InCharge™

# InCharge IP Discovery Guide

## Version 6.2

smarts

Cisco Part Number: OL-6132-01

the technology of Crystal Decisions. The use of all Crystal Decisions software products is subject to a separate license agreement included with the Software electronically, in written materials, or both. YOU MAY NOT USE THE CRYSTAL DECISIONS SOFTWARE UNLESS AND UNTIL YOU READ, ACKNOWLEDGE AND ACCEPT THE TERMS AND CONDITIONS OF THE CRYSTAL DECISIONS' SOFTWARE LICENSE AGREEMENT. IF YOU DO NOT ACCEPT THE TERMS AND CONDITIONS OF THE CRYSTAL DECISIONS' SOFTWARE LICENSE, YOU MAY RETURN, WITHIN THIRTY (30) DAYS OF PURCHASE, THE MEDIA PACKAGE AND ALL ACCOMPANYING ITEMS (INCLUDING WRITTEN MATERIALS AND BINDERS OR OTHER CONTAINERS) RELATED TO THE CRYSTAL DECISIONS' TECHNOLOGY, TO SMARTS FOR A FULL REFUND; OR YOU MAY WRITE, CRYSTAL WARRANTIES, P.O. BOX 67427, SCOTTS VALLEY, CA 95067, U.S.A.

GNU eTeks PJA Toolkit



The eTeks PJA Toolkit is resident on the CD on which the Software was delivered to you. Additional information is available at eTEks' web site: http://www.eteks.com. The eTeks PJA Toolkit program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License (GPL) as published by the Free Software Foundation; version 2 of the License. The full text of the applicable GNU GPL is available for viewing at http://www.gnu.org/copyleft/gpl.txt. You may also request a copy of the GPL from the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA. The eTeks PJA Toolkit program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

For a period of three years from the date of your license for the Software, you are entitled to receive under the terms of Sections 1 and 2 of the GPL, for a charge no more than SMARTS' cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code for the GNU eTeks PJA Toolkit provided to you hereunder by requesting such code from SMARTS in writing: Attn: Customer Support, SMARTS, 44 South Broadway, White Plains, New York 10601.

IBM Runtime for AIX



HP-UX Runtime Environment for the Java™ 2 Platform

The Software contains the HP-UX Runtime for the Java™ 2 Platform, distributed pursuant to and governed by Hewlett-Packard Co. ("HP") software license terms set forth in detail at: http://www.hp.com. Please check the Software to determine the version of Java runtime distributed to you.

DataDirect Technologies



NetBSD



RSA Data Security, Inc.



AES



License Terms:

The free distribution and use of this software in both source and binary form is allowed (with or without changes) provided that:

1. distributions of this source code include the above copyright notice, this list of conditions and the following disclaimer;
2. distributions in binary form include the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other associated materials;
3. the copyright holder's name is not used to endorse products built using this software without specific written permission.

ALTERNATIVELY, provided that this notice is retained in full, this product may be distributed under the terms of the GNU General Public License (GPL), in which case the provisions of the GPL apply INSTEAD OF those given above.

Disclaimer: This software is provided 'as is' with no explicit or implied warranties in respect of its properties, including, but not limited to, correctness and/or fitness for purpose. Issue Date: 26/08/2003

# Contents

# Preface

This document describes how to configure and use the InCharge discovery functions for network elements, both logical and physical, at Layer 2 and Layer 3.

The InCharge IP products that provide discovery include:

- InCharge IP Availability Manager
- InCharge IP Performance Manager
- InCharge Discovery Manager

InCharge Discovery Manager includes only the discovery capabilities described in this document and does not perform any availability or performance analysis.

## Intended Audience

This document is intended to be read by IT managers seeking to understand how the discovery process works, and by system administrators responsible for the administration, configuration, or use of InCharge applications that use ICMP and SNMP polling to provide discovery over IP networks.

## Prerequisites

Before performing procedures in this document, one of the InCharge IP products that provide discovery must be installed. InCharge Service Assurance Manager must also be installed because the Global Console is required to perform certain configuration tasks. For information about installing these products, see the *InCharge IP Management Suite Installation Guide* and the *InCharge Service Assurance Management Suite Installation Guide*.

# Document Organization

This document consists of the following chapters.

| | |
|---|---|
| 1. ICIM Network Concepts | Describes ICIM classes and element types and their relationships. |
| 2. Polling Used During Discovery | Describes the ICMP and SNMP polling engines used for discovery. |
| 3. Understanding the Discovery Process | Explains the discovery process, including the four phases of discovery. |
| 4. Overview of the Domain Manager Administration Console | Describes the basic layout of the Domain Manager Administration Console. |
| 5. Methods for Adding Topology | Explains how to configure and initiate autodiscovery, manual discovery, and topology import. |
| 6. Understanding Discovery Results | Describes errors that can occur during discovery and how to resolve them. |
| 7. Working with the Managed Topology | Describes functions available through the Domain Manager Administration Console for managing topology. |
| 8. Discovery Configuration Files | Describes how to edit and use configuration files associated with discovery. |
| A. Wildcards | Describes the wildcards used to create matching patterns. |

**Table 1:** Document Organization

# Documentation Conventions

Several conventions may be used in this document as shown in Table 2.

| Convention | Explanation |
|---|---|
| `sample code` | Indicates code fragments and examples in Courier font |
| **keyword** | Indicates commands, keywords, literals, and operators in bold |
| % | Indicates C shell prompt |
| # | Indicates C shell superuser prompt |

| CONVENTION | EXPLANATION |
|---|---|
| <parameter> | Indicates a user-supplied value or a list of non-terminal items in angle brackets |
| [option] | Indicates optional terms in brackets |
| */InCharge* | Indicates directory path names in italics |
| ***yourDomain*** | Indicates a user-specific or user-supplied value in bold, italics |
| *File > Open* | Indicates a menu path in italics |
| ▼▲ | Indicates a command that is formatted so that it wraps over one or more lines. The command must be typed as one line. |

**Table 2:** Documentation Conventions

Directory path names are shown with forward slashes (/). Users of the Windows operating systems should substitute back slashes (\) for forward slashes.

Also, if there are figures illustrating consoles in this document, they represent the consoles as they appear in Windows. Under UNIX, the consoles appear with slight differences. For example, in views that display items in a tree hierarchy such as the Topology Browser, a plus sign displays for Windows and an open circle displays for UNIX.

Finally, unless otherwise specified, the term InCharge Manager is used to refer to InCharge programs such as Domain Managers, Global Managers, and adapters.

# InCharge Installation Directory

In this document, the term ***BASEDIR*** represents the location where InCharge software is installed.

- For UNIX, this location is: */opt/InCharge<n>/<productsuite>*.
- For Windows, this location is: *C:\InCharge<n>\<productsuite>*.

The *<n>* represents the InCharge software platform version number. The *<productsuite>* represents the InCharge product suite that the product is part of.

Table 3 defines the *<productsuite>* directory for each InCharge product.

| PRODUCT SUITE | INCLUDES THESE PRODUCTS | DIRECTORY |
|---|---|---|
| InCharge IP Management Suite | • IP Availability Manager<br>• IP Performance Manager<br>• IP Discovery Manager<br>• InCharge Adapter for HP OpenView NNM<br>• InCharge Adapter for IBM/Tivoli NetView | /IP |
| InCharge Service Assurance Management Suite | • Service Assurance Manager<br>• Global Console<br>• Business Dashboard<br>• Business Impact Manager<br>• Report Manager<br>• SAM Failover System<br>• Notification Adapters<br>• Adapter Platform<br>• SQL Data Interface Adapter<br>• SNMP Trap Adapter<br>• Syslog Adapter<br>• XML Adapter<br>• InCharge Adapter for Remedy<br>• InCharge Adapter for TIBCO Rendezvous<br>• InCharge Adapter for Concord eHealth<br>• InCharge Adapter for InfoVista<br>• InCharge Adapter for NetIQ AppManager | /SAM |
| InCharge Application Management Suite | • Application Services Manager<br>• Beacon for WebSphere<br>• Application Connectivity Monitor | /APP |
| InCharge Security Infrastructure Management Suite | • Security Infrastructure Manager<br>• Firewall Performance Manager<br>• InCharge Adapter for Check Point/Nokia<br>• InCharge Adapter for Cisco Security | /SIM |
| InCharge Software Development Kit | • Software Development Kit | /SDK |

**Table 3: Product Suite Directory for InCharge Products**

For example, on UNIX operating systems, InCharge IP Availability Manager is, by default, installed to */opt/InCharge6/IP/smarts*. This location is referred to as ***BASEDIR****/smarts*.

Optionally, you can specify the root of ***BASEDIR*** to be something other than */opt/InCharge6* (on UNIX) or *C:\InCharge6* (on Windows), but you cannot change the *<productsuite>* location under the root directory.

For more information about the directory structure of InCharge software, refer to the *InCharge System Administration Guide*.

# Additional Resources

In addition to this manual, SMARTS provides the following resources.

## InCharge Commands

Descriptions of InCharge commands are available as HTML pages. The *index.html* file, which provides an index to the various commands, is located in the ***BASEDIR****/smarts/doc/html/usage* directory.

## Documentation

Readers of this manual may find other SMARTS documentation (also available in the ***BASEDIR****/smarts/doc/pdf* directory) helpful.

### InCharge Documentation

The following SMARTS documents are product independent and thus relevant to users of all InCharge products:

- *InCharge Release Notes*
- *InCharge Documentation Roadmap*
- *InCharge System Administration Guide*
- *InCharge ICIM Reference*
- *InCharge ASL Reference Guide*
- *InCharge Perl Reference Guide*

### InCharge IP Management Documentation

The following SMARTS documents are relevant to users of the InCharge IP Management product suite.

- *InCharge IP Management Suite Installation Guide*
- *InCharge IP Deployment Guide*
- *InCharge IP Discovery Guide*
- *InCharge IP Availability Manager User's Guide*
- *InCharge IP Performance Manager User's Guide*
- *InCharge IP Adapters User's Guide*

## Common Abbreviations and Acronyms

The following lists common abbreviations and acronyms that are used in the InCharge guides.

| | |
|---|---|
| ASL | Adapter Scripting Language |
| CDP | Cisco Discovery Protocol |
| ICIM | InCharge Common Information Model |
| ICMP | Internet Control Message Protocol |
| IP | Internet Protocol |
| MSFC | Multilayer Switch Feature Card |
| MIB | Management Information Base |
| MODEL | Managed Object Definition Language |
| RSFC | Router Switch Feature Card |
| RSM | Router Switch Module |
| SNMP | Simple Network Management Protocol |
| TCP | Transmission Control Protocol |
| VLAN | Virtual Local Area Network |

# Technical Support

SMARTS provides technical support by e-mail or phone during normal business hours (8:00 A.M.—6:00 P.M. U.S. Eastern and Greenwich Mean Time). In addition, SMARTS offers the InCharge Express self-service web tool. The web tool allows customers to access a personalized web page and view, modify, or create help/trouble/support tickets. To access the self-service web tool, point your browser to:

*https://websupport.smarts.com/SelfService/smarts/en-us*

### U.S.A Technical Support

E-Mail: *support@smarts.com*
Phone: +1.914.798.8600

### EMEA Technical Support

E-Mail: *support-emea@smarts.com*
Phone: +44 (0) 1753.878140

### Asia-Pac Technical Support

E-Mail: *support-asiapac@smarts.com*

You may also contact SMARTS at:

| | U.S.A World Headquarters | United Kingdom |
|---|---|---|
| Address | SMARTS<br>44 South Broadway<br>White Plains, New York 10601 U.S.A | SMARTS<br>Gainsborough House<br>17-23 High Street<br>Slough<br>Berkshire SL1 1DY<br>United Kingdom |
| Phone | +1.914.948.6200 | +44 (0)1753.878110 |
| Fax | +1.914.948.6270 | +44 (0)1753.878111 |

For sales inquiries, contact SMARTS Sales at:
*sales@smarts.com*.

SMARTS is on the World Wide Web at:
*http://www.smarts.com*

# 1

# ICIM Network Concepts

This chapter describes the InCharge Common Information Model (ICIM) hierarchy and the topology elements created by InCharge discovery. It also provides several topology examples.

## The Two ICIM Hierarchies

An essential feature of ICIM is that it describes the managed network in terms of two parallel, but distinct, hierarchies: physical and logical. The former describes the real-world components that make up your managed system. A physical component is something to which you could attach an inventory tag.

A logical component, on the other hand, is visible through network protocols, provides some network service, or connects logical components. It is created through the operation of hardware and software—you cannot touch it.

### The Physical Hierarchy

The physical hierarchy consists of subclasses of the class ICIM_PhysicalElement, which describe any elements of a system that have distinct physical identities. All current ICIM physical elements are actually drawn from subclasses of ICIM_PhysicalPackage, which represents physical elements that can contain or host other physical elements.

The most common class in the physical hierarchy is Card. Cards are used to model any element capable of carrying signals or providing a mounting point for other physical components. Other classes you will see include Chassis, which represents the physical elements that enclose other elements and provide some definable functionality, and Rack, which represents an enclosure in which Chassis are placed.

The most basic relationship in the physical hierarchy is ComposedOf, which relates a physical object to its component parts (which in turn are PartOf the enclosing object). A Chassis is ComposedOf the Cards placed in it; a Card used as a motherboard may in turn be ComposedOf Cards plugged into it; and so on.

## The Logical Hierarchy

The logical hierarchy consists of subclasses of the class ICIM_LogicalElement. These describe abstract system components, ranging from ProtocolEndpoints and LogicalLinks to logical devices. Logical components are creations of software; they exist only as long as the software implementing their capabilities continues to run.

### Computer Systems

One important type of logical element is the computer system. Important classes of the computer system include the Host (a workstation or server) and the following relay devices: Hub, Switch, Bridge, and Router. The relay devices forward packets at Layer 1 (a Hub), Layer 2 (a Switch or Bridge), or Layer 3 (a Router) in a network.

### Network Adapters

Network adapters are elements that represent Layer 2 connection points. Two subclasses are common: Interfaces, such as Ethernet interfaces that connect routers to an Ethernet; and Ports, which connect Interfaces to Layer 2 devices. An Ethernet connection to an Ethernet Switch is an example of a Port.

### Service Access Points

A service access point describes a logical endpoint that can be used to gain access to some network service. Important service access points are IP, which represents the point at which an IP address can be reached; and Media Access Control (MAC), which similarly represents the point at which a Layer 2 address can be reached. Layer 3 service access points are often described as logical nodes.

A service access point is related to some lower level network object by the LayeredOver relationship. Thus, an IP address would be LayeredOver an Interface.

### Logical Links

Logical links model the paths that connect logical nodes to each other. The most common logical link class is IPNetwork, which represents an IP (sub)network. ConnectedVia describes the relationship between a logical node and a logical link. Two IP addresses can communicate directly when they are ConnectedVia the same IPNetwork. A connection between two IP addresses that are not on the same IPNetwork must go through one or more Routers: IP ConnectedVia IPNetwork; Interface HostedBy Router; Router HostsAccessPoints include IP and MAC addresses.

An example of a Layer 2 logical link is a VLAN.

### Connections

A network device connection represents the path between a pair of network adapters. The most important examples are Cables, TrunkCables, and NetworkConnections.

ConnectedVia plays a role for network device connections that is similar to its role for logical links. A network relay device such as a Switch can be connected via a Cable to another relay device such as a Router, or via a TrunkCable to another relay device such as a Switch. Two Routers joined by a virtual circuit whose intermediate devices are not included in the topology (for example, a Frame Relay PVC) are connected by a NetworkConnection.

## Joining the Two Hierarchies

The physical and logical hierarchies are not independent: The elements of the physical hierarchy exist exactly to provide implementations for logical devices that implement features of interest in the network. The Realizes relationship joins a physical element to the logical devices for which it acts as the physical embodiment or implementation.

# Element Types

Discovery creates elements and defines the dependencies between elements according to the data model specified by ICIM.

ICIM classifies network elements according to the following categories:

- Unitary computer systems and management agents
- Network adapters and network connections
- Chassis and cards
- Protocol endpoints and logical links
- Redundancy groups

## Unitary Computer Systems and Management Agents

A unitary computer system contains cards and network adapters, and hosts protocol endpoints and management agents. As a logical entity it may represent the abstraction of a physical device such as a switch or a logical entity such as a virtual router within a MultiProtocol Label Switching (MPLS) Virtual Private Network (VPN) router.

InCharge discovers the following system elements:

- *Bridge* — A bridge is a protocol-independent network element that connects two LAN segments.
- *Firewall* — A firewall is a device that controls the flow of traffic between networks.
- *Host* — A host is a general purpose computer, such as a workstation or server.
- *Hub* — A hub is a device that connects multiple physical segments. Active hubs are multi-port repeaters, which means that they repeat signals received on any port to all the other ports.
- *Multilayer Switch Feature Card (MSFC)* — A card in a switch that performs routing between VLANs.
- *Node* — A node identifies a system that is monitored using generic network management instrumentation. Nodes are probed for standard MIB-II information but not for enterprise-specific information such as device resources.

- *Probe* — A probe is a system that monitors networks or other systems. An example is a Remote Monitoring (RMON) probe.
- *Router* — A router is a device or, in some cases, software in a computer that determines the next network point to which a packet should be forwarded as it travels toward its destination. A router is connected to at least two networks and decides which way to send each information packet based on its current understanding of the state of the networks to which the router is connected.

  A router may also be a *virtual router*, which is a software emulation of a router implemented within a physical router or switch. Each virtual router has its own independent IP routing and forwarding tables, which permit the same routing and forwarding of packets as with a standard router. Virtual routers are often used with VPNs to allow a greater separation of VPN traffic while using the same equipment.
- *Router Switch Feature Card (RSFC)* — A card in a Catalyst switch that runs Cisco IOS router software and is used to perform routing between VLANs.
- *Router Switch Module (RSM)* — An RSM is a router installed as a card in a switch to perform routing between VLANs.
- *Switch* — A switch is a network element that switches packets, typically at wire speeds, between physically separate network segments.
- *Terminal Server* — A specialized system that connects terminals to a network.
- *Unsupported* — An unsupported element identifies a system that is not supported. This class is no longer used but is retained for backward compatibility.

A unitary computer system may be associated with the physical package, chassis or card, in which it is housed. An example is shown in Figure 1 where a router is packaged in a chassis. The chassis packages a system, the router, and the router hosts a management agent and is composed of network adapters.

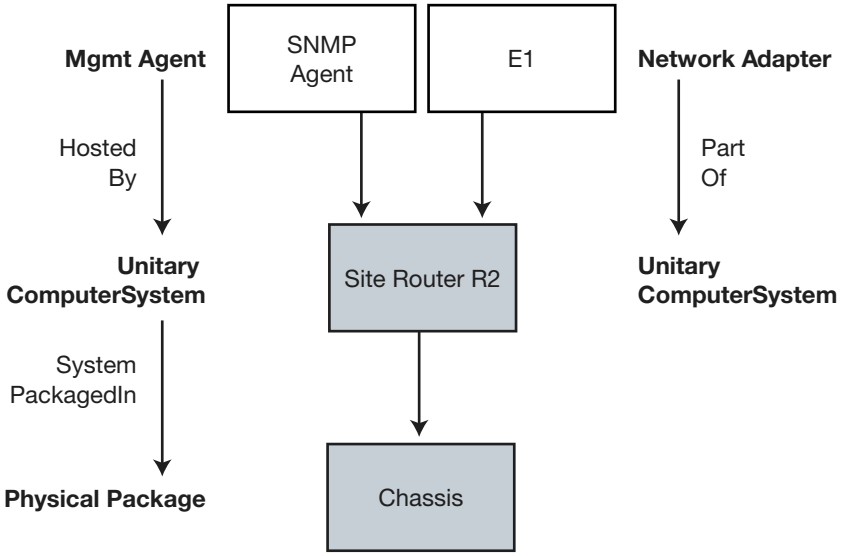


**Figure 1:** Packaging of a System

A management agent is an application that provides monitoring and other management functions for a unitary computer system. The discovery process creates instances of management agents because they are important sources of information.

## Network Adapters and Network Connections

A network adapter represents a connection point such as an Ethernet interface on a host or an access port on a switch. Instances of the Port or Interface class can represent a physical interface or port, one with a connector to which you attach a cable, or a logical adapter such as a Frame Relay sub-interface. In the case of a physical interface or port, the network adapter is a logical abstraction of the physical adapter.

InCharge discovers the following network adapter elements:

- *Port* – A port is where the physical connection to a network segment is made. A port may have a Media Access Control (MAC) address layered over it.
- *Interface* – An interface is where the physical connection to a network is made. An interface may have a MAC address, an IP address, or both layered over it.

- *Sub-interface* – A sub-interface is a logical division of a physical interface. A physical interface can be divided into one or more sub-interfaces. For example, in a typical Frame Relay network, a physical interface is configured with multiple virtual circuits called data link connection identifier (DLCI) interfaces, and each virtual circuit is associated with a sub-interface.

**Note:** Sub-interface elements, similar to interfaces, are instances of the Interface class.

A network adapter may be layered over another network adapter. The layering represents (models) the dependency between network adapters at different protocol layers. For example, in Figure 2, the Frame Relay sub-interfaces S.1 and S.2 are layered over the physical interface S.

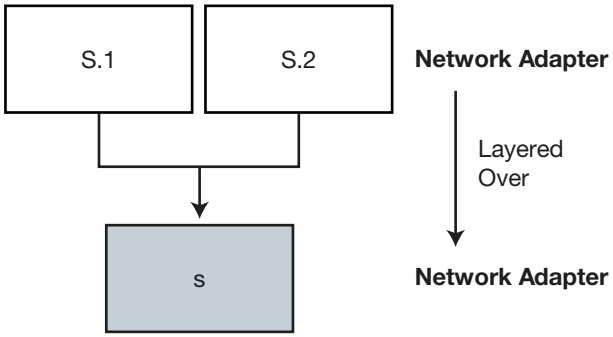


**Figure 2:** Layering of Network Adapters

Another example of layered interfaces is a channelized DS1 interface where each channel is an interface, and each channel is layered over the DS1 interface.

A network adapter is also classified according to its intended use, indicated by the Mode attribute. The mode can be one of normal, on-demand, or backup. A value of *normal* describes an active and operational network adapter, such as those of type Ethernet or serial. An *on-demand* adapter becomes active only when it is needed. An example of such an adapter is a PPP or SLIP interface that hangs up when idle to reduce dialup charges. Such an adapter fluctuates between up and down under normal operating conditions. A *backup* adapter is one that activates only when there is a failure on a primary link. Backup adapters are inactive under normal operating conditions. An example of a backup adapter is an ISDN adapter where the physical interface remains up but the logical adapters layered over the physical interface are down until the link is activated.

A network adapter attaches to the network through a *network connection*. The connection may be the abstraction of a physical entity such as a cable, or it may represent a purely logical entity such as a virtual connection over a Wide Area Network. A network connection is typically attached to two network adapters, but ICIM does not preclude it from attaching to more. An example might be a point-to-multipoint virtual connection.

**Note:** InCharge discovers and monitors both point-to-point and point-to-multipoint Frame Relay virtual circuit connections.

InCharge discovers the following network connection elements:

- *Cable* – A cable is a connection between a port and an interface.
- *Trunk Cable* – A trunk cable is a connection between two ports.
- *Network Connection* – A network connection is a connection between two interfaces. A network connection can be a logical connection or a physical connection. An example of a logical connection is when routers are connected via a virtual circuit and none of the intermediate network devices are included in the topology. An example of a physical connection is when routers are connected via a serial or point-to-point connection.

Network connections may exist at different network adapter layers. Consider the case of a Frame Relay interface on an access router, as shown in Figure 3. The connection at interface S represents the physical connection to the Frame Relay access device, while the connection at sub-interface S.2 is a virtual connection across the Wide Area Network to the peer sub-interface at the remote site.

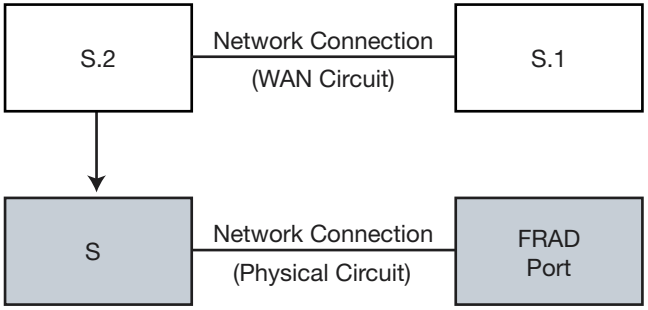


**Figure 3:** Layering of Network Connections

Network connections may be layered over other network connections. This enables the discovery process to build a topological representation of the complex dependencies between connections at different layers. For example, the Wide Area Network circuit connection from Figure 3 could be layered over the various logical and physical connections it traverses within the Wide Area Network.

## Chassis and Cards

A card is a physical element that plugs into a chassis. (See Figure 4.) A chassis is also a physical element. Card and chassis are modeled as physical elements (rather than as logical abstractions of physical elements) because there are aspects of the physical manifestation that are useful to represent, namely asset information such as the serial number. A card is associated with one or more network adapters that are realized by the card. A card may also be part of another card as in the case of a daughter card.

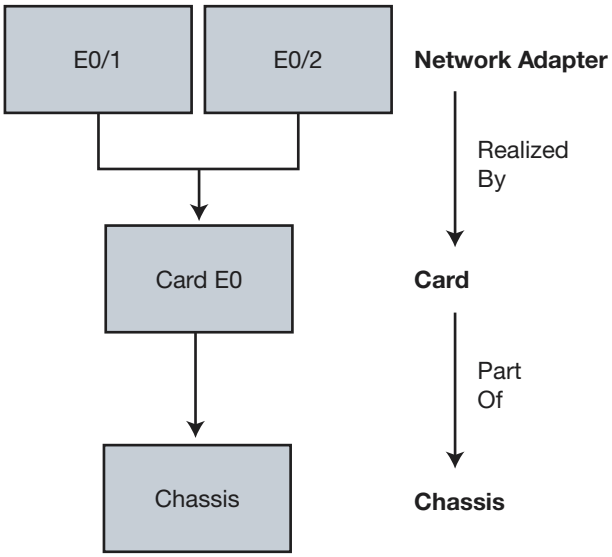


**Figure 4:** Chassis, Card, and Network Adapters

## Protocol Endpoints and Logical Links

A protocol endpoint is a higher layer service access point such as an IP endpoint, a LAN endpoint, or a PVC endpoint. A protocol endpoint is distinguished by a layer-specific protocol address. An IP endpoint is identified by its IP address, a LAN endpoint by its Media Access Control (MAC) address, a PVC endpoint by its DLCI number. Protocol endpoints may be layered over

network adapters or other protocol endpoints and hosted by systems. (See Figure 5.) An IP endpoint on a LAN device will usually be layered over a network adapter and a MAC endpoint. This represents the dependency between the IP layer and the data link layer and the dependency between the IP layer and the interface through which it is accessed.

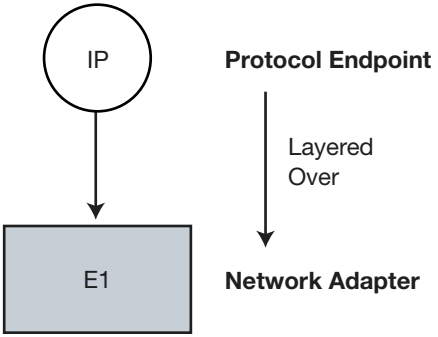


**Figure 5:** IP Protocol Endpoint Layered Over a Network Adapter

Protocol endpoints connect to logical links. A logical link is a purely logical connection at the layer of the protocol endpoint. An IP network is a layer 3 logical link, a VLAN is a layer 2 logical link, and a PVC is a layer 2 logical link connecting its DLCI endpoints. Logical links may be layered over other logical links or network connections.

## Redundancy Groups

A redundancy group is a logical entity that contains two or more elements that participate in a redundant configuration. For example, a remote site accessed through two routers may be modeled as a redundancy group that contains the two routers. If one router fails then the redundancy is at risk. If both fail then all capability is lost. ICIM supports redundancy groups of network adapters, network connections, cards, and unitary computer systems.

For information about creating redundancy groups, see the *InCharge IP Availability Manager User's Guide*.

## HSRP Groups

An HSRP group is a redundancy group of routers using a proprietary protocol from Cisco called Hot Standby Routing Protocol. The group provides router backup in the event of a router device failure or an interface failure.

In an HSRP group, several routers connected to the same segment of an Ethernet, FDDI, or token-ring network work together to present the appearance of a single virtual router on the LAN. Because the routers share the same IP and MAC addresses, the hosts on the LAN are able to continue forwarding packets to a consistent IP and MAC address in the event of a router device failure. The process of transferring the routing responsibilities from one device to another is transparent to the user.

For information about creating HSRP groups, see the *InCharge IP Availability Manager User's Guide*.

# Topology Examples

The following topology examples combine some of the elements previously described and include the relationships between the different elements. All of these elements and relationships are created automatically by the discovery process.

## Host Connected to a Switch

The first example, shown in Figure 6, shows all the basic elements and relationships created to represent a connection between two elements: a single-homed host (H1) and a switch (S1).

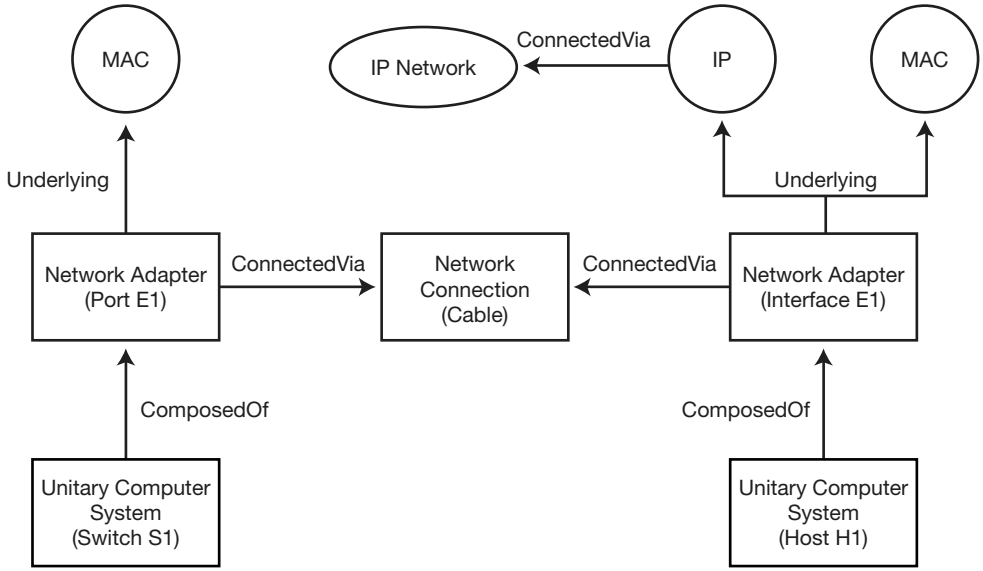


**Figure 6:** Topology of a Connection Between a Host and Switch

The interface on the host and the port on the switch are both network adapters. Both have an associated MAC endpoint, and the interface has an associated IP endpoint logically connected to an IP subnet. The network connection between the two network adapters is modeled as a cable, which, in ICIM, represents a connection between an interface and a port.

## Wide Area Network

The second example, shown in Figure 7, shows a Wide Area Network connection between Hub Router H1 and Site Router R2. Typically, such a topology would include additional sub-interfaces, which are not shown to simplify the diagram.

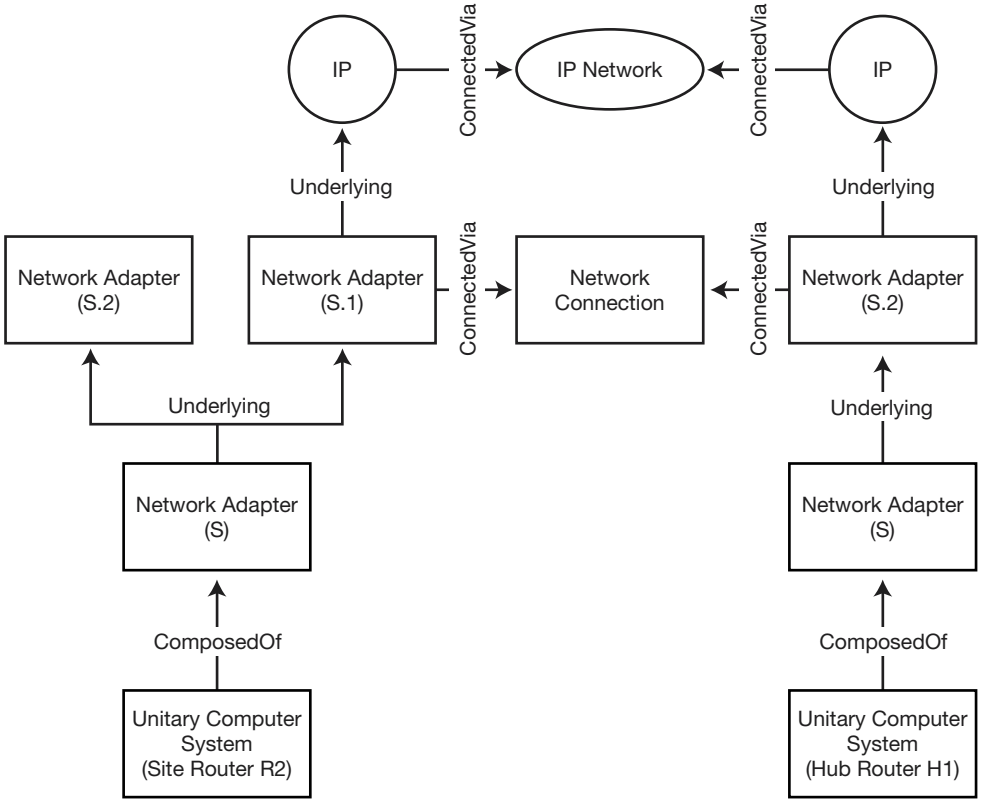


**Figure 7:** Topology of a Connection Over a Wide Area Network

This topology example illustrates the layering between the physical interface (S) and the sub-interfaces (S.1 and S.2). The sub-interfaces have associated IP endpoints while the physical interface does not. This example also shows the connectivity at the different layers: the network connection represents a layer 2 circuit connection between sub-interfaces, and the IP network represents the point-to-point IP subnet riding on the circuit. The IP layer would be omitted for an unnumbered connection.

## Switch With Routing Module

The last example, shown in Figure 8, shows a switch with a routing blade. It illustrates how ICIM represents (models) both the physical and logical elements of a network and creates relationships between them. The physical elements include the switch chassis and router switch module, while the logical systems include the switch, router, and logical interface.

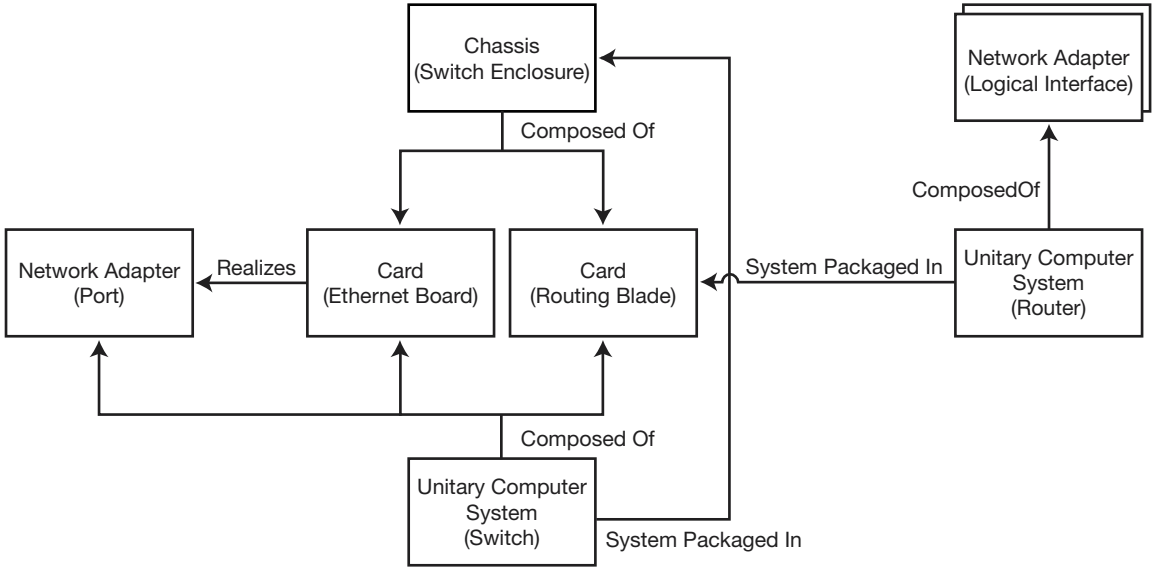


**Figure 8:** Topology of Physical and Logical Elements

# 2

# Polling Used During Discovery

InCharge uses the Internet Control Message Protocol (ICMP) and the Simple Network Management Protocol (SNMP) to poll elements. Polling is performed for two distinct purposes: discovery, including the initial device discovery, and correlation analysis.

## What Is ICMP?

Internet Control Message Protocol is tightly integrated with IP. ICMP messages, delivered in IP packets, are used for out-of-band messages related to network operation. Some of ICMP's functions are to announce network errors, announce network congestion, announce timeouts, and to assist in troubleshooting network problems.

ICMP supports an Echo function that sends a packet on a round–trip between two hosts. Ping, which is a common network management tool based on the Echo function, transmits a series of packets to measure average round–trip times and to compute loss percentages. Ping is central to InCharge ICMP polling.

# What Is SNMP?

The Simple Network Management Protocol is used by network management applications for discovery, fault management, and performance management. SNMP is an application-layer protocol that exchanges management information between a network management application, such as InCharge IP Availability Manager, and one or more managed network elements, called agents.

SNMP, itself, does not provide network management. What SNMP does provide is a framework, or infrastructure, on which network management applications can be built.

## SNMP Key Components

The following figure illustrates the SNMP key components: management station, agent, management information base, and SNMP protocol.

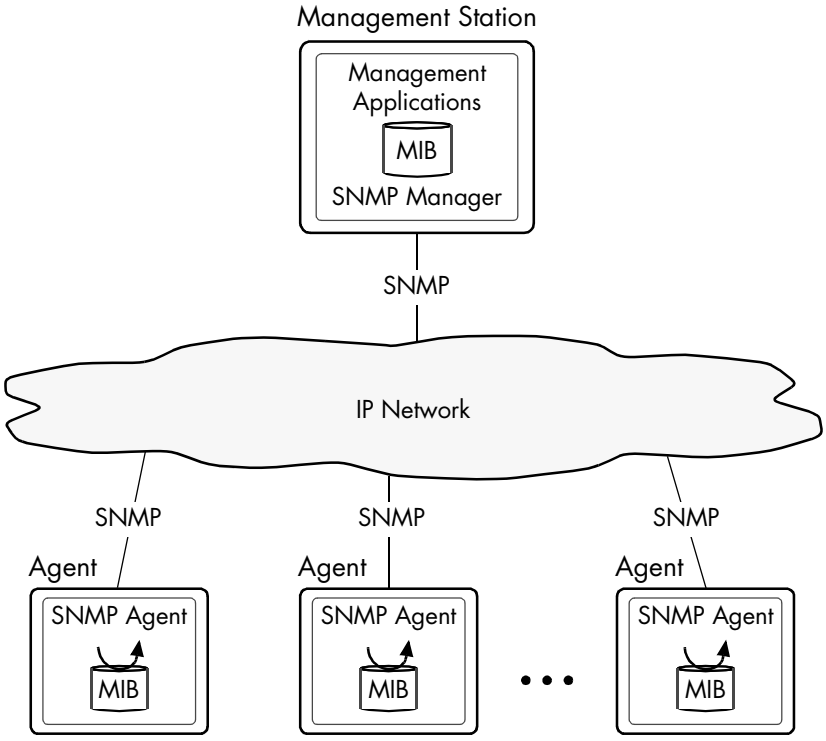


**Figure 9:** SNMP-Managed Network

Each SNMP agent or SNMP manager maintains a MIB. An agent MIB contains information about the network device managed by the SNMP agent and about the agent itself. A manager MIB contains network management information extracted by the SNMP manager from each agent MIB.

## SNMP Basic Operations

SNMP includes the following basic operations:

- GET: enables the SNMP manager to retrieve (read) the value of MIB objects at the SNMP agent.
- SET: enables the SNMP manager to set (write) the value of MIB objects at the SNMP agent.
- NOTIFY: enables an SNMP agent to asynchronously notify the SNMP manager of significant but unsolicited events (traps).
- INFORM: enables an SNMP manager to asynchronously send an alert (similar to a trap) to another SNMP manager.

## SNMP Versions

Currently, there are three versions of the SNMP protocol: V1, V2C, and V3. SNMP V1 and V2C offer rudimentary authentication and authorization schemes, whereas SNMP V3 offers robust authentication and authorization schemes plus data privacy.

### SNMP V1 Security

SNMP V1 security consists of a pairing of an SNMP agent with some arbitrary set of SNMP managers to form an SNMP community. Each SNMP community is given a name called the community name or community string for the community. Common community names are *public* (read-only), *private* (read-write), and *trap*. The permissions (read-only, read-write) for a community name indicate the read-write permissions of an SNMP manager when using that community name to access an agent's MIB. Community names can be thought of as passwords to SNMP V1 agents.

### SNMP V2C Security

Like SNMP V1, SNMP V2C uses the notion of communities to establish trust between SNMP managers and SNMP agents. Unlike SNMP V1, SNMP V2C uses an updated version of the *structure of management information* (SMI), which extends the MIB object tree, allows several new data types, and makes a number of other changes. SNMP V2C is more efficient than SNMP V1 and has better error-handling capabilities.

### SNMP V3 Security

SNMP V3 provides integrity, authenticity, data privacy, and access control for SNMP messages exchanged between an SNMP manager and the managed SNMP agents. Unlike the community-based administrative model of SNMP V1 and V2C, SNMP V3 unambiguously identifies the source and destination of each SNMP message. (An SNMP V3 manager or agent may have multiple party—user—identities.) And instead of using community names to establish trust between SNMP managers and SNMP agents, SNMP V3 uses the following security-related services to establish trust:

- Authentication

  The source includes information in each sent message that identifies the source as authentic, and performs the required functions to ensure message integrity. A typical authentication scheme requires that the source and destination parties share the same authentication key.

- Privacy

  Messages are encrypted to achieve privacy. The encryption is done in such a way that only the intended destination can perform the decryption. A typical privacy scheme requires that the source and destination parties share the same privacy key.

- Access control

  Both the source and destination play a part in access control. Each destination may have a distinct access policy for each potential source, which gives an administrator considerable flexibility in setting up an SNMP management system and assigning various levels of authorization to different users.

## InCharge SNMP Version Support

The InCharge IP products fully support SNMP V1 and V2C. They also provide the following support for SNMP V3:

- Support SNMP V3 authentication and access control but not data encryption
- Support manual discovery but not autodiscovery of candidate systems having SNMP V3 agents
- Consider all received SNMP V3 traps as genuine and therefore do not authenticate V3 traps

# Controlling InCharge ICMP and SNMP Polling

The parameters for controlling InCharge ICMP and SNMP polling during discovery are contained in the *discovery.conf* file and described in *Increasing Timeout Values for Discovery Polling* on page 89 and *Description of discovery.conf* on page 105.

The parameters for controlling ICMP and SNMP polling for correlation analysis are accessed through the Polling and Thresholds Console and described in the *InCharge IP Availability Manager User's Guide* and the *InCharge IP Performance Manager User's Guide*.

# 3

# Understanding the Discovery Process

This chapter describes the InCharge discovery process and the methods available for discovery. The discovery process uses ICMP and SNMP polling to collect topology information about the managed environment. This chapter also describes how to field-certify topology elements that do not support SNMP.

## Overview of Discovery

Stated simply, InCharge discovery is the process of creating an ICIM representation of the managed network topology within a Domain Manager's repository. Data is collected by SNMP to create instances of the managed network systems (switches, routers, hubs, bridges, hosts), their dependencies, their internal components, and their connections.

When a system is added to the managed environment, InCharge performs discovery to determine the system's configuration and its relationships to other managed systems and components. When a system is removed from the managed environment and manually deleted from the InCharge topology, InCharge removes the system and all of its components from the topology.

Systems can be added to or deleted from the InCharge topology at any time. Adding new systems does not impede the ability of InCharge to monitor elements that are already managed.

# Discovery Process and Discovery Methods

Figure 10 and Figure 11 show the discovery process and the discovery methods for adding systems to the InCharge topology. The discovery methods are:

- Autodiscovery
- Manual discovery
- Topology import

## Autodiscovery

With autodiscovery (Figure 10), InCharge automatically discovers candidate systems based on the systems initially added to the InCharge topology using manual discovery. Autodiscovery is useful when topology information is incomplete or unavailable or when manual updates cannot be maintained. For information regarding autodiscovery, see *Adding Topology with Autodiscovery* on page 54.

When enabled, autodiscovery takes IP addresses obtained from discovered systems to find additional discovery candidate systems. The initial IP addresses correspond to the neighboring systems of the systems discovered using manual discovery.

For systems that match the discovery filters and are added to the topology, they too are probed for IP addresses of their neighbors. The autodiscovery cycle continues until no more new IP addresses match the discovery filters. The discovery filters, which are configurable, ensure that the InCharge IP manager only sweeps the networks that the customer systems are configured to access.

## Manual Discovery

With manual discovery (Figure 10), InCharge discovers a set of systems specified in a *seed file*, or discovers an individual system specified in an InCharge **Add Agent** command. This method proves useful when topology information is available from a well-maintained database. For information regarding manual discovery, see *Creating Topology with Manual Discovery* on page 67.

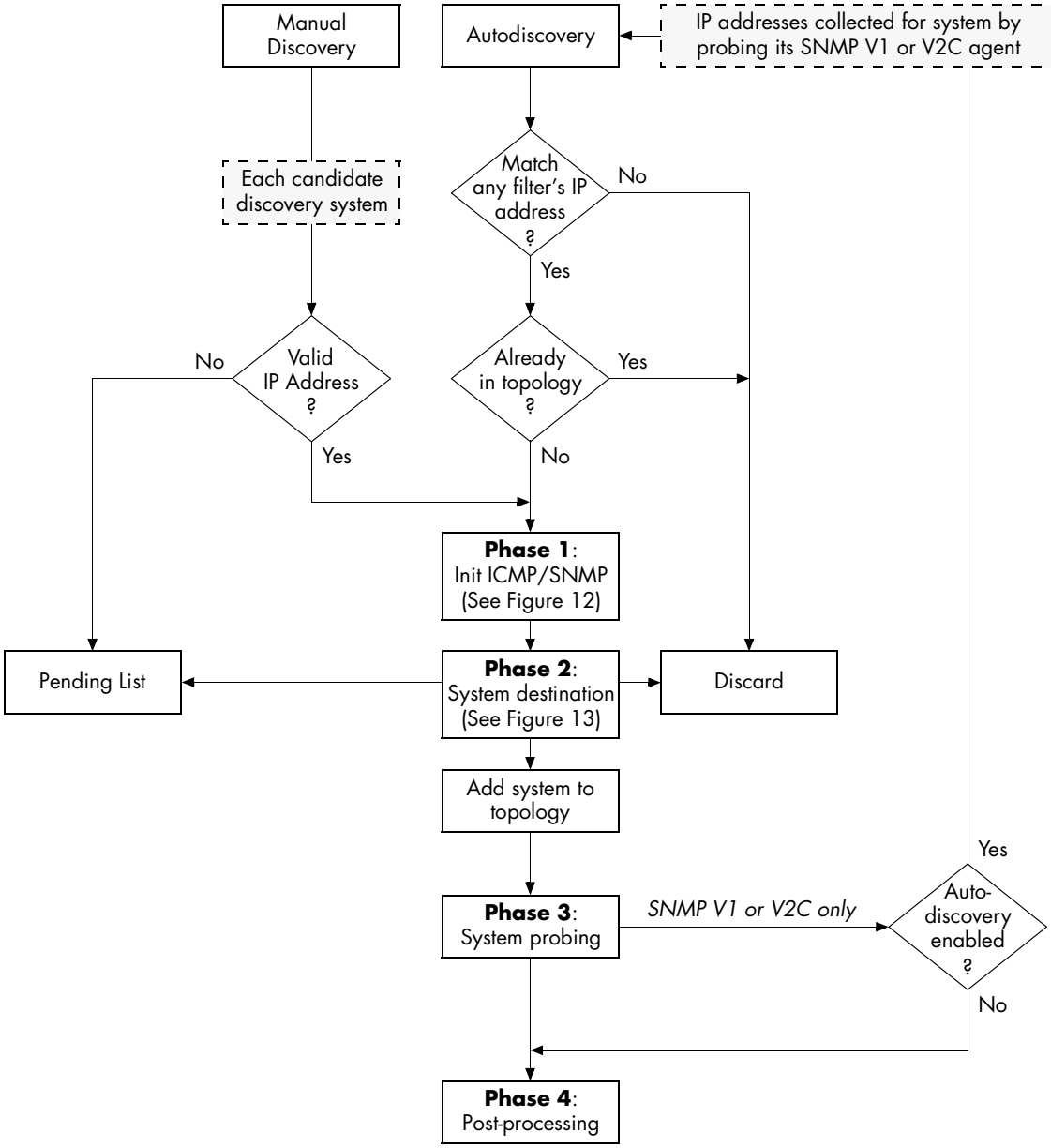


**Figure 10:** Discovery Process Using Autodiscovery and Manual Discovery

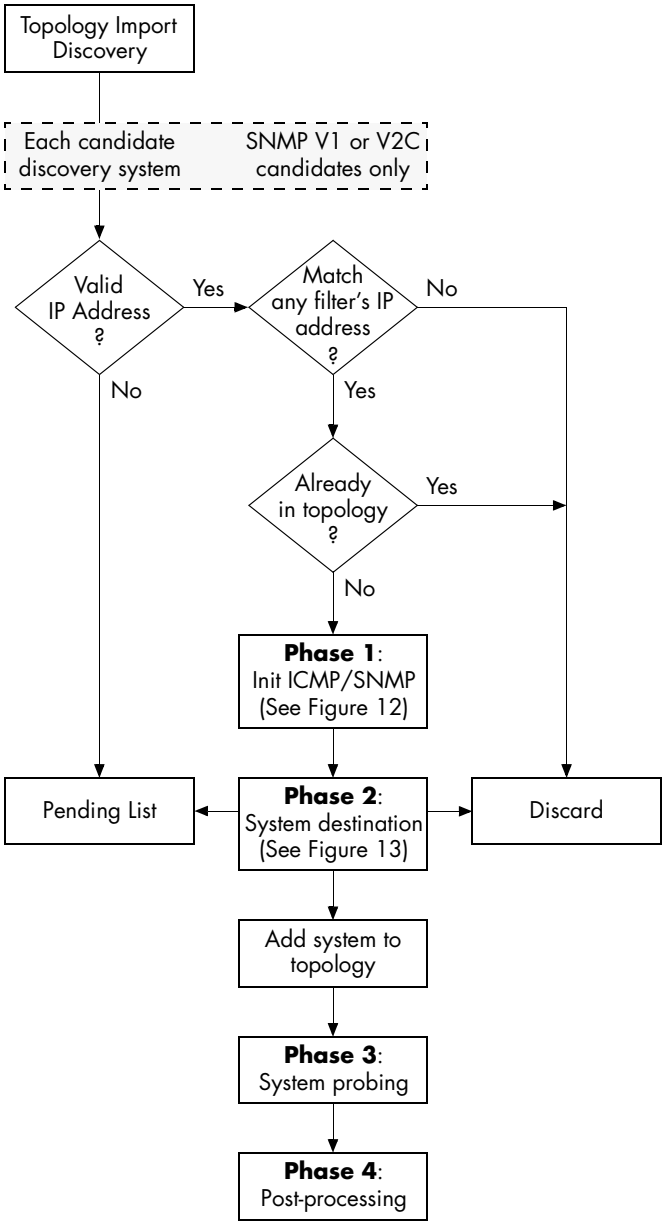


**Figure 11:** Discovery Process Using Topology Import

### Topology Import

With topology import (Figure 11), InCharge discovers a set of candidate systems imported from a third-party source through an InCharge IP adapter. InCharge IP adapters are described in the *InCharge IP Adapters User's Guide*. For information regarding topology import, see *Importing Topology* on page 76.

## Hostname Resolution and IP Address Checking

At the beginning of manual discovery (Figure 10), the discovery process checks whether a candidate system is identified by a hostname or an IP address. If the system is identified by a hostname, the discovery process attempts to resolve the hostname to one or more IP addresses. If the hostname does not resolve, the discovery process places the system, identified by its hostname, on the Pending Devices List.

If the candidate system is identified by an IP address, the discovery process checks whether the IP address conforms with the IP address format. If the IP address does not conform, the discovery process places the system, identified by its invalid IP address, on the Pending Devices List.

For information about the Pending Devices List, see *Pending Devices List* on page 37.

## Phases of Discovery

Discovery is divided into four phases, shown in Figure 10 and Figure 11, that are performed sequentially for each discovery candidate system:

1. Perform initial polling of the candidate system
2. Determine the destination of the candidate system: add to topology, place on Pending Devices List, or discard
3. Probe the system added to the topology
4. Post-process the discovery information

## Phase 1: Perform Initial Polling of the Candidate System

The first phase of the discovery process, shown in Figure 12, determines if the discovery candidate system is reachable and sends an initial SNMP request to identify the system.

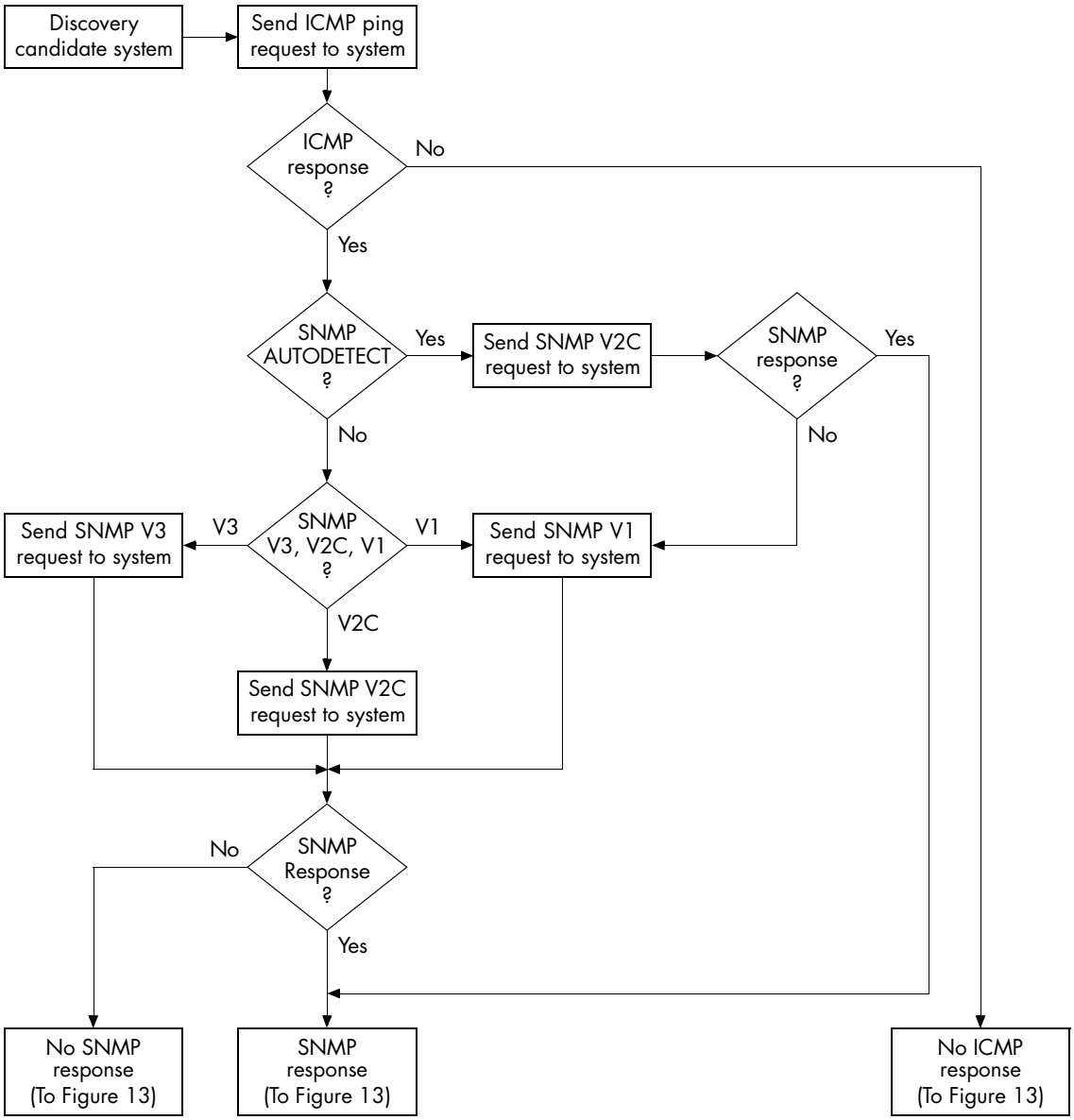


**Figure 12:** Phase 1: Initial Polling of a Candidate System

In Phase 1, the discovery process pings the IP address of the discovery candidate system to see if the address is a reachable address. If the address is reachable, the discovery process sends an SNMP request to the system requesting the following system-related information:

- sysDescr
- sysObjectID
- sysContact
- sysName
- sysLocation

By default, the discovery process sends an SNMP V2C request to the candidate system; if that request fails after the allotted number of retries, the discovery process sends an SNMP V1 request. If the SNMP version is explicitly specified (V1, V2C, V3) for the candidate system (via a discovery filter, a seed file, or an **Add Agent** command), the discovery process only sends an SNMP request using the specified version.

For an SNMP V1 or V2C request to the candidate system having multiple read community strings, the discovery process sends multiple SNMP requests to the system *simultaneously*, each containing a different community string. As described in *Adding and Removing Read Community Strings* on page 93, you can specify multiple read community strings for a candidate system.

The results of the ICMP and SNMP polling serve as input to the second phase of the discovery process.

## Phase 2: Determine the Destination of the Candidate System

The second phase of the discovery process, shown in Figure 13, determines the destination of a discovery candidate system: add to topology, place on the Pending Devices List, or discard.

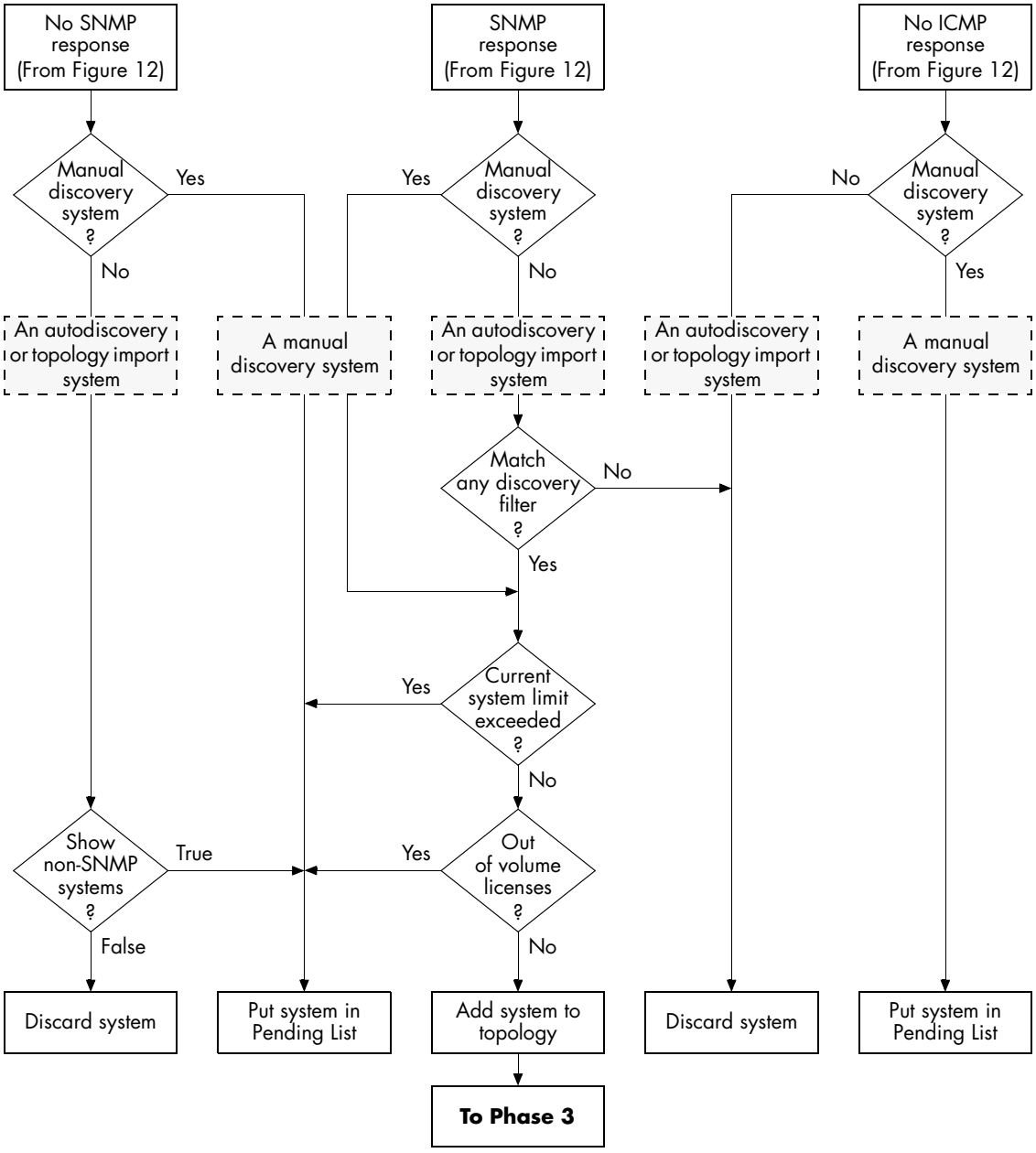


**Figure 13:** Phase 2: Determining the Destination of a Candidate System

As shown in the figure, the destination of the candidate system depends on many factors, including:

- Whether the candidate system is an autodiscovery, manual, or topology import candidate system
- Whether the system passes ICMP/SNMP polling
- Whether the system matches a discovery filter
- Whether the current system limit is exceeded
- Whether any volume licenses are available
- Whether ShowPendingNONSNMP in the ***BASEDIR***/*smarts/conf/discovery.conf* file is set to True

The following table summarizes how these factors determine the destination of a candidate system when sufficient volume licenses are available. For information about volume licenses and the OutOfLicense notification, see *Insufficient Number of Volume Licenses* on page 86.

| Candidate System | Initial Polling | | Match Filter | Current System Limit Exceeded* | ShowPendingNONSNMP | Destination |
|---|---|---|---|---|---|---|
| | ICMP | SNMP | | | | |
| Autodiscovery or topology import | Passed | Passed | Yes | No | N/A | Topology |
| | | | | Yes | N/A | Pending List |
| | | | No | N/A | N/A | Discard |
| | | Failed | N/A | N/A | True | Pending List |
| | | | | | False (default) | Discard |
| | Failed | N/A | N/A | N/A | N/A | Discard |
| Manual Discovery | Passed | Passed | N/A | No | N/A | Topology |
| | | | | Yes | N/A | Pending List |
| | | Failed | N/A | N/A | N/A | Pending List |
| | Failed | N/A | N/A | N/A | N/A | Pending List |

* 50 by default; to change the default, see *Setting a Current System Limit for the Topology* on page 62. Table assumes that sufficient volume licenses are available.

**Table 4:** Factors Affecting the Destination of a Candidate System

## Phase 3: Probe the System Destined for the Topology

During the third phase of the discovery process, each system destined for the topology is probed to gather additional information for the creation of the topology, and to gather IP addresses of neighboring systems for the autodiscovery process. Autodiscovery, if enabled, will compare each of these IP addresses against one or more *discovery filters* and add each IP address matching a filter to the discovery queue. Discovery filters are described in *Adding Topology with Autodiscovery* on page 54.

**Note:** Autodiscovery is not supported for systems having SNMP V3 agents.

The discovery process uniquely identifies the system within the topology and determines the system's level of certification. These tasks are accomplished using the following information retrieved from the initial SNMP poll: sysDescr, sysObjectID, sysContact, sysName, and sysLocation.

### Matching sysObjectID to OID in oid2type Files

The discovery process matches the sysObjectID to an OID in the *oid2type* files to identify the system's type (Switch, Router, Hub, Bridge, Host), its model, its certification level, and what discovery probes to use to probe the system. The discovery process searches for the probes in *.import* files, which are located in the same directory as the *oid2type* files: ***BASEDIR****/smarts/conf/discovery*. For more information about the *oid2type* files, and about how to add support for new systems, see *Certifying Non-Network Devices* on page 44.

### Level of Certification

Certification indicates the level of support that InCharge provides for the system.

- CERTIFIED indicates that the system has the highest level of certification and is discovered using standard MIB-II data and proprietary MIBs.
- TEMPLATE indicates that the OID is recognized but that no information is known about the MIBs that this system supports.
- GENERIC indicates that the OID is not recognized. The system is classified as GENERIC and added to the Node class. Analysis is performed using MIB-II data.

### Standard and Proprietary MIBs

The discovery process uses proprietary MIBs (also known as enterprise or vendor-specific MIBs), in addition to standard MIBs, because standard MIBs do not contain information for certain elements that InCharge is able to discover and manage. Standard MIBs, for example, do not represent the processor or memory elements necessary for performance analysis. In these cases, vendor-specific MIBs offer more detailed information.

### Discovery Probes

The discovery probes determine the configuration and connectivity of the system. Available discovery probes include:

- Name resolution probes
- Containment probes
- VLAN membership probes
- Virtual router probes
- HSRP probes
- Bridge probes
- IP network probes
- Neighbor probes
- Health probes

Some of the probes are vendor-specific. Of these probes, autodiscovery collects the hostnames and IP addresses gathered by the containment, IP network, and neighbor probes.

The probes operate in series, one right after the other. In the event that one of the discovery probes is unable to complete, the discovery process does not stop. Instead, discovery proceeds with the next probe. Events that can prevent a discovery probe from completing include the encountering of an SNMP error, the timing out of an SNMP request, and the accessing of a MIB object that is not supported by the system being probed. For more information about resolving these types of errors, see *DiscoveryError Notifications* on page 81.

### Name Resolution Probe

The Name Resolution Probe provides a unique name for the device. The process of determining a system's name is described in *How Discovery Names a System* on page 41.

### Containment Probe

The Containment Probe discovers the components of a system including ports, interfaces, IP endpoints, MAC endpoints, and any cards or modules.

The Containment Probe queries the following MIB variables: ifType, ifDescr, ifSpeed, ifMtu, ifPhysAddress, ifAlias, ifName, and other variables. If the Containment Probe has access to vendor-specific MIBs, it queries additional variables to gather additional information, such as the duplex mode (full-duplex, half-duplex) of ports and interfaces, or the data link connection identifiers for Frame Relay virtual connections. For switches or bridges, the probe may also query the dot1dBasePortIfIndex MIB variable. The Containment Probe uses standard and vendor-specific MIB information.

### VLAN Probe

The VLAN Probe collects VLAN information from switches including VLAN identifiers, VLAN trunks, and VLAN port memberships. The VLAN probe uses vendor-specific MIB information.

### Virtual Router Probe

The Virtual Router Probe discovers the virtual routers in a physical router or switch that contains virtual routers. The Virtual Router Probe discovers the following additional information: the physical card on which the virtual router resides (if such information is available) and the virtual router's interfaces and their associated physical ports. The Virtual Router Probe uses vendor-specific MIB information.

### HSRP Probe

The HSRP Probe collects information from a redundancy group of routers involved in an HSRP group; the routers communicate with one another using the proprietary Cisco Hot Standby Router Protocol. The HSRP Probe discovers the group number, the virtual IP address, and the virtual MAC address for an HSRP group; the priority number and status for each router (endpoint) participating in the group; and the currently active router in the group. The HSRP Probe uses vendor-specific MIB information.

### Bridge Probe

The Bridge Probe collects bridge-forwarding tables and Spanning Tree Protocol (STP) information. Discovery uses this information to create MAC objects and to build the bridging relationships between MAC endpoints and ports. The Bridge Probe uses standard and vendor-specific MIB information.

### IP Network Probe

The IP Network Probe collects the IP addresses and netmask for each interface on a system. Discovery uses this information to create IP Network elements and to build the connected relationships between IP endpoints and IP networks. The IP Network Probe uses standard MIB-II information.

### Neighbor Probe

The Neighbor Probe collects information about a system's neighbors. A neighboring system can be any of the following:

- A system that is directly connected
- A system or systems on the same IP network
- A system connected through a bridging relationship

The Neighbor Probe uses vendor-specific MIBs.

### Health Probe

The Health Probe collects performance-related information for system elements such as ports, interfaces, processors, memory, fans, power supplies, voltage sensors, and temperature sensors. It also collects metrics about host disks and file systems as well as additional information about host processors and memory. The Health Probe uses standard and vendor-specific MIB information.

The Health Probe is used only by InCharge IP Performance Manager and InCharge Discovery Manager. In addition, using the Health Probe to collect metrics about host disks and file systems requires *InCharge IP Server Performance Manager*, which is a specialized version of InCharge IP Performance Manager.

### Discovered SNMP Agent

In addition to discovering the system, the discovery process also discovers the system's SNMP agent and adds it to the topology. The discovery process sets the following attributes for the SNMP agent:

- Sets the AgentAddress attribute to the IP address used by the initial ICMP poll to reach the system.
- Sets the PortNumber attribute (161 by default) to the port number used by the first SNMP poll to successfully communicate with the SNMP agent.

- Sets the SNMPVersion attribute (AUTODETECT by default) to the version used by the first SNMP poll to successfully communicate with the SNMP agent.
- (SNMP V1 or V2C only): Sets the ReadCommunity attribute (*public* by default) to the read community string used by the first SNMP poll to successfully communicate with the SNMP V1 or V2C agent.
- (SNMP V3 only): Sets the following attributes to the values used by the first SNMP poll to successfully communicate with the SNMP V3 agent:

| ATTRIBUTE | DESCRIPTION |
|---|---|
| User | Name of user included in SNMP request sent to this agent. |
| AuthProtocol | Protocol used to authenticate SNMP request sent to this agent: MD5 (default), SHA, or NONE. |
| EngineID | Identifier that uniquely identifies SNMP engine of this agent; for example, 800002b804616263. |
| EngineBoots* | Number of times that SNMP engine of this agent has initialized or reinitialized since its last configuration. |
| EngineBootTime* | Date and time that SNMP engine of this agent last initialized or reinitialized. |
| * Retrieved from SNMP polling of this agent. | |

- Sets the sysDescr, sysObjectID, sysContact, sysName, and sysLocation attributes to the sysDescr, sysObjectID, sysContact, sysName, and sysLocation values retrieved from the first SNMP poll to successfully communicate with the SNMP agent.
- Sets many other SNMP agent attributes according to the values found by the discovery probes.

### Discovered VR Agents

For a physical router or switch containing virtual routers, the discovery process also discovers the VR agents for the virtual routers and adds those agents to the topology. Because VRAgent is a subclass of SNMPAgent, the attributes of a VR agent are the same as the attributes of an SNMP agent with the following exception: the VR agent has an additional attribute called RouterID.

## Phase 4: Post-Process the Discovery Information

During the final phase of discovery, the information collected from the various probes is processed to create the connections between systems in the topology. The post-processing basic functions include:

- Mapping IP and MAC access points retrieved from the ARP Cache to the appropriate systems. For non-SNMP systems, create IP and MAC access points and layer them over a physical interface. Because non-SNMP systems are discovered, discovery creates an *artificial* physical interface for each IP address associated with the system.
- Removing (pruning) MAC access points that do not belong to systems in the topology. Any MAC access point bridged by a port but not hosted by any system is removed.
- Creating connections based on bridge information, which includes creating cables between ports and interfaces, creating trunk cables between ports, and removing MAC access points that are bridged by access ports from the BridgedVia relationships of other ports.
- Creating trunk cables based on STP information.
- Creating network connections to represent WAN, or logical, connections.
- Creating connections based on discovery protocols.
- Creating any user-defined connections specified in the *user-defined-connections.conf* file located in the ***BASEDIR****/smarts/local/conf/discovery* directory. For information about user-defined connections, see *Description of user-defined-connections.conf* on page 112.
- Creating partitions, if any. For information about creating partitions, see *Description of partition.conf* on page 112.

# Specifying Additional Post-Processing Steps With ASL

In addition to the post-processing steps performed automatically during the fourth phase of the discovery process, you can specify additional post-processing steps to occur at various stages of post-processing. For example, you could use a post-processing script to manage or unmanage IP objects during discovery.

You specify an additional post-processing step by editing one of the provided ASL rule sets. Each ASL rule set is configured in such a way that it is invoked during a specific phase of the discovery. The ASL scripts are invoked at the following points during discovery:

- Before a full discovery (*custom-start-fulldisc.asl*)
- Before a system is discovered (*custom-start-system.asl*)
- After a system is discovered (*custom-end-system.asl*)
- Before discovery post-processing (*custom-start-post.asl*)
- After discovery post-processing (*custom-end-post.asl*)

The ASL scripts are installed in the ***BASEDIR****/smarts/rules/discovery/custom* directory. By default, the ASL scripts do not perform any discovery processing, meaning that you must provide the ASL code that performs the actual processing.

The following example shows how the *custom-start-system.asl* and *custom-end-system.asl* scripts identify systems for processing. The discovery process automatically assigns the name of the SNMP agent hosted by the system currently being discovered to the AgentName variable. The system scripts use the value of AgentName to obtain the name of the system by invoking the `getsystem()` operation on the SNMP agent object.

```
/*

* Passed in argument
*/
default AgentName = "";

/*
* To obtain SNMPAgent object
*/
agentObj = object();
agentObj = object("SNMPAgent", AgentName)?IGNORE;
if (agentObj->isNull()) {
    stop();
}

/*
* To obtain system object
*/
systemObj = agentObj->getSystem();
if (systemObj->isNull()) {
stop();
}

/*******************************************************/
```

```
START {
/*
* Add Parsing rules here if needed.
*/
.. eol
} do {
/*
* Add processing rules here.
*/
stop();
}
```

To open a user-modifiable ASL file, invoke the *sm_edit* utility from ***BASEDIR****/smarts/bin* as shown in the following example:

```
% ./sm_edit rules/discovery/custom/custom-start-system.asl
```

The modified ASL script is saved to the ***BASEDIR****/smarts/local/rules/discovery/custom* directory.

# Scheduled Discovery

After initial discovery, you can schedule two different types of discovery: full discovery and pending discovery.

Scheduled full discovery enables you to regularly invoke a full discovery of the existing topology. For information about scheduled full discovery, see *Full Discovery Interval* on page 97.

Scheduled pending discovery determines how often systems on the Pending Devices List are discovered. For information about scheduled pending discovery, see the following section and *Pending Discovery Interval* on page 97.

# Pending Devices List

The Pending Devices List, which is displayed in the Discovery Progress window, lists systems that are waiting to be discovered. Systems on the Pending Devices List are discovered at the next Pending Discovery Interval or when a Pending Discovery is manually invoked.

To view the Pending Devices List, select *Show Discovery Progress* from the *Topology* menu of the Domain Manager Administration Console. The following figure is an example of a Pending Devices List.

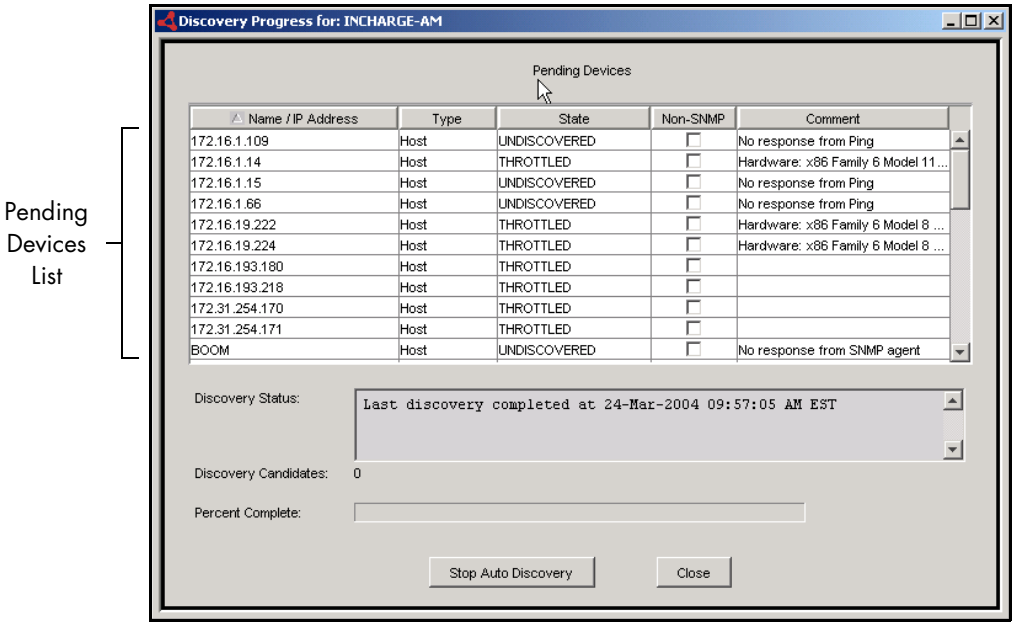


**Figure 14:** Pending Devices List Containing Multiple Pending Systems

A system is added to the Pending Devices List when one of the following occurs:

- The system does not respond to ICMP polls during a discovery.
- The system does not respond to SNMP polls during a discovery.
- The system is processed by autodiscovery and the manual accept mode is enabled. For information about manual accept mode, see *Using Manual Accept Mode* on page 61.
- The number of discovered systems exceeds the current system limit. For information about setting the current system limit, see *Setting a Current System Limit for the Topology* on page 62.
- The number of discovered systems exceeds the number of *volume licenses* available to the InCharge application. For information about volume licenses and the OutOfLicense notification, see *Insufficient Number of Volume Licenses* on page 86.

- The system does not support SNMP, meaning that the system responded to ICMP polls but that the discovery process could not contact an SNMP agent for the system. For information about systems that do not support SNMP, see *System Does Not Support SNMP* on page 84.
- InCharge receives an SNMP trap indicating that the configuration of the system might have changed. A system is placed on the Pending Devices List when one of the following traps is received:
  - ColdStart (RFC 1215 MIB)
  - WarmStart (RFC 1215 MIB)
  - Module Inserted (CISCO-STACK-MIB)
  - Module Inserted (3COM-CB9000-MIB)
- The topology is autodiscovered, or imported through one of the InCharge IP adapters described in the *InCharge IP Adapters User's Guide*, and the ShowPendingNONSNMP parameter in the *discovery.conf* file is set to True. For information about the *discovery.conf* file, see *Description of discovery.conf* on page 105.

## Information Provided by Pending Devices List

The Pending Devices List contains five columns that provide the following information for each system:

- Name or IP address of the system
- Type, which is one of the system types listed in *Unitary Computer Systems and Management Agents* on page 4.
- State is either UNDISCOVERED or THROTTLED, as described in Table 5 and in *Autodiscovery Safeguards* on page 61.
- Non-SNMP is a checkbox for indicating that the system does not support SNMP.
- Comment describes why the system is on the Pending Devices List, or contains the system's SysDescr value of the system if the current system limit has been exceeded.

Table 5 describes the discovery states and the actions you can take to resolve the situation.

| DISCOVERY STATE | DESCRIPTION | AVAILABLE ACTIONS |
|---|---|---|
| UNDISCOVERED | Indicates that the system was not successfully discovered. Another attempt to discover the system will occur when the Pending Devices List is next processed. | • Designate the system as one that does not support SNMP. The system will be re-classified as a Host during the next Pending Discovery Interval.<br>• Rediscover the system.<br>• Remove the system. |
| THROTTLED | Indicates that the system was autodiscovered with manual accept mode enabled and placed on the Pending Devices List. | • Accept the system and it will be discovered at the next full discovery, pending discovery, or rediscovery.<br>• Remove the system. |

**Table 5:** Discovery States

## Managing Systems on the Pending Devices List

Depending on a system's state, you can invoke one of the following four actions on a system on the Pending Devices List. Right-click the system to see the available actions.

- *Accept* pertains to systems in the THROTTLED state, which means that the system matched an autodiscovery filter configured for manual accept mode. If you accept the system, InCharge changes the state of the system to UNDISCOVERED.
- *Remove* is available for all systems on the Pending Devices List. Selecting this action removes the system from the Pending Devices List. If you remove a system, InCharge prompts you to confirm the removal.
- *Mark Non-SNMP* is available for all systems on the Pending Devices List. When you designate a system as Non-SNMP, InCharge reclassifies the system as a Host and adds it to the topology at the next discovery interval *assuming* that the system responds to an InCharge ICMP request. Systems marked as non-SNMP are treated similarly to systems marked as ICMPONLY.

  Once a system has been marked Non-SNMP in the Pending Devices List, you cannot change its status. You must rediscover the system manually by using the **Add Agent** command or by importing the system from a seed file.
- *Rediscover* initiates a discovery of the system.

# How Discovery Names a System

When a system is discovered, the Name Resolution Probe determines the name of a system to identify it within the topology. The name resolution process has the following goals:

- Provide a unique name for the system.
- Use a consistent naming convention to ensure that the system is assigned the same name, if, for example, it is discovered using a different IP address.
- Update the name during a rediscovery if the name of the system has changed.

The name assigned to a system is the value of the system's Name attribute. The value of the Name attribute depends on the setting of the NameFormat parameter in the *discovery.conf* file. That setting may be TM_USESEEDNAME, in which case InCharge uses the seed name to name the system, or TM_USEAUTONAME, in which case InCharge uses the name resolution service of the local operating system to name the system.

**Note:** The values of a system's Name attribute and DisplayName attribute may be different, depending on the setting of the DisplayNameFormat parameter in the *discovery.conf* file. For information about the *discovery.conf* file, see *Description of discovery.conf* on page 105.

## Using the Seed Name to Name a System

When the value of NameFormat is TM_USESEEDNAME, InCharge uses the seed name of the system for the system name. The seed name can be one of the following:

- Hostname or IP address specified in the seed file.
- Hostname or IP address specified in the Add Agent dialog box for the **Add Agent** command.
- If topology is imported from an external source (for example, through one of the InCharge IP adapters described in the *InCharge IP Adapters User's Guide*), the name provided by the external source for the system name.

For an autodiscovery candidate system, for which a seed name is not available, InCharge uses the first non-private IP address to name the system.

The range of private IP addresses is defined as follows:

- Any IP address with the value of 10 in the first octet (10.*.*.*).
- Any IP address with a value of 172 in the first octet and a value between 16 and 31 in the second octet (172.<16-31>.*.*).
- Any IP address with a value of 192 in the first octet and a value of 168 in the second octet (192.168.*.*).

If all the IP addresses for the system are private IP addresses, the first IP address *in the ipAddrTable table retrieved from the system's SNMP agent* is used as the system name.

## Using Name Resolution to Name a System

When the value of the NameFormat attribute is set to TM_USEAUTONAME (the default), InCharge uses the name resolution service (a Domain Name System server, for example) provided by the local operating system to determine the system name. Figure 15 shows the name resolution process.

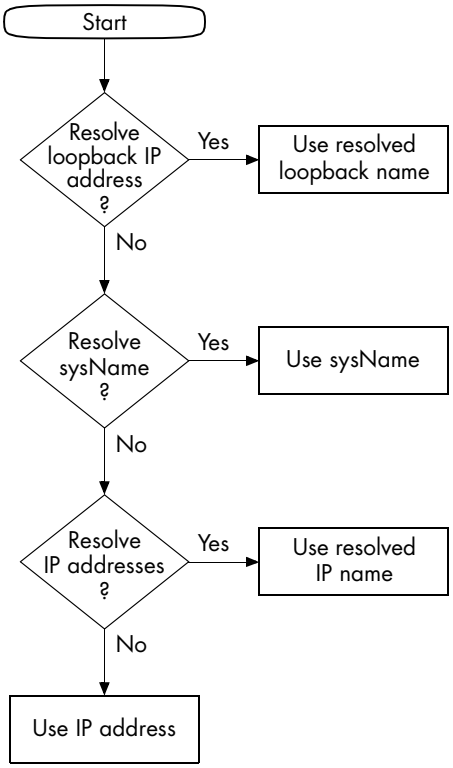


**Figure 15:** Process of Determining a System's Name

The following steps describe the system name resolution process shown in Figure 15.

1. InCharge first tries to resolve the loopback IP address to a name.

   The loopback IP address for the system is identified by the ifType MIB variable of the MIB-II interface table. When there are many loopback addresses, InCharge uses the first loopback address *in the ipAddrTable table retrieved from the system's SNMP agent* that resolves to a name.

2. If unsuccessful, InCharge then tries to resolve the sysName for the system to one or more IP addresses that match one or more IP addresses in ipAddrTable.

   If any of the IP addresses returned from the name resolution service match any of the IP addresses in ipAddrTable, InCharge uses sysName as the system name.

3. If unsuccessful, InCharge then tries to resolve the IP addresses in ipAddrTable, one by one, to a name.

**Note:** If the DNS server is not configured properly, this operation can take a long time because InCharge waits for the DNS request to timeout.

   InCharge tries to resolve the first non-private IP address in ipAddrTable to a name by performing a reverse DNS lookup (IP to name). If the non-private IP address resolves to a name, InCharge uses that name as the system name.

   The range of private IP addresses is defined as follows:

   - Any IP address with the value of 10 in the first octet (10.*.*.*).
   - Any IP address with a value of 172 in the first octet and a value between 16 and 31 in the second octet (172.<16-31>.*.*).
   - Any IP address with a value of 192 in the first octet and a value of 168 in the second octet (192.168.*.*).

   If all the IP addresses in ipAddrTable are private, InCharge tries to resolve the first private IP address in ipAddrTable to a name by performing a reverse DNS lookup. If the IP address resolves to a name, InCharge uses that name as the system name.

4. If unsuccessful, InCharge uses the private IP address as the system name.

# Certifying Non-Network Devices

After your network has been discovered, you may want to keep certain devices such as personal computers or printers in your topology. Because these are non-network devices, they are classified as GENERIC by InCharge's discovery process. You can, however, add such devices to a field certification file so that InCharge properly recognizes these devices.

For InCharge version 4.0.1 or later, you should add non-network devices to the field certification file *oid2type_Field.conf*. Any changes made to this file are not overwritten by a subsequent IDS (ICIP in 6.0 or later) package. This file is located in the ***BASEDIR****/smarts/conf/discovery* directory.

### Editing oid2Type_Field.conf

To certify non-network devices, edit this file as follows. For more information about editing configuration files using the *sm_edit* utility, see *Modifying InCharge Files* on page 104.

1. Open the *oid2Type_Field.conf* file using the *sm_edit* utility in the ***BASEDIR****/smarts/bin* directory.

   ```
   % ./sm_edit conf/discovery/oid2Type_Field.conf
   ```

2. Add the systems that you wish to certify and save the file. The modified version of the file is saved to the ***BASEDIR****/smarts/local/conf/discovery* directory.

   For information regarding the syntax of this file, see *Syntax of oid2type_Field.conf* on page 44.

3. Restart the InCharge application to make the changes take effect.

### Syntax of oid2type_Field.conf

The following example illustrates the syntax of a device entry in the *oid2type_Field.conf* file. Each entry contains the sysObjectID of the device followed by several fields that describe the device. You can use a pound sign (#) to add comments to the file.

```
# Structure of device entry
<sysObjID> {
    TYPE = Host
    VENDOR = <manufacturer>
    MODEL = [<model_name>]
    CERTIFICATION = TEMPLATE

INSTRUMENTATION:
    Interface-Fault = MIB2
    Interface-Performance = MIB2
}
```

Table 6 describes each field.

| FIELD | DESCRIPTION |
|---|---|
| TYPE | Type of system. This value should always be *Host*, regardless of the actual device type. |
| VENDOR | Manufacturer of device. This value must be a single word. |
| MODEL | Optional field that if present usually contains the model name of the device. This field can be more than one word. Note that the field name is required. |
| CERTIFICATION | Level of certification for device. This value should always be *TEMPLATE*.<br>Template indicates that the device is recognized by InCharge but that it has not been tested by SMARTS. |
| Interface-Fault | MIB used for analysis. This value should always be *MIB2*. |
| Interface-Performance | MIB used for analysis. This value should always be *MIB2*. |

**Table 6: Field Descriptions for oid2type_Field.conf**

The following example shows the type of certification information added to certify a host and a printer. Note that both devices are classified as hosts.

```
# Dell Personal Computer
.1.3.6.1.4.1.311.1.1.3.1.1 {
    TYPE = Host
    VENDOR = Dell
    MODEL = Latitude
    CERTIFICATION = TEMPLATE

INSTRUMENTATION:
    Interface-Fault = MIB2
    Interface-Performance = MIB2
}

# This is an example of a printer (TYPE = Host)
# IBM Network Printer 24
.1.3.6.1.4.1.2.3.6 {
     TYPE = Host
     VENDOR = IBM
     MODEL = Network Printer 24
     CERTIFICATION = TEMPLATE

INSTRUMENTATION:
     Interface-Fault = MIB2
     Interface-Performance = MIB2
}
```

# 4

# Overview of the Domain Manager Administration Console

This chapter briefly describes the layout and features of the Domain Manager Administration Console, including how to open the console.

## The Domain Manager Administration Console

The primary tool for configuring discovery and managing topology for an InCharge application is the Domain Manager Administration Console. You use the Domain Manager Administration Console to perform tasks such as:

- Configure and enable autodiscovery
- Add or remove elements from the managed topology
- Manage or unmanage elements in the managed topology
- Browse the managed topology
- Access the Polling and Thresholds Console

The groups and settings configured through the Polling and Thresholds Console are application-specific.

# Opening the Domain Manager Administration Console

Attaching to a Domain Manager with the Domain Manager Administration Console requires an InCharge user account with the following privileges and permissions:

- *All* privileges, specified in the *serverConnect.conf* file (or its equivalent) read by the Domain Manager.
- Permission to use the console operation *Configure Domain Manager Admin Console*. Through the Global Manager Administration Console, this permission is specified in the Console Operations section of the user profile.

For information about configuring access privileges, see the *InCharge System Administration Guide*. For information about configuring permissions to perform specific console operations, see the *InCharge Service Assurance Manager Configuration Guide*.

To open the Domain Manager Administration Console, follow these steps:

1. Attach to a Global Manager with the Global Console. The Notification Log Console opens.
2. In the Notification Log Console, select *Configure > Domain Manager Administration Console*. The Domain Manager Administration Console Console opens.

# Layout of the Domain Manager Administration Console

The Domain Manager Administration Console contains two panels. The left panel displays a topology tree that lists the attached Domain Managers and classes, or types, of managed elements.

When the Domain Manager is selected in the left panel, the right panel displays the following tabs:

- Correlation tab shows parameters related to correlation.
    - Correlation Interval determines how often InCharge correlates events to find problems.

  - Codebook Radius determines the level of noise (delayed, lost, or spurious events) that InCharge can tolerate and still guarantee correct correlation results.
  - Correlation Radius determines how exact of a match must occur between symptoms known to be caused by a problem and the symptomatic events received by InCharge.
  - Number of Problems determines how many concurrent problems that InCharge should consider possible when it correlates events.
  - Loss Symptom Probability is the likelihood that InCharge will not receive notice of a symptom that actually occurred.
  - Spurious Symptom Probability is the likelihood that InCharge will falsely receive notice that a symptom has occurred when it actually has not.
- Modules tab shows the libraries and programs that are loaded into the Domain Manager.
- Threads tab lists each running thread and its status.
- Topology tab shows the Pending Discovery Interval and provides controls for enabling regular full discoveries as well as autodiscovery. This tab is also where you specify default read community strings and set the system limit.
- Discovery Filters tab is where you specify filters for autodiscovery. At least one filter is required to enable autodiscovery.

When a class, such as Host, is selected in the left panel, the right panel displays the properties of the class under two tabs:

- Description tab provides a brief description of the class.
- Attributes tab lists attributes of the class. When a class is selected, the attributes do not contain values.

An example of a Domain Manager Administration Console is shown in Figure 16.

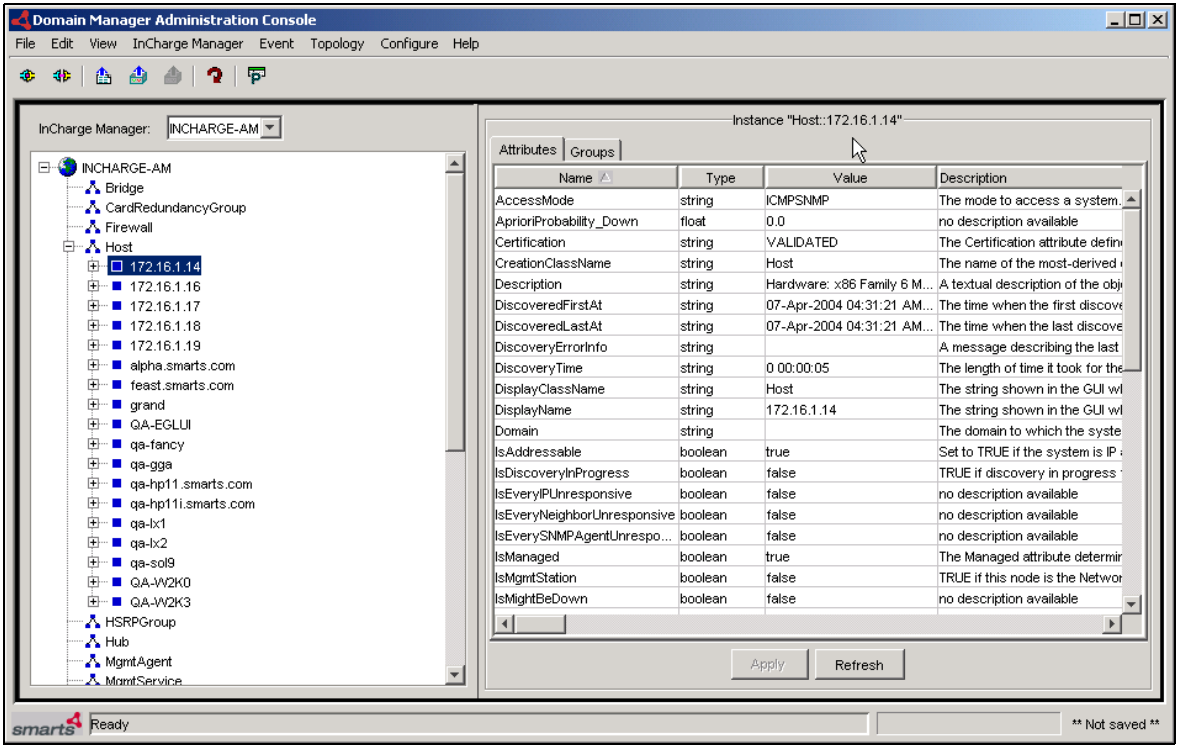


**Figure 16:** Domain Manager Administration Console

In addition, you can expand the classes listed in the left panel when there are instances of that class in the topology. Expanding a class lists the instances of that class. When an instance is expanded, the relationships that the instance participates in with other instances are revealed. Expanding the relationship shows the classes for which there are instances participating in the relationship.

When an instance (managed element) of a class is selected in the left panel, the right panel displays the properties of the instance under two tabs:

- Attributes tab lists the attributes and their values.
- Groups tab lists the Polling and Threshold groups to which the instance belongs plus the settings from those groups that apply to the instance.

**Note:** If a line appears in place of an attribute value, the value is derived by polling an external device that is currently unavailable. The line may appear in either the Details tab of the Notification Properties window or the Attribute tab of a property sheet in the Topology Browser view.

## Domain Manager Administration Console Toolbar Buttons

In the Domain Manager Administration Console, certain menu commands can be invoked through toolbar buttons. The toolbar buttons are summarized in Table 7.

| BUTTON | DESCRIPTION |
|---|---|
|  | Attach to a Domain Manager |
|  | Detach from a Domain Manager |
|  | Import topology from seed file |
|  | Add a new SNMP agent to topology |
|  | Update instrumentation and recompute codebook |
|  | Open the Polling and Thresholds Console |

**Table 7:** Domain Manager Administration Console Toolbar Buttons

# 5

# Methods for Adding Topology

This chapter describes different methods for providing discovery candidate systems to the discovery process. The discovery process is described in *Understanding the Discovery Process* on page 21.

When a candidate system is successfully discovered, the system is added to the topology. If the discovery of a system does not complete, one of the following occurs:

- The system is added to the Pending Devices List. A candidate system that was not previously discovered, and therefore is not already in the topology, is added to the Pending Devices List. For information about the Pending Devices List, see *Pending Devices List* on page 37.
- A DiscoveryError notification appears in the Notification Log. A DiscoveryError is notified for a system that is already in the topology, but was not successfully re-discovered. For information about discovery errors and their solutions, see *Understanding Discovery Results* on page 79.

# Adding Topology with Autodiscovery

Autodiscovery, shown in the figure *Discovery Process Using Autodiscovery and Manual Discovery* on page 23, automatically discovers candidate systems from one or more *seed systems* added to the topology through the manual discovery method. When autodiscovery is enabled, the topology and IP address table MIBs of the seed systems are probed to determine what other systems are known to the seed systems. As these systems are discovered, they too are probed for candidate systems.

Each candidate system found in this manner must then match a discovery filter and any additional autodiscovery safeguards before it is fully discovered and added to the topology. You use the Domain Manager Administration Console to create discovery filters, as described in *Using Autodiscovery* on page 62.

**Note:** All that is known about a candidate system at the beginning of the autodiscovery process is its IP address.

## Discovery Filters

Discovery filters control what systems are added to the discovery process. You must specify at least one discovery filter to use autodiscovery. In addition, a discovery filter is also required when importing topology through one of the InCharge IP adapters described in the *InCharge IP Adapters User's Guide*.

You specify filters using the five discovery fields identified in the following table.

| Discovery Filter Fields | Discovery Filter Example |
|---|---|
| IP Address Range | * (matches any IP address) |
| System Name† | * (matches any system name) |
| System Description† | * (matches and system description) |
| SystemOID | * (matches any system OID) |
| System Type | Switch |
| † The System Name and System Description fields are case-sensitive. | |

**Table 8:** Fields of a Discovery Filter

The discovery filter example shown in the table will match any candidate system that is a Switch. The asterisk (*) in the IP address range, system name, system description, and system OID fields is a wildcard character that matches any arbitrary string of characters. The string can be empty.

Discovery filters are *inclusive*, meaning that if you do not create any filters, no candidate systems are accepted. A system is accepted only if it matches a filter.

Consider the following points before creating discovery filters.

- When creating a filter, you must specify a value for each of its five fields. A candidate system must pass each field to match the filter.
- You can create multiple filters and arrange them in a high-to-low priority. A candidate system is compared against the highest priority filter first.
- When a candidate system's IP address matches an IP address of a filter, InCharge checks whether the IP address is already in the topology; if yes, InCharge discards the candidate system; if no, InCharge uses the IP address to perform the initial ICMP/SNMP polling in Phase 1 of the discovery process (shown in the figure *Phase 1: Initial Polling of a Candidate System* on page 26) to learn the system name, system description, system OID, and system type of the candidate system. Once this information is retrieved, InCharge determines whether the candidate system matches the filter during Phase 2 of the discovery process (shown in the figure *Phase 2: Determining the Destination of a Candidate System* on page 28).
- When a candidate system matches a filter and passes any additional autodiscovery safeguards, it enters Phase 3 of the discovery process and bypasses any lower priority discovery filters.
- To exclude a specific candidate system, specify the system in the ipExcludeList in the *discovery.conf* file. (Do not create filters to exclude specific systems.) For information about the *discovery.conf* file, see *Description of discovery.conf* on page 105.
- Seed systems, as listed in a seed file or entered through the **Add Agent** command, are not subject to the discovery filters. For information about seed files and the **Add Agent** command, see *Creating Topology with Manual Discovery* on page 67.

For systems that match a discovery filter and are added to the topology, they too are probed for the IP addresses of their neighboring systems. The autodiscovery cycle continues until no more new IP addresses match the discovery filters.

## Extended Filter Information

In addition to the five fields of a discovery filter, you can specify additional information for a filter to control the polling of any system that matches an IP address of the filter. The additional fields, referred to as extended options, are identified in the following table.

| DISCOVERY FILTER FIELDS |
|---|
| *Core Fields* |
| IP Address Range |
| System Name |
| System Description |
| SystemOID |
| System Type |
| *Extended Option Fields* |
| SNMP Port |
| SNMP Version* |
| Access Mode |
| Community String |
| * SNMP V3 is not currently supported for autodiscovery. |

**Table 9: Applying Extended Options to a Discovery Filter**

When a candidate system's IP address matches an IP address of an *extended* discovery filter, InCharge uses the IP address and the following user-specified extended options of the filter to perform the initial ICMP/SNMP polling of the candidate system:

- SNMP port number (161 by default)
- SNMP version (V1, V2, AUTODETECT; AUTODETECT by default)
- Access mode (ICMPSNMP or ICMPONLY; ICMPSNMP by default)
- Read community string (public by default)

For a successful SNMP polling, InCharge reads the SNMP response to learn the system name, system description, system OID, and system type of the candidate system.

Upon learning the system-related information for the candidate system, InCharge determines whether the candidate system matches the filter by checking the first five fields, or core fields, of the filter. If the candidate system matches the filter, it enters Phase 3 of the discovery process. If the candidate system does not match the filter, the system is compared against the core fields of the next lower priority filter. Whether the system matches this filter or another filter in the stack of discovery filters, it will keep the extended options assigned to it during the initial ICMP/SNMP polling. If the system does not match any filter, it is discarded.

The two filters in the following table help clarify this behavior.

| DISCOVERY FILTER FIELDS | FILTER 1 VALUES | FILTER 2 VALUES |
|---|---|---|
| IP Address Range | * | * |
| System Name | * | * |
| System Description | * | * |
| SystemOID | * | * |
| System Type | Switch | Router |
| SNMP Port | 165 | 0 |
| SNMP Version† | AUTODETECT | V2C |
| Access Mode | ICMPSNMP | ICMPSNMP |
| Community String | public1 | public2 |
| † SNMP V3 is not currently supported for autodiscovery. | | |

**Table 10:** Two Discovery Filters Having Extended Options

The candidate system to be compared with Filters 1 and 2 is a router. Although the router matches Filter 2 but not Filter 1, the router keeps the Filter 1 extended options included in the initial SNMP poll of the system: the router's SNMP port is 165, its SNMP version is AUTODETECT, its access mode is ICMPSNMP, and its read community string is public1.

**Note:** For a candidate system that matches the IP address of a *non-extended* filter, InCharge uses the default values for the extended options when contacting the system.

Other important information relevant to extended options:

- SNMP Port enables you to specify an alternate port on which systems receive SNMP requests. The default value is 0, which indicates that InCharge uses the port number value (initially 161) of the defaultSNMPPort parameter in the *discovery.conf* file. For information about the *discovery.conf* file, see *Description of discovery.conf* on page 105.
- SNMP Version enables you to control what version of SNMP that InCharge uses to communicate with the system. Valid values for autodiscovery are V1, V2C, and AUTODETECT. The default is AUTODETECT, meaning that discovery tries SNMP V2C first and then SNMP V1 if necessary. If you specify an explicit SNMP version, V2C or V1, the discovery process tries only the SNMP version that you specified. If the explicit version is different from the actual version supported by the system, a DiscoveryError with a message of *No response from SNMP Agent* is notified.
- Access Mode determines what protocols are used to both discover and monitor the system and its components. Valid values for autodiscovery are ICMPSNMP and ICMPONLY. The default is ICMPSNMP, meaning that both ICMP and SNMP protocols are used to discover and manage the system. When ICMPONLY is selected, discovery does not send SNMP polls to the system and cannot, therefore, discover the system and its components. Instead, the system is added to the topology as a Host and is monitored only for connectivity.

**Note:** An additional Access Mode option, SNMPONLY, is available through the manual discovery method described in *Creating Topology with Manual Discovery* on page 67.

- Community String enables you to specify the one or more read community strings that should be used to contact systems that match this filter. If no community strings are specified, autodiscovery uses the default read community string. For information about default community strings, see *Adding and Removing Read Community Strings* on page 93.

## Examples of Discovery Filters

A discovery filter can be broad, to discover a broad range of elements, or restrictive, to limit the number or type of discovered systems.

For a filter's IP address range, system name, system description, and system OID fields, you can use wildcards to create a matching pattern. For information regarding the syntax of wildcards, see *Wildcards* on page 115.

The following examples illustrate different types of discovery filters.

### Filtering a Range of IP Addresses

To exclude a particular range of systems, for example 22 in the second octet of an IP address range, use a filter that includes every other possibility as shown in Table 11.

| DISCOVERY FILTER FIELDS | VALUE |
|---|---|
| IP Address Range | 172.<16-21>.* \| 172.<23-31>.55.* |
| System Name | * |
| System Description | * |
| SystemOID | * |
| System Type | Select all |

**Table 11:** Filtering by IP Address

### Filtering by System Type, Name, and Description

To narrow the range of a filter, specify criteria for multiple fields, as shown in Table 12. Only those systems that are routers, within a specified IP address range, whose system name begins with *ny*, and whose system description begins with *Cisco* match this filter.

| DISCOVERY FILTER FIELDS | VALUE |
|---|---|
| IP Address Range | 172.21.55.* |
| System Name | ny* |
| System Description | Cisco* |
| SystemOID | * |
| System Type | Router |

**Table 12:** Filtering By System Type, Name, and Description

### Filtering with Manual Accept Mode

To manually accept each candidate system, enable manual accept mode by checking the *Ask before adding new systems* checkbox. In the example shown in Table 13, routers and switches that match Filter 1 are automatically added to the topology. Hosts, however, that match Filter 2 are automatically added to the Pending Devices List.

| DISCOVERY FILTER FIELDS | FILTER 1 VALUES | FILTER 2 VALUES |
|---|---|---|
| IP Address Range | 172.21.55.* | 172.21.55.* |
| System Name | * | * |
| System Description | * | * |
| SystemOID | * | * |
| System Type | Routers<br>Switches | Hosts |
| Ask before adding new systems | Unchecked | Checked |

**Table 13:** Filtering with Accept Mode

### Filter for InCharge IP Adapter

When you import topology through one of the InCharge IP adapters described in the *InCharge IP Adapters User's Guide*, use an open filter and select all system types as shown in Table 14.

| DISCOVERY FILTER FIELD | VALUE |
|---|---|
| IP Address Range | * |
| System Name | * |
| System Description | * |
| SystemOID | * |
| System Type | Select all |

**Table 14:** Filter for Importing Topology

### Filtering by System OID

To limit discovered systems by vendor, you can filter by the enterprise OID. The filter shown in Table 15 only matches systems of type Switch with an OID of 1.3.6.1.4.9, which is the enterprise OID for devices made by Cisco Systems, Inc.

| DISCOVERY FILTER FIELD | VALUE |
|---|---|
| IP Address Range | * |
| System Name | * |
| System Description | * |
| SystemOID | 1.3.6.1.4.1.9.* |
| System Type | Switch |

**Table 15:** Filtering by OID

## Autodiscovery Safeguards

You can use one or both of the following safeguards to ensure that autodiscovery does not add systems (to the topology) that you do not intend to manage:

- Manual accept mode
- Current system limit

### Using Manual Accept Mode

You can configure a discovery filter for manual accept mode so that a candidate system matching the filter is added to the Pending Devices List in the THROTTLED state. By default, a candidate system that matches a filter is automatically discovered and added to the topology.

To enable manual accept mode for a filter, follow these steps:

1. In the left panel of the Domain Manager Administration Console, click the Domain Manager name to display a multiple-tab window in the right panel of the console.
2. Select the Discovery Filters tab.
3. Select the filter for which you want to enable the manual accept mode.
4. Click the *Ask before adding new systems* checkbox to enable manual accept mode for the filter.
5. Click **Apply**.

To see what systems passed the filter and are waiting to be accepted, follow these steps:

1. In the Domain Manager Administration Console, Select *Topology > Show Discovery Progress* to open the Discovery Progress window.

2. In the Discovery Progress window, find systems listed as THROTTLED.
3. Right-click a system marked THROTTLED and select **Accept** from the drop-down menu.

   Accepting a THROTTLED system changes the state of the system to UNDISCOVERED; the system will be discovered at the next Pending Discovery Interval. For information about the Pending Devices List, see *Pending Devices List* on page 37. For information about the Pending Discovery Interval, see *Pending Discovery Interval* on page 97.

### Setting a Current System Limit for the Topology

You can limit the number of systems that are added to the topology by setting a current system limit. By default, the current system limit is set to 50 systems.

To change the current system limit, follow these steps:

1. In the left panel of the Domain Manager Administration Console, click the Domain Manager name to display a multiple-tab window in the right panel of the console.
2. Select the Topology tab.
3. Enter a new value for the Current System Limit field.
4. Click **Apply**.

When the current system limit is exceeded, InCharge will complete the discovery of any system (and its components) in the midst of being discovered, including all virtual routers configured for a router or switch that implements virtual routers. When a router or switch implementing virtual routers is discovered, the chassis-based router/switch is counted as a system, and each virtual router is counted as a system. Any new candidate discovery systems are added to the Pending Devices List in the UNDISCOVERED state, and the message *System limit exceeded* appears in the Discovery Status section of the Discovery Progress window.

**Note:** The current system limit applies to all discovery methods: autodiscovery, manual discovery, and topology import.

## Using Autodiscovery

You configure and initiate autodiscovery through the Domain Manager Administration Console. The procedure for using autodiscovery requires three steps:

1. Create one or more discovery filters.
2. Enable autodiscovery.
3. Initiate autodiscovery by providing one or more seed systems.

### Creating a Discovery Filter

For autodiscovery and topology import, discovery filters determine which systems are discovered and added to the topology. You create a filter by specifying a combination of characters and wildcards and selecting the types of systems to which the filter is applied. For information regarding the syntax of wildcards, see *Wildcards* on page 115.

To create a discovery filter, follow these steps.

1. In the left panel of the Domain Manager Administration Console, click the Domain Manager name to display a multiple-tab window in the right panel of the console.
2. Select the Discovery Filters tab.
3. Specify values for each of the discovery filter fields:
   - IP Address Range
   - System Name
   - System Description
   - SystemOID
   - System Types

   You must specify a value for each field. The System Name and System Description fields are case-sensitive.
4. Decide whether newly discovered systems matching this filter should be added to the topology or to the Pending Devices List. For information about automatically or manually accepting new systems, see *Filtering with Manual Accept Mode* on page 60.
5. Optional: Check the View Extended Filter checkbox to view the extended options of the filter. The extended options enable you to specify the SNMP port, the SNMP version, the access mode, and the read community string(s) to be used when polling the systems that match an IP address within the IP Address Range of this filter.
6. Click **Add**, then **Apply**.

When you create multiple filters, systems are compared against the filters in the order in which the filters are arranged. When a system matches a filter, it is added to the discovery queue. Because of this behavior, put your more limiting filters first. You can arrange the order of the filters by selecting a filter and clicking the up or down arrows under the list of filters.

### Enabling Autodiscovery

After creating one or more discovery filters, enable autodiscovery by following these steps.

1. In the left panel of the Domain Manager Administration Console, click the Domain Manager name to display a multiple-tab window in the right panel of the console.
2. Select the Topology tab.
3. Click the *Enable Auto Discovery* checkbox.
4. Select the data sources that discovery should use to find additional candidate systems. By default, autodiscovery uses the topology and IP address table MIBs of the discovered systems (initially, of the discovered seed systems).

   For information about autodiscovery data sources, see *Autodiscovery Data Sources* on page 65.
5. In the Community String field, provide any additional read community strings necessary to communicate with systems on your network. You may add up to four read community strings; the default is *public*.

   The community strings specified here are global default community strings that apply to autodiscovery, manual discovery, and topology import. For information about adding community strings, see *Adding and Removing Read Community Strings* on page 93.
6. In the Current System Limit field, enter the total number of systems you want to allow into the topology. If the number of discovered systems exceeds the system limit, the remaining candidate systems to be discovered are placed on the Pending Devices List with a state of UNDISCOVERED.
7. Click **Apply**.

### Initiating Autodiscovery

After you have created at least one discovery filter and enabled autodiscovery, you are ready to initiate autodiscovery. To start the autodiscovery process, you need to provide InCharge with one or more *seed systems*, which serve as the starting point for building the InCharge topology. Seed systems are not matched against the discovery filters.

InCharge probes the topology and IP address table MIBs of the seed systems to find IP addresses of additional systems; those IP addresses become candidate systems for the autodiscovery process. Routers and switches are good choices for seed systems because their MIBs contain the IP addresses of many additional systems in the network.

You can provide one or more seed systems using the **Import from seed file** command, or you can provide a single seed system using the **Add Agent** command. To use the **Import from seed file** command, see *Using the Import From Seed File Command* on page 74. To use the **Add Agent** command, see *Using the Add Agent Command* on page 74.

Each time you invoke a discovery, the Discovery Progress window displays. This window shows the progress of discovery and contains the Pending Devices List. For information about the Pending Devices List, see *Pending Devices List* on page 37.

### Stopping Autodiscovery

You can stop the autodiscovery process from the Discovery Progress window. (If the Discovery Progress window is not open, select *Topology > Show Discovery Progress* in the Domain Manager Administration Console to open the window.) In the Discovery Progress window, click **Stop Autodiscovery** to stop autodiscovery.

Clicking the **Stop Autodiscovery** button stops the finding of candidate systems for autodiscovery during Phase 3 of the discovery process. However, any systems already in the discovery queue continue to be processed, and the progress of this processing displays in the Discovery Progress window.

Stopping autodiscovery also disables future autodiscovery action until autodiscovery is re-enabled. To re-enable autodiscovery, see *Enabling Autodiscovery* on page 64.

## Autodiscovery Data Sources

The autodiscovery process can use three different sources of information to discover candidate systems:

- Topology MIB
- IP address table MIB
- ARP table MIB

By default, discovery uses the topology and IP address table MIBs because they are very reliable sources for topology information.

To select which data source(s) are used for autodiscovery, follow these steps:

1. In the left panel of the Domain Manager Administration Console, click the Domain Manager name to display a multiple-tab window in the right panel of the console.
2. Select the Topology tab.
3. Select the data sources used by autodiscovery to find additional candidate systems. The data sources are listed underneath the *Enable Autodiscovery* checkbox.
4. Click **Apply**.

### Topology MIBs Data Source

Topology MIBs are an important data source for discovering neighboring systems. When the topology MIB data source is enabled, autodiscovery uses the IP addresses from these sources to discover related systems. Examples of topology MIB data sources include:

- IP addresses stored in the Cisco Discovery Protocol (CDP) tables
- IP addresses stored in the MIB-II atmInterfaceConfTable table

In some cases, however, a system is not discovered for each IP address, if, for example, there is no route to the network from the host on which InCharge is running. This situation can occur when security restrictions are designed to prevent access between two separate networks.

If you disable the topology MIB data source, autodiscovery may not be able to find all the systems in the managed network.

### IP Address Table MIB Data Source

The MIB-II IP address table lists all of the IP interfaces of the system being discovered as well as the subnet masks associated with these IP networks. Using this information, autodiscovery derives all of the possible IP addresses within each subnet. InCharge pings each of these addresses to find candidate systems for subsequent SNMP probing.

By default, InCharge pings the addresses in networks with an *8-bit host address* (254 hosts), to prevent the indiscriminate pinging of large numbers of addresses. If necessary, you can change the default 8-bit host address setting to a larger value by editing the MaximumHostBits parameter in the *discovery.conf* file. For information about the MaximumHostBits parameter, see *Description of discovery.conf* on page 105.

**Note:** It can take InCharge a long time, depending on your network configuration, to ping the entire list of addresses when the MaximumHostsBits is set to a large value, such as 16 bits (more than 65,000 addresses). With a value of 24 bits, for example, it could take InCharge up to several hours to ping all the available addresses.

### ARP Table MIB Data Source

The MIB-II ARP table provides a mapping of IP addresses (Layer 3) to MAC addresses (Layer 2) for any neighboring system that has recently communicated with the system with this MIB. However, because the ARP MIB is only populated when the system resolves the addresses of neighboring systems connected with a broadcast network protocol such as Ethernet, the ARP MIB is not appropriate for use in Wide Area Networks, which typically use a non-broadcast protocol.

By default, this data source is not enabled because autodiscovery does not typically gain many IP addresses using the ARP table MIB.

## Creating Topology with Manual Discovery

Manual discovery, shown in the figure *Discovery Process Using Autodiscovery and Manual Discovery* on page 23, provides two options for manually adding systems to the InCharge topology. Both options are initiated through the Domain Manager Administration Console.

- Option 1: Import a seed file that lists one or more seed systems to be added to the topology. This option is best for adding a set of systems.
- Option 2: Add seed systems to the topology one at a time using the **Add Agent** command. This option is best for adding a small number of systems.

When autodiscovery is enabled, using either option initiates the autodiscovery process.

## Preparing a Seed File

A seed file identifies top-level network systems such as switches, routers, hubs, bridges, and hosts that are to be discovered and monitored by InCharge. InCharge discovers the internal structure of these systems. For example, in the case of a router, you provide the hostname or IP address of one interface, and InCharge discovers the rest of the interfaces. In the case of a switch, you provide the hostname or IP address of the switch's SNMP agent, and InCharge determines the switch's configuration and how the switch connects to other systems listed in the topology.

The seed file must be stored on the host on which the Domain Manager runs. By default the Domain Manager expects to find the seed file in the ***BASEDIR****/smarts/local/conf* directory. An example of a seed file named *seedfile* is provided in ***BASEDIR****/smarts/conf*.

### Format of a Seed File

A seed file consists of one or more seed entries, each of which represents a candidate discovery system. Each seed entry begins with the hostname or IP address for the candidate system followed by options for the system; a seed entry may span several lines. Most of the options are specified using the keywords described in Table 16. The <keyword>=<value> options may be arranged in any order.

For a seed entry, you can specify the port number of the SNMP agent running on the candidate system in one of two ways:

- By typing a colon (:) and the port number after the hostname or IP address
- By using the SNMPPORT keyword described in Table 16

For a candidate system having an SNMP V1 or V2C agent, the second field of a seed entry is the optional read community string. If you do not provide a community string for a V2C or V1 candidate system, InCharge uses the default read community string(s) specified in the Read Community Strings field (Topology tab) of the Domain Manager Administration Console—see *Adding and Removing Read Community Strings* on page 93 for details. If the community string starts with the pound sign (#), you must use the COMMUNITY keyword (Table 16) to specify the read community string.

For a candidate system having an SNMP V3 agent, the only keyword option required is the USER keyword (Table 16), which identifies the user name for the system. If you do not provide a user name for a V3 candidate system, InCharge throws an exception.

| KEYWORD | VALUE | DEFINITION |
|---|---|---|
| *Global SNMP Options* | | |
| SNMPPORT= | 0 to 65535<br>Default: port in *discovery.conf* | SNMP port used to access the SNMP agent of the candidate system. If the value is omitted or 0, the port specified in the defaultSNMPPort parameter (161 by default) in the *discovery.conf* file is used.<br>For information about the defaultSNMPPort parameter, see *Description of discovery.conf* on page 105. |
| SNMPVERSION= | V1, V2C, V3, or AUTODETECT<br>Default: AUTODETECT | SNMP protocol version supported by the SNMP agent of the candidate system:<br>• For AUTODETECT, send V2C request; if no response, send V1 request.<br>• For V1, send V1 request only.<br>• For V2C, send V2C request only.<br>• For V3, send V3 request only. |
| ACCESSMODE= | ICMPONLY, SNMPONLY, or ICMPSNMP<br>Default: ICMPSNMP | Protocols used to access the candidate system:<br>• For ICMPSNMP, both ICMP and SNMP are used to discover and monitor the system.<br>• For ICMPONLY, the system is not discovered or monitored. Instead, the system is added to the topology as a Host, which limits InCharge to only availability analysis using ICMP.<br>• For SNMPONLY, the system is discovered using SNMP, but ICMP is not used to determine whether the system is reachable or to monitor the system's availability. |
| ADDRESSFORMAT= | AUTOADDRESS or LOOPBACKADDRESS<br>Default: value in *discovery.conf* | Address used to access the candidate system for discovery and polling. If the value is omitted, the value specified in the AccessAddressFormat parameter (AUTOADDRESS by default) in the *discovery.conf* file is used.<br>For information about the AccessAddressFormat parameter, see *Description of discovery.conf* on page 105. |
| *SNMP V1 and V2C Options* | | |
| COMMUNITY= | String of unspecified length<br>Default: string(s) in Read Community Strings field* | Read community string included in a request sent to the SNMP V1/V2C agent of the candidate system. This keyword is required if the first character of the string is #. |

| KEYWORD | VALUE | DEFINITION |
|---|---|---|
| *SNMP V3 Options* | | |
| USER= | String consisting of 1 to 32 characters<br>Default: null string (empty) | Name of the user included in a request sent to the SNMP V3 agent of the candidate system. User name is required. |
| ENGINEID= | An even-length string between 10 and 64 hex characters; e.g., 800002b804616263<br>Default: null string (empty) | Identifier, within an administrative domain, that uniquely identifies the SNMP engine of the SNMP V3 agent. The engine ID is required for communication to the SNMP V3 agent:<br>• When specified, InCharge uses the specified value when attempting to communicate with the SNMP agent.<br>• If empty, InCharge learns the engine ID through the engine ID discovery mechanism described in RFC 3414 and then uses the learned value when attempting to communicate with the SNMP agent. |
| AUTHPROTOCOL= | MD5, for Message Digest 5, SHA, for Secure Hash Algorithm, or NONE<br>Default: MD5 | Protocol used for authentication when sending a request to the SNMP V3 agent. Authentication protocol is recommended but not required. |
| AUTHPASS= | String consisting of 1 to 64 characters; using at least eight characters is recommended<br>Default: null string (empty) | Password used as the basis for the creation of the localized authentication key used with the selected authentication protocol and the SNMP V3 agent. The password is required if the authentication protocol is MD5 or SHA. Otherwise, the password is ignored.<br>The password may be encrypted (recommended) or appear as plain text in the seed file.† |
| CONTEXT= | String of no specific length<br>Default: null string (empty) | Name that together with the user name determines the access permissions of a request sent to the SNMP V3 agent. Context name is optional.<br>InCharge must have permissions to access the standard and proprietary MIBs identified in Appendix A of the *InCharge IP Availability Manager User's Guide*. |

* Specified in the Topology tab of the Domain Manager Administration Console.

† Explained in *How To Encrypt Passwords in the Seed File* on page 73.

**Table 16:** Keywords For Seed File

Other keywords used to specify additional information for a candidate system are also listed in Table 16. Blank spaces before and after the equal sign (=) are not permitted.

To make your seed file more readable, you can include blank lines and comments. The first character of a comment line is the pound sign (#).

### Seed File Examples for V1/V2C Candidate Systems

The following examples of seed file entries illustrate the use of the keywords (listed in Table 16) for an SNMP V1 or V2C candidate system.

The first seed entry spans multiple lines and specifies the SNMP version and the read community string.

```
Core-ROUTER-1 SNMPVERSION=V1
       COMMUNITY=xxxxyyyy#%@*&^)(:"=|\][}{
```

The second seed entry specifies the SNMP port (2161) that InCharge should use to access the system.

```
ROUTER2:2161 SNMPVERSION=V2C COMMUNITY=public1
```

The third seed entry requires the keyword COMMUNITY because the first character of the read community string is a pound sign (#).

```
router-2.example.com COMMUNITY=#abcdefg SNMPPORT=2222
```

The fourth seed entry can be specified without using keywords. The only two fields are the system identifier and the read community string. The COMMUNITY keyword is not required even though the first character of the community string is the equal sign (=).

```
192.168.2.2  =abcdefg
```

### Seed File Examples for V3 Candidate Systems

The following examples of seed file entries illustrate the use of the keywords (listed in Table 16) for an SNMP V3 candidate system.

The first seed entry spans multiple lines and specifies the SNMP version, the user name, and no authentication.

```
Core-SWITCH-1 SNMPVERSION=V3 USER=inch_n
       ENGINEID=800002b804616263 AUTHPROTOCOL=NONE
```

The second seed entry specifies the SNMP port (2161) that InCharge should use to access the system.

```
SWITCH2:2161 SNMPVERSION=V3 USER=inch_n AUTHPROTOCOL=NONE
```

The third seed entry specifies that the default authentication protocol MD5 should be used to access the system. The prefix `<E-1.0>` indicates that the authentication password is encrypted.

```
switch-2.example.com SNMPVERSION=V3 USER=smarts SNMPPORT=2222
       ENGINEID=6acff2d4098338 AUTHPROTOCOL=MD5
       AUTHPASS=<E-1.0>869D64B72C38FE49E881BC38BCCE6A0B
       1DE597A5871636C4B0F139FA7EB1100C
```

Including AUTHPROTOCOL=MD5 in this entry is optional because MD5 is the default authentication protocol. Also, the encrypted password, which wraps around in the display but is actually one line, was created using the procedure in *How To Encrypt Passwords in the Seed File* on page 73.

The fourth seed entry specifies that the authentication protocol SHA should be used to access the system and that access control is in effect. The authentication password is not encrypted.

```
192.154.1.1 SNMPVERSION=V3 USER=smarts_2 AUTHPROTOCOL=SHA
       AUTHPASS=sally121#% CONTEXT=incharge_west
```

### How To Determine What To Put in a Seed File

The seed file should list the top-level network systems that are to be managed by InCharge. To create the seed entries in a seed file, you can usually find the information that you need from existing computer-accessible sources.

- If you are currently using CiscoWorks with Cisco Resource Manager, you can dump the hardware inventory to a text file. That text file can be converted into a seed file. Similar operations may be possible with other network management platforms.
- If you are currently using one of the InCharge IP adapters described in the *InCharge IP Adapters User's Guide*, you can obtain the network information from the *ovsnmp.conf* file. This configuration file is maintained by the xnmsnmpconf configuration utility.
- If you store backup router or switch configuration files at a central location, you can construct a seed file from them.
- Domain name resolution files, such as *hosts.txt* or zone files, are often a useful source. Note that, at sites where multiple administrative authorities choose different read community strings, the hosts listed in a given zone file often share an administrative authority and hence a read community string. Terminal Access Controller Access Control System (TACACS) configuration files may be useful for similar reasons.
- Large sites usually maintain hardware inventories in a database or at least in some computer-readable form. These inventories often contain the network information that you need to create the seed file.

- Some sites run network *sniffer* tools to keep track of active IP addresses. The lists of addresses that they generate can form the basis for a seed file.

After you have identified which systems to include in the seed file, you may want to include multiple seed entries for systems having multiple IP addresses (routers, for example). The discovery process will realize that the multiple IP addresses actually belong to a single system. Accordingly, InCharge will be able to reach the system using an alternate address if the primary address is inaccessible (due to a failure) during the initial probe of the system. After a system is discovered and all its IP addresses are known, InCharge will automatically try up to ten different IP addresses to route around failures and reach the system.

### How To Create a Seed File

You typically use the template file named *seedfile*, located in the ***BASEDIR****/smarts/conf* directory, as the basis for creating a seed file.

To create a seed file using the *seedfile* template, follow these steps:

1. Open the *seedfile* template using the *sm_edit* utility in the ***BASEDIR****/smarts/bin* directory.

   ```
   % ./sm_edit conf/seedfile
   ```

2. Add the seed entries for the systems that you want to discover and save the file. The modified version of the file is saved to the ***BASEDIR****/smarts/local/conf* directory.

### How To Encrypt Passwords in the Seed File

When viewing the seed file with *sm_edit*, add the following line (including #) as the first line in the seed file to direct *sm_edit* to encrypt the authentication passwords in the file:

```
#<encrypted seed>:1.0:AUTHPASS
```

When you close the seed file, *sm_edit* will automatically encrypt the authentication passwords in the file.

To determine whether a password in a seed file has been encrypted, look for the prefix `<E-1.0>` in the password. If `<E-1.0>` is present, the password is encrypted; if `<E-1.0>` is not present, the password is plain text. Although seed files containing plain-text passwords are supported, they are not recommended.

The *sm_edit* utility does not encrypt a password that is already encrypted, determined by the presence of the `<E-1.0>` prefix. Once a password is encrypted, you cannot use *sm_edit* to decrypt the password. If you wish to change an encrypted password, you must delete the password (including the `<E-1.0>` prefix), enter the new password, and save the file.

During the installation of InCharge products, the installation program uses the case-sensitive secret phrase *Not a secret* to create a secret key, to be used by *sm_edit* and other InCharge utilities to encrypt passwords. The secret phrase and secret key, themselves, are encrypted and stored in the ***BASEDIR****/local/conf/imk.dat* file. After the installation, an InCharge administrator should use the *sm_rebond* utility to change the secret phrase and secret key.

For information about *sm_rebond* and InCharge encryption, see the *InCharge System Administration Guide*. For more information about *sm_edit*, see *Modifying InCharge Files* on page 104.

## Using the Import From Seed File Command

To import a list of seed systems from a seed file, follow these steps:

1. In the Domain Manager Administration Console, select *Topology > Import from seed file*, or click the **Import from seed file** toolbar button, to display the Import From Seed File dialog box.
2. In the dialog box, specify the complete path and name of the seed file. By default, the path to the seed file is ***BASEDIR****/smarts/local/conf*. Note that the seed file must be stored on the host where the Domain Manager runs.
3. Click **OK** to import the seed systems and to start the discovery process. The Discovery Progress window displays.

Each time you invoke a discovery, the Discovery Progress window appears. This window shows the progress of discovery and contains the Pending Devices List. For information about the Pending Devices List, see *Pending Devices List* on page 37. For information regarding discovery errors and their solutions, see *Understanding Discovery Results* on page 79.

## Using the Add Agent Command

You use the **Add Agent** command to add seed systems to the topology one at a time. **Add Agent** is intended for adding a small number of seed systems.

### Adding an SNMP V1/V2C Seed System

To add an SNMP V1 or V2C seed system, follow these steps:

1. In the Domain Manager Administration Console, select *Topology > Add Agent*, or click the **Add Agent** toolbar button, to display the Add Agent dialog box.
2. In the dialog box, specify the hostname or IP address of the V1/V2C seed system in the Agent Name field.

   You can also specify the port on which the SNMP agent of the seed system receives requests by typing a colon (:) and the port number after the hostname or IP address. If you do not specify a port, the default port specified in the *discovery.conf* file is used.
3. Optional: If the read community string of the V1/V2C seed system is not one of the strings in the Read Community Strings field (Topology tab) of the Domain Manager Administration Console, enter a read community string in the Read Community field. For more information about read community strings, see *Adding and Removing Read Community Strings* on page 93.
4. Optional: Check the Advanced Options checkbox to display the Add Agent Advanced Options panel.

   In the Advanced Options panel, you can optionally specify the SNMP port, the SNMP version, and the access mode for the V1/V2C seed system. For a description of these options and their values, see the table *Keywords For Seed File* on page 70.
5. Click **OK** to add the V1/V2C seed system and to start the discovery process. The Discovery Progress window displays.

For information regarding the Pending Devices List that appears in the Discovery Progress window, see *Pending Devices List* on page 37. For information regarding discovery errors and their solutions, see *Understanding Discovery Results* on page 79.

### Adding an SNMP V3 Seed System

To add an SNMP V3 seed system, follow these steps:

1. In the Domain Manager Administration Console, select *Topology > Add Agent*, or click the **Add Agent** toolbar button, to display the Add Agent dialog box.
2. In the dialog box, specify the hostname or IP address of the V3 seed system in the Agent Name field.

You can also specify the port on which the SNMP agent of the seed system receives requests by typing a colon (:) and the port number after the hostname or IP address. If you do not specify a port, the default port specified in the *discovery.conf* file is used.

3. Check the Advanced Options checkbox to display the Add Agent Advanced Options panel.

   In the Advanced Options panel, you can optionally specify the SNMP port and the access mode for the V3 seed system. For a description of these options and their values, see the table *Keywords For Seed File* on page 70.

4. In the Advanced Options panel, select SNMP version V3 to display the Add Agent SNMP V3 Specifications panel.

   In the SNMP V3 Specifications panel, you must specify the user name for the V3 seed system; you can optionally specify the SNMP engine ID, the authentication protocol, and the context name. If you specify MD5 or SHA, you must specify an authentication password; the password that you type is masked with asterisks (*) for security reasons. For a description of these options and their values, see the table *Keywords For Seed File* on page 70.

5. Click **OK** to add the V3 seed system and to start the discovery process. The Discovery Progress window displays.

For information regarding the Pending Devices List that appears in the Discovery Progress window, see *Pending Devices List* on page 37. For information regarding discovery errors and their solutions, see *Understanding Discovery Results* on page 79.

# Importing Topology

Topology import, shown in the figure *Discovery Process Using Topology Import* on page 24, enables InCharge to discover a set of candidate systems imported from a third-party source through an InCharge IP adapter. For information regarding the configuration and operation of an InCharge IP adapter, see the *InCharge IP Adapters User's Guide*.

If you have previously discovered a network topology through a third-party source, InCharge can use an InCharge IP adapter to import that information—a list of systems having SNMP V1 or V2C agents—to build its own topology. To import systems having SNMP V3 agents, list those systems in a seed file and use manual discovery to add them to the InCharge topology, as described in *Preparing a Seed File* on page 68.

To import topology using an InCharge IP adapter, you must specify at least one discovery filter. You should create an open discovery filter that allows a broad range of elements to be added to the topology. For an example, see *Filter for InCharge IP Adapter* on page 60.

Keep in mind that the open discovery filter created for topology import will adversely affect the discovery filter(s) created for autodiscovery in the following regard: No matter where you place the topology import filter in the high-to-low priority stack of discovery filters, it will allow all autodiscovery candidate systems to pass through and be added to the discovery queue. Accordingly, you should disable autodiscovery during the topology import process.

After the topology import process completes, you may want to enable autodiscovery and manually initiate a full discovery to autodiscover the neighboring systems of the imported systems added to the topology. For information about full discovery, see *Rediscovering Topology Elements* on page 96.

# 6

# Understanding Discovery Results

When InCharge attempts to discover a system, one of the following results occur:

- Discovery successfully completes and the system and its components are added to the InCharge topology.
- Discovery starts but one of the discovery probes encounters an error during the discovery process.
- Discovery starts but cannot complete because of an insufficient number of volume licenses.
- Discovery is unable to start or successfully complete.

This chapter describes what InCharge does when the discovery process encounters an error. For an overview of the discovery process, see the flow diagrams in *Discovery Process and Discovery Methods* on page 22 and *Phases of Discovery* on page 25.

Table 17 describes different discovery scenarios and error conditions. Initial discovery refers to an attempt to discover a system that has not been previously discovered—not included in the InCharge topology. Rediscovery refers to an attempt to discover a system that is already included in the topology.

| DISCOVERY PHASE | CONDITION | RESULT |
|---|---|---|
| Initial discovery | Cannot resolve hostname to IP address. | Add system to Pending Devices List. See *Discovering a System with a Hostname That Does Not Resolve to an IP Address* on page 88. |
| | No response from ICMP ping. | For a seed system (manual discovery), add system to the Pending Devices List as UNDISCOVERED.<br>For a system found through autodiscovery or topology import, discard system.<br>Also, see *Increasing Timeout Values for Discovery Polling* on page 89. |
| | No response from SNMP agent. | For a seed system (manual discovery), add system to the Pending Devices List as UNDISCOVERED.<br>For a system found through autodiscovery or topology import, and ShowPendingNONSNMP set to TRUE, add system to the Pending Devices List as UNDISCOVERED.<br>For a system found through autodiscovery or topology import, and ShowPendingNONSNMP set to FALSE, discard system.<br>Also, see *System Does Not Support SNMP* on page 84. |
| | ICMP response and SNMP response; current system limit exceeded. | For a seed system (manual discovery) or a system found through autodiscovery or topology import, display *System limit exceeded* message in the Discovery Progress window and add system to the Pending Devices List as UNDISCOVERED. |
| | ICMP response and SNMP response; manual accept mode enabled for discovery filter. | For a system matching the discovery filter, add system to the Pending Devices List as THROTTLED. |
| | SNMP agent encounters a loop, preventing the system from being discovered. | Classify system according to its sysObjectID, add to topology, and send a DiscoveryError notification. |
| | Insufficient number of volume licenses, preventing complete discovery of systems. | Stop discovery process and send an OutOfLicense notification.<br>Display *Out of license* message in the Discovery Progress window.<br>Add any new discovery candidate systems to the Pending Devices List as UNDISCOVERED. |
| | Discovery completes successfully. | Classify system according to its sysObjectID and add to topology. If the OID is not recognized, classify system as a Node. |

| DISCOVERY PHASE | CONDITION | RESULT |
|---|---|---|
| Rediscovery | No response from ICMP ping. | Send notification, typically *Down*, based on the correlation analysis, and send a DiscoveryError notification. |
| | Receive no response from SNMP agent: SNMP request times out. | Send notification, typically *Agent NotResponding*, based on the correlation analysis. |
| | Discovery not able to contact SNMP agent on known address. | Send DiscoveryError notification. |

**Table 17:** Discovery Scenarios and Results

# DiscoveryError Notifications

Discovery errors are generated for systems that support SNMP. When an error is encountered during the discovery of such a system, InCharge notifies a DiscoveryError. If the AccessMode for a system is ICMPONLY and the system is not reachable by ICMP during discovery, InCharge does not notify a DiscoveryError.

To obtain more information about a DiscoveryError notification, double-click the notification in the Notification Log Console to open the Notification Properties dialog. Select the Details tab to see the DiscoveryErrorInfo attribute. A Details tab is displayed for each domain that notified the event.

The following is a list of errors that result in a DiscoveryError notification:

- SNMP request times out
- SNMP agent loops
- System Down
- Qualified access address not found
- System previously discovered fails authentication

## SNMP Request Times Out

A common error is that an SNMP request times out before it completes. There are a number of reasons why this error may occur.

- The system might be busy and therefore drops SNMP packets.
- The network might be busy and therefore drops or delays SNMP packets.

- A large bridge forwarding table might be causing the SNMP agent not to respond in time. Discovery uses the bridge forwarding table to determine connectivity.

If the error occurs because the system or network is busy, you should rediscover the system when the system or network is less busy.

If you notice an excessive number of timeouts during the discovery process, you can change the default values for timeouts in the *discovery.conf* file. For information regarding this procedure, see *Increasing Timeout Values for Discovery Polling* on page 89.

## SNMP Agent Loops

If a system has an SNMP agent with an improper MIB implementation, SNMP requests from the discovery process might loop over a table. Looping over a table is a violation of the standard protocol and prevents InCharge from retrieving the necessary information to fully discover a system.

When InCharge encounters such a problem, it notifies a DiscoveryError.

You can identify the location of the agent loop by invoking the following command from the ***BASEDIR****/smarts/bin* directory:

```
% ./sm_snmpwalk <Host or IP address>
```

You should see an error message similar to the following:

```
SNMP-E-Agent 198.49.114.168, port 161 loops in response to
Get-Next/Bulk(.1.3.6.1.4.1.9.9.84.1.2.1.1.3.1.1.0.176.208.
108.29.255)
```

Discovery continues with the next probe until the discovery process completes. However, if the error is the result of a problem with the system's MIB, you should contact the system vendor to see if an update is available.

## System Down

In some cases, discovery fails because the system is Down (unreachable). When this situation occurs, InCharge notifies a DiscoveryError and lists the DiscoveryError as an impact of the Down problem. The value of the DiscoveryErrorInfo attribute is *No response from ping*.

## Qualified Access Address Not Found

This error occurs during discovery or rediscovery of a system when no IP address object exists in the topology for an SNMP agent or each of the IP addresses associated with the SNMP agent is:

- An unmanaged IP address, or
- An IP address within the range of the ipExcludeList parameter in the *discovery.conf* file, or
- An IP address associated with an interface that is either in BACKUP or ONDEMAND mode.

Because InCharge has no valid IP address to access the SNMP agent, InCharge notifies a DiscoveryError.

## System Previously Discovered Fails Authentication

If, during a rediscovery, the SNMP agent of a system fails authentication, InCharge notifies a DiscoveryError. In the Notification Log Console, you can determine the reason for the error by double-clicking the DiscoveryError notification and viewing the Details tab in the Notification Properties dialog. Or, in the Topology Browser Console, you can determine the reason for the error by double-clicking the failed system and viewing the Attributes tab in the system's properties dialog.

For an SNMP V1 or V2C agent, the most likely reason for this error is that the read community string for the agent has changed. For errors concerning an incorrect read community string, see *Incorrect Read Community String* on page 85.

For an SNMP V3 agent, the most likely reason for this error is that the authentication password, authentication protocol, or user name for the agent has changed. For information about SNMP V3 parameters, see the table *Keywords For Seed File* on page 70.

# SNMP Agent Violates SNMP Protocol

SMARTS implementation of the SNMP protocol is strict in that it only sends or receives SNMP messages that conform to the SNMP standard. Unfortunately, not all SNMP implementations are as strict. The result is that SMARTS programs may receive non-conforming SNMP messages. In some cases, SMARTS software can successfully interpret and handle non-conforming messages.

One such case is when an SNMP agent returns the values of an SNMP request in the wrong order. For example, InCharge requests the values for *var1*, *var2*, and *var3*, and the SNMP agent returns *var1*, *var3*, and *var2*. InCharge writes a message to the Domain Manager's log file:

```
SWFE-W-Agent violates lexicographic ordering in response to
GET_NEXT: Agent:172.16.1.94, First OID:
    .1.3.6.1.2.1.17.4.3.1.2.0.48.171.12.15.223
```

Although this problem does not hinder discovery, you may wish to contact the system vendor to see if a fix is available for the vendor's implementation of the SNMP protocol.

**Note:** By default, InCharge is configured to accept additional non-conforming SNMP messages because, in the *runcmd_env.sh* file, the value of the SM_SNMP_BUG_COMPATIBLE environment variable is set to TRUE. For information about environment variables, see the *InCharge System Administration Guide*.

# System Does Not Support SNMP

Certain systems, such as desktop computers or printers, do not usually support SNMP. The discovery process assumes that a system supports SNMP. If a system does not support SNMP, the system is either not discovered or is placed on the Pending Devices List with a description that the SNMP request failed.

By default, systems found using autodiscovery that do not support SNMP are removed from the discovery queue. Systems imported from a seed file or through **Add Agent** that do not support SNMP are always added to the Pending Devices List.

If you wish to include autodiscovered, non-SNMP systems in the InCharge topology, set the value of the ShowPendingNONSNMP parameter to TRUE in the *discovery.conf* file. For information about editing the *discovery.conf* file, see *Description of discovery.conf* on page 105.

**Note:** Note that when autodiscovery is enabled, setting ShowPendingNONSNMP to TRUE can result in a large number of non-SNMP systems being added to the Pending Devices List.

When the value of ShowPendingNONSNMP is set to TRUE, candidate systems that do not support SNMP are placed on the Pending Devices List. For information about managing systems on the Pending Devices List, see *Managing Systems on the Pending Devices List* on page 40.

# Incorrect Read Community String

For errors concerning an incorrect read community string, you should examine the read community string and correct it if necessary. In the Notification Log Console, you can determine what community string is being used by double-clicking the notification and viewing the Details tab in the Notification Properties dialog.

- For autodiscovery, one or more read community strings can be specified in a discovery filter or in the Topology tab of the Domain Manager Administration Console. For information about how to change community strings, see *Adding and Removing Read Community Strings* on page 93.
- For manual discovery, a read community string can be specified in the seed file or in the Add Agent dialog box for each seed system, or one or more read community strings can be specified in the Topology tab of the Domain Manager Administration Console.

After correcting the read community string, add the system to the topology with the **Add Agent** command. You do not have to restart autodiscovery or re-import the entire seed file just to discover this one system.

# Duplicate IP Address Errors

When two or more systems with the same IP address are discovered, InCharge initiates the following actions:

- Classifies the IP address as a DuplicateIP
- Disables all other IP-related notifications for the IP address
- Generates a notification regarding the duplicate IP address

In the Notification Log Console, you can determine which systems have the duplicate IP address by double-clicking the DuplicateIP notification and viewing the Details tab in the Notification Properties dialog.

To correct a duplicate IP address error, follow these steps:

1. Reconfigure your systems to use unique IP addresses (no duplicates).
2. Use the Domain Manager Administration Console to delete the DuplicateIP object from the Topology Browser view: select the object, right-click, and choose **Delete**.
3. Rediscover all systems that contained duplicate IP addresses.

If you choose not to correct the problem, you can clear the DuplicateIP notification by unmanaging the duplicate IP address. To do so, select the object from the Topology Browser view, right-click, and choose **Unmanage**. For more information about unmanaging a network element, see *Managing and Unmanaging Topology Elements* on page 98.

**Note:** If an IP address has already been discovered on one system and the same IP address is then found to be configured as the only SNMP agent address on another system, InCharge will not discover the second system.

# Insufficient Number of Volume Licenses

A license file for the InCharge IP Management Suite may contain three different types of licensing: application licensing, feature licensing, and volume licensing. Volume licensing requires a license for each discovered system. Volume licensing applies to InCharge IP Availability Manager but not to InCharge IP Performance Manager or InCharge Discovery Manager.

Availability Manager will retrieve blocks of volume licenses from the license server as needed when discovering systems. If Availability Manager discovers more systems than it is licensed to discover, it will generate the OutOfLicense notification to indicate that no more volume licenses are available and that no additional systems will be discovered.

For Availability Manager and Performance Manager running as a single process, Availability Manager and Performance Manager will retrieve blocks of volume licenses from the license server as needed when discovering systems. If together Availability Manager and Performance Manager discover more systems than Availability Manager is licensed to discover, Availability Manager will generate the OutOfLicense notification to indicate that no more volume licenses are available and that no additional systems will be discovered.

**Note:** When Availability Manager and Performance Manager are running as separate processes, there is no limit on the number of systems that Performance Manager can discover.

When the OutOfLicense is notified, Availability Manager (or Performance Manager) will complete the discovery of any system (and its components) in the midst of being discovered, including all virtual routers configured for a router or switch that implements virtual routers. When a router or switch implementing virtual routers is discovered, the chassis-based router/switch is counted as a system, and each virtual router is counted as a system. Any new candidate discovery systems are added to the Pending Devices List in the UNDISCOVERED state, and the message *Out of license* appears below the Discovery Status section of the Discovery Progress window.

If you receive an OutOfLicense notification, contact SMARTS Professional Services to acquire a new license file having a greater number of volume licenses. For the procedure to install a license file, see the *InCharge IP Management Suite Installation Guide*. For more information about license management, see the *InCharge System Administration Guide*.

# Discovering a System with a Hostname That Does Not Resolve to an IP Address

For a discovery candidate system identified by its hostname, the hostname may resolve to multiple IP addresses. If InCharge cannot resolve the hostname to one of the first several IP addresses that the name resolution service—provided by the local operating system—finds, InCharge does not discover the system. Instead, the system is placed on the Pending Devices List, and subsequent attempts to discover it will also fail.

Discovery cannot resolve the hostname for a system when one of the following is true:

- The name resolution service returns more IP addresses than the operating system allows.
- None of the IP addresses returned by the operating system result in InCharge reaching the system's SNMP agent.

If you are using a seed file, you can solve the hostname resolution problem by simply replacing a system's hostname with an accessible IP address of the system's SNMP agent. If making this replacement is impractical because your seed file is generated automatically or because you are importing topology from an external source, read the following discussions for UNIX and Windows to resolve the problem

## UNIX

All supported UNIX systems return up to 35 addresses for a given name. To avoid the name resolution problem, you must configure your system so that at least one usable IP address is among the 35 addresses returned.

UNIX systems translate names using one of three mechanisms: DNS, NIS or NIS+, or a hosts file. The mechanisms supported on a given system are specified by a configuration file.

The most controllable mechanism is the hosts file, */etc/hosts*. All UNIX implementations that SMARTS has tested will return (up to) the first 35 translations in order of appearance in the file.

DNS translations are more difficult to control. First, they are often updated automatically. Second, DNS servers automatically sort translations based both on internally defined and externally specified criteria.

SMARTS has not been able to determine with certainty how translations made through NIS or NIS+ are ordered.

## Windows

Windows systems provide up to six different sources for specifying name translations. Windows has different limits on the number of translations that the operating system will return, based on the source. Windows systems attempt to translate names from the following six sources:

- HOSTS file - Only one translation, the first one to appear in the file is returned.
- DNS — Multiple translations can be returned. SMARTS has not been able to determine if there is a limit, and if so, what it is.
- WINS — Only one translation is returned; it is not clear how that translation is determined.
- Local broadcast — This source is not generally useful in IP-based networks.
- LMHOSTS file — The treatment of this source of translations appears to be equivalent to that of the HOSTS file.
- NetBIOS names — Windows-based hosts or Windows-based router cards may be named using their NetBIOS names when a DNS lookup fails. To avoid naming confusion, SMARTS recommends that you disable NetBIOS over TCP/IP on Windows-based servers where InCharge applications are running. If it is not possible to disable NetBIOS over TCP/IP, then NetBIOS names should be consistent with DNS names.

If you run InCharge on a Windows system, SMARTS recommends that you avoid discovering systems using names with large numbers of translations. If unavoidable, and you are unable to discover the system, create a single entry for the system in the HOSTS file that specifies the IP address for the SNMP agent.

# Increasing Timeout Values for Discovery Polling

The *discovery.conf* file contains several settings that you can edit to improve performance during topology discovery. However, changing these values beyond the recommended values can adversely affect the discovery process. These settings include:

- defaultICMPAutoRetries
- defaultICMPAutoTimeout

- defaultSNMPAutoRetries
- defaultSNMPAutoTimeout
- defaultRetries
- defaultTimeout

When a timeout occurs because of packet loss in the network, you should increase the number of retries. If the timeout occurs because the network is slow or the system is busy, you should increase the value of the timeout.

**Note:** The defaultICMPAutoRetries, defaultICMPAutoTimeout, defaultSNMPAutoRetries, and defaultSNMPAutoTimeout settings apply to discovery. These settings do not affect the operation of the ICMP poller or the SNMP poller that poll elements to perform analysis.

## ICMP Polling Discovery Settings

Before sending an SNMP request, discovery sends an ICMP poll to the system to make sure it is reachable. The defaultICMPAutoRetries and defaultICMPAutoTimeout control the ICMP poll used during the initial stage of discovery.

Retries is the number of times an ICMP poll is sent to a system after the initial attempt fails. The default value is three. Timeout is the amount of time that should elapse before considering that the previous ICMP poll has failed or timed out. The default value is 500 milliseconds.

The value of the timeout is constant, meaning that the timeout value stays the same between retries. In contrast, the value of the timeout for the SNMP poller doubles after each retry.

## SNMP Polling Discovery Settings

The defaultSNMPAutoRetries and defaultSNMPAutoTimeout settings control SNMP polling during the first phase of the discovery process.

The defaultSNMPAutoRetries is the number of times that the SNMP poller should try to reach a system after the initial attempt fails. The default value is three. The defaultSNMPAutoTimeout is the amount of time that the SNMP poller should wait before it considers an SNMP request to have failed or timed out. The default value is 500 milliseconds.

The defaultRetries and defaultTimeout settings control the SNMP poller during the third phase of the discovery process, when the systems are fully probed and the topology is created.

The defaultRetries is the number of times that the SNMP poller should try to reach a system after the initial attempt fails. The default value is five. The defaultTimeout is the amount of time that the SNMP poller should wait before it considers an SNMP request to have failed or timed out. The default value is 1000 milliseconds.

SMARTS recommends adjusting the timeout and retries settings as follows:

- If random systems are experiencing excessive timeouts, a slow system response time is the most probable cause. In this case, SMARTS recommends increasing the timeout value by 50% and decreasing the number of retries by 1.
- If several systems within the same subnet are experiencing excessive timeouts, a high packet loss rate is the most probable cause. In this case, SMARTS suggests increasing the number of retries by 1, and decreasing the timeout value by 50%.

You should not increase the value of the timeout beyond 8000 milliseconds. The timeout doubles with each retry. Therefore, an initial value of 8000 milliseconds with the default number of retries (five) results in a cumulative timeout interval of 504000 milliseconds (more than eight minutes) during the discovery process. In addition, because increasing the number of retries by one effectively doubles the total timeout, you should not increase both the timeout and the number of retries.

# 7

# Working with the Managed Topology

This chapter describes additional configuration tasks related to InCharge topology and discovery. These tasks include:

- Adding and removing read community strings
- Extracting a seed file from a Domain Manager
- Saving the topology to a file
- Rediscovering elements in the topology
- Managing or unmanaging topology elements

All topology related operations described in this chapter are performed using the Domain Manager Administration Console.

## Adding and Removing Read Community Strings

You can specify the read community strings used by discovery in several different places:

- Discovery filter
- Seed file
- **Add Agent** command

If one or more read community string are specified for a discovery filter, those community strings are used when polling the systems that match an IP address within the IP address range of the discovery filter. For information about specifying read community strings for a filter, see *Creating a Discovery Filter* on page 63.

If no read community string is specified in a discovery filter, or no community string is specified in the seed file or through the **Add Agent** command, or the specified community string is not correct, InCharge chooses a community string in the following manner:

1. InCharge tries any community string(s) listed in the Read Community Strings field (Topology tab) of the Domain Manager Administration Console.
2. If there are no community strings in the Read Community Strings field, InCharge reads the default read community string in the *discovery.conf* file.
3. If no read community strings are found for a system, InCharge notifies a DiscoveryError or places the system on the Pending Devices List.

## How To Add Read Community Strings

To add a community string to the Read Community Strings field, follow these steps:

1. In the left panel of the Domain Manager Administration Console, click the Domain Manager name to display a multiple-tab window in the right panel of the console.
2. Select the Topology tab.
3. Type a community string in the Community String field.
4. Click **Add**. By default, you can list up to four community strings in the Read Community Strings field.
5. Click **Apply** to make the new string take effect.

The changes that you make to the Read Community Strings field are applied to systems discovered after the changes are made. Systems already discovered are not affected.

## How To Delete Read Community Strings

To delete a community string from the Read Community Strings field, follow these steps:

1. In the left panel of the Domain Manager Administration Console, click the Domain Manager name to display a multiple-tab window in the right panel of the console.
2. Select the Topology tab.
3. Select the community string to be removed in the Read Community Strings field.
4. Click **Remove**, then **Apply**, to remove the selected community string.

The default community string *public* is automatically included in the Read Community Strings field. If you do not want InCharge to use the *public* community string, you must remove *public* here and remove *public* from the defaultReadCommunity parameter in the *discovery.conf* file. For information about the defaultReadCommunity parameter, see *Description of discovery.conf* on page 105.

## How To Specify Additional Read Community Strings

If you need to specify additional read community strings, see the MaximumCommunities parameter described in *Description of discovery.conf* on page 105.

# Extracting a Seed File From a Domain Manager

You can retrieve a list of the IP addresses and read community strings of the SNMP agents managed by InCharge.

To obtain a list of SNMP agents, invoke the *sm_tpmgr* utility on the host where the Domain Manager is running. This utility produces a text file named *seedfile.log* that lists the IP address of the SNMP agents in one column and their corresponding read community strings in a second column. For SNMP V3 agents, of course, no read community strings appear in the second column.

To obtain a list of SNMP agents and their read community strings, invoke the following command from the ***BASEDIR****/smarts/bin* directory:

```
% ./sm_tpmgr -s INCHARGE-AM --dump-agents --output=seedfile
```

The `-s` argument specifies the name of the Domain Manager. The `--dump-agents` argument extracts the information regarding the SNMP agents from the Domain Manager. The `--output=seedfile` argument creates a file named *seedfile.log* in the ***BASEDIR****/smarts/local/logs* directory. More information regarding the *sm_tpmgr* utility is available with the `--help` argument.

# Automatic Saving of Topology

Every six hours, an InCharge IP application automatically saves important topology and configuration information to a repository file in the ***BASEDIR****/smarts/local/repos/icf* directory. The configuration information pertains to the groups and settings applied to the managed systems. When an InCharge application is restarted, it loads the saved information from the repository file.

The name of the repository file is taken from the name of the Domain Manager. For example, if INCHARGE-AM is the name of the Domain Manager, its repository file would be *INCHARGE-AM.rps*.

To manually save the topology to the repository file, select *Topology > Save Topology* in the Domain Manager Administration Console.

# Rediscovering Topology Elements

Discovery can be configured to automatically rediscover the managed topology. Scheduling a regular full discovery reveals any changes in the configuration or connectivity of the managed systems and keeps the topology current. For example, if a card is added to a switch, InCharge must rediscover the switch so that it can add the new card to the topology.

In addition to scheduled discovery updates, you can also manually rediscover a selected system.

## Discovery Intervals

The parameters that control discovery are located on the Topology tab of the Domain Manager Administration Console. These parameters determine how often the Domain Manager discovers systems on the Pending Devices List and how often the Domain Manager performs a full discovery of the entire topology.

### Pending Discovery Interval

The Pending Discovery Interval controls how often InCharge attempts to discover the systems on the Pending Devices List. The interval is calculated from the time the previous discovery started.

To change the Pending Discovery Interval, follow these steps:

1. In the left panel of the Domain Manager Administration Console, click the Domain Manager name to display a multiple-tab window in the right panel of the console.
2. Select the Topology tab.
3. Type a new value in the Discovery Pending Interval field.
4. Select a time interval from the drop-down menu.
5. Click **Apply**.

To manually invoke discovery of systems on the Pending Devices List, select *Topology > Discover Pending* in the Domain Manager Administration Console.

### Full Discovery Interval

The Full Discovery Interval determines how often InCharge attempts to rediscover the entire managed topology. Full Discovery is disabled by default.

To enable Full Discovery and specify a Full Discovery Interval, follow these steps:

1. In the left panel of the Domain Manager Administration Console, click the Domain Manager name to display a multiple-tab window in the right panel of the console.
2. Select the Topology tab.
3. Click the *Enable Full Discovery* checkbox.
4. Type an interval, and select the time interval from the drop-down list.
5. Click **Apply**.

To manually invoke a full discovery, select *Topology > Discover All* in the Domain Manager Administration Console.

## Manual Discovery Update

There are several methods for manually invoking a discovery from the Domain Manager Administration Console.

- Choose a system and then select *Topology > Rediscover* to initiate a discovery of the selected system. You can also rediscover a system by right-clicking it and selecting **Rediscover** from the drop-down menu.
- Select *Topology > Discover Pending* to initiate a discovery of the systems on the Pending Devices List.
- Select *Topology > Discover All* to initiate a discovery of all managed systems, including those on the Pending Devices List.
- A single system on the Pending Devices List can be rediscovered by right-clicking the system and selecting **Rediscover** from the drop-down menu.

You can also invoke a full discovery and a pending discovery using the *sm_tpmgr* utility. Doing so is useful if you want to schedule rediscovery or a pending discovery at a regular time using the *sm_sched* utility. Invoke both commands from the ***BASEDIR****/smarts/bin* directory on the host where the Domain Manager is running:

```
% ./sm_tpmgr -s <domain_manager> --discover-all
```

```
% ./sm_tpmgr -s <domain_manager> --discover-pending
```

# Managing and Unmanaging Topology Elements

A *managed element* refers to an element whose IsManaged attribute has a value of TRUE, and an *unmanaged element* refers to an element whose IsManaged attribute has a value of FALSE. A managed element is monitored using ICMP and SNMP (unless a different access mode is specified) to determine the element's status and connectivity. An unmanaged element is not polled or rediscovered but remains present in the topology.

The Manage and Unmanage commands enable you to control whether an element is monitored by the Domain Manager. Unmanaging a system may be useful when, for example, a switch or a card is taken offline for maintenance.

**Note:** The Manage and Unmanage commands are recursive. For example, if you unmanage a switch, all of the ports and cards that belong to the switch are unmanaged.

There are two methods you can use to control the managed status of elements.

- You can set the managed status of individual elements using the Manage and Unmanage commands. For information regarding these commands, see *Managing and Unmanaging Individual Elements* on page 101.
- You can use the Interface Management Policy provided with InCharge IP Availability Manager and InCharge IP Performance Manager to set the managed status for interfaces that belong to System Resource Groups. The Interface Management Policy, accessible through the Polling and Thresholds Console, enables you to set a system-level policy to unmanage interfaces. By default, this policy is not active and all interfaces in the topology are managed. For information regarding this setting, see the *InCharge IP Availability Manager User's Guide* or the *InCharge IP Performance Manager User's Guide*.

**WARNING:** If you are using one of the InCharge IP adapters described in the *InCharge IP Adapters User's Guide*, you must allow the third-party source to control the managed status of network elements. When the adapter's topology reader synchronizes the topology of InCharge with the topology of the third-party source, the topology reader must be able to overwrite the managed status of elements in the InCharge topology.

## Determining the Managed Status of an Element

The value of the IsManaged attribute shows the managed status of an element. IsManaged is a Boolean attribute: TRUE when an element is managed, and FALSE when the element is unmanaged.

To check the value of an element's IsManaged attribute, follow these steps:

1. In the Domain Manager Administration Console, select the element in the left panel of the console to display the element's properties window in the right panel of the console.
2. In the element's properties window, select the Attributes tab and locate the IsManaged attribute. The value of this attribute indicates the managed status of the element.

## Types of Elements That Can Be Managed or Unmanaged

You can set the managed or unmanaged status for the following types of elements:

- Services such as SNMP agents
- Service Access Points such as IP interfaces
- Systems such as routers, switches, and hosts
- Cards
- Logical devices such as processors, interfaces, and ports

The ability to unmanage IP interfaces enables you to develop flexible management policies. When you unmanage an IP interface, the underlying physical interface continues to be managed because the status of the IP interface is determined using ICMP, and the status of the physical interface is determined using SNMP.

You can unmanage the IP interfaces on systems where ICMP pings are not allowed because, for example, of a firewall. When you unmanage these IP interfaces, InCharge no longer polls them using ICMP. More importantly, however, is that InCharge still manages the underlying physical interfaces through SNMP.

## Rules Governing Manage and Unmanage

The Manage and Unmanage operations are recursive. When you unmanage an element, all the services and other elements that are functionally dependent on that element become unmanaged.

The following list of rules determines whether an element can be managed or unmanaged and explains the recursive nature of the Manage and Unmanage commands.

- When you unmanage a system, a service that is hosted by or is a part of that system also becomes unmanaged. For example, a system such as a router hosts an SNMP agent service. If you unmanage a router that hosts an SNMP agent service, the SNMP agent service is also unmanaged.
- When you unmanage a system, a logical device that is a part of that system becomes unmanaged. Logical devices include elements such as fans, temperature and voltage sensors, processors, ports, and interfaces. If you unmanage a system such as a router, the interfaces and sensors that are part of that router are not managed nor can they be managed while the parent object is unmanaged. Likewise, if you manage a system, all of its components are managed except those that are explicitly unmanaged.
- When you add a logical device or service to an unmanaged system, that logical device or service automatically becomes unmanaged. For example, if an interface is added to an unmanaged host, the new interface is also unmanaged.
- When you unmanage a service, the service access point that is associated with that service becomes unmanaged.
- When you unmanage a logical device or a service access point, any other service access point that is layered over that logical device or service access point becomes unmanaged. For example, if you unmanage an interface, the sub-interfaces or IP addresses layered over that interface are also unmanaged.
- When you unmanage a card, the managed state of any sub-cards, network adapters realized by the card, or systems packaged by the card remain managed. As such, monitoring and analysis of these items continues.

## Managing and Unmanaging Individual Elements

To manage or unmanage an element using the Manage and Unmanage commands, follow these steps:

1. In the Domain Manager Administration Console, right-click the element that you want to manage or unmanage.
2. Select **Manage** or **Unmanage** from the drop-down menu.
3. Select *Topology > Rediscover*.

# Removing Systems from the Topology

To remove a system from the topology, follow these steps:

1. In the Domain Manager Administration Console, right-click the system to be deleted.
2. Select **Delete** from the drop-down menu.

When you remove a system, all of its components and connections are also removed.

# 8

# Discovery Configuration Files

This chapter provides the proper method for editing configuration files, lists the configuration files relevant to discovery, and describes their parameters. These configuration files include:

- *discovery.conf*
- *partition.conf*
- *user-defined-connections.conf*

These files are installed in the ***BASEDIR****/smarts/conf/discovery* directory. For all InCharge configuration files, lines that begin with a pound sign (#) are comments.

# Modifying InCharge Files

As part of the InCharge deployment and configuration process, you will need to modify certain files. User modifiable files include InCharge tool scripts, configuration files, rule set files, and templates. Original versions of these files are installed into appropriate subdirectories under the ***BASEDIR****/smarts/* hierarchy. For example, on UNIX operating systems the original versions of Global Manager configuration files are installed to */opt/InCharge6/SAM/smarts/conf/ics.*

To edit a user modifiable file, create a local copy of the file in ***BASEDIR****/smarts/local* or one of its subdirectories. For example, a modified *ics.conf* file should be saved to */opt/InCharge6/SAM/smarts/local/conf/ics.* InCharge software is designed to first search for user modifiable files in ***BASEDIR****/smarts/local* or one of its subdirectories. If a modified version of a file is not found in the local area, InCharge software then searches appropriate nonlocal directories.

**Note:** Original versions of files may be changed or updated as part of an InCharge software upgrade. However, files located in ***BASEDIR****/smarts/local* are always retained during an upgrade.

To facilitate proper file editing, SMARTS provides the *sm_edit* utility with every InCharge product suite. When used to modify an original version of a file, this utility automatically creates a local copy of the file and places it in the appropriate location under ***BASEDIR****/smarts/local.* This ensures that the original version of the file remains unchanged. In both UNIX and Windows environments, you can invoke *sm_edit* from the command line. Optionally, you can configure Windows so that *sm_edit* is automatically invoked when user-modifiable files are double-clicked in Windows Explorer.

To invoke the *sm_edit* utility from the command line, specify the path and the name of the file you want to edit under ***BASEDIR****/smarts.* If multiple InCharge products are running on the same host, you should ensure that you invoke *sm_edit* from the *bin* directory of the product suite whose files you wish to edit. For example, to edit the configuration file for the Global Manager, you invoke the *sm_edit* utility as follows:

```
# /opt/InCharge6/SAM/smarts/bin/sm_edit conf/ics/ics.conf
```

The *sm_edit* utility automatically creates a local copy of the *ics.conf* file in the ***BASEDIR****/smarts/local/conf/ics* directory, if necessary, and opens the file in a text editor. If a local version of the file already exists, the *sm_edit* utility opens the local version in a text editor. In addition, *sm_edit* creates any necessary directories.

For more information about how to properly edit user modifiable InCharge files and how to use the *sm_edit* utility, refer to the *InCharge System Administration Guide*.

# Description of discovery.conf

The *discovery.conf* file includes a number or parameters, the values of which affect the whole discovery process. In some cases, for parameters such as SNMPPort, you can override the value specified in *discovery.conf* by specifying an alternate value in a discovery filter, seed file, or the **Add Agent** command.

To open *discovery.conf* and create a local copy, invoke the following command from the ***BASEDIR****/smarts/bin* directory.

```
% ./sm_edit conf/discovery/discovery.conf
```

After you edit the *discovery.conf* file, the Domain Manager must be restarted for the changes to take effect.

Table 18 describes the parameters in *discovery.conf*.

| PARAMETER | DEFAULT VALUE | DESCRIPTION |
|---|---|---|
| ipExcludeList | | Use the ipExcludeList to prevent specific systems, or a range of systems identified by the IP network, from being added to the topology. You can use wildcards, described in *Wildcards* on page 115, to specify a matching pattern.<br>The following example excludes all systems from the specified IP network:<br>`ipExcludeList += "10.10.9.*"`<br>The following example excludes system from one IP network and a range of systems from a second IP network:<br>`ipExcludeList += "10.10.1.*\|10.10.<2-10>.*"` |
| ShowPendingNONSNMP | FALSE | Controls whether non-SNMP systems found by autodiscovery or topology import are added to the Pending Devices List or removed from the discovery queue. A non-SNMP system is one that responds to ICMP polls but does not respond to SNMP polls.<br>A value of FALSE removes non-SNMP systems from the discovery queue.<br>A value of TRUE adds non-SNMP systems to the Pending Devices List. For more information, see *System Does Not Support SNMP* on page 84. |
| LogDiscoveryProgress | FALSE | Enables verbose logging of discovery-related messages to the servers log file. |
| defaultSNMPPort | 161 | Specifies the default port used by the SNMP poller. You can override this value in a discovery filter, seed file, or **Add Agent** command. |
| defaultRetries | 5 | Number of SNMP retry polls to use during Phase 3 of discovery.<br>For a description of Phase 3, see *Phase 3: Probe the System Destined for the Topology* on page 30.<br>For recommendations about changing this value, see *SNMP Polling Discovery Settings* on page 90. |
| defaultTimeout | 1000 milliseconds | Timeout for an SNMP poll during Phase 3 of discovery.<br>For a description of Phase 3, see *Phase 3: Probe the System Destined for the Topology* on page 30.<br>For recommendations about changing this value, see *SNMP Polling Discovery Settings* on page 90. |

| PARAMETER | DEFAULT VALUE | DESCRIPTION |
|---|---|---|
| defaultICMPAutoRetries | 3 | Number of ICMP polls to try during Phase 1 of discovery.<br>For a description of Phase 1, see *Phase 1: Perform Initial Polling of the Candidate System* on page 26.<br>For recommendations about changing this value, see *ICMP Polling Discovery Settings* on page 90. |
| defaultICMPAutoTimeout | 500 milliseconds | Timeout for an ICMP poll during Phase 1 of discovery.<br>For a description of Phase 1, see *Phase 1: Perform Initial Polling of the Candidate System* on page 26.<br>For recommendations about changing this value, see *ICMP Polling Discovery Settings* on page 90. |
| defaultSNMPAutoRetries | 3 | Number of SNMP polls to try during Phase 1 of discovery.<br>For a description of Phase 1, see *Phase 1: Perform Initial Polling of the Candidate System* on page 26.<br>For recommendations about changing this value, see *SNMP Polling Discovery Settings* on page 90. |
| defaultSNMPAutoTimeout | 500 milliseconds | Timeout for an SNMP poll during Phase 1 of discovery.<br>For a description of Phase 1, see *Phase 1: Perform Initial Polling of the Candidate System* on page 26.<br>For recommendations about changing this value, see *SNMP Polling Discovery Settings* on page 90. |
| numberProbeThreads | 10 | Controls the number of discovery threads used by the discovery process. |

| PARAMETER | DEFAULT VALUE | DESCRIPTION |
|---|---|---|
| ICMPSleepTime | 0 | Controls the number of ICMP polls sent during Phase 1 of discovery.<br>During the initial discovery phase, discovery sends large bursts of ICMP polls. If ICMP has a low priority, the network is congested, or the routers on the path are experiencing high processor utilization, a number of these ICMP packets may be dropped. As a result, discovery receives a limited number of ICMP responses which causes a number of systems that might otherwise be successfully discovered to be placed on the Pending Devices List as UNDISCOVERED.<br>Other symptoms of packet loss include DiscoveryError notifications for systems that were previously discovered successfully and the full topology is not discovered after two full discoveries.<br>Increasing the sleep time increases the amount of time between ICMP polls. You can specify a value between 0 and 100 milliseconds. A value of 100 milliseconds translates to 10 ICMP polls per second.<br>Specifying a long sleep time can significantly prolong the discovery process. Estimate the total additional time by multiplying the sleep time by the number of IP addresses to be processed. |
| defaultReadCommunity | public | Read community string used by InCharge when no community string is specified in the Read Community Strings field described in *Adding and Removing Read Community Strings* on page 93.<br>If you do not want discovery to use a default community string during discovery, you must remove *public* from the Read Community Strings field and from this parameter. |
| MaximumCommunities | 4 | Number of community strings that can be specified in the Read Community Strings field described in *Adding and Removing Read Community Strings* on page 93. |

| PARAMETER | DEFAULT VALUE | DESCRIPTION |
|---|---|---|
| MaximumHostBits | 8 | Determines the number of host bits in the netmask used by autodiscovery to find additional discovery candidate systems. The default value of 8 corresponds to a netmask of 255.255.255.0.<br>If, for example, your network includes systems with a netmask of 255.255.254.0, these systems will not be found using autodiscovery.<br>Modify the default value of the MaximumHostBits parameter with caution because as you increase the value, so do you increase the number of ICMP polls generated during every discovery.<br>SMARTS recommends setting the value only as high as needed. If necessary, modify the value during the initial topology discovery and once the devices are discovered lower the value so that subsequent pending and full discoveries do not attempt to look beyond the default netmask (this will not affect systems already discovered).<br>For example, the default value of 8 provides a maximum of 254 host IP addresses. A value of 16 provides a maximum of 65,534 host IP addresses.<br>192.168.1.1/24 = 8 host bits<br>192.168.1.1/18 = 14 host bits<br>192.168.1.1/16 = 16 host bits<br>192.168.1.1/8 = 24 host bits<br>Valid values range from 2 - 24. If a value greater than 24 is specified, a value of 8 is used. |

| PARAMETER | DEFAULT VALUE | DESCRIPTION |
|---|---|---|
| NameFormat | TM_USEAUTONAME | Controls how discovery determines the Name attribute of a system. Valid values are TM_USEAUTONAME and TM_USESEEDNAME.<br>TM_USEAUTONAME means that discovery uses the host operating system to determine the system name. TM_USESEEDNAME means that discovery uses the seed name to determine the system name.<br>For details, see *How Discovery Names a System* on page 41. |
| AccessAddressFormat | AUTOADDRESS | Determines how the discovery process obtains an IP address to communicate with discovered systems. This address is used for ICMP and SNMP polls during discovery and for monitoring and analysis. Valid values are AUTOADDRESS and LOOPBACKADDRESS.<br>AUTOADDRESS means that all the IP addresses of a device are polled using ICMP. If a device has more than ten IP addresses, only ten of the IP addresses are polled using SNMP. The addresses to be polled by SNMP are chosen as follows:<br>• IP address is managed and is not a duplicate<br>• InterfaceMode of the IP, if any, is NORMAL<br>• If the system has more than ten IP addresses, the first ten IP addresses are chosen in the following manner: IP addresses of type SOFTWARELOOPBACK, ETHERNET, TOKENRING, and GENERIC are chosen first in the order that the IP addresses appear in the ipAddrTable. IP addresses of type GENERIC must not have a netmask of 255.255.255.252. The remaining IP addresses are chosen in the order that they appear in the ipAddrTable.<br>LOOPBACKADDRESS means that discovery uses the loopback address for both ICMP and SNMP for monitoring and discovery. (MIB-II ifType 24). Note that discovery cannot determine the loopback address until after the system has been discovered.<br>If a qualified IP address is not found, the last known SNMP address is used and a DiscoveryError is notified. The DiscoveryError contains the message: Qualified access address not found. |

| PARAMETER | DEFAULT VALUE | DESCRIPTION |
|---|---|---|
| DisplayNameFormat | AUTOASSIGNED | Controls how discovery determines the DisplayName attribute of a system. Valid values are AUTOASSIGNED and MIBIISYSNAME.<br>AUTOASSIGNED means that the DisplayName of a system is set to the value of the Name attribute of the system. (Using the NameFormat parameter, described previously in this table, you can determine how the value of the Name attribute is determined.) Also, for AUTOASSIGNED, you can change the value of DisplayName using the Domain Manager Administration Console, dmctl, or an ASL script; the value of DisplayName is retained across any subsequent discoveries.<br>MIBIISYSNAME means that the DisplayName of a system is set to the value of the MIB-II sysName variable. Any change to the DisplayName using the Domain Manager Administration Console, dmctl, or an ASL script is not retained across subsequent discoveries. |
| DuplexAssumed | FALSE | Controls whether to calculate the current utilization for *all* network adapters (ports or interfaces) in the topology. Valid values are TRUE and FALSE.<br>A value of FALSE means that InCharge will calculate *only* current utilization for those ports or interfaces that can be determined as full-duplex or half-duplex by reading the Enterprise MIB, ETHERLIKE-MIB, and Neighbor MIB. Any port or interface having an unspecified duplex mode setting will be ignored.<br>A value of TRUE means that InCharge will calculate current utilization for all ports and interfaces in the topology regardless of whether their duplex mode can be determined by checking the MIBs. For any port or interface whose duplex mode cannot be determined by checking the MIBs, InCharge sets the port's/interface's DuplexMode attribute to FULLDUPLEX and its DuplexSource attribute to ASSUMED. The one exception is the 10 megabit (Mb) Ethernet adapter, for which InCharge sets the adapter's DuplexMode attribute to HALFDUPLEX.<br>*Be aware that calculating current utilization for all ports and interfaces requires considerable system resources. Ensure that your server is sized properly to accommodate the size of your environment, as described in the InCharge IP Deployment Guide.*<br>For more information about duplex mode and current utilization calculations, see the *InCharge IP Performance Manager User's Guide*. |

**Table 18:** Parameters of discovery.conf

# Description of partition.conf

You can use the *partition.conf* file to set the DisplayName of Partition elements in the managed topology. By default, discovery uses the value of the Name attribute to set the value of DisplayName. Discovery names the partitions by assigning them a number, starting with 0, and incrementing this value by one for each partition. The partition number is followed by the name of the Domain Manager that created the partition. For example, the name of the first partition created by the INCHARGE-AM Domain Manager is Partition-0/INCHARGE-AM, the second is Partition-1/INCHARGE-AM, and so on.

To set the DisplayName of a partition, specify the hostname or IP address of a partition member followed by its name. You should specify a name or IP address that is already known to discovery. Specify one hostname or IP address and name combination on each line.

The following example sets the DisplayName of the partition that includes the system with an IP address of 10.64.1.1.

```
10.64.1.1 Servers in LA
```

If you specify multiple names for a partition, which is done by providing separate entries, the last name listed is used.

Changes to *partition.conf* do not require that you restart the Domain Manager. The *partition.conf* file is loaded each time discovery performs post-processing.

# Description of user-defined-connections.conf

You can use the *user-defined-connections.conf* file to add Layer and Layer 3 connections to the managed topology. Doing so may be useful in cases where these connections are not discovered because of incomplete information in the MIB. If you know the necessary information, you can create these connections by specifying them in this file. The types of connections that are created include Cable, TrunkCable, and NetworkConnection.

A user-defined connection contains the following four fields, each separated by a line (|):

- The first field is the local network. You can specify the local network by name or IP address.
- The second field is the network adapter on the local device. You can specify the network adapter by its ifIndex, ifDescr, or ifName, as defined in RFC 1213. You can also use the <n>/<m> or <n>.<m> syntax, where <n> is the module number, and <m> is the port number.
- The third field is the remote network device. You can specify the remote network device by name or IP address.
- The fourth field is the network adapter on the remote device. You can specify the network adapter by its ifIndex, ifDescr, or ifName, as defined in RFC 1213. You can also use the <n>/<m> or <n>.<m> syntax, where <n> is the module number, and <m> is the port number.

**Note:** Any elements specified in these four fields must be in the managed topology.

The Domain Manager searches for the network adapter in the order of ifName, ifDescr, and ifIndex. If a network adapter is not found, the specified string is used as interfaceKey/PortKey to find the network adapter.

The following are two examples of user-defined connections:

```
192.168.1.200|Serial0|router2.smarts.com|5|
10.64.1.1|2.4|Core-switch.smarts.com|2/4|
```

Note that the network adapter field for the remote device must end with a line.

# A

# Wildcards

A wildcard pattern is a series of characters that are matched against incoming character strings. You can use these patterns when you define pattern matching criteria.

Matching is done strictly from left to right, one character or basic wildcard pattern at a time. Basic wildcard patterns are defined in Table 19. Characters that are not part of match constructs match themselves. The pattern and the incoming string must match completely. For example, the pattern *abcd* does not match the input *abcde* or *abc*.

A compound wildcard pattern consists of one or more basic wildcard patterns separated by ampersand (&) or tilde (~) characters. A compound wildcard pattern is matched by attempting to match each of its component basic wildcard patterns against the entire input string. For compound wildcard patterns, see Table 20.

If the first character of a compound wildcard pattern is an ampersand (&) or tilde (~) character, the compound is interpreted as if an asterisk (*) appeared at the beginning of the pattern. For example, the pattern ~*[0-9]* matches any string not containing any digits. A trailing instance of an ampersand character (&) can only match the empty string. A trailing instance of a tilde character (~) can be read as “except for the empty string.”

**Note:** Spaces are interpreted as characters and are subject to matching even if they are adjacent to operators like "&".

| CHARACTER | DESCRIPTION |
|---|---|
| Note: Spaces specified before or after wildcard operators are interpreted as characters and are subject to matching. | |
| ? | Matches any single character.<br>For example, *server?.smarts.com* matches *server3.smarts.com* and *serverB.smarts.com*, but not *server10.smarts.com*. |
| * | Matches an arbitrary string of characters. The string can be empty.<br>For example, *server*.smarts.com* matches *server-ny.smarts.com* and *server.smarts.com* (an empty match). |
| [set] | Matches any single character that appears within [set]; or, if the first character of [set] is (^), any single character that is *not* in the set. A hyphen (-) within [set] indicates a range, so that [*a-d*] is equivalent to [*abcd*]. The character before the hyphen (-) must precede the character after it or the range will be empty. The character (^) in any position except the first, or a hyphen (-) at the first or last position, has no special meaning.<br>Example, *server[789-].smarts.com* matches *server7.smarts.com* through *server9.smarts.com*, but not *server6.smarts.com*. It also matches *server-.smarts.com*.<br>Example: *server[^12].smarts.com* does not match *server1.smarts.com* or *server2.smarts.com*, but will match *server8.smarts.com*. |
| <n1-n2> | Matches numbers in a given range. Both *n1* and *n2* must be strings of digits, which represent non-negative integer values. The matching characters are a non-empty string of digits whose value, as a non-negative integer, is greater than or equal to *n1* and less than or equal to *n2*. If either end of the range is omitted, no limitation is placed on the accepted number.<br>For example, *192.168.<1-100>.10* matches a range of IP addresses from *192.168.1.10* through *192.168.100.10*.<br>Example of an omitted high end of the range: *<50->* matches any string of digits with a value greater than or equal to 50.<br>Example of an omitted low end of the range: *<-150>* matches any value between zero and 150.<br>A more subtle example: The pattern *<1-10>** matches 1, 2, up through 10, with * matching no characters. Similarly, it matches strings like 9x, with * matching the trailing x. However, it does not match 11, because <1-10> always extracts the longest possible string of digits (11) and then matches only if the number it represents is in range. |
| \| | Matches alternatives. For example,"*ab\|bc\|cd*" without spaces matches exactly the three following strings: "*ab*", "*bc*", and "*cd*". A \| as the first or last character of a pattern accepts an empty string as a match.<br>Example with spaces "*ab \| bc*" matches the strings "*ab* " and " *bc*". |
| \ | Removes the special status, if any, of the following character. Backslash (\) has no special meaning within a set ([set]) or range (<n1-n2>) construct. |

**Table 19:** Basic Wildcard Patterns

Special characters for compound wildcard patterns are summarized below.

| CHARACTER | DESCRIPTION |
|---|---|
| & | "And Also" for a compound wildcard pattern. If a component basic wildcard pattern is preceded by & (or is the first basic wildcard pattern in the compound wildcard pattern), it *must* successfully match.<br>Example: *NY*&*Router* matches all strings which contain NY and also contain Router.<br>Example: <1-100>&*[02468] matches even numbers between 1 and 100 inclusive. The <1-100> component only passes numbers in the correct range and the *[02468] component only passes numbers that end in an even digit.<br>Example: *A*|*B*&*C* matches strings that contain either an A or a B, and also contain a C. |
| ~ | "Except" for a compound wildcard pattern (opposite function of &).If a component basic wildcard pattern is preceded by ~, it *must not* match.<br>Example: 10.20.30.*~10.20.30.50 matches all devices on network 10.20.30 except 10.20.30.50.<br>Example: *Router*~*Cisco*&*10.20.30.*~10.20.30.<10-20>* matches a Router, except a Cisco router, with an address on network 10.20.30, except not 10.20.30.10 through 10.20.30.20. |

**Table 20:** Compound Wildcard Patterns

# Index

## E



## F



## G



## H



## I



## L



## M



## N

## O



## P



## R



## S

## T



## U



## V



## W