information, and system configuration. |
| Domain Repository | The master database that contains configuration information for each domain element. |
| DPI | Deep Packet Inspection Protocol. This is network packet filtering that examines packet *data*, searching for nonprotocol compliance or predefined criteria, to decide if the packet can pass. This is in contrast to shallow packet inspection (called packet inspection), which checks only the packet *header*. |

Table A-1. Terms and Definitions.

| Term | Definition |
|---|---|
| DSL | Digital Subscriber Line. DSL technologies use sophisticated modulation schemes to pack data onto copper wires. |
| DSLAM | Digital Subscriber Line Access Multiplexer. This mechanism links customer DSL connections to a single high-speed ATM line. |
| **E** | |
| Element | An object with the BPM: package; agent configuration; service instance; shared object. |
| Enumeration | In the BPDS Object manager tool, enumeration is contained within a simple type. |
| Ethernet | The Ethernet is a large and diverse family of frame-based computer networking technologies for local area networks (LANs). It defines a number of wiring and signaling standards for the physical layer, two means of network access at the Media Access Control (MAC)/Data Link Layer, and a common addressing format. Ethernet has been standardized as IEEE 802.3. |
| ETSI | European Telecommunications Standards Institute. ETSI is an independent, non-profit organization, whose mission is to produce telecommunications standards for today and for the future. |
| European Telecommunications Standards Institute | ETSI. ETSI is an independent, non-profit organization, whose mission is to produce telecommunications standards for today and for the future. |
| **F** | |
| Field Replaceable Unit | FRU. An FRU represents an element (e.g., entire system, BPDS client software, agent) within the Broadband Policy Managers (BPM) that has a version associated with it. A FRU is a subset of an element. |
| Flow | The movement of data or control between agents. It is a collection of one or more operators and zero or more routes. The designer uses flows to define services and applications. |
| FRU | Field Replaceable Unit. An FRU represents an element (e.g., entire system, BPDS client software, agent) within the Broadband Policy Managers (BPM) that has a version associated with it. A FRU is a subset of an element. |
| Function | The element that performs an operation, based on inputs and returns the results of the operation via its outputs. The designer drags and drops a function into a flow in the BPDS. An agent, interacting with a device, performs the actual operation. |
| **G** | |
| Graphical User Interface | GUI. A program interface that takes advantage of the computer's graphics capabilities to make the program easier to use. For the BPM, the GUI is the BPDS. |
| GUI | Graphical User Interface. A program interface that takes advantage of the computer's graphics capabilities to make the program easier to use. For the BPM, the GUI is the BPDS. |
| **H** | |

**Table A-1. Terms and Definitions.**

| Term | Definition |
|---|---|
| Handler | A handler enables flow of control between the PIF, DAF, and SMF interfaces. It includes details about the appropriate service flow to call under specific conditions. |
| Handler Flow | A Handler Flow normalizes protocol-specific parameters before forwarding them to an application.  An application can indirectly invoke a Handler Flow using the DHDS. |
| Head Version | The latest version of an element. |
| Hypertext Preprocessor | PHP. PHP is an open source, server-side, HTML embedded scripting language used to create dynamic Web pages. |
| **I** | |
| Implementation | An instruction set for executing a specification. |
| Instance | An executing type (agent, service), created from a specification, implementation, and configuration. An agent instance is a specific implementation of that agent type. |
| Interface | A collection of functions. |
| Internet Service Provider | ISP. An ISP is a company that provides access to the Internet. For a monthly fee, the company provides a software package, username, password and access phone number. In addition to serving individuals, ISPs also serve large companies, providing a direct connection from the company network to the Internet. |
| IP address | The address that identifies a computer. The IP address format is a 32-bit numeric address written as four numbers (0 to 255) separated by periods. |
| ISP | Internet Service Provider. An ISP is a company that provides access to the Internet. For a monthly fee, the company provides a software package, username, password and access phone number. In addition to serving individuals, ISPs also serve large companies, providing a direct connection from the company network to the Internet. |
| **J** | |
| **K** | |
| Key | A key is an identifier used in conjunction with network sessions. |
| **L** | |
| LAN | Local Area Network. A LAN is computer network that spans a relatively small area. Most LANs are confined to a single building or group of buildings. A LAN connect workstations and personal computers. This allows users to share devices and data and communicate via email. |
| L2TP | Layer Two Tunneling Protocol. L2TP is an extension to the PPP protocol that enables ISPs to operate VPNs. |
| Layer Two Tunneling Protocol | L2TP. L2TP is an extension to the PPP protocol that enables ISPs to operate VPNs. |
| Link | A link is a line or channel over which data is transmitted. |

**Table A-1. Terms and Definitions.**

| Term | Definition |
|---|---|
| Local Area Network | LAN. A LAN is computer network that spans a relatively small area. Most LANs are confined to a single building or group of buildings. A LAN connect workstations and personal computers. This allows users to share devices and data and communicate via email. |
| **M** | |
| Management Protocol | A management protocol is the mechanism for managing a network element. Common management protocols are RADIUS and SNMP. |
| Metadata | In the BPDS, this is the data structure. A customer can import metadata to invoke a structure for his or her database. |
| MPLS | Multiprotocol Label Switching. MPLS integrates Layer 2 network link information into Layer 3 within an autonomous system or ISP. It improves IP-packet exchange and allows operators to divert and route traffic around link failures, congestion, and bottlenecks. |
| Multiprotocol Label Switching | MPLS. MPLS integrates Layer 2 network link information into Layer 3 within an autonomous system or ISP. It improves IP-packet exchange and allows operators to divert and route traffic around link failures, congestion, and bottlenecks. |
| **N** | |
| N + 1 Redundancy | The ability for service engines to use one service engine as a backup. |
| NAF | Network Adaptation Function. NAF. The NAF dynamically resizes network links and queue sizes, based on the ability of the underlying network to adapt after a request from the ACF. |
| Namespace | A namespace helps distinguish two or more values that otherwise would conflict with each other. |
| NAS | Network Attached Storage. A NAS device is a server dedicated to file sharing, allowing more hard disk storage space to be added to a network that already utilizes servers without shutting them down for maintenance and upgrades. A NAS device can exist anywhere in a LAN and can be made up of multiple networked NAS devices. |
| NAV | Network Admin view. In the BPS and BPDS graphical user interfaces to the BPM, this is the network view where you can perform administration tasks. |
| Network | A network is a group of two or more computer systems linked together. Local-area networks (LANs), wide-area networks (WANs), and metropolitan-area networks MANs are typical networks. |
| Network Adaptation Function | NAF. The NA) dynamically resizes network links and queue sizes, based on the ability of the underlying network to adapt after a request from the ACF. |
| Network Admin View | NAV. In the BPS and BPDS graphical user interfaces to the BPM, this is the network view where you can perform administration tasks. |
| Network Attached Storage | NAS. A NAS device is a server dedicated to file sharing, allowing more hard disk storage space to be added to a network that already utilizes servers without shutting them down for maintenance and upgrades. A NAS device can exist anywhere in a LAN and can be made up of multiple networked NAS devices. |
| Network Event | A network event is a set of install and uninstall rules, contained within a profile, that are performed in sequence. |

Table A-1. Terms and Definitions.

| Term | Definition |
|---|---|
| Network Manager | NM. The NM product provides a framework for controlling and querying the element configurations in the broadband network. |
| Network Policy | A network policy is a device rule entry. The device rule contains commands to configure a network device to apply a network policy. See also Device Rule, Policy Rule. |
| Network Realm | The Network Realm stores specific network adaptation information, such as the devices active on a particular Resource Controller, profiles, and handlers. The Network Realm is centrally provisioned on the Topology Database Server, and it is distributed to all Resource Controllers. |
| Network Session | A network session represents a single point-to-point connection in the network, for example, a VoIP call. |
| Network Storage Function | NSF. The Network Storage Function provides access to the Network Information Model. |
| NM | Network Manager. The NM product provides a framework for controlling and querying the element configurations in the broadband network. |
| Node | In networks, a processing location. A node can be a computer or some other device, such as a printer. Every node has a unique network address, sometimes called a Data Link Control (DLC) address or Media Access Control (MAC) address. |
| NSF | Network Storage Function. The NSF provides access to the Network Information Model. |
| **O** | |
| Object | An agent, controller, function, service, switch, or service within the Broadband Policy Manager (BPM). |
| Object Dependency | An exact object type, for example a Cisco 2500 router agent, that a service depends on. The service designer adds the object type to the dependency list of the service. All Interfaces supported by the object type are then available for use with the service. |
| Object Type | In the BPDS, an object type is defined with attributes. It can own contain, and associate with other object types. |
| OC | Orchestration Controller. That portion of the Broadband Policy Managers (BPM) that controls processes such as username and password authentication. |
| Operation and Support System | OSS. OSS refers to a suite of programs that enable an enterprise to monitor, analyze, and manage a network system. The term originally referred to a management system that controlled telephone and computer networks. It now applies to the business world to mean a system that supports network operations. |
| Operator | A representation of actions to be undertaken on a system networked to a Broadband Policy Managers (BPM). |
| Orchestration Controller | OC. That portion of the Broadband Policy Managers (BPM) that controls processes such as username and password authentication. |
| Orchestration Network | The process for handling service calls over a network. It defines the flow of control and information between work units. |

**Table A-1. Terms and Definitions.**

| Term | Definition |
|---|---|
| OSS | Operation and Support Systems. OSS refers to a suite of programs that enable an enterprise to monitor, analyze, and manage a network system. |
| **P** | |
| Pad | A collection of pins on an operator. This appears as a box along the edge of an operator. |
| Path Computation Function | PCF. The PCF determines the path through the topology for any given end-to-end session, as requested by the ACF. |
| PCF | Path Computation Function. The PCF determines the path through the topology for any given end-to-end session, as requested by the ACF. |
| PDP | Policy Decision Point. The PDP is a component of policy-based management. When a user tries to access a file or other resource on a system using policy-based access management, the PDP decides whether or not to authorize the user based on user attributes. |
| PE | Policy Engine. The software that stores and manages user profile information, subscriber access records, policy rules; also known as the policy database. |
| PEP | Policy Enforcement Point. The PEP is the logical entity or place on a server that makes admission control and policy decisions in response to a request from a user wanting to access a resource on a computer or network server. |
| PHP | Hypertext Preprocessor (PHP). PHP is an open source, server-side, HTML embedded scripting language used to create dynamic Web pages. |
| PIF | Protocol Interface Function. A PIF service encapsulates an interface with an external device or service |
| PIF Agent | An Agent that acts an adaptor between the system and an external device or service. |
| Pin | An input or output from an operator. The pin serves as a route endpoint and holds a single input or output value. For example, an operator that needs a username and password as input has two input pins; one for the username; the other, the password. |
| PMF | Profile Management Function. The Profile Management Function (PMF) activates and deactivates network profiles on subscriber sessions. |
| Point-to-Point Protocol Over ATM | PPPoA. PPPoA relies on two widely accepted standards: PPP and ATM. It is an end-to-end asymmetric digital subscriber line (ADSL) architecture. |
| Point-to-Point Termination Aggregation | PTA. This is a method of aggregating IP traffic by terminating PPP sessions and amassing the IP traffic into a single routing domain. |
| Policy | A flow comprising a rule or set of rules that take a specific action provided by an ISP for its subscribers. For example, a policy for subscriber access directs how the system identifies a subscriber via user id, access type, and log in location. A policy performs an operation, based on input and returns the results of its action as output. |

**Table A-1. Terms and Definitions.**

| Term | Definition |
|---|---|
| Policy Database | The database of policy objects that services access to make policy decisions. |
| Policy Decision Point | PDP. The PDP is a component of policy-based management. When a user tries to access a file or other resource on a system using policy-based access management, the PDP decides whether or not to authorize the user based on user attributes. |
| Policy Enforcement Point | PEP. The PEP is the logical entity or place on a server that makes admission control and policy decisions in response to a request from a user wanting to access a resource on a computer or network server. |
| Policy Engine | PE. The software that stores and manages user profile information, subscriber access records, policy rules; also known as the policy database. |
| Policy Function | Policy rules encapsulated in a TCL agent *execute* function. |
| Policy Repository | The Policy Repository BPM stores all persistent data associated with customers and services. It utilizes industry-standard database technology that allows any of the underlying system elements to interrogate it. |
| Pool | A pool represents a range of IP addresses. A BRAS handles one or more address ranges. A Resource Controller potentially handles multiple BRASs. So a typical Resource Controller can handle multiple ranges of IP addresses (multiple pools). |
| PPPoA | Point-to-Point Protocol Over Asynchronous Transfer Mode. PPPoA relies on two widely accepted standards: PPP and ATM. It is an end-to-end asymmetric digital subscriber line (ADSL) architecture. |
| Presence Director | The Presence Director is an optional, modified, Director service that handles receives session requests and distributes them to the appropriate Resource Controllers. |
| Profile | A profile is a procedure for changing a set of related network elements for a given purpose, for example, increasing the bandwidth associated with a network session. |
| Profile Management Function | PMF. The Profile Management Function (PMF) activates and deactivates network profiles on subscriber sessions. |
| Property | The parameter or characteristic of an agent or device. |
| Protocol Interface Function | PIF. A PIF service encapsulates an interface with an external device or service. |
| PTA | Point-to-Point Termination Aggregation. This is a method of aggregating IP traffic by terminating PPP sessions and amassing the IP traffic into a single routing domain. |
| **Q** | |
| QoS | Quality of Service. QoS specifies a guaranteed throughput level that allows providers to guarantee to their customers that end-to-end latency will not exceed a specified level. |
| Quality of Service | QoS. QoS specifies a guaranteed throughput level that allows service providers to guarantee to their customers that end-to-end latency will not exceed a specified level. |

**Table A-1. Terms and Definitions.**

| Term | Definition |
|---|---|
| **R** | |
| RACS | Resource and Admission Control Subsystem. RACS consists of the Policy Decision Function (PDF) and Access-RAC Function (A-RACF), which controls QoS within the access network. |
| RADIUS | Remote Authentication Dial-In User Service. RADIUS is a client/server protocol enabling remote access server communication with a central server to authenticate dial-in users and authorize their access to the requested system or service. RADIUS allows a company to maintain user profiles in a central database and set up a policy that can be applied at a single administered network point. |
| Realm | A realm represents a collection of information, stored in the database, that should be transferred, as a unit, between BPM systems. The realm defines a unit for intersystem communication and improves performance by restricting lookups and updates against smaller data sets. |
| Remote Authentication Dial-in User Service | RADIUS. RADIUS is a client/server protocol enabling remote access server communication with a central server to authenticate dial-in users and authorize their access to the requested system or service. RADIUS allows a company to maintain user profiles in a central database and set up a policy that can be applied at a single administered network point. |
| Remote Method Invocation | RMI. RMI is the basis of distributed object computing in the Java environment. It defines how Java components can interoperate in a Java environment. |
| Resource | A resource is any device or other item that can be used. Devices such as printers and disk drives are resources. Memory is also a resource. In many operating systems, a resource is specifically data or routines that are available to programs. These are also called system resources. |
| Resource and Admission Control Subsystem | RACS. RACS consists of the Policy Decision Function (PDF) and Access-RAC Function (A-RACF), which controls QoS within the access network. |
| Resource Controller | A Resource Controller is a stateful installation that tracks resource utilization for the system. |
| Resource Realm | A Resource Realm represents a BRAS device and its connected CPE equipment. The Resource Realm is provisioned on the Topology Database Server and distributed to the Resource Controller that coordinates activity for that BRAs. At runtime, the Resource Realm stores capacity and usage information required to perform CAC decisions. |
| RMI | Remote Method Invocation. RMI is the basis of distributed object computing in the Java environment. It defines how Java components can interoperate in a Java environment. |
| Role | A role is as a functional category for device types and device instances. For example, *bras* and *dpi* are roles. |

Table A-1. Terms and Definitions.

| Term | Definition |
|---|---|
| Role-based Dependency | A dependency in which a service designer indicates that multiple service elements support the same interface. The designer defines different roles and assigns the required service interfaces to each. The different roles are added to the dependency list for the service and operators are clearly marked to indicate their assigned role. |
| Route | A route is a path between operators. |
| Rule | Criteria applied to the objects and methods of a business system to determine how objects and methods are used by, or for, a given system subscriber. A flow comprises a rule or set of rules. Rules prescribe terms and conditions for a specific action provided by an ISP for its subscribers. One rule can call another rule. |
| **S** | |
| S-VLAN | Stacked VLAN. An S-VLAN provides a two-level S-VLAN tag structure that extends the VLAN ID space to more than 16 million VLANs. |
| SAV | Service Admin view. In the BPS and BPDS graphical user interfaces to the BPM, this is the view where you can perform service tasks. |
| Schema | A set of rules and syntax for storing data. |
| SDV | Service Design view. In the BPS and BPDS graphical user interfaces to the BPM, this is the view where you can design services. |
| SE | Service Engine. SE is an unassigned and unconfigured system. It is also known as the backend. |
| Service | An application, created by the BPM designer, that maps business models and operational procedures directly into IP services, executable by their customers, for example, video on demand or automatic backup. A service comprises objects (agent, controller, function, switch, or other service) and can comprise one or more flows. |
| Service Admin View | SAV. In the BPS and BPDS graphical user interfaces to the BPM, this is the view where you can perform service tasks. |
| Service Configuration | The information needed to construct a service. The service configuration specifies agent configurations for each function in the service type. The BPM designer creates the service configuration. |
| Service Dependency | The dependencies of a service, created by the service designer. The designer builds a service by defining data-flows that use operators from multiple objects, including agents and other services. The designer builds a service upon a concrete set of agents and services.<br>If a service is portable across different agents and services, the designer specifies any constraints on the concrete instances and specifies the interfaces that those concrete instances must support. |
| Service Design View | SDV. In the BPS and BPDS graphical user interfaces to the BPM, this is the view where you can design services. |
| Service Engine | SE. SE is the generic term for an unassigned and unconfigured system. it is also known as the backend. |
| Service Interface Dependency | If a service uses a particular service interface, but does not require that a specific object provide the service interface, the service designer can add the service interface as a dependency. Here, the service interface operators are available for use in the current service, but the object that provides the interface is determined later. |

Table A-1. Terms and Definitions.

| Term | Definition |
|---|---|
| Service Instance | The running of a service type created by the subscriber. |
| Service Level Agreement | SLA. An SLA is a contract between an ASP and the end user that stipulates the required level of service and its fee. |
| Service Palette | The agent types available to a service. |
| Service Profile | A collection of services and information about service execution. |
| Service Provider | SP. This is the provider of Internet connectivity services. |
| Service Type | The definition of what agent types are required for a service; the defined flow of data between functions of agent types. The service designer creates the service type. |
| Servlet | An applet that runs on a server. Usually refers to a Java applet that runs within a Web server environment. Analogous to a Java applet that runs within a Web browser environment. |
| Session Management Application | SMA. Within the Session Manager, the SMA encapsulates customer-specific business logic for managing network sessions. |
| Session Management Function | SMF. The SMF encapsulates customer-specific business logic applied to network sessions. Abstracted from specific protocols and devices used in the network through the DAF and PIF layers, the SMF notifies applications of session state changes. |
| Session Manager | SM. The SM provides a framework for tracking user sessions connecting to the network. |
| Session Realm | A Session Realm stores Session Manager contexts and assists in the decision-making process during network adaptation. |
| Session Storage Function | SSF. The SSF provides access to the Session Information Model. |
| SF | Statistics Function. The SF records and queries system statistics and provides a location for various components to store runtime state statistics. |
| Shared Secret | An authentication string that ensures security between devices. KERBEROS is an instance of a shared-secret authentication protocol. |
| SIF | Signaling Interface Function (SIF): The SIF sends QoS requests from an application to the Director ACF. If more than one Director exists, an external Load Balancer selects a Director. The SIF receives replies from Director ACFs and forwards them to the application. |
| Signaling Interface Function | SIF. The SIF sends QoS requests an application to the Director ACF. If more than one Director exists, an external Load Balancer selects a Director. The SIF receives replies from Director ACFs and forwards them to the application. |
| Simple Object Access Protocol | SOAP. This is a lightweight XML-based messaging protocol that encodes the information in Web service request and response messages before sending them over a network. SOAP messages are independent of any operating system or protocol and may be transported using a variety of Internet protocols, including SMTP, MIME, and HTTP. |
| Simple Type | In the BPDS Object manager tool, a simple type is similar to data type, except it can express with enumerations. |

**Table A-1. Terms and Definitions.**

| Term | Definition |
|---|---|
| Simple Network Management Protocol | SNMP. A protocol by which networked devices are periodically polled for information as part of a network management system. |
| SLA | Service Level Agreement. An SLA is a contract between an ASP and the end user that stipulates a required level of service and its fee. |
| SM | Session Manager. The SM provides a framework for tracking user sessions connecting to the network. |
| SMA | Session Management Application. Within the Session Manager, the SMA encapsulates customer-specific business logic for managing network sessions. |
| SMF | Session Management Function. The SMF encapsulates customer-specific business logic applied to network sessions. Abstracted from specific protocols and devices used in the network through the DAF and PIF layers, the SMF notifies applications of session state changes. |
| SNMP | Simple Network Management Protocol. A protocol by which networked devices are periodically polled for information as part of a network management system. |
| SOAP | Simple Object Access Protocol. This is a lightweight XML-based messaging protocol that encodes the information in Web service request and response messages before sending them over a network. SOAP messages are independent of any operating system or protocol and may be transported using a variety of Internet protocols, including SMTP, MIME, and HTTP. |
| SP | Service Provider. This is the provider of Internet connectivity services. |
| Specification | A type definition that includes interface definitions, configuration schemas, and binding information. |
| SQL | Structured Query Language. SQL is a standardized query language for requesting information from a database. SQL enables several users on a local-area network to access the same database simultaneously. |
| SSF | Session Storage Function. The SSF provides access to the Session Information Model. |
| Stacked VLAN | S-VLAN. An S-VLAN provides a two-level S-VLAN tag structure that extends the VLAN ID space to more than 16 million VLANs. |
| Standby BPM | In a pair of BPMs, the standby BPM constantly monitors the health of the active BPM to assess its ability to process requests. If the active BPM is not viable, the standby BPM becomes the active. |
| Statistics Function | SF. The SF records and queries system statistics and provides a location for various components to store runtime state statistics. |
| Status | A status is a condition used in conjunction with network sessions. |
| Structured Query Language | SQL. SQL is a standardized query language for requesting information from a database. SQL enables several users on a local-area network to access the same database simultaneously. |
| Subscriber | A customer of a service provider. The service provider delivers a variety of online services, including e-mail, stock quotes, news, and online forums. |
| Subscriber Profile | A table entry containing information, such as authentication, authorization, and location on a specific subscriber. |

Table A-1. Terms and Definitions.

| Term | Definition |
|---|---|
| Super Operator | A reusable flow that other flows can call. To the other flows, the super operator appears as an operator that they can call and insert on any route. |
| Switch | A device that filters and forwards packets between LAN segments. Switches operate at the data link layer and the network layer of the OSI Reference Model. |
| Super User | The term denotes the highest level of user privilege. It allows unlimited access to a system. Usually, super user is the highest level of privilege for applications, as opposed to operating or network systems. |
| **T** | |
| TAF | Topology Awareness Function. The TAF extracts and reacts to changes in the underlying network. The information can be read from provisioning files or received from the TDS. |
| TISPAN | Telecommunications and Internet Services and Protocol for Advanced Networking. TISPAN is the ETSI core competence center for fixed networks and for migration from switched circuit networks to packet-based networks with an architecture that can serve in both. TISPAN is responsible for all aspects of standardization for present and future converged networks. |
| Telecommunications and Internet Services and Protocol for Advanced Networking | TISPAN. TISPAN is the ETSI core competence center for fixed networks and for migration from switched circuit networks to packet-based networks with an architecture that can serve in both. TISPAN is responsible for all aspects of standardization for present and future converged networks. |
| Topology Awareness Function | TAF. The TAF extracts and reacts to changes in the underlying network. The information can be read from provisioning files or received from the TDS. |
| Topology Database Server | In resilient pairs, Topology Database Servers maintain the global topology database for the system as a whole. The Director detects delayed response times or dropped requests and notifies the Topology Database Server. The Topology Database Server initiates Resource Controller failover when necessary. |
| Topology Store Function | TSF. The TSF maintains the TIM for a given BP Resource Controller system component. |
| Transaction Remote Procedure Call | TRPC. The TRPC protocol is the interface between Cisco BPM components. |
| TRPC | Transaction Remote Procedure Call.The TRPC protocol is the interface between Cisco BPM components. |
| TSF | Topology Store Function. The TSF maintains the TIM for a given BP Resource Controller system component. |
| Type | A BPM component group that has a unique specification. It may have an implementation, and it may have one or more configurations and instances. |
| **U** | |
| **V** | |

Table A-1. Terms and Definitions.

| Term | Definition |
|---|---|
| VC | Virtual Circuit. A connection between two devices that acts as though it's a direct connection even though it may physically be circuitous. |
| Virtual Circuit | VC. A VC is a connection between two devices that acts as though it's a direct connection even though it may physically be circuitous. |
| Virtual LAN | VLAN. A network of computers that behave as if connected to the same wire even though they can be physically located on different segments of a LAN. VLANs are configured through software rather than hardware and extremely flexible. |
| Virtual Path | VP.  A VC is a set of links across an ATM network between two specified end points. |
| Virtual Private Network | VPN. A VPN is constructed using public wires to connect nodes. A number of systems exist that enable the creation of networks using the Internet as the medium for transporting data. They use security mechanisms to ensure that only authorized users can access the network and data cannot be intercepted. |
| VLAN | Virtual LAN. A network of computers that behave as if they are connected to the same wire even though they may be physically located on different segments of a LAN. VLANs are configured through software rather than hardware and are extremely flexible. |
| Voice-over-IP | VoIP. Voice delivered using the Internet Protocol. |
| VoIP | Voice-over-IP. Voice delivered using the Internet Protocol. |
| VP | Virtual Path. VP.   A VP is a set of link across an ATM network between two specified end points. |
| VPN | Virtual Private Network. A VPN is constructed using public wires to connect nodes. A number of systems exist that enable the creation of networks using the Internet as the medium for transporting data. They use security mechanisms to ensure that only authorized users can access the network and data cannot be intercepted. |
| **W** | |
| WDSL | Wireless Digital Subscriber Line. WDSL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. It is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate. |
| Wireless Digital Subscriber Line | WDSL. WDSL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. It is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate. |
| Workspace | The BPDS area where the designer visually programs services. |

# Appendix B - Creating and Provisioning R-Realms

## Overview

The Topology Database Server Broadband Policy Manager (BPM) provisions all topology. It automatically creates and modifies topology data, as it receives events from operations on Director BPMs and Resource Controller BPMs, and it automatically distributes updates to all target systems. You manually provision other topology data, for example, the Resource Realm and IP Pool data, onto the Topology Database Server using the tash **Provision** command on a file that defines topology changes. The Topology Database Server also automatically distributes these changes to the component systems.

## Realms

Topology, resources, active sessions, and active contexts exist in an information *realm*. The realm improves performance by restricting lookups and updates against smaller data sets, providing less lock contention and faster search times. It also allows a Resource Controller to comprehend the realms for which it is responsible. If a request involves a realm that the Resource Controller does not own, it can ignore the request (if that is its configured behavior). When a new Resource Controller is introduced, BRAS responsibility migrates from one Resource Controller to a new Resource Controller. The realm concept allows the information model to consider this a block movement of a realm. The migration affects only the realm that is moving. When scaling the BPM to accommodate more hardware and repartitioning the realms, only the realm/BRAS being moved is unavailable to quality of service (QoS) requests. Calls originating or terminating in other realms remain uninterrupted.

A BRAS defines a realm of self-contained information. The mapping of Resource Controllers to BRASs allows a single Resource Controller to handle multiple BRASs and their access. The state maintenance of various components is specified at the granularity of the Resource Controller. Thus, a single Topology Store Function (TSF) element handles more than one BRAS.

## Topology Information File

The topology information file allows topology additions, deletions, and updates of nodes, links, and resources. The file format includes the <TIM> record, and information about Domain Realms, Director Realms, and Resource Controller Realms.

### <TIM> Record Definition

The topology information file contains one or more <TIM> records, whose attributes define how the content of that record should be processed. Each <TIM> directive has a form similar to this.

```
<tim realms='<realm_svc>' resources='<resource_svc>'
   nodes='<nodes_svc>' links='<links_svc>' action='<action>' >
```

where:

`realm_svc` is any of the following:

- `taf_realms`: Applies realm changes only to the local Topology Database Server system

- `tds_manage_realms`: Applies realm changes to the Topology Database Server, propagates changes to the impacted Resource Controller, and distributes changes to all Director systems

Two forms exist for each of the other attribute values (`resource_svc`, `node_svc` and `link_svc`). They have similar names and impacts:

- `taf_nodes`, `taf_links`, and `taf_resources` update resource/node/link information only on the Topology Database Server.
- `tds_taf_rc_nodes`, `tds_taf_rc_links`, `tds_taf_rc_resources` distributes changes to the impacted Resource Controller, as well as all Directors, and applies them to the local Topology Database Server topology.

`action` defines what to do with the TIM data, as follows:

- `add`: Adds the TIM information to the topology. Data that exists in the topology generates errors. However, processing of the file continues if an error occurs.
- `remove`: Removes the TIM information from the topology.
- `update`: Updates the TIM information in the existing topology.
- ‘’ (empty string): Takes no action.

**Note:** *Do not set the TIM action to* `''`.

- <not set>: If the TIM directive does not specify the action, then it is set on the command line to the tash **Provision** command that loads the file.

Within a <TIM> record, there is one or more realm definition records, for the three types of realms to define.

## Domain Realm Topology Definition

Figure 11 shows a sample Domain Realm.

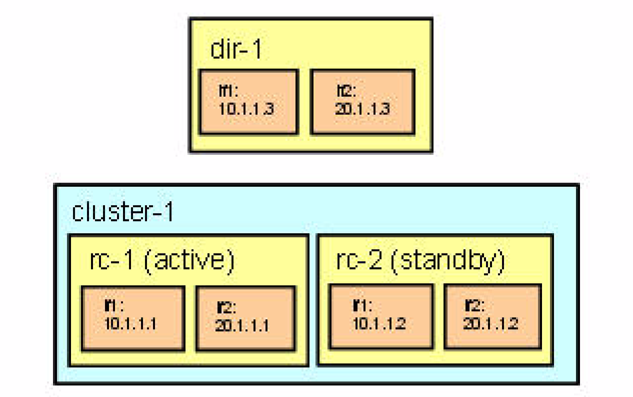


**Figure 9. Sample Domain Realm**

A Domain Realm record contains directives that define the domain structure. The following is a sample Domain Realm directive:

```
<realm id='<id>' type=‘tsf:domain’ action='<action>'>
```

where:

- id: The unique ID of the realm.
- action:
  - add: Adds the realm to the topology. Data that exists in the topology generates errors. However, processing of the file continues if an error occurs.
  - remove: Removes the realm from the topology.
  - update: Updates the realm in the topology.
  - ‘’ (empty string): Takes no action for the realm itself. Component resources of the realm can be added/removed/updated, as specified by the content of this directive.
  - <Unspecified>: If action is not specified, the default action, as specified in the <TIM> directive, or in the tash **Provision** command used to load this file, is taken.

Within a Domain Realm record, the following subrecords are allowed:

<resource>: This defines a resource of the domain, used to represent the network interfaces, system health, clusters, and configurations of each system. The <resource> directive appears as follows:

```
<resource id=‘<id>‘ type=‘<type>‘ action='<action>'>
```

where:

- id: ID of the resource within the realm. This string is used elsewhere in the topology to refer to this resource.
- type: One of the following values:
  - interface (In this case, id should represent an IP address)
  - health (In this case, id should be <IP-addr>-health, for example, 10.1.1.2-health)
  - cluster (In this case, id should be a unique ID for the cluster, for example, cluster-1)
  - configuration(Inthiscase,idshouldbe<IP-addr>:<port>-configuration, for example, 10.1.1.2:10000-configuration)
- action:
  - add: Adds the realm to the topology. Data that exists in the topology generates errors. However, processing of the file continues if an error occurs.
  - remove: Removes the realm from the topology.
  - update: Updates the realm in the topology.
  - ‘’ (empty string): Takes no action for the realm itself. Component resources of the realm can be added/removed/updated, as specified by the content of this directive.
  - <Unspecified>: If action is not specified, the default action, as specified in the <TIM> directive, or in the tash **Provision** command used to load this file, is taken.

Within a resource record, the following subrecords are allowed:

- `<property>`: Represents a property of a resource. The `<property>` directive appears as follows:

  `<property name='<name>' value='<value>' action='<action>'>`

  The name/value pairs for a Domain Realm property are as in Table 4. The directive `action` has a similar meaning as for resources.

**Table 4. Domain Realm Properties**

| Resource Type | Property Name | Meaning | Property Value |
|---|---|---|---|
| Interface | name | The name of the interface. | Interface name, such as `ip1`. |
| Interface | address | The IP address of the interface. | IP address, such as `192.9.200.50`. |
| Interface | subnet | The subnet mask of the interface. | Subnet mask, such as `255.255.255.0`. |
| Interface | state | The state of the interface. | `1` for up and `0` for down. |
| Interface | nodekey | The IP/port combination for the interface. | String of the form `<IP>:<port>`, for example `192.9.200.15:10000`. |
| Health | health | The health of the system. | `0` for healthy or `-1` for unhealthy. |
| Cluster | Type | The type of cluster. | `hot` or `warm`. |
| Cluster | Name | The name of the cluster. This allows persistence of a cluster beyond the actual lifetime of a BPM cluster (which dissolves on failover). | String uniquely identifying the cluster. |
| Cluster | Active | The IP address and port of the active cluster member. | String of the form `<IP>:<port>`, for example `192.9.200.15:10000`. |
| Cluster | standby | The IP address and port of the standby cluster member. | String of the form `<IP>:<port>`, for example `192.9.200.15:10000`. |
| configuration | configuration | A propertyListType that allows configuring a system over a Vista connection. | List, for example, `{name rc-1}`. |

- `<node>`: Identifies a node within the topology. In a Domain Realm, the nodes represent the component BPM systems of the overall installation. The `<node>` directive appears as follows:

  `<node id='<id>' active='<flag>' action='<action>'>`

  where:

  - `id`: The ID of a given node: a string of the form `<IP>:<port>` that represents the IP address and connection port for the given BPM system.
  - `active`: A string that indicates if the given node is a member of the BPM domain. This should always be true.
  - The directive `action` has a similar meaning as for resources above.

Within a node record, the following subrecords are allowed:

- `<resourceid>`: Represents a resource associated with the node. The `<resourceid>` directive appears as follows:

  ```
  <resourceid id='<id>' type='<type>' action='<action>'>
  ```

  The type/ID pairs for a Domain Realm resourceid are as in Table 5. The directive `action` has a similar meaning as for resources above.

**Table 5. Type/ID Pairs for Domain Realm**

| Type | ID | Meaning |
|---|---|---|
| Role | The string `aracf` or `spdf`. | The role that the system plays in the installation. |
| configuration | An existing ID of a resource of type configuration, in the domain. Default configurations include `aracf` and `spdf`. | Defines a system configuration so the Topology Database Server can automatically configure it over a Vista interface. |
| Cluster | An existing ID of a resource of type cluster, in the domain. | Indicates the cluster of which this node is a member. |
| interface | An existing ID of a resource of type interface, in the domain. | An interface connected to the system defined by this node. |
| Health | The subnet mask of the interface. | a subnet mask, such as `255.255.255.0`. |

- `<link>`: Identifies a link within the topology. In a Domain Realm, the links represent the connectivity between cluster members. The `<link>` directive appears as follows:

  ```
  <link id='<id>' src='<source>' dst='<dest>' active='<flag>'
     action='<action>'>
  ```

  where:

  - `id`: The ID of a given link. Within a Domain Realm, this is the ID of a cluster resource.
  - `source`: The source of a link. In a Domain Realm, this is the active node of the cluster (IP and port), such as `192.9.200.10:10000`.
  - `dest`: The destination of a link. In a Domain Realm, this is the standby node of the cluster (IP and port), such as `192.9.200.12:10000`.
  - `active`: A string that indicates if the given cluster is active. The possible values are `true` or `false`.
  - The directive `action` has a similar meaning as for resources above.

## Director Realm Topology Definition

Figure 11 shows a sample Director Realm.

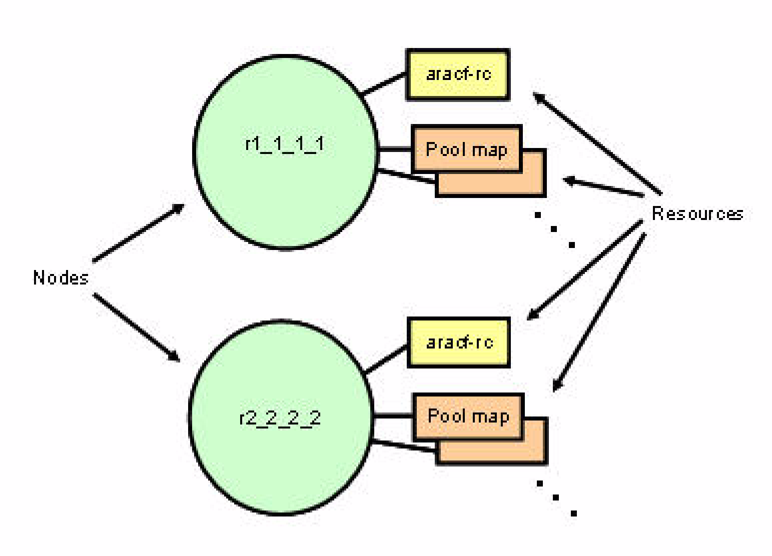


**Figure 10. Sample Director Realm**

A Director Realm record contains directives that define the Director structure. The following is a sample Director Realm directive:

```
<realm id='<id>' type='tsf:director' action='<action>'>
```

where:

- `id`: The unique ID of the realm.
- `action`:
  - `add`: Adds the realm to the topology. Data that exists in the topology generates errors. However, processing of the file continues if an error occurs.
  - `remove`: Removes the realm from the topology.
  - `update`: Updates the realm in the topology.
  - ‘’ (empty string): Takes no action for the realm itself. Component resources of the realm can be added/removed/updated, as specified by the content of this directive.
  - <Unspecified>: If `action` is not specified, the default action, as specified in the <TIM> directive, or in the tash **Provision** command used to load this file, is taken.

Within a Director Realm record, the following subrecords are allowed:

- `<resource>`: This defines a resource of the director, used to represent the Resource Controller Realms of the installation, and the IP pools of the installation. The `<resource>` directive appears as follows:

  ```
  <resource id='<id>' type='<type>' action='<action>'>
  ```

  where:

  - `id`: the unique ID of the realm.

- type: One of the following values:
  - aracf ( In this case, id should be an identifier for a Resource Controller, for example, r192_9_200_17-RC)
  - pool (In this case, id should be an identifier for an IP pool, for example, 192.9.200.0/24/CUSTREALM)
- action:
  - add: Adds the data to the topology. Data that exists in the topology generates errors. However, processing of the file continues if an error occurs.
  - remove: Removes the data from the topology.
  - update: Updates the data in the topology.
  - ‘’ (empty string): Takes no action for the data itself. Component resources can be added/removed/updated, as specified by the content of this directive.
  - <Unspecified>: If action is not specified, the default action, as specified in the <TIM> directive, or in the tash **Provision** command used to load this file, is taken.

Within a resource record, the following subrecords are allowed:

- <property>: Represents a property of a resource. The <property> directive appears as follows:

  <property name='*<name>*' value='*<value>*' action='*<action>*'>

  Table 6 shows name/value pairs for a Domain Realm property. The directive action has a similar meaning as for resources above.

**Table 6. Name/Value Pairs for a Domain Realm Property**

| Resource Type | Property Name | Meaning | Property Value |
|---|---|---|---|
| Aracf | health | The health of the active member of the cluster (or standalone if unclustered). | 0 for healthy, -1 for unhealthy. |
| aracf | host | The IP address of the active member of the cluster (or standalone if unclustered). | IP address, such as 192.9.200.50. |
| Aracf | port | The port for connectivity to the active member of the cluster (or standalone if unclustered). | Base port plus 100, such as 10100. |
| Aracf | qos | Service to handle QoS requests on the active member of the cluster (or standalone if unclustered). | Generally, aracf_sif. |
| Aracf | saf | Service to handle the session awareness function for the active member of the cluster (or standalone if unclustered). | Generally, aracf_saf. |
| Aracf | fof | Service to handle failover requests on the active member of the cluster (or standalone if unclustered). | Generally, aracf_fof. |
| Aracf | _health | The health of the standby member of the cluster. | 0 for healthy, -1 for unhealthy. |
| Aracf | _host | The IP address of the standby member of the cluster. | IP address such as 192.9.200.50. |
| Aracf | _port | The port for connectivity to the standby member of the cluster. | Base port plus 100, such as 10100. |
| Aracf | _qos | Service to handle QoS requests on the standby member of the cluster. | Generally, aracf_sif. |

**Table 6. Name/Value Pairs for a Domain Realm Property**

| Resource Type | Property Name | Meaning | Property Value |
|---|---|---|---|
| Aracf | _saf | Service to handle the session awareness function for the standby member of the cluster. | Generally, `aracf_saf`. |
| Aracf | _fof | Service to handle failover requests on the standby member of the cluster. | Generally, `aracf_fof`. |
| Pool | ip | IP address for the pool. | IP address, such as `192.9.200.50`. |
| Pool | mask | IP mask (i.e., significant left-most number of bits in the IP address). | Integer, such as 24. |
| Pool | vpn | Service to handle the session awareness function for the active member of the cluster (or standalone if unclustered). | String, currently `app1`. |
| Pool | data | Service to handle failover requests on the active member of the cluster (or standalone if unclustered). | String. |

- `<node>`: Identifies a node within the topology. In a Director Realm, the nodes represent the BRASs. The `<node>` directive appears as follows:

  `<node id=‘<id>‘ active=‘<flag>‘ action='<action>'>`

  where:

  - `id`: The ID of a given BRAS: a string of the form `rA_B_C_D`, where `A`, `B`, `C` and `D` are the components of the IP address, such as `r192_9_200_50`.
  - `active`: A string that indicates if the given BRAS or resource realm is online or offline.
  - The directive `action` has a similar meaning as for resources above.

  Within a node record, the following subrecords are allowed:

  - `<resourceid>`: Represents a resource associated with the node. The `<resourceid>` directive appears as follows:

  `<resourceid id=<id>' type='<type>' action='<action>'>`

    The type/ID pairs for a Domain Realm resourceid are as in Table 7. The directive `action` has a similar meaning as for resources above.

**Table 7. Type/ID Pairs for a Domain Realm Resourceid**

| Type | ID | Meaning |
|---|---|---|
| Aracf | The ID of a Resource Controller, such as `r192_9_200_50-RC`. | The Resource Controller that hosts this BRAS. |
| Pool | A string of the form `<IP>/<bits>/<realm>`, where `<IP>` is the IP address of the pool, `<bits>` is the number of leftmost non-mask bits in the IP pool address, and `<realm>` represents the realm to which the pool belongs. | Defines an IP pool. |

## Resource Controller Realm Topology Definition

Figure 11 shows a sample Resource Controller Realm.

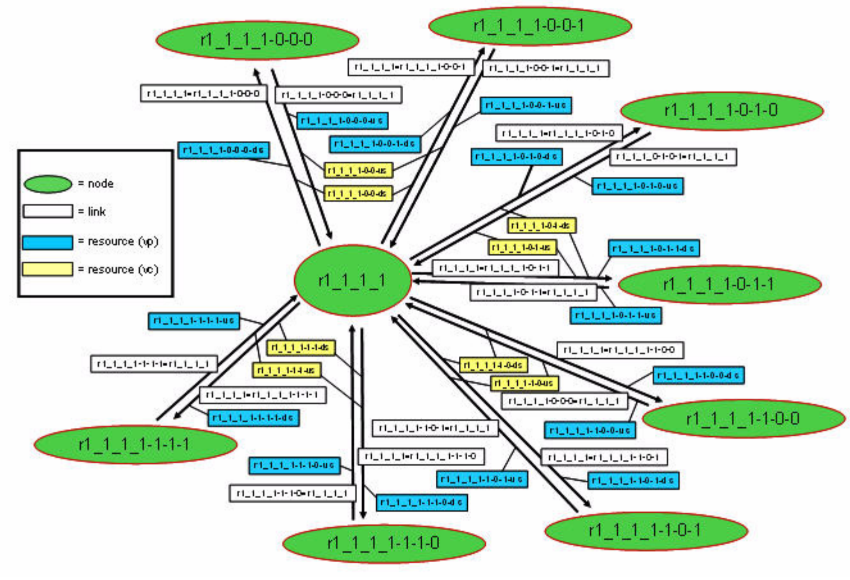


**Figure 11. Sample Resource Controller Realm**

A Resource Controller Realm record contains directives that define the Resource Controller structure. The following is a sample Resource Controller Realm directive:

```
<realm id='<id>' type=‘tsf:realm’ action='<action>'>
```

where:

- `id`: The unique ID of the realm. In the case of a Resource Controller Realm, this identifies a BRAS device.
- `action`:
    - `add`: Adds the realm to the topology. Data that exists in the topology generates errors. However, processing of the file continues if an error occurs.
    - `remove`: Removes the realm from the topology.
    - `update`: Updates the realm in the topology.
    - ‘’ (empty string): Takes no action for the realm itself. Component resources of the realm can be added/removed/updated, as specified by the content of this directive.
    - <Unspecified>: If `action` is not specified, the default action, as specified in the <TIM> directive, or in the tash **Provision** command used to load this file, is taken.

Within a Resource Controller Realm record, the following subrecords are allowed:

- `<resource>`: This defines a resource of the topology, used to represent virtual path (VP) capacity and virtual circuit (VC) capacity. The `<resource>` directive appears as follows:

```
<resource id=‘<id>‘ type=‘<type>‘ action='<action>'>
```

where:

- `id`: ID of the resource within the realm. The value depends on the type of the resource. For VPs, this takes the form `<bras>-<port>-<vp>-us` for upstream capacity, and `<bras>-<port>-<vp>-ds` for downstream capacity. For VCs, this takes the form `<bras>-<port>-<vp>-<vc>-us` for upstream capacity, and `<bras>-<port>-<vp>-<vc>-ds` for downstream capacity. In each of these, `<bras>` represents the ID of the BRAS node in the topology. `<port>`, `vp`, and `vc` are integers.
- `type`: One of the following values:
  - `vp` (for a resource used for VP capacity)
  - `vc` (for a resource used for VC capacity)
- `action`:
  - `add`: Adds the realm to the topology. Data that exists in the topology generates errors. However, processing of the file continues if an error occurs.
  - `remove`: Removes the realm from the topology.
  - `update`: Updates the realm in the topology.
  - ‘’ (empty string): Takes no action for the realm itself. Component resources of the realm can be added/removed/updated, as specified by the content of this directive.
  - <Unspecified>: If `action` is not specified, the default action, as specified in the <TIM> directive, or in the tash **Provision** command used to load this file, is taken.

Within a resource record, the following subrecords are allowed:

- `<property>`: Represents a property of a resource. The `<property>` directive appears as follows:

  ```
  <property name=‘<name>‘ value=‘<value>‘ action='<action>'>
  ```

  The name/value pairs for a Domain Realm property are as in Table 8. The directive `action` has a similar meaning as for resources above.

**Table 8. Name/Value Pairs for a Domain Realm Property**

| Resource Type | Property Name | Meaning | Property Value |
|---|---|---|---|
| vp or vc | used | This is always zero on the Topology Database Server. A running Resource Controller sets it during normal operation. | Should be zero on a Topology Database Server. On a running Resource Controller, this represents the in-use capacity. |
| vp or vc | capacity | The capacity of the VP or VC | An integer that represents the capacity. Note that no units are defined. The units are whatever the requesting interface requires, for example bits per second or Kbits per second. Provision this with correct values. |

- `<node>`: Identifies a node within the topology. In a Resource Controller Realm, the nodes represent the BRAS and the connected CPEs. The `<node>` directive appears as follows:

  ```
  <node id=‘<id>‘ active=‘<flag>‘ action='<action>'>
  ```

  where:

  - `id`: The ID of a given node. For a BRAS, this is of the form, `rA_B_C_D`, where `A`, `B`, `C`, and `D` are the components of the IP address of the BRAS, `A.B.C.D`. For a CPE, the node ID is of the form `<BRAS-ID>-<port>-<vp>-<vc>,` where `<BRAS-ID>` is the ID of the BRAS for the realm, and `<port>`, `<vp>`, and `<vc>` represent how the BRAS connects to the CPE.
  - `active`: A string that indicates if the given node, or the link between them is up (`true`) or down (`false`).
  - The directive `action` has a similar meaning as for resources above.

- `<link>`: Identifies a link within the topology. In a Resource Controller Realm, the links represent the connectivity between the BRAS and CPE devices. The BRAS connects to each CPE device by two links, one for upstream bandwidth and one for downstream bandwidth. The `<link>` directive appears as follows:

  ```
  <link id=‘<id>‘ src=‘<source>‘ dst=‘<dest>‘ active=‘<flag>‘
     action='<action>'>
  ```

  where:

  - `id`: The ID of a given link. Within a Resource Controller Realm, this takes the form `<source>=<dest>`, where both `source` and `dest` represent either the node ID of the BRAS or of one of the CPEs, depending on whether the link represents an upstream or downstream capacity. For example, `r192_9_200_15=r192_9_200_15-0-1-1`.
  - `source`: The source of a link. This represents the source of the link – the ID of either the BRAS or a CPE. For example, `r192_9_200_15`.
  - `dest`: The destination of a link. This represents the source of the link – the ID of either the BRAS or a CPE. For example, `r192_9_200_15-2-1-2`.
  - `active`: A string that indicates if the given clusterlink is active. The possible values are `true` or `false`.
  - The directive `action` has a similar meaning as for resources above.

  Within a link record, the following subrecords are allowed:

  - `<resourceid>`: Represents a resource associated with the link. The `<resourceid>` directive appears as follows:

  ```
  <resourceid id=<id>' type='<type>' action='<action>'>
  ```

  The type/ID pairs for a Domain Realm resourceid are as in Table 9. The directive `action` has a similar meaning as for resources above.

**Table 9. Type/ID Pairs for a Domain Realm Resourceid**

| Type | ID | Meaning |
|---|---|---|
| vp | String, such as `r192_9_200_15-3-4-ds`. | ID of a VP resource. |
| vc | String, such as `r192_9_200_15-3-4-2-ds`. | ID of a VC resource. |

# Creating the R-Realm Topology File

On the Topology Database Server, create a topology information file to define the R-Realm topology that the Directors and Resource Controllers use. You can have many topology information files. Give each topology information file an extension of `.tim`. You can place topology information files in any directory on the Topology Database Server system.

**Note:** *You need to provision only the R-Realm information. The content of the Domain and Director Realms is created automatically.*

### *Example: Topology Information File*

The following is a sample R-Realm topology information file:

```
<tim realms='tds_manage_realms'
 resources='tds_taf_rc_resources'
 nodes='tds_taf_rc_nodes' links='tds_taf_rc_links'>

   <realm id='r1_1_1_1' type='tsf:realm'>
      <!-- VPs -->
      <resource id='r1_1_1_1-0-0-us' type='vp'>
         <property name='used' value='0'/>
         <property name='capacity' value='64000'/>
      </resource>
      . . .

      <!-- VCs -->
      <resource id='r1_1_1_1-0-0-0-us' type='vc'>
         <property name='used' value='0'/>
         <property name='capacity' value='32000'/>
      </resource>
      . . .

      <!-- nodes -->
      <node id='r1_1_1_1' active='true'/>
      <node id='r1_1_1_1-0-0-0' active='true'/>
      . . .

      <!-- links -->
      <link id='r1_1_1_1=r1_1_1_1-0-0-0' src='r1_1_1_1'
       dst='r1_1_1_1-0-0-0' active='true'>
         <resourceid id='r1_1_1_1-0-0-ds' type='vp'/>
         <resourceid id='r1_1_1_1-0-0-0-ds' type='vc'/>
      </link>
      . . .
   </realm>
</tim>
```

This file uses a naming scheme for realm IDs, where `rA_B_C_D` represents a BRAS with IP address `A.B.C.D`. However, any naming scheme is acceptable. The realm `type` is `tsf:realm` for Resource Realms.

The `VPs` section defines the VPs on the system. There are one or more VPs per port on the device, and each has a property for upstream capacity and one for downstream capacity. A naming scheme for a VP resource takes the form `<Realm>-<Port>-<VP>-<Dir>`, where `Realm` is the ID of the realm, `Port` is the integer ID of the port containing the VP, `VP` is the integer ID of the VP itself, and `Dir` is the direction: `us` for upstream and `ds` for downstream. However, any naming scheme is acceptable.

The `VCs` section defines the VCs on the system. There are one or more VCs per VP on the device, and each has a property for upstream capacity and one for downstream capacity. A naming scheme for a VC resource takes the form `<Realm>-<Port>-<VP>-<VC>-<Dir>`, where `Realm`, `Port`, and `VP` are defined above (`Port` and `VP` being the containers of the VC being defined), `VC` is the integer ID of the VC itself, and `Dir` is the direction: `us` for upstream and `ds` for downstream. However, any naming scheme is acceptable.

Both VP and VC resources have two properties: `Used` and `Capacity`. `Used` is usually `0` at initial system provisioning. `Capacity` defines the maximum bandwidth allowed on the VP or VC. It is in units defined by the SIP interface to which the QoS Server connects (usually, bits per second).

The `nodes` section defines the nodes in the realm. There is one node for the BRAS, and one node for each CPE (as defined by the cross product of available ports, VPs, and VCs on the BRAS). A naming scheme for nodes is for the ID of the BRAS node to be the same as the realm ID, and for the ID of each CPE to take the form `<Realm>-<Port>-<VP>-<VC>`, where `Realm`, `Port`, `VP`, and `VC` are defined above. Set the `active` field of nodes to `true`.

The `links` section defines the link connectivity between the BRAS and the CPEs. A naming scheme for links is for the ID to be `<src>=<dst>`, where `src` is the source node of the link (depending on whether it is upstream or downstream) and `dst` is the destination. Set `active` to true. Each link has two `resourceid` tags, to declare the VP and VC resources that are associated with the link.

### *Example: Another Topology Information File*

The following topology information file defines a topology consisting of a Domain Controller, one Director, and a clustered pair of Resource Controllers. The Resource Controllers serve one BRAS device, which contains two ports. Each port has two VPs, and each VP has two VCs.

```
<global>

  <!-- Domain, (taf_ services only affect local store) -->
  <tim realms='taf_realms' resources='taf_resources'
   nodes='taf_nodes' links='taf_links'>
     <!-- example domain information: 3 systems -->
     <!-- 2 RCs (.1, .2) that are clustered, and 1 dir (.3) -->

     <!-- domain realm should already exist (from initial -->
     <!-- setup of domain), hence action='' -->
     <realm id='domain' type='tsf:domain' action=''>

        <!-- interface resources -->
        <resource id='192.168.100.163' type='interface'>
           <property name='name'      value='if1'/>
           <property name='address'   value='192.168.100.163'/>
           <property name='subnet'    value='255.255.0.0'/>
           <property name='state'     value='1'/>
           <property name='nodekey'   value='192.168.100.163:10000'/>
        </resource>
```

```
<resource id='200.1.1.1' type='interface'>
   <property name='name'      value='if2'/>
   <property name='address'   value='200.1.1.1'/>
   <property name='subnet'    value='255.255.0.0'/>
   <property name='state'     value='1'/>
   <property name='nodekey'   value='192.168.100.163:10000'/>
</resource>
<resource id='192.168.100.164' type='interface'>
   <property name='name'      value='if1'/>
   <property name='address'   value='192.168.100.164'/>
   <property name='subnet'    value='255.255.0.0'/>
   <property name='state'     value='1'/>
   <property name='nodekey' value='192.168.100.164:10000'/>
</resource>
<resource id='200.1.1.2' type='interface'>
   <property name='name'      value='if2'/>
   <property name='address'   value='200.1.1.2'/>
   <property name='subnet'    value='255.255.0.0'/>
   <property name='state'     value='1'/>
   <property name='nodekey' value='192.168.100.164:10000'/>
</resource>
<resource id='192.168.100.162' type='interface'>
   <property name='name'      value='if1'/>
   <property name='address'   value='192.168.100.162'/>
   <property name='subnet'    value='255.255.0.0'/>
   <property name='state'     value='1'/>
   <property name='nodekey' value='192.168.100.162:10000'/>
</resource>
<resource id='200.1.1.3' type='interface'>
   <property name='name'      value='if2'/>
   <property name='address'   value='200.1.1.3'/>
   <property name='subnet'    value='255.255.0.0'/>
   <property name='state'     value='1'/>
   <property name='nodekey' value='192.168.100.162:10000'/>
</resource>

<!-- health resources -->
<resource id='192.168.100.163:10000' type='health'>
   <property name='health'    value='0'/>
</resource>
<resource id='192.168.100.164:10000' type='health'>
   <property name='health'    value='0'/>
</resource>
<resource id='192.168.100.162:10000' type='health'>
   <property name='health'    value='0'/>
</resource>

<!-- cluster resources -->
<resource id='cluster-1' type='cluster'>
   <property name='type'      value='hot'/>
   <property name='name'      value='cluster-1'/>
   <property name='active' value='192.168.100.163:10000'/>
   <property name='standby' value='192.168.100.164:10000'/>
</resource>

<!-- configuration resources -->
<resource id='192.168.100.163:10000' type='configuration'>
```

```
         <property name='configuration'  value='{name rc-1}'/>
      </resource>
      <resource id='192.168.100.164:10000' type='configuration'>
         <property name='configuration'  value='{name rc-2}'/>
      </resource>
      <resource id='192.168.100.162:10000' type='configuration'>
         <property name='configuration' value='{name dir-1}'/>
      </resource>

      <!-- nodes -->
      <node id='192.168.100.163:10000' active='true'>
         <resourceid id='aracf'                    type='role'/>
         <resourceid id='aracf'          type='configuration'/>
         <resourceid id='cluster-1'            type='cluster'/>
         <resourceid id='200.1.1.1'          type='interface'/>
         <resourceid id='192.168.100.163'   type='interface'/>
         <resourceid id='192.168.100.163:10000' type='health'/>
         <resourceid id='192.168.100.163:10000'
 type='configuration'/>
      </node>
      <node id='192.168.100.164:10000' active='true'>
         <resourceid id='aracf'                    type='role'/>
         <resourceid id='aracf'          type='configuration'/>
         <resourceid id='cluster-1'            type='cluster'/>
         <resourceid id='200.1.1.2'          type='interface'/>
         <resourceid id='192.168.100.164'   type='interface'/>
         <resourceid id='192.168.100.164:10000' type='health'/>
         <resourceid id='192.168.100.164:10000'
 type='configuration'/>
      </node>
      <node id='192.168.100.162:10000' active='true'>
         <resourceid id='spdf'                     type='role'/>
         <resourceid id='spdf'           type='configuration'/>
         <resourceid id='200.1.1.3'          type='interface'/>
         <resourceid id='192.168.100.162'   type='interface'/>
         <resourceid id='192.168.100.162:10000'  type='health'/>
         <resourceid id='192.168.100.162:10000'
 type='configuration'/>
      </node>

      <!-- links -->
      <link id='cluster-1' src='192.168.100.163:10000'
       dst='192.168.100.164:10000' active='true'/>

   </realm>
</tim>

<!-- Director (tds_taf_dir_ services auto-distribute to directors)
 -->
<tim realms='tds_taf_dir_realms' resources='tds_taf_dir_resources'
 nodes='tds_taf_dir_nodes' links='tds_taf_dir_links'>
   <!-- example domain information: 2 realms r1_1_1_1 r2_2_2_2 -->

   <!-- app1 realm should already exist (from initial setup of -->
   <!-- domain), hence action='' -->
   <realm id='app1' type='tsf:director' action=''>
```

```
      <!-- aracf resources -->
      <resource id='r1_1_1_1' type='aracf'>
         <property name='health'    value='0'/>
         <property name='host'      value='192.168.100.163'/>
         <property name='port'      value='10100'/>
         <property name='qos'       value='aracf_sif'/>
         <property name='saf'       value='aracf_saf'/>
         <property name='fof'       value='aracf_resiliency'/>
         <property name='_health'   value='0'/>
         <property name='_host'     value='192.168.100.164'/>
         <property name='_port'     value='101000'/>
         <property name='_qos'      value='aracf_sif'/>
         <property name='_saf'      value='aracf_saf'/>
         <property name='_fof'      value='aracf_resiliency'/>
      </resource>
      <resource id='r2_2_2_2' type='aracf'>
         <property name='health'    value='0'/>
         <property name='host'      value='192.168.100.163'/>
         <property name='port'      value='10100'/>
         <property name='qos'       value='aracf_sif'/>
         <property name='saf'       value='aracf_saf'/>
         <property name='fof'       value='aracf_resiliency'/>
         <property name='_health'   value='0'/>
         <property name='_host'     value='192.168.100.164'/>
         <property name='_port'     value='101000'/>
         <property name='_qos'      value='aracf_sif'/>
         <property name='_saf'      value='aracf_saf'/>
         <property name='_fof'      value='aracf_resiliency'/>
      </resource>

      <!-- pool resources -->
      <resource id='1.1.1.1/24/app1' type='pool'>
         <property name='ip'     value='1.1.1.1'/>
         <property name='mask'   value='24'/>
         <property name='vpn'    value='app1'/>
         <property name='data'   value='r1_1_1_1'/>
      </resource>
      <resource id='2.2.2.2/24/app1' type='pool'>
         <property name='ip'     value='2.2.2.2'/>
         <property name='mask'   value='24'/>
         <property name='vpn'    value='app1'/>
         <property name='data'   value='r2_2_2_2'/>
      </resource>

      <!-- nodes -->
      <node id='r1_1_1_1' active='true'>
         <resourceid id='r1_1_1_1'           type='aracf'/>
         <resourceid id='1.1.1.1/24/app1' type='pool'/>
      </node>
      <node id='r2_2_2_2' active='true'>
         <resourceid id='r2_2_2_2'           type='aracf'/>
         <resourceid id='2.2.2.2/24/app1' type='pool'/>
      </node>

   </realm>
</tim>
```

```
<!-- Resource Realms (for RCs), (tds_taf_rc services push all -->
<!-- updates to any assigned RCs [best effort]) -->
<tim realms='tds_manage_realms' resources='tds_taf_rc_resources'
 nodes='tds_taf_rc_nodes' links='tds_taf_rc_links'>

   <!-- realm r1_1_1_1, has one BRAS and 8 HGWs-->
   <!--  r1_1_1_1 -->
   <!--     port 0 -->
   <!--        vpi 0 -->
   <!--           vci 0 -->
   <!--           vci 1 -->
   <!--        vpi 1 -->
   <!--           vci 0 -->
   <!--           vci 1 -->
   <!--     port 1 -->
   <!--        vpi 0 -->
   <!--           vci 0 -->
   <!--           vci 1 -->
   <!--        vpi 1 -->
   <!--           vci 0 -->
   <!--           vci 1 -->
   <!--  vp, vc resources for upstream (-us) and downstream (-ds)
 links -->

   <realm id='r1_1_1_1' type='tsf:realm' action=''>
      <!-- VPs -->
      <resource id='r1_1_1_1-0-0-us' type='vp'>
         <property name='used' value='0'/>
         <property name='capacity' value='64000'/>
      </resource>
      <resource id='r1_1_1_1-0-0-ds' type='vp'>
         <property name='used' value='0'/>
         <property name='capacity' value='64000'/>
      </resource>
      <resource id='r1_1_1_1-0-1-us' type='vp'>
         <property name='used' value='0'/>
         <property name='capacity' value='64000'/>
      </resource>
      <resource id='r1_1_1_1-0-1-ds' type='vp'>
         <property name='used' value='0'/>
         <property name='capacity' value='64000'/>
      </resource>
      <resource id='r1_1_1_1-1-0-us' type='vp'>
         <property name='used' value='0'/>
         <property name='capacity' value='64000'/>
      </resource>
      <resource id='r1_1_1_1-1-0-ds' type='vp'>
         <property name='used' value='0'/>
         <property name='capacity' value='64000'/>
      </resource>
      <resource id='r1_1_1_1-1-1-us' type='vp'>
         <property name='used' value='0'/>
         <property name='capacity' value='64000'/>
      </resource>
      <resource id='r1_1_1_1-1-1-ds' type='vp'>
         <property name='used' value='0'/>
         <property name='capacity' value='64000'/>
```

```
</resource>

<!-- VCs -->
<resource id='r1_1_1_1-0-0-0-us' type='vc'>
   <property name='used' value='0'/>
   <property name='capacity' value='32000'/>
</resource>
<resource id='r1_1_1_1-0-0-0-ds' type='vc'>
   <property name='used' value='0'/>
   <property name='capacity' value='32000'/>
</resource>
<resource id='r1_1_1_1-0-0-1-us' type='vc'>
   <property name='used' value='0'/>
   <property name='capacity' value='32000'/>
</resource>
<resource id='r1_1_1_1-0-0-1-ds' type='vc'>
   <property name='used' value='0'/>
   <property name='capacity' value='32000'/>
</resource>
<resource id='r1_1_1_1-0-1-0-ds' type='vc'>
   <property name='used' value='0'/>
   <property name='capacity' value='32000'/>
</resource>
<resource id='r1_1_1_1-0-1-0-us' type='vc'>
   <property name='used' value='0'/>
   <property name='capacity' value='32000'/>
</resource>
<resource id='r1_1_1_1-1-0-0-us' type='vc'>
   <property name='used' value='0'/>
   <property name='capacity' value='32000'/>
</resource>
<resource id='r1_1_1_1-1-0-0-ds' type='vc'>
   <property name='used' value='0'/>
   <property name='capacity' value='32000'/>
</resource>
<resource id='r1_1_1_1-0-1-1-us' type='vc'>
   <property name='used' value='0'/>
   <property name='capacity' value='32000'/>
</resource>
<resource id='r1_1_1_1-0-1-1-ds' type='vc'>
   <property name='used' value='0'/>
   <property name='capacity' value='32000'/>
</resource>
<resource id='r1_1_1_1-1-0-1-us' type='vc'>
   <property name='used' value='0'/>
   <property name='capacity' value='32000'/>
</resource>
<resource id='r1_1_1_1-1-0-1-ds' type='vc'>
   <property name='used' value='0'/>
   <property name='capacity' value='32000'/>
</resource>
<resource id='r1_1_1_1-1-1-0-us' type='vc'>
   <property name='used' value='0'/>
   <property name='capacity' value='32000'/>
</resource>
<resource id='r1_1_1_1-1-1-0-ds' type='vc'>
   <property name='used' value='0'/>
   <property name='capacity' value='32000'/>
</resource>
<resource id='r1_1_1_1-1-1-1-us' type='vc'>
   <property name='used' value='0'/>
   <property name='capacity' value='32000'/>
</resource>
<resource id='r1_1_1_1-1-1-1-ds' type='vc'>
   <property name='used' value='0'/>
   <property name='capacity' value='32000'/>
</resource>

<!-- nodes -->
<node id='r1_1_1_1' active='true'/>
<node id='r1_1_1_1-0-0-0' active='true'/>
<node id='r1_1_1_1-0-0-1' active='true'/>
<node id='r1_1_1_1-0-1-0' active='true'/>
<node id='r1_1_1_1-0-1-1' active='true'/>
<node id='r1_1_1_1-1-0-0' active='true'/>
<node id='r1_1_1_1-1-0-1' active='true'/>
<node id='r1_1_1_1-1-1-0' active='true'/>
<node id='r1_1_1_1-1-1-1' active='true'/>

<!-- links -->
<link id='r1_1_1_1=r1_1_1_1-0-0-0' src='r1_1_1_1'
 dst='r1_1_1_1-0-0-0' active='true'>
   <resourceid id='r1_1_1_1-0-0-ds' type='vp'/>
   <resourceid id='r1_1_1_1-0-0-0-ds' type='vc'/>
</link>
<link id='r1_1_1_1-0-0-0=r1_1_1_1' src='r1_1_1_1-0-0-0'
 dst='r1_1_1_1' active='true'>
   <resourceid id='r1_1_1_1-0-0-us' type='vp'/>
   <resourceid id='r1_1_1_1-0-0-0-us' type='vc'/>
</link>
<link id='r1_1_1_1=r1_1_1_1-0-0-1' src='r1_1_1_1'
 dst='r1_1_1_1-0-0-1' active='true'>
   <resourceid id='r1_1_1_1-0-0-ds' type='vp'/>
   <resourceid id='r1_1_1_1-0-0-1-ds' type='vc'/>
</link>
<link id='r1_1_1_1-0-0-1=r1_1_1_1' src='r1_1_1_1-0-0-1'
 dst='r1_1_1_1' active='true'>
   <resourceid id='r1_1_1_1-0-0-us' type='vp'/>
   <resourceid id='r1_1_1_1-0-0-1-us' type='vc'/>
</link>
<link id='r1_1_1_1=r1_1_1_1-0-1-0' src='r1_1_1_1'
 dst='r1_1_1_1-0-1-0' active='true'>
   <resourceid id='r1_1_1_1-0-1-ds' type='vp'/>
   <resourceid id='r1_1_1_1-0-1-0-ds' type='vc'/>
</link>
<link id='r1_1_1_1-0-1-0=r1_1_1_1' src='r1_1_1_1-0-1-0'
 dst='r1_1_1_1' active='true'>
   <resourceid id='r1_1_1_1-0-1-us' type='vp'/>
   <resourceid id='r1_1_1_1-0-1-0-us' type='vc'/>
</link>
<link id='r1_1_1_1=r1_1_1_1-0-1-1' src='r1_1_1_1'
 dst='r1_1_1_1-0-1-1' active='true'>
   <resourceid id='r1_1_1_1-0-1-ds' type='vp'/>
   <resourceid id='r1_1_1_1-0-1-1-ds' type='vc'/>
   </link>
   <link id='r1_1_1_1-0-1-1=r1_1_1_1' src='r1_1_1_1-0-1-1'
    dst='r1_1_1_1' active='true'>
      <resourceid id='r1_1_1_1-0-1-us' type='vp'/>
      <resourceid id='r1_1_1_1-0-1-1-us' type='vc'/>
   </link>
   <link id='r1_1_1_1=r1_1_1_1-1-0-0' src='r1_1_1_1'
    dst='r1_1_1_1-1-0-0' active='true'>
      <resourceid id='r1_1_1_1-1-0-ds' type='vp'/>
      <resourceid id='r1_1_1_1-1-0-0-ds' type='vc'/>
   </link>
   <link id='r1_1_1_1-1-0-0=r1_1_1_1' src='r1_1_1_1-1-0-0'
    dst='r1_1_1_1' active='true'>
      <resourceid id='r1_1_1_1-1-0-us' type='vp'/>
      <resourceid id='r1_1_1_1-1-0-0-us' type='vc'/>
   </link>
   <link id='r1_1_1_1=r1_1_1_1-1-0-1' src='r1_1_1_1'
    dst='r1_1_1_1-1-0-1' active='true'>
      <resourceid id='r1_1_1_1-1-0-ds' type='vp'/>
      <resourceid id='r1_1_1_1-1-0-1-ds' type='vc'/>
   </link>
   <link id='r1_1_1_1-1-0-1=r1_1_1_1' src='r1_1_1_1-1-0-1'
    dst='r1_1_1_1' active='true'>
      <resourceid id='r1_1_1_1-1-0-us' type='vp'/>
      <resourceid id='r1_1_1_1-1-0-1-us' type='vc'/>
   </link>
   <link id='r1_1_1_1=r1_1_1_1-1-1-0' src='r1_1_1_1'
    dst='r1_1_1_1-1-1-0' active='true'>
      <resourceid id='r1_1_1_1-1-1-ds' type='vp'/>
      <resourceid id='r1_1_1_1-1-1-0-ds' type='vc'/>
   </link>
   <link id='r1_1_1_1-1-1-0=r1_1_1_1' src='r1_1_1_1-1-1-0'
    dst='r1_1_1_1' active='true'>
      <resourceid id='r1_1_1_1-1-1-us' type='vp'/>
      <resourceid id='r1_1_1_1-1-1-0-us' type='vc'/>
   </link>
   <link id='r1_1_1_1=r1_1_1_1-1-1-1' src='r1_1_1_1'
    dst='r1_1_1_1-1-1-1' active='true'>
      <resourceid id='r1_1_1_1-1-1-ds' type='vp'/>
      <resourceid id='r1_1_1_1-1-1-1-ds' type='vc'/>
   </link>
   <link id='r1_1_1_1-1-1-1=r1_1_1_1' src='r1_1_1_1-1-1-1'
    dst='r1_1_1_1' active='true'>
      <resourceid id='r1_1_1_1-1-1-us' type='vp'/>
      <resourceid id='r1_1_1_1-1-1-1-us' type='vc'/>
   </link>
</realm>

<!-- realm r2_2_2_2, has one BRAS and 8 HGWs-->
<!--  r1_1_1_1 -->
<!--     port 0 -->
<!--        vpi 0 -->
<!--           vci 0 -->
<!--           vci 1 -->
<!--        vpi 1 -->
<!--           vci 0 -->
<!--           vci 1 -->
```

```
  <!--     port 1 -->
  <!--        vpi 0 -->
  <!--           vci 0 -->
  <!--           vci 1 -->
  <!--        vpi 1 -->
  <!--           vci 0 -->
  <!--           vci 1 -->
  <!--  vp, vc resources for upstream (-us) and downstream (-ds)
links -->

  <realm id='r2_2_2_2' type='tsf:realm' action=''>
     <!-- VPs -->
     <resource id='r2_2_2_2-0-0-us' type='vp'>
        <property name='used' value='0'/>
        <property name='capacity' value='64000'/>
     </resource>
     <resource id='r2_2_2_2-0-0-ds' type='vp'>
        <property name='used' value='0'/>
        <property name='capacity' value='64000'/>
     </resource>
     <resource id='r2_2_2_2-0-1-us' type='vp'>
        <property name='used' value='0'/>
        <property name='capacity' value='64000'/>
     </resource>
     <resource id='r2_2_2_2-0-1-ds' type='vp'>
        <property name='used' value='0'/>
        <property name='capacity' value='64000'/>
     </resource>
     <resource id='r2_2_2_2-1-0-us' type='vp'>
        <property name='used' value='0'/>
        <property name='capacity' value='64000'/>
     </resource>
     <resource id='r2_2_2_2-1-0-ds' type='vp'>
        <property name='used' value='0'/>
        <property name='capacity' value='64000'/>
     </resource>
     <resource id='r2_2_2_2-1-1-us' type='vp'>
        <property name='used' value='0'/>
        <property name='capacity' value='64000'/>
     </resource>
     <resource id='r2_2_2_2-1-1-ds' type='vp'>
        <property name='used' value='0'/>
        <property name='capacity' value='64000'/>
     </resource>

     <!-- VCs -->
     <resource id='r2_2_2_2-0-0-0-us' type='vc'>
        <property name='used' value='0'/>
        <property name='capacity' value='32000'/>
     </resource>
     <resource id='r2_2_2_2-0-0-0-ds' type='vc'>
        <property name='used' value='0'/>
        <property name='capacity' value='32000'/>
     </resource>
     <resource id='r2_2_2_2-0-0-1-us' type='vc'>
        <property name='used' value='0'/>
        <property name='capacity' value='32000'/>
```

```
</resource>
<resource id='r2_2_2_2-0-0-1-ds' type='vc'>
   <property name='used' value='0'/>
   <property name='capacity' value='32000'/>
</resource>
<resource id='r2_2_2_2-0-1-0-us' type='vc'>
   <property name='used' value='0'/>
   <property name='capacity' value='32000'/>
</resource>
<resource id='r2_2_2_2-0-1-0-ds' type='vc'>
   <property name='used' value='0'/>
   <property name='capacity' value='32000'/>
</resource>
<resource id='r2_2_2_2-0-1-1-us' type='vc'>
   <property name='used' value='0'/>
   <property name='capacity' value='32000'/>
</resource>
<resource id='r2_2_2_2-0-1-1-ds' type='vc'>
   <property name='used' value='0'/>
   <property name='capacity' value='32000'/>
</resource>
<resource id='r2_2_2_2-1-0-0-us' type='vc'>
   <property name='used' value='0'/>
   <property name='capacity' value='32000'/>
</resource>
<resource id='r2_2_2_2-1-0-0-ds' type='vc'>
   <property name='used' value='0'/>
   <property name='capacity' value='32000'/>
</resource>
<resource id='r2_2_2_2-1-0-1-us' type='vc'>
   <property name='used' value='0'/>
   <property name='capacity' value='32000'/>
</resource>
<resource id='r2_2_2_2-1-0-1-ds' type='vc'>
   <property name='used' value='0'/>
   <property name='capacity' value='32000'/>
</resource>
<resource id='r2_2_2_2-1-1-0-us' type='vc'>
   <property name='used' value='0'/>
   <property name='capacity' value='32000'/>
</resource>
<resource id='r2_2_2_2-1-1-0-ds' type='vc'>
   <property name='used' value='0'/>
   <property name='capacity' value='32000'/>
</resource>
<resource id='r2_2_2_2-1-1-1-us' type='vc'>
   <property name='used' value='0'/>
   <property name='capacity' value='32000'/>
</resource>
<resource id='r2_2_2_2-1-1-1-ds' type='vc'>
   <property name='used' value='0'/>
   <property name='capacity' value='32000'/>
</resource>

<!-- nodes -->
<node id='r2_2_2_2' active='true'/>
<node id='r2_2_2_2-0-0-0' active='true'/>
```

```
<node id='r2_2_2_2-0-0-1' active='true'/>
<node id='r2_2_2_2-0-1-0' active='true'/>
<node id='r2_2_2_2-0-1-1' active='true'/>
<node id='r2_2_2_2-1-0-0' active='true'/>
<node id='r2_2_2_2-1-0-1' active='true'/>
<node id='r2_2_2_2-1-1-0' active='true'/>
<node id='r2_2_2_2-1-1-1' active='true'/>

<!-- links -->
<link id='r2_2_2_2=r2_2_2_2-0-0-0' src='r2_2_2_2'
 dst='r2_2_2_2-0-0-0' active='true'>
   <resourceid id='r2_2_2_2-0-0-ds' type='vp'/>
   <resourceid id='r2_2_2_2-0-0-0-ds' type='vc'/>
</link>
<link id='r2_2_2_2-0-0-0=r2_2_2_2' src='r2_2_2_2-0-0-0'
 dst='r2_2_2_2' active='true'>
   <resourceid id='r2_2_2_2-0-0-us' type='vp'/>
   <resourceid id='r2_2_2_2-0-0-0-us' type='vc'/>
</link>
<link id='r2_2_2_2=r2_2_2_2-0-0-1' src='r2_2_2_2'
 dst='r2_2_2_2-0-0-1' active='true'>
   <resourceid id='r2_2_2_2-0-0-ds' type='vp'/>
   <resourceid id='r2_2_2_2-0-0-1-ds' type='vc'/>
</link>
<link id='r2_2_2_2-0-0-1=r2_2_2_2' src='r2_2_2_2-0-0-1'
 dst='r2_2_2_2' active='true'>
   <resourceid id='r2_2_2_2-0-0-us' type='vp'/>
   <resourceid id='r2_2_2_2-0-0-1-us' type='vc'/>
</link>
<link id='r2_2_2_2=r2_2_2_2-0-1-0' src='r2_2_2_2'
 dst='r2_2_2_2-0-1-0' active='true'>
   <resourceid id='r2_2_2_2-0-1-ds' type='vp'/>
   <resourceid id='r2_2_2_2-0-1-0-ds' type='vc'/>
</link>
<link id='r2_2_2_2-0-1-0=r2_2_2_2' src='r2_2_2_2-0-1-0'
 dst='r2_2_2_2' active='true'>
   <resourceid id='r2_2_2_2-0-1-us' type='vp'/>
   <resourceid id='r2_2_2_2-0-1-0-us' type='vc'/>
</link>
<link id='r2_2_2_2=r2_2_2_2-0-1-1' src='r2_2_2_2'
 dst='r2_2_2_2-0-1-1' active='true'>
   <resourceid id='r2_2_2_2-0-1-ds' type='vp'/>
   <resourceid id='r2_2_2_2-0-1-1-ds' type='vc'/>
</link>
<link id='r2_2_2_2-0-1-1=r2_2_2_2' src='r2_2_2_2-0-1-1'
 dst='r2_2_2_2' active='true'>
   <resourceid id='r2_2_2_2-0-1-us' type='vp'/>
   <resourceid id='r2_2_2_2-0-1-1-us' type='vc'/>
</link>
<link id='r2_2_2_2=r2_2_2_2-1-0-0' src='r2_2_2_2'
 dst='r2_2_2_2-1-0-0' active='true'>
   <resourceid id='r2_2_2_2-1-0-ds' type='vp'/>
   <resourceid id='r2_2_2_2-1-0-0-ds' type='vc'/>
</link>
<link id='r2_2_2_2-1-0-0=r2_2_2_2' src='r2_2_2_2-1-0-0'
 dst='r2_2_2_2' active='true'>
   <resourceid id='r2_2_2_2-1-0-us' type='vp'/>
```

```
           <resourceid id='r2_2_2_2-1-0-0-us' type='vc'/>
        </link>
        <link id='r2_2_2_2=r2_2_2_2-1-0-1' src='r2_2_2_2'
         dst='r2_2_2_2-1-0-1' active='true'>
           <resourceid id='r2_2_2_2-1-0-ds' type='vp'/>
           <resourceid id='r2_2_2_2-1-0-1-ds' type='vc'/>
        </link>
        <link id='r2_2_2_2-1-0-1=r2_2_2_2' src='r2_2_2_2-1-0-1'
         dst='r2_2_2_2' active='true'>
           <resourceid id='r2_2_2_2-1-0-us' type='vp'/>
           <resourceid id='r2_2_2_2-1-0-1-us' type='vc'/>
        </link>
        <link id='r2_2_2_2=r2_2_2_2-1-1-0' src='r2_2_2_2'
         dst='r2_2_2_2-1-1-0' active='true'>
           <resourceid id='r2_2_2_2-1-1-ds' type='vp'/>
           <resourceid id='r2_2_2_2-1-1-0-ds' type='vc'/>
        </link>
        <link id='r2_2_2_2-1-1-0=r2_2_2_2' src='r2_2_2_2-1-1-0'
         dst='r2_2_2_2' active='true'>
           <resourceid id='r2_2_2_2-1-1-us' type='vp'/>
           <resourceid id='r2_2_2_2-1-1-0-us' type='vc'/>
        </link>
        <link id='r2_2_2_2=r2_2_2_2-1-1-1' src='r2_2_2_2'
         dst='r2_2_2_2-1-1-1' active='true'>
           <resourceid id='r2_2_2_2-1-1-ds' type='vp'/>
           <resourceid id='r2_2_2_2-1-1-1-ds' type='vc'/>
        </link>
        <link id='r2_2_2_2-1-1-1=r2_2_2_2' src='r2_2_2_2-1-1-1'
         dst='r2_2_2_2' active='true'>
           <resourceid id='r2_2_2_2-1-1-us' type='vp'/>
           <resourceid id='r2_2_2_2-1-1-1-us' type='vc'/>
        </link>
     </realm>

  </tim>
</global>
```

## Provisioning the Resource Realms

As part of the installation process, you provision the system components, pool mappings, and topology information on the Topology Database Server.

After you know what Resource Realms to provision on your system, create the Resource Realms using the following appropriate tash commands. A command file can simplify this operation.

**Note:** *Note that the* **AddResourceRealm** *and* **AssignResourceRealm** *operations are lengthy. Set the timeout value to* `30` *(seconds).*

- Add the Resource Realms that are defined in the `.tim` files into the system, using tash **AddResourceRealm** commands.
- Add appropriate IP pools to the Resource Realms, using tash **AddPool** commands

- Assign each Resource Realm to a Resource Controller, using tash **AssignResourceRealm** commands
- Provision <file>: This command populates the new realm with data.

### *Example: Provisioning*

For example, the following commands add four Resource Realms into the system, then add one IP pool to each one, then assign two of those R-Realms to a Resource Controller, and the other two to a different Resource Controller.

```
prompt% tash
%AddResourceRealm r192_9_21_1 30
%AddResourceRealm r192_9_21_2 30
%AddResourceRealm r192_9_21_3 30
%AddResourceRealm r192_9_21_4 30
%AddPool app1 192.170.1.1 24 r192_9_21_1
%AddPool app1 192.170.5.1 24 r192_9_21_2
%AddPool app1 192.154.10.1 24 r192_9_21_3
%AddPool app1 192.154.12.1 24 r192_9_21_4
%AssignResourceRealm r192_9_21_1 192.168.111.82:10000 30
%AssignResourceRealm r192_9_21_2 192.168.111.82:10000 30
%AssignResourceRealm r192_9_21_3 192.168.111.92:10000 30
%AssignResourceRealm r192_9_21_4 192.168.111.92:10000 30
```

# Index