InCharge™

Service Assurance Manager Configuration Guide

Version 5.0.1

smarts

Cisco Part Number: OL-6118-01

IBM Runtime for AIX

HP-UX Runtime Environment for the Java™ 2 Platform

DataDirect Technologies

Contents

Preface

The purpose of this guide is to provide instructions for the configuration of InCharge Service Assurance Manager (Service Assurance). This document describes the configuration files associated with Service Assurance components and provides detailed instructions regarding the configuration of the Global Manager component.

Intended Audience

This guide is intended for service providers and system administrators who are responsible for configuring Service Assurance. It is not intended for operations personnel responsible for monitoring the results displayed in the Global Console. These users should refer to the *Service Assurance Manager Operator's Guide*.

In addition to the configuration guides for specific components, administrators should also read the *InCharge System Administration Guide*.

Prerequisites

This guide assumes you have the administrative privileges and the necessary experience to properly install and configure network management software. Before you begin an installation, we recommend that you first read *Deploying InCharge Service Assurance Manager* on page 7.

Document Organization

This guide consists of the following chapters:

1. OVERVIEW OF INCHARGE SERVICE ASSURANCE MANAGER	Describes the components of Service Assurance
2. DEPLOYING INCHARGE SERVICE ASSURANCE MANAGER	Describes various deployment scenarios for Service Assurance.
3. CONFIGURATION OVERVIEW OF THE GLOBAL MANAGER	Lists the configuration files associated with Service Assurance and provides a configuration roadmap.
4. SYSTEM CONFIGURATION FOR THE GLOBAL MANAGER	Describes how to perform system configuration tasks.
5. NOTIFICATION CONFIGURATION FOR THE GLOBAL MANAGER	Describes notification lists and how to create them.
6. TOPOLOGY CONFIGURATION FOR THE GLOBAL MANAGER	Describes how to organize topology by creating groups.
7. TOOL CONFIGURATION FOR THE GLOBAL MANAGER	Describes how to create programs that respond to notifications.
A. ICIM CLASSES USED WITH MATCHING CRITERIA	Lists and describes the attributes for the ICIM_Notification and system classes.
B. WILDCARD PATTERNS	Describes wildcards used to create matching patterns.

Table 1: Description of Document Chapters

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
<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

In this document, the term ***BASEDIR*** represents the location where InCharge software is installed. The term ***BASEDIR*** represents the */opt/InCharge<n>* directory for UNIX, the *C:\InCharge<n>* directory for Windows (where <n> represents the InCharge software version number), or your specified path. The InCharge software resides in the ***BASEDIR****/smarts* subdirectory.

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.

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 Installation Guide*
- *InCharge System Admininistration Guide*
- *InCharge Notification Adapters User's Guide*

InCharge Service Assurance Manager Documentation

The following SMARTS documents are relevant to users of InCharge Service Assurance Manager:

- *An Introduction to InCharge Service Assurance Manager*
- *InCharge Service Assurance Manager Operator's Guide*
- *InCharge Service Assurance Manager Configuration Guide*
- *InCharge Service Assurance Manager Open Integration Configuration Guide*
- *InCharge Service Assurance Manager Failover System User's Guide*
- *InCharge Service Assurance Manager User's Guide for Business Impact Manager*
- *InCharge Service Assurance Manager User's Guide for Report Manager*

- *InCharge Service Assurance Manager Web Portal Operator's Guide*
- *InCharge Service Assurance Manager Web Portal Configuration Guide*

InCharge Application Services Manager Documentation

The following SMARTS documents are relevant to users of InCharge Application Services Manager:

- *InCharge Application Services Manager Deployment Guide*
- *InCharge Application Services Manager User's Guide*
- *InCharge Application Services Manager SMART 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 Information Model
ICMP	Internet Control Message Protocol
IDS	Incremental Device Support
IP	Internet Protocol
MIB	Management Information Base
MODEL	Managed Object Definition Language
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 (9:00 A.M. - 6:00 P.M. U.S. Eastern Time).

TECHNICAL SUPPORT:	*support@smarts.com*
SALES:	*sales@smarts.com*
WORLD WIDE WEB:	*http://www.smarts.com*
TELEPHONE:	+1.914.948.6200
FAX:	+1.914.948.6270

You may also contact us at:

SMARTS
44 South Broadway
White Plains, New York 10601 U.S.A

1

Overview of InCharge Service Assurance Manager

This chapter provides a brief overview of the InCharge Service Assurance Manager (Service Assurance) components and modules. In addition, this chapter describes all of the configuration files related to Service Assurance components and modules.

Overview of Service Assurance Components

The successful deployment of Service Assurance requires knowledge of your operations environment and the management tools already in place. You can integrate Service Assurance with third-party tools and existing installations of InCharge applications.

Service Assurance Components

InCharge Service Assurance Manager is composed of the following components.

- Global Manager is the software component at the heart of Service Assurance. The Global Manager consolidates topology and event information from multiple underlying domains and provides a central point for monitoring and managing distributed network domains.

- Global Console is the primary tool for operators. The console enables operators to monitor the state of the managed environment and quickly respond to notifications. The Global Console also provides map and topology views. As the primary tool for operators, the Global Console is typically installed on many hosts.
- Open Integration is a software component that imports topology and event information from third-party applications or products, normalizes the imported data to the InCharge Information Model (ICIM), and provides this data to the Global Manager.

Note: Please refer to the *InCharge Installation Guide* for the hardware and software requirements as well as the procedures for installing Service Assurance.

Service Assurance Modules

You can also add the following modules to enhance the analysis or extend the capabilities of Service Assurance.

- InCharge Business Impact Manager extends the capabilities of Service Assurance to calculate the business impact of events and propagate the impacts to affected business elements (service offerings and subscribers) as discrete notifications that are linked to network elements.
- InCharge Report Manager enables you to save notifications into a relational database for historical reporting. It includes a database schema that is designed to store the information encapsulated in a notification, provides a daily summary of device availability, and generates reports from the database that can be viewed in a Web browser. In addition, Report Manager includes a reporting engine, default reports, and a customized Web application for requesting reports.
- InCharge Web Portal provides a facility for organizations to share needed management data with their end users, and for service providers to share limited information with their customers about their customers' managed network services. It can provide a secure channel of communication and displays only those notifications that are relevant to that specific user. Summary views and notification logs can be made accessible to any user with a Web browser and Internet access.

Configuration Files for Service Assurance

As part of the InCharge deployment and configuration process, you will need to modify certain files. User modifiable files include InCharge startup scripts, 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, original versions of Global Manager configuration files are installed to ***BASEDIR****/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 ***BASEDIR****/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. 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.* For example, to edit the configuration file for the Global Manager, you invoke the *sm_edit* utility as follows:

```
% BASEDIR/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*.

Table 3 lists the configuration files for the Service Assurance components and Table 4 lists the configuration files for Service Assurance modules. Configuration files are located in directories that correspond to the component or module that they configure. For example, the ***BASEDIR****/smarts/local/conf/ics* directory contains configuration files related to the Global Manager and ***BASEDIR****/smarts/local/conf/icoi* directory contains configuration files related to Open Integration.

Table 3 lists the configuration files for Service Assurance components.

Directory	File(s)	Component	Description
actions/client	*SmGetEnv* *SmLaunchPerlScript* *adminconsole* *browser* *ciscoworks* *newconsole* *newicconsole* *pinger* *Reporting*	Global Console	Example client tool scripts. Client tools are invoked from the Global Console by an operator. For more information regarding the configuration of client tools, see *Configuring a Client Tool* on page 99.
actions/server	*ics-closetkt* *ics-opentkt* *ics-ping-IP* *ics-ping-all* *ics-ping-device* *ics-ping-interface* *ics-ping* *ics-telnet* *remedy-closetkt* *remedy-config* *remedy-gettktk* *remedy-opentkt*	Global Manager	Example server and automated tool scripts. Server and automated tools are executed by the Global Manager. Server tools are invoked through the Global Console while automated tools are invoked through sm_adapter. For more information regarding the configuration of server tools, see *Configuring a Server Tool* on page 93 and *Configuring an Automated Tool* on page 97.
classes/unzipped	*globalConsole.html*	Global Console	HTML file used to load the Global Console as a Java applet. This file must be edited to enable maps through the applet. For more information regarding requirements and configuration, see the *InCharge Installation Guide*.
conf	*serverConnect.conf* *clientConnect.conf* *brokerConnect.conf*	All	Security files for configuring access to Service Assurance. For information regarding security, see the *InCharge System Administration Guide*.
	runcmd_env.sh	All	Used to set environment variables for SMARTS software.
	SMARTS.licserv	All	License file that points to the location of the FLEXlm license server.

DIRECTORY	FILE(S)	COMPONENT	DESCRIPTION
conf/console	*tools.conf*	Global Console	File for specifying client tools. Should be configured on each host where an operator invokes tools through the Global Console.
conf/icoi	*icoi.conf*	Open Integration	Configuration file for Open Integration server.
	trap_mgr.conf	Open Integration	File used to define and map incoming SNMP traps to notifications. This file is only required when running the SNMP Trap Adapter.
	trapd.conf	Open Integration	Configuration file for the SNMP Trap Adapter.
conf/ics	*ics.conf*	Global Manager	Configuration file for the Global Manager.
	topology-group.data.template	Global Manager	Example data file for importing groups into the Global Manager.
conf/notifier	*file-notify.conf* *mail-notify.conf* *script-notify.conf* *tibrv-notify.conf* *trap-notify.conf*	Global Manager	Configuration files for file, mail, script, TIBCO, and trap notification adapters.
conf/remedy	*IC_SAM_Final.def*	Global Manager	Sample Remedy schema that includes Service Assurance information.
conf/trapd	*trapd.conf*	Global Manager	Configuration file for the SNMP Trap Forwarder.

Table 3: Configuration Files for Service Assurance Components

For more information regarding the Global Console or Open Integration, refer to the following documents:

- For the Global Console, see the *InCharge Service Assurance Manager Operator's Guide*.
- For Open Integration, see the *InCharge Service Assurance Manager Open Integration Configuration Guide*.

Table 4 lists the configuration files for Service Assurance modules.

DIRECTORY	FILE(S)	MODULE	DESCRIPTION
conf/ics	*service.data.template*	Business Impact Manager	Example data file for importing service topology into the Global Manager.
	weights.conf	Business Impact Manager	File for specifying a weight value, or impact, to topology elements.
conf/sdi/crystal-reports	*availability_summary.rpt* *critical_customers.rpt* *critical_devices.rpt* *event_detail.rpt* *event_detail_groups.rpt* *event_detail_servers.rpt* *event_detail_userdef.rpt* *event_summary.rpt* *incharge_effectiveness.rpt* *open_events.rpt*	Report Manager	Default reports included with InCharge Report Manager
conf/sdi/sdi	*sdi_ics.conf*	Report Manager	File that configures the connection to the Global Manager.
	sdi_odbc.conf	Report Manager	Configuration file that specifies connection to the ODBC driver.
conf/sdi/summary	*sdi_odbc.conf*	Report Manager	Configuration file for the Report Manager's Summary Adapter.
conf/wp	*wp.conf*	Web Portal	Configuration file for InCharge Web Portal.

Table 4: Configuration Files for Service Assurance Components

For more information regarding InCharge Business Impact Manager, InCharge Report Manager, or the InCharge Web Portal, refer to the following documents:

- For Business Impact Manager, see the *InCharge Service Assurance Manager User's Guide for Business Impact Manager*.
- For Report Manager, see the *InCharge Service Assurance Manager User's Guide for Report Manager*.
- For the Web Portal, see the *InCharge Service Assurance Manager Web Portal Configuration Guide* and the *InCharge Service Assurance Manager Web Portal Operator's Guide*.

2

Deploying InCharge Service Assurance Manager

This chapter presents deployment scenarios for InCharge Service Assurance Manager (Service Assurance) and describes the steps required to achieve each deployment. For detailed configuration information, refer to the product-specific documentation.

The components of Service Assurance are described in *Overview of Service Assurance Components* on page 1. This chapter assumes that you are familiar with the purpose of each component and are prepared to install, configure, and operate your InCharge software.

Compatibility with Previous Versions of InCharge Software

This section describes compatibility requirements for version 5.0 or later of Service Assurance with regard to previous versions of Service Assurance and InCharge applications.

Global Manager

The version 5.0 or later Global Manager supports version 4.0.4 or later of InCharge applications and version 1.1 or later of Ciscoworks Device Fault Manager. SMARTS recommends that you upgrade any underlying domains to the most recent versions.

In addition, you can integrate a version 5.0 or later Global Manager with Global Managers from versions 4.5 and 4.6. However, if the Global Managers are deployed in a hierarchical configuration, you should use the 4.5 or 4.6 versions of the Global Manager to feed events and topology to the 5.0 or later version of the Global Manager.

Global Console

Because of improvements to the access control mechanism used for security, version 5.0 or later Global Consoles cannot connect to pre-5.0 Global Managers. Previous versions of the Global Console can connect to version 5.0 or later Global Managers, but are not fully compatible. SMARTS recommends that you maintain two versions of the Global Console until your migration is complete.

Only version 5.0 or later of the Global Console is compatible with the secure broker.

InCharge Broker

Improvements to the access control mechanism used by InCharge require that version 5.0 or later components of Service Assurance use at least version 5.0 of the InCharge Broker. When the version 5.0 or later broker operates in a non-secure mode, the default, it is fully backward compatible with all supported versions of InCharge applications and CiscoWorks Device Fault Manager.

A new feature of the InCharge Broker is that you can run it in secure mode. When the broker runs in secure mode, only version 4.1 or later InCharge applications are supported. If you operate in this environment, you must use the brcontrol utility to register the version 4.1 or later InCharge servers with the InCharge Broker.

In addition, the version 5.0 or later InCharge Broker uses a new file, *brokerConnect.conf*, for providing credentials when it connects to servers.

For more information about security and the InCharge Broker, see the *InCharge System Administration Guide*.

InCharge Report Manager

In versions 4.5 and 4.6 of Service Assurance, InCharge Report Manager was called the SQL Data Interface. Because of changes to the schema and the architecture of Report Manager, you should contact SMARTS Technical Support for help when migrating your existing data to the new schema.

Service Assurance Deployment Scenarios

The following scenarios describe how to setup and configure different Service Assurance components so that they operate as a system. Each scenario describes a slightly different deployment.

Scenario 1: Deploying the Basic Components

The first deployment scenario includes many of the building blocks that appear in the second scenario.

For best results, SMARTS recommends that you install each component on a separate host system. This ensures that each component has enough available resources, and, if a host system were to malfunction, you do not need to install and configure two or more components.

The components used in the first scenario are divided into the following categories:

- Service Assurance Components
  - InCharge Broker
  - Global Manager
  - Global Console
  - InCharge Report Manager
- Underlying Domains
  - InCharge Availability Manager
  - InCharge Performance Manager
  - InCharge Application Services Manager
  - SMARTS Adapters for Application Services Manager

SMARTS recommends that you deploy items from the above list in the following order:

1. InCharge Broker
2. SMART Adapters
3. Underlying domains
4. Global Manager
5. Global Console
6. InCharge Report Manager

The general rule of thumb is to install and configure items from the bottom of the hierarchy first, moving up to the Global Manager. After the Global Manager is installed and configured, you can install components that use the Global Manager as a source, such as the Global Console and InCharge Report Manager. The InCharge Broker is the exception to this rule. The broker should always be installed first as it is used by all InCharge software.

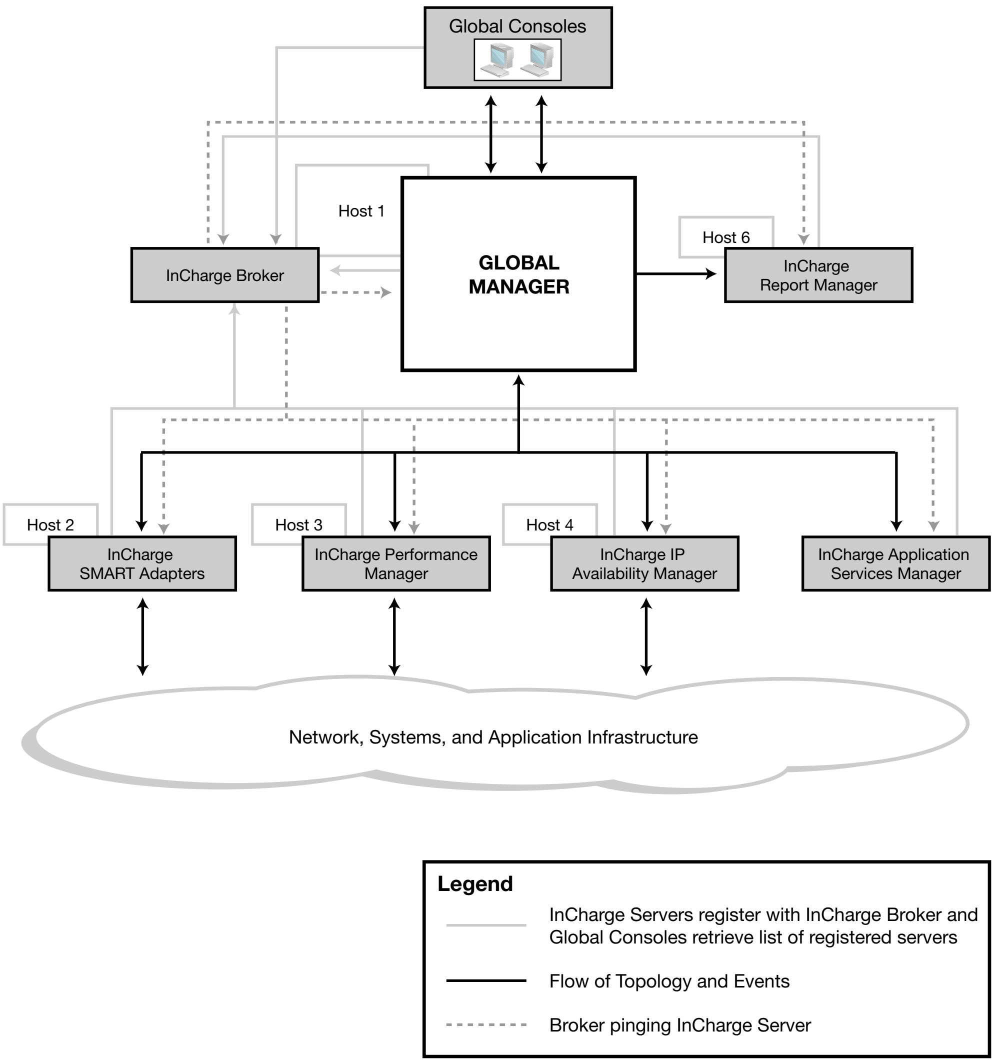
Global Consoles
Host 1
GLOBAL MANAGER
InCharge Broker
Host 6
InCharge Report Manager
Host 2
InCharge SMART Adapters
Host 3
InCharge Performance Manager
Host 4
InCharge IP Availability Manager
InCharge Application Services Manager
Network, Systems, and Application Infrastructure
Legend
InCharge Servers register with InCharge Broker and Global Consoles retrieve list of registered servers
Flow of Topology and Events
Broker pinging InCharge Server

Figure 1: Deployment Scenario 1

Deploying the InCharge Broker

As noted earlier, the latest version of the Global Manager works with a version 5.0 or later InCharge Broker. Because of this, if you are upgrading, you must use at least version 5.0 of the InCharge Broker. SMARTS recommends, however, that you always use the latest version of the broker.

For a new installation, with no existing broker, you can install the broker on any host. The option to run the broker is provided when you install any Service Assurance component.

In this scenario, the InCharge Broker is installed on the same host as the Global Manager.

1. During the installation for the Global Manager, you are prompted for the location where the InCharge Broker is running. To run the broker on this host, enter the default values of "localhost:426".

   If you specify a different host, the broker is not installed.

2. After the installation is complete, you can manually start the broker. Instructions are provided in both the *InCharge Installation Guide* and the *InCharge System Administration Guide*.

If you are already running an InCharge Broker from a previous installation of Service Assurance or an InCharge application, you must use at least version 5.0 of the InCharge Broker.

To upgrade to the latest version of the broker, complete the following steps:

1. Install the latest version of the InCharge Broker, but do not start it.
2. Stop the old InCharge Broker.
3. Copy the repository file (*broker.rps*) from the old InCharge Broker to the ***BASEDIR****/smarts/local/repos/broker* directory on the host where the latest version of the InCharge Broker is installed.
4. Start the latest version of the InCharge Broker.

Using this technique preserves the registration information for any InCharge servers already registered with the previously running broker.

If you decide to deploy a secure broker, read *InCharge System Administration Guide* for instructions on configuring the broker to run in this mode.

Deploying SMART Adapters

SMART Adapters provide an interface between InCharge software and sources of topology and events. A SMART adapter may perform one or more of the following tasks: discover topology as well as event-processing tasks such as normalization, de-duplication, topology association, and aggregation.

In Scenario 1, SMART adapters are deployed to provide events and topology for InCharge Application Services Manager. In this case, the SMART adapters send the topology and event information they collect to the Global Manager. InCharge Application Services Manager retrieves this event and topology information from the Global Manager and sends back the results of its analysis. For more information about deploying InCharge Application Services Manager, see the *InCharge Application Services Manager Deployment Guide*.

Deploying Underlying Domains

When you deploy an underlying domain, or if your underlying domains are already running, check for the following:

- The host(s) that the underlying domain(s) run on are reachable from the host where the InCharge Broker is running.
- The host(s) that the underlying domain(s) run on is reachable from the host where the Global Manager is running.
- All underlying domains are a supported version.
- If two or more underlying domains manage the same topology element(s), read *Ensuring a Consistent Representation of Topology* on page 62.

Deploying the Global Manager

The Global Manager is responsible for monitoring and managing topology and events from multiple distributed domains. Deploying the Global Manager requires the following:

- Specifying the correct data exchange files for each underlying domain in the *ics.conf* file
- Adding a notification list to the *ics.conf* file for InCharge Report Manager.

You specify the data exchange files in the *ics.conf* file, located in the ***BASEDIR****/smarts/local/conf/ics* directory.

InCharge Report Manager uses a notification list to receive events from the Global Manager. You must create a notification list that matches those events on which you want to perform analysis and specify it in the *ics.conf* file. Typically, this would be a notification list that matches all events. For information about creating notification lists, see *Overview of Notification Lists* on page 50.

Deploying the Global Console

The Global Console is used by operators to view the results of the Global Manager's and by administrators to perform tasks such as creating topology groups and creating or modifying the topology for Services or Applications maps. As such, the Global Console is typically deployed on many different hosts. You also have the option of loading the Global Console as an applet, which allows you to access the Global Manager through a supported Web browser.

Once installed the Global Console requires no additional configuration unless you want to create and distribute customized consoles.

Deploying InCharge Report Manager

InCharge Report Manager consists of two adapters, both of which must be configured. In addition, Report Manager requires an SQL database with which it can connect and insert records.

For information about configuring InCharge Report Manager, see the *InCharge Service Assurance Manager User's Guide for Report Manager*.

Configuring Security

How you configure security will depend on the security policies and procedures you already have in place. Several different security scenarios, and their requirements, are described in the *InCharge System Administration Guide*.

To enforce a basic level of security requires that you create unique InCharge user names and passwords for each user that connects to the Global Manager. In this scenario, you would need to add these user names and passwords to the *serverConnect.conf* file on the host where the Global Manager is running.

In addition, you need to create a similar record for administrators and comment out or remove the following authentication records from the *serverConnect.conf* file:

```
* : admin: changeme: All
* : main : maint : Monitor
* : oper : oper : Monitor
```

It is important to note that this level of security does not provide unique user names and passwords for the interaction between client and server programs.

Table 5 lists the server programs, and each of their clients, for the deployment scenario. Each client program listed in the left column would have a unique user name and password that it uses to connect to the listed servers. These user names and passwords must be specified in the appropriate *brokerConnect.conf*, *clientConnect.conf,* and *serverConnect.conf* files.

SERVER	CLIENT
InCharge Broker	Global Manager
	Underlying Domains
	SMART Adapters
	Global Consoles
	InCharge Report Manager adapters
	Command line utilities
Underlying Domains	InCharge Broker
	Global Manager
SMART Adapters	InCharge Broker
	Global Manager
Global Manager	InCharge Broker
	InCharge Report Manager
	Global Consoles
	Command line utilities

Table 5: Server and Client Programs For Scenario 1

Scenario 2: Extending Service Assurance

This second scenario extends the configuration of the first deployment scenario with two notable additions: Open Integration and the InCharge Web Portal. Open Integration functions as an underlying domain, feeding events and topology from disparate sources and sending them to the Global Manager. The Web Portal enables you to send notifications to a user who views them through a Web browser.

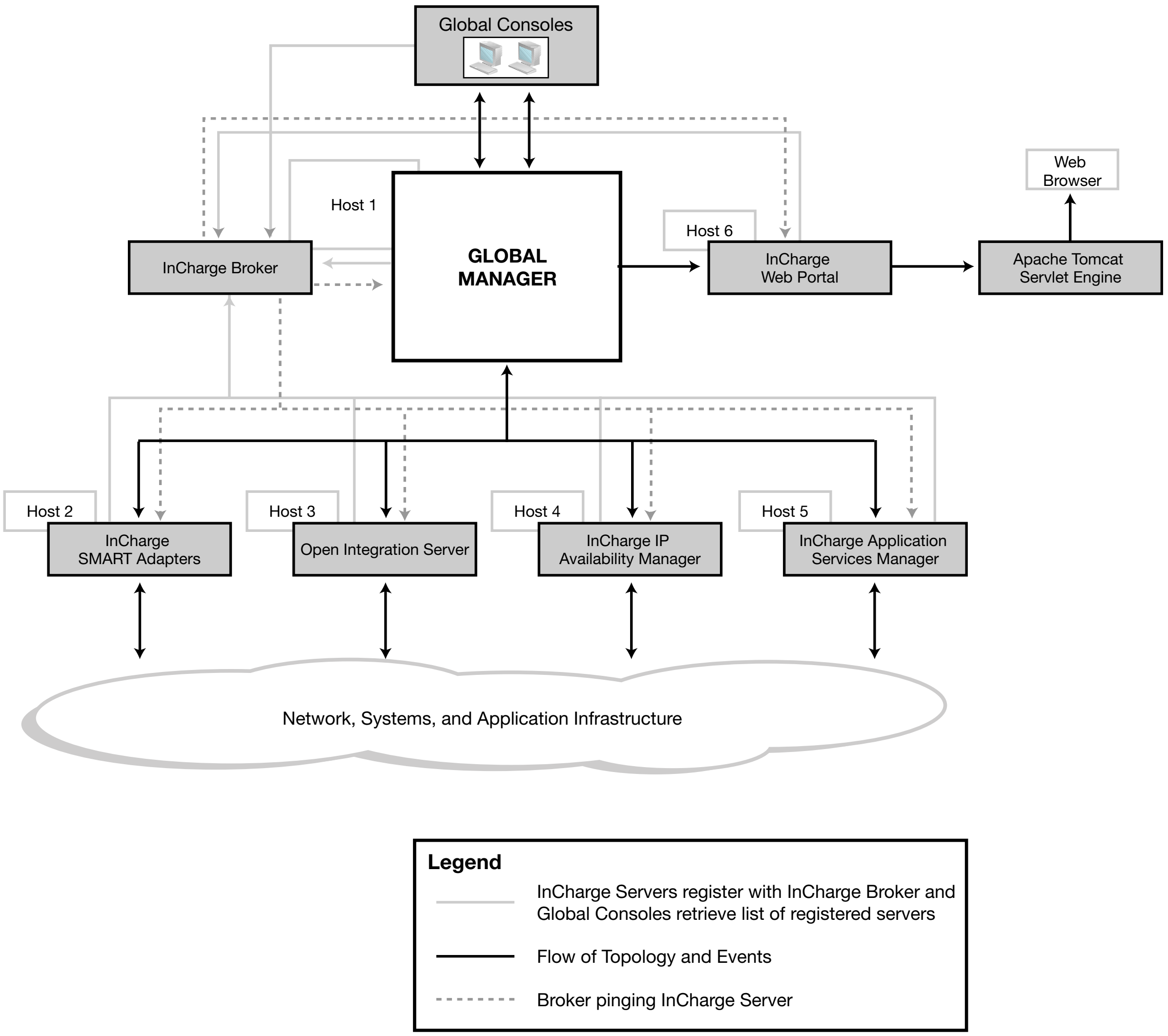
Global Consoles
Host 1
GLOBAL MANAGER
Web Browser
Host 6
InCharge Broker
InCharge Web Portal
Apache Tomcat Servlet Engine
Host 2
InCharge SMART Adapters
Host 3
Open Integration Server
Host 4
InCharge IP Availability Manager
Host 5
InCharge Application Services Manager
Network, Systems, and Application Infrastructure
Legend
InCharge Servers register with InCharge Broker and Global Consoles retrieve list of registered servers
Flow of Topology and Events
Broker pinging InCharge Server

Figure 2: Deployment Scenario 2

Deploying Open Integration

You deploy an Open Integration server at the same time you are deploying underlying domains. Open Integration, however, involves additional configuration and deployment considerations. The purpose of Open Integration is receive events and topology from a third-party source and process them so that the Global Manager can use them to diagnose authentic problems. Open Integration can receive events and topology from the following sources:

- SNMP traps – To use process SNMP traps, you must configure the InCharge SNMP Trap Adapter to define the traps you want to send to the Open Integration server. Defining traps means that you map the incoming traps to a known notification by the InCharge Information Model (ICIM).
- System log files – To produce notifications from a system log file, you must configure the InCharge Syslog Adapter. You must specify the system log file that you want to process and make sure that the format of the log file matches the format described in the Syslog Adapter's ASL rule set.
- Third-party sources with *sm_ems* – To process events from a third party source, you must configure the event source to invoke the *sm_ems* utility. You must configure *sm_ems* so that it can notify or clear events, update the values of attributes, or create topology elements.

For information about configuring these adapters and utilities, as well as the Open Integration server, see the *InCharge Service Assurance Manager Open Integration Configuration Guide*.

Deploying the Open Integration server involves the following tasks:

- Specifying the correct name and data exchange file for the Open Integration server in the Global Manager's configuration file.
- Creating a custom notification list so that the Open Integration Server sends the appropriate notifications to the Global Manager. This step is not required as a default notification list is provided. The default notification list sends all notifications to the Global Manager.

Deploying the InCharge Web Portal

The Web Portal Server receives notifications from a Global Manager, passes them to a Web servlet engine, which makes the information available to users of the InCharge Web Console.

For more information about configuring the InCharge Web Portal, see the *InCharge Service Assurance Manager Web Portal Configuration Guide*.

InCharge Web Portal includes all of the necessary software, including the Web servlet engine, for deployment. The deployment steps include:

- Configuring the Web Portal Server. The configuration file for the Web Portal is named *wp.conf*.
  - You must specify the Global Manager as an underlying domain in the DomainType section of the *wp.conf* file.

    - Define user profiles in the *wp.conf* file for each user, or group of users, that view notifications through the Web Console.
- Configuring the Web Portal Servlet, which includes specifying Web Console session settings.
- Configuring the Apache Tomcat Servlet Engine.
- Creating InCharge Web Consoles, which display in a user's Web browser.

Configuring Security for Open Integration and Web Portal

If you have already created appropriate accounts for users, who access Service Assurance, it is unlikely you will need to create new users accounts for Open Integration. However you need to create user accounts for the Web Console users. Web Console users are authenticated by the Web Portal server and must have accounts defined in the *serverConnect.conf* file used by the Web Portal server.

If you are securing the access of client applications, you will need to create accounts for the client programs listed in Table 6.

SERVER	CLIENT
InCharge Broker	Open Integration server
	InCharge SNMP Trap Adapter
	InCharge Syslog Adapter
	sm_ems
	Web Portal server
Global Manager	Web Portal server
Open Integration server	InCharge Broker
	InCharge SNMP Trap Adapter
	InCharge Syslog Adapter
	sm_ems
	Global Manager
Web Portal server	Web Portal servlet Web Console users

Table 6: Server And Client Programs For Scenario 2

3

Configuration Overview of the Global Manager

The Global Manager serves as the central point of Service Assurance, responsible for monitoring and managing multiple distributed domains. This chapter describes the basic steps needed to configure the Global Manager and summarizes the configuration options by functional area.

Configuration Steps

After you have installed the Global Manager, you will need to complete one or more of the following steps before starting the Global Manager. Default options are included for most configuration options. Required steps are indicated.

1. [Required] Specify the name of one or more underlying domains and the appropriate data exchange file for each as described in *Defining Domain Parameters* on page 31.
2. Create one or more notification lists as described in *Defining Notification List Parameters* on page 51.
3. Create one or more user profiles as described in *Defining User Profiles* on page 27.
4. Specify topology elements that are managed by two or more underlying domains as described in *Managing Overlapping Elements From Separate Underlying Domains* on page 37.

5 Start the Global Manager as described in the *InCharge Installation Guide*. Detailed information about starting, stopping, and configuring InCharge processes is provided in the *InCharge System Administration Guide*.

Note: These steps assume that security configuration for Service Assurance allows for the necessary connectivity between components and permits access for authorized users.

ics.conf: the Global Manager's Configuration File

Much of the configuration specific to the Global Manager is defined in the *ics.conf* configuration file. The original version is located in the ***BASEDIR****/smarts/conf/ics* directory. When edit this file, use the *sm_edit* utility to open the *ics.conf* file and to create a local copy of the file.

The Global Manager reads this file at startup to determine its configuration. If you edit this file while the Global Manager is running, you need to reconfigure the Global Manager as described in *Reconfiguring the Global Manager* on page 24.

The *ics.conf* file is divided into the following sections, each of which corresponds to one or more topics in this document:

- UserSection associates a user name with a notification list and one or more saved consoles. For more information, see *Defining User Profiles* on page 27.
- NotificationListSection defines one or more notification lists. A notification list determines the set of notifications a Global Manager client can receive. For more information, see *Notification Configuration for the Global Manager* on page 43.
- DomainSection specifies the underlying domains from which the Global Manager receives event and topology information and associates each underlying domain with the proper data exchange file. For more information, see *Defining Domain Parameters* on page 31.
- TagSection defines IP elements that are managed by two or more underlying domains that the Global Manager must keep topologically distinct. For more information, see *Managing Overlapping Elements From Separate Underlying Domains* on page 37.

- ToolSection specifies the configuration of automated and server tools. For more information, see *Tool Configuration for the Global Manager* on page 83.
- SystemDefaultsSection describes system configuration options for the Global Manager. For more information, see *Defining System Defaults* on page 34.
- BusinessSection specifies data files used to import group information into the Global Manager. For more information, see *Topology Configuration for the Global Manager* on page 61.

Syntax Conventions of the ics.conf File

The following section describes the syntax of the *ics.conf* configuration file. Configuration files for related Service Assurance components follow the same syntactical conventions. These files include:

- *icoi.conf* for Open Integration
- *wp.conf* for Web Portal

Each section of the *ics.conf* file starts with a name, such as UserSection, followed by a pair of curly braces ({ }). The configuration information for the named section is enclosed by the curly braces. If a section is divided into one or more subsections, each subsection is also named and followed by a pair of curly braces. The configuration information for the subsection is enclosed by the curly braces.

The configuration fields are declared inside of a section or subsection. A configuration field is composed of a parameter, an equals sign (=), and the value assigned to the parameter. All non-numeric values must be enclosed in double quotes (""). Each configuration field must end with a semicolon (;).

A line that starts with a pound sign (#) is considered a comment and is ignored.

The following example illustrates the syntax for one section of the *ics.conf* file. The UserSection is composed of two User subsections. Each User subsection defines a separate user profile.

```
UserSection
{
    User                 # Default user profile
    {
        Name             = "default";
        NotificationList = "Default";
        Console          = "NotificationLog";
      }
    User                 # Maintenance user profile
    {
        Name             = "maint";
        NotificationList = "Maintenance";
        Console          = "NotificationLog";
    }
}
```

Note: Field names and their values are case sensitive.

Functional Description of the Global Manager Configuration Tasks

This section categorizes configuration tasks by the functional area, grouping similar or related configuration options. You can continue to modify and refine the configuration of the Global Manager while it is running.

The major functional categories for the Global Manager include:

- System
- Notifications
- Topology

System Configuration

System configuration includes tasks that define the operation of the Global Manager such as specifying underlying domains and creating user profiles. System configuration can be further divided into the following categories:

- Users
- Domain
- System defaults
- Tools

Users

A user is any operator, administrator, or client program that requires access to the Global Manager. For operators, you need to create a user profile and associate it with a notification list and, optionally, a saved console. For information about creating user profiles, see *Defining User Profiles* on page 27.

Notification lists determine which notifications the Global Manager shows to a particular user or client. You define one or more filters in a notification list to determine which notifications are available through that notification list. For more information about notification lists, see *Notification Configuration for the Global Manager* on page 43.

By creating a saved console, you can provide operators with a defined view of the network. This can include which columns are visible in a notification log or which classes are listed in the topology browser. For more information about creating a saved console, see *Creating and Saving a Remote Console* on page 29.

Domain

Domain refers to the underlying systems, typically InCharge analysis applications, that are the sources of events and topology for the Global Manager. The Global Manager can receive events and topology from multiple underlying domains. For each underlying domain, you need to specify the proper data exchange configuration file. In addition, you can define the following parameters:

- Specify whether two or more underlying domains manage overlapping topology elements. This is referred to as *tagging*.
- Define a smoothing interval and a minimum certainty for notifications received from each underlying domain.
- Optionally, create an ASL script to modify the notifications maintained by the Global Manager.

For information about setting domain parameters, see *Defining Domain Parameters* on page 31.

System Defaults

System defaults are parameters that are applied system wide to the Global Manager. These settings control various features related to the management of notifications, the interaction between the Global Manager and Global Consoles, and the interaction between the Global Manager and underlying domains. For information about the system defaults, see *Defining System Defaults* on page 34.

Tools

Tools provide a means by which users or an adapter can respond to a notification. Such a response might include pinging a device to see if it is reachable or opening a trouble ticket. For information regarding the purpose and configuration of tools, see *Tool Configuration for the Global Manager* on page 83.

Notification Configuration

Notification configuration includes creating notification lists for Global Manager clients. A notification list includes one or more filters that determine which notifications are exported by the notification list. For more information regarding notification lists, see *Notification Configuration for the Global Manager* on page 43.

Topology Configuration

Topology configuration involves organizing topology elements into groups and editing the topology for Services and Applications maps. Topology editing enables you to create new topology elements as well as create relationships between topology elements. For information regarding topology configuration, see *Topology Configuration for the Global Manager* on page 61.

Reconfiguring the Global Manager

The Global Manager reads the *ics.conf* file during startup and configures itself accordingly. If you change the *ics.conf* file after the Global Manager is running, you need to invoke a command so that the Global Manager will reload its configuration file. We refer to this procedure as *reconfiguring* the Global Manager. Reconfiguring the Global Manager requires administrative privileges.

To reconfigure the Global Manager, invoke the following command from the ***BASEDIR**/smarts/bin* directory:

```
% sm_adapter -s <global_manager> ics/ICS_RemoteConfig.asl
```

Depending on your security configuration, you may be prompted for your InCharge user name and password.

When the Global Manager reloads the *ics.conf* file, it sends output to the terminal or its log file verifying that it was able to read each section of the configuration file. The following example shows the output when *ics.conf* is successfully reloaded.

```
% ./sm_adapter -s INCHARGE-SA ics/ICS_RemoteConfig.asl
Server INCHARGE-SA User: admin
admin's Password: XXXXX
ICNF-N-Processing configuration file
'local/conf/ics/ics.conf'
ICNF-N-Successfully processed SystemDefaultsSection
ICNF-N-Successfully registered Notification List 'Default'
ICNF-N-Successfully registered Notification List
'Maintenance'
```

If the Global Manager encounters an error while reloading the *ics.conf* file, it sends an error to the terminal or log file and continues to function using its previous configuration. The following example shows the output if the Global Manager encounters a syntax error when reading its configuration file.

```
% ./sm_adapter -s INCHARGE-SA ics/ICS_RemoteConfig.asl
Server INCHARGE-SA User: admin
admin's Password: XXXXX
ICNF-N-Processing configuration file
    '/opt/smarts/local/conf/ics/ics.conf'
ICNF-E-Line 152: Syntax Error 2007: No matching right brace
'}' found for
    container 'DomainSection' with left brace on line 41
```

Note: Changes to the *ics.conf* file that affect the Global Console, including changes to user profiles, notification lists, and tools may require the operator to restart the Global Console.

4

System Configuration for the Global Manager

This chapter describes system configuration tasks for Service Assurance Manager. System configuration topics include creating user profiles, configuring settings for the underlying domains, and setting system parameters for the Global Manager.

Defining User Profiles

User profiles are used by operators who attach to the Global Manager with the Global Console. A user profile includes an InCharge user name, a notification list, and, optionally, one or more saved consoles. You can create user profiles for each operator or you can organize them by function so that operators with the same function share a user profile.

An advantage of creating individual user profiles is that you can track an operator's response to notifications through the Audit Log. The Audit Log records user actions such as acknowledging and taking ownership of notifications.

Because a user profile is associated with a user name, you must ensure that you have provided appropriate access through the Service Assurance security mechanism. For information regarding security, see the *InCharge System Administration Guide*.

User profiles are defined in the *ics.conf* configuration file. The following example shows the syntax of the UserSection of the *ics.conf* file. Each user profile is defined in a separate User subsection.

```
UserSection
{
    User                    # Default user profile
    {
        Name                = "default";
        NotificationList = "Default";
        Console             = "NotificationLog";
    }

    User                    # Maintenance user profile
    {
        Name                = "maint";
        NotificationList = "Maintenance";
        Console             = "NotificationLog";
    }
}
```

Table 7 describes the fields of a User subsection.

FIELD	DESCRIPTION
Name	The user name associated with this user profile. This value must match an authentication record defined in the Global Manager's *serverConnect.conf* file.
NotificationList	The notification list associated with this user profile. The notification list must be defined in the NotificationListSection of the *ics.conf* file.
Console	The saved console that opens when this user attaches to the Global Manager. NotificationLog is the default console provided with Service Assurance. You can specify multiple consoles for a user with additional Console fields.

Table 7: Fields Defining the UserSection

If you edit the *ics.conf* file when the Global Manager is running, you must reconfigure the Global Manager to make the changes take effect. To reconfigure the Global Manager, invoke the following command from the ***BASEDIR***/*smarts/bin* directory:

```
% sm_adapter -s <global_manager> ics/ICS_RemoteConfig.asl
```

For more information regarding this command, see *Reconfiguring the Global Manager* on page 24.

Note: Changes to the UserSection may not be available to console users until they restart the Global Console.

About the Default User Profiles

Service Assurance comes with two user profiles: “default” and “maint”. The “default” user profile specifies the Default notification list and the NotificationLog console. This ensures that an operator without a user profile can attach to Global Manager and receive results from Service Assurance. (The user must still authenticate with the Global Manager.)

In addition, the “default” user profile provides another useful feature. If you want to distribute a saved console to a number of users, you can specify it in the Console field of the default user profile. This saves you from having to create separate profiles for each user and specifying the console in each profile. A limitation of this approach is that you cannot specify different notification lists for users. If you require separate notification lists, then you need to create multiple user profiles.

The “maint” user profile also has a special purpose, which is to mark notifications that are not relevant to operators. For example, if notifications are generated for interfaces that should not be managed, or for components that are known to be faulty but are not scheduled to be fixed, an administrator can log in as the “maint” user and take ownership of these notifications.

The “Default” notification list specified in the *ics.conf* file filters out notifications owned by the user “maint”. When “maint” takes ownership of a notification, that notification is removed from the display of users of the “Default” notification list.

Note that there is a difference between the “maint” user taking ownership of a notification and acknowledging a notification. An acknowledged notification becomes unacknowledged if the event is recurs. If “maint” takes ownership of a notification, with the default notification lists, the notification remains filtered out from the display of operators even if it is recurs.

For a description of the default notifications lists used by the “default” and “maint” user profiles, see *About the Default Notification Lists* on page 58.

Creating and Saving a Remote Console

You can create a console layout for the Global Console, save it to a directory on the Global Manager, and make it available to operators.

For information regarding the different ways you can modify the layout of the Global Console, see the *InCharge Service Assurance Manager Operator's Guide*.

You can use two methods to distribute a saved console:

- If you specify the console name in the Console parameter of a user profile, the saved console opens automatically when the user starts the Global Console. The saved console must be located in the ***BASEDIR****/smarts/local/consoles* directory.
- You can copy a console to the ***BASEDIR****/smarts/local/consoles* directory. Console users can use the **Open Remote As** command to open consoles in this directory.

Operators can modify a saved console and save their modified console to a local directory on their system or save it remotely to the host where the Global Manager is running.

Consoles saved to the Global Manager are saved to the ***BASEDIR****/smarts/local/consoles/<USER>* directory, where <USER> is the InCharge username of the operator. An operator cannot see or open the consoles saved by another operator unless it is manually copied into their own console directory or the ***BASEDIR****/smarts/local/consoles* directory.

Providing a Default Console in a User Profile

To provide a default console in a user profile, complete the following steps:

1. Configure a console as described in the *InCharge Service Assurance Manager Operator's Guide*.
2. Select the **Save Remote As** command and save the console to the host where the Global Manager is running.

   The console will be saved to a directory that corresponds to your user name under the ***BASEDIR****/smarts/local/consoles* directory. For example, if you are logged in as the user "admin", the console will be saved to the ***BASEDIR****/smarts/local/consoles/admin* directory.
3. Move the saved console from its current directory to the ***BASEDIR****/smarts/local/consoles* directory. Consoles stored in this directory, are available to all Global Console operators.

   When a saved console is located in this directory, operators can also open the console using the **Open Remote** command. Specifying the console in a user profile makes it open automatically when an operator starts the Global Console and logs in to the Global Manager.

4 Specify the default console name in the Console field of the user profile.

5 Reconfigure the Global Manager by invoking the following command from the ***BASEDIR****/smarts/bin* directory. This command requires administrative privileges.

```
% sm_adapter -s <global_manager> ics/ICS_RemoteConfig.asl
```

Note: The command must be typed as one line.

Operators have to restart the console for these changes to take effect.

Defining Domain Parameters

A domain refers to an underlying InCharge analysis application, Open Integration server, SMART Adapter, CiscoWorks 2000 Device Fault Manager, or other Global Manager that serves as a source of event and topology data. You specify these domains and the parameters that control the import of event and topology data in the DomainSection of the *ics.conf* configuration file.

The DomainSection is divided into one or more DomainType subsections, each of which defines the configuration for one or more underlying domains. You can specify up to 40 underlying domains for a Global Manager. The following example shows the syntax of a DomainSection when the underlying domain is InCharge Availability Manager.

```
DomainSection
{
    DomainType
    {
        ConfFile          = "dxa-conn.conf";
        MinimumCertainty  = 0.0;
        SmoothingInterval = 65;
        HookScript        = "ics/dxa-sample-hook.asl";
        Name              = "INCHARGE";
    }
}
```

Table 8 describes the fields of DomainType subsections.

FIELD	DESCRIPTION
ConfFile	The data exchange file that corresponds to the underlying domain. A data exchange file ensures that Global Manager receives the correct event and topology data. These files should not be edited. • *dxa-app-poller.conf* for InCharge Application Connection Monitor • *dxa-asm.conf* for InCharge Application Services Manager. • *dxa-bmc.conf* for InCharge SMART Adapter for BMC Patrol. • *dxa-conn-perf.conf* for an InCharge application running both Availability Manager and Performance Manager. • *dxa-conn.conf* for InCharge Availability Manager. • *dxa-dfm.conf* for CiscoWorks Device Fault Manager. • *dxa-oi.conf* for InCharge Open Integration server. • *dxa-perf.conf* for InCharge Performance Manager. • *dxa-sam.conf* for another Global Manager. • *dxa-vhm.conf* for CiscoWorks Voice Health Monitor.
MinimumCertainty	Minimum value the Certainty attribute must have before an event is sent to the Global Manager from the underlying domain. Events with a Certainty value below the threshold are discarded. This value must be a number between 0.0 and 0.99. The default value is 0.24.
SmoothingInterval	Time, in seconds, an event must be active before it is sent to the Global Manager. The default value is 65 seconds. Note that the smoothing interval does not apply to underlying Open Integration servers or other Global Managers.
HookScript	[Optional] Name of an ASL script that modifies a notification before it is created by the Global Manager. Typically, this is used to add information to one of the user-defined fields of a notification. Hook scripts must be located in the ***BASEDIR****/smarts/local/rules/ics* directory. You must prefix the name of the script with the directory in which it is located, typically this is the *ics* directory.
Name	Name of the underlying domain. You can specify multiple domains with the same configuration by adding Name fields to the DomainType subsection. However, the value of each Name field within a DomainSection must be unique.

Table 8: Fields Defining the DomainSection

Examples of DomainSection Configurations

This section provides an example of a DomainSection. As shown in the example, you can specify two or more underlying domains within a single DomainType by using additional Name fields. To do this, the underlying domains must be of the same type and use the same settings, such as MinimumCertainty. When you specify two or more underlying domains, the name of each domain must be unique within the DomainSection.

DomainSection with Multiple Underlying Domains

```
DomainSection
{
    DomainType
    {
        ConfFile            = "dxa-perf.conf";
        MinimumCertainty    = 0.24;
        SmoothingInterval = 65;
#        HookScript          = "ics/dxa-sample-hook.asl";
        Name                = "INCHARGE";
        Name                = "INCHARGE-2"'
    }

    DomainType
    {
        ConfFile            = "dxa-conn-perf.conf";
        MinimumCertainty    = 0.24;
        SmoothingInterval = 65;
#        HookScript          = "ics/dxa-sample-hook.asl";
        Name                = "INCHARGE-3";
    }

    DomainType
    {
        ConfFile            = "dxa-oi.conf";
        MinimumCertainty    = 0.24;
        SmoothingInterval = 65;
#        HookScript          = "ics/dxa-sample-hook.asl";
        Name                = "INCHARGE-OI";
    }

    DomainType
    {
        ConfFile            = "dxa-dfm.conf";
        MinimumCertainty    = 0.24;
        SmoothingInterval = 65;
#        HookScript          = "ics/dxa-sample-hook.asl";
        Name                = "DFM";
    }
    DomainType
    {
        ConfFile          = "dxa-sam.conf";
        MinimumCertainty  = 0.24;
        SmoothingInterval = 65;
#        HookScript        = "ics/dxa-sample-hook.asl";
        Name              = "INCHARGE-SA2";
    }
}
```

By default, the Hookscript fields are commented out. The default ASL hook scripts are placeholders for a script you must provide. A hook script is not required unless you need to perform additional processing on the notifications coming from the underlying domain.

Defining System Defaults

System defaults are system-wide settings that affect the Global Manager and its clients. The following example illustrates the syntax of the SystemDefaultsSection.

```
SystemDefaultsSection
{
   AutoAcknowledgementInterval = 300;

   InactiveAutoArchiveInterval = 14400;

   AuditTrailSizeLimit   = 100;

   SMTPServer = "localhost";

   FetchLocalNotificationProperties = false;

   RMITimeOut = 60;

}
```

Table 9 describes the fields of the SystemDefaultsSection.

FIELD	DESCRIPTION
AutoAcknowledgementInterval	Interval, in seconds, after which an inactive and unowned notification is acknowledged. Notifications that are acknowledged by the Global Manager are owned by the user SYSTEM. Default is 300 seconds.
InactiveAutoArchiveInterval	Interval, in seconds, after which an inactive and acknowledged notification is archived. Default is 14400 seconds (4 hours). If this value is set to zero, archiving is disabled and notifications will not be deleted, causing Global Manager to use more memory.
AuditTrailSizeLimit	Number of audit log entries for each notification that are saved in the Global Console before the log contents are archived. When this limit is reached, half of the entries are written to the notification archive. Default is 100 entries.
SMTPServer	Name of the SMTP mail server through which mail messages sent by the Global Console's Mail tool are sent. This mail server must be reachable from the host where the Global Manager is running. The default is "localhost". This value can be overridden in the Global Console.
FetchLocalNotificationProperties	Controls how the properties of a particular notification are retrieved when requested by a client application. If set to "true", properties are retrieved from the Global Manager. If set to "false", properties are retrieved from the Global Manager and from the underlying domain. The default value is false. Client applications include the Global Console, the dmctl command-line utility, and adapters.
RMITimeout	Maximum amount of time, in seconds, the Global Manager is allotted to fetch Notification Properties, Find System, or get System Containment information from an underlying InCharge domain. If a negative value is specified, no time-out value is set. The default value is 60 seconds.

Table 9: Fields Defining SystemDefaultsSection

About Acknowledging Notifications

When a notification is acknowledged, an entry is appended to the notification's Audit Log. In addition, the values of two notification attributes are set: Owner and Acknowledged. The value of the Owner attribute is set to the InCharge user name of whomever acknowledged the event. The value of the Acknowledged attribute is set to TRUE.

If a notification is unacknowledged, another entry is appended to the notification's audit log. The value of the Owner attribute is set to the InCharge user name of whomever unacknowledged the notification and the value of the Acknowledged attribute is set to FALSE.

Autoacknowledgement is a facility designed for notifications that clear before an operator is able to acknowledge them. It is not uncommon for a notification to clear shortly after it appears. Instead of requiring operators to manually acknowledge such notifications, the Global Manager automatically acknowledges unowned notifications after they remain inactive (clear) for the time specified by the AutoAcknowledgementInterval. Notifications acknowledged by the Global Manager have their Owner attribute set to SYSTEM.

After a notification is acknowledged, it is eligible for archiving. The Global Manager will not archive an active or unacknowledged notification.

Archiving Notifications

Old notifications are periodically removed from the Global Manager's repository and archived to a flat file. The notification archive includes the values of the notification's attributes at the time it is archived, including the contents of the audit log. This archive is intended to provide a record of information before it is deleted.

If you wish to generate historical reports based on notification data, see the *InCharge Service Assurance Manager User's Guide for Report Manager*. InCharge Report Manager does not use the notification archive for reporting, but maintains the notification information in a relational database.

Archived notifications are written to file named *<global_manager>.archive*, where *<global_manager>* is the name of the Global Manager. By default, this is INCHARGE-SA so the resulting notification archive is named *INCHARGE-SA.archive*. This file is written to the ***BASEDIR****/smarts/local/logs* directory.

Note: Over a period of time, a busy Global Manager can generate a sizeable archive file. It may be necessary to periodically rotate the notification archive file.

Managing Overlapping Elements From Separate Underlying Domains

A Global Manager collects topology information from multiple underlying domains. In some cases, two or more of these domains can manage elements with the same name. Two such examples are IP networks, which are named using the network address, and partitions, which are assigned names by the underlying domain.

For example, a service provider might use a private IP network address, such as 10.0.0.0, to provide IP addresses to different customers. When the same range of IP addresses are assigned to multiple customers, different topology elements may have the same IP address, and thus the same name.

Partitions are created and named by InCharge IP Availability Manager. All Availability Manager domain managers use the same convention to name partitions. If the topologies of two domain managers each include partitions, it is possible that one or more of the partitions have the same name, as defined by the Name attribute. For more information about partitions, see the *InCharge User's Guide for Availability Manager*.

By default, the Global Manager treats elements with the same name from different underlying domains as a single instance, consolidating the relationships and attributes of the two elements to a single topological instance. The values for the attributes are taken from the underlying domain that performs the most recent topology synchronization.

You can, however, configure the Global Manager to manage elements of the same name from different domains as distinct elements. By specifying a *tag* for one or both of the underlying domains, the Global Manager will create two separate topology elements, each containing the attributes and relationships of the respective objects in the underlying domains. The tag that you specify is appended to the name of the objects in the tagged domain and displayed in the Global Console.

To tag managed elements, you need to complete two tasks:

- Specify a tag in the DomainSection.
- Specify a matching pattern that will match the IP addresses and partitions to be tagged.

The following example adds the tagging syntax to the DomainSection where four underlying domains are specified. The four underlying domains use the same configuration values for the ConfFile, MinimumCertainty, and SmoothingInterval fields. However, no tags are applied to the INCHARGE_1 and INCHARGE_2 domains. A tag, "tag-3", is applied to instances of the INCHARGE_3 domain that match the specified matching pattern. In addition, a tag, "tag-4" is applied to the instances of the INCHARGE_4 domain that match the specified matching pattern. Examples of matching patterns for these tags follow.

```
DomainSection
{
    DomainType
    {
        ConfFile          = "dxa-conn.conf";
        MinimumCertainty  = 0.24;
        SmoothingInterval = 65;
#        HookScript        = "ics/dxa-sample-hook.asl";
        Name              = "INCHARGE_1";
        Name              = "INCHARGE_2";
        Tagging
        {
            Name    = "INCHARGE_3";
            TagType = "Private IP 192";
            Tag     = "tag-3";
        }
        Tagging
        {
            Name    = "INCHARGE_4";
            TagType = "Private IP 10";
            Tag     = "tag-4";
        }

    }
```

Table 10 describes the Tagging fields of the DomainSection.

FIELD	DESCRIPTION
Tagging	Identifies this as the tagging section of a DomainType.
Name	Name of the underlying domain to which the tag is applied. The values of Name fields within the DomainSection must be unique.
TagType	The TagType applied to this domain. This must match a Name field in the TagType subsection of the TagSection in the *ics.conf* file. The TagSection specifies the matching criteria against which instances of the underlying domain are compared.
Tag	The string of characters that are applied as the tag.

Table 10: Tagging Fields of DomainSections

The following example illustrates what the TagSection for the preceding example might look like. The name of TagType "Private IP 192" and "Private IP 10" must match a TagType field in the Tagging section of the DomainSection.

- Matching patterns that begin with "Partition*" match instances of the Partition class.
- Matching patterns that begin with "IP*" match instances of the IP Network class that belong to the 192.168.0.0 and 10.0.0.0 IP networks.
- Matching patterns with an IP address range match both IP addresses and system elements named with a IP address.

```
TagSection
{
  TagType
  {
     Name    = "Private IP 192";
     Pattern = "Partition*|IP*-192.168.*|192.168.<0-255>.
                <0-255>";
  }
  TagType
  {
     Name    = "Private IP 10";
     Pattern = "Partition*|IP*-10.*|10.<0-255>.<0-255>.
                <0-255>";
  }

}
```

Note: The tag pattern must be specified on a single line.

Table 11 describes the fields of the TagSection.

FIELD	DESCRIPTION
TagType	Defines a TagType subsection of the TagSection.
Name	The name of the appropriate TagType section.
Pattern	Matching pattern used to identify managed elements to which the tag is applied.

Table 11: Fields Defining the TagSection

How the Global Manager Applies Tags

The Global Manager can apply tags to both topology elements and notifications. The Global Manager finds matching elements by comparing the matching pattern against the Name and DisplayName attributes of both topology elements and notifications from the relevant underlying domain. For matching elements, any additional attributes that match the pattern are also tagged.

As a result, you should construct a matching pattern to match the values of attributes you want tagged. For topology elements, this could include attributes such as Name and DisplayName. For notifications, this could include attributes such as DisplayName, InstanceDisplayName, and InstanceName.

For example, the matching pattern "IP*-172.16.*|172.16.*" could match the following:

- Elements of the IPNetwork class, which are prefixed with "IPNET-". For IPNET-172.16.0.0, this would result in a DisplayName of `172.16.0.0 [tag-3]` and a Name of `IPNET-172.16.0.0_tag-3`.
- Elements of the system classes, such as Host, that are named using the system's IP address. For host 172.16.1.107, this would result in a DisplayName of `172.16.1.107 [tag-3]` and a Name of `172.16.1.107_tag-3`.
- Notifications generated for these elements would also be tagged. For a Host Down notification, this would result in a DisplayName of `Host Down 100%: 172.16.1.107 [tag-3]`, an InstanceDisplayName of `172.16.1.107 [tag-3]`, and an InstanceName of `172.16.1.107_tag-3`.

Note, however, that if the system was named using its host name, this matching pattern would not apply.

5

Notification Configuration for the Global Manager

This chapter describes notifications, the attributes the define a notification's state, and how a notification's state affect the acknowledgement and archival of notifications. This chapter also describes notification lists: what they are, how they are used, and how to create them.

Overview of Notifications

The Global Manager stores the topology and event information that it receives from the underlying domains in its repository. The topology and event information are stored as objects; instances of the classes defined in the InCharge Information Model (ICIM). Because of this, the notifications that Global Manager sends to clients are themselves objects in the Global Manager's repository.

Notification objects, similar to other objects in the Global Manager's repository, have attributes that describe their properties. For notifications, these attributes include the time the event occurred, the type of event, the name of the object where the event occurred, and much more information. For the complete list of notification attributes, see *Attributes for Matching Notification Properties* on page 112.

Notification attributes are used throughout Service Assurance for a variety of purposes:

- Notification attributes correspond to the columns in the Notification Log of the Global Console. Note that attributes names are not exactly the same as the column headings of the Notification Log.
- You can specify a matching pattern against the values of notification attributes for filtering a notification list.
- Ten notification attributes are user defined. You can write an ASL script that populates these attributes with additional information.

Understanding and Managing Notifications

This section describes the states of an InCharge notification. It also describes the attributes of a notification and how a change in the notification's state affects the values of these attributes. Finally, this section describes the acknowledgement and archival of notifications.

The following notification attributes are related to a notification's state:

- Event State
- First Notify Time
- Last Notify Time
- Last Change Time
- Count

Uniquely Identifying Notifications

Before we describe the states of a notification, it is important to note that each InCharge notification has a unique name. A notification's name is created from the name of the class and instance where the event occurred and the name of the event itself. For example, the notification NOTIFICATION-Router_R1_Down identifies the event Down, which occurred in the instance R1 of the Router class. The Notification Log will not list more than one notification with this name. Instead, if this event occurs again, the Count field is increased accordingly.

States of a Notification

A notification's state is defined by the Event State attribute, which has five possible values. Table 12 describes each possible value of the Event State attribute and shows its relationship to the Active notification attribute.

EVENT STATE VALUE	DESCRIPTION	ACTIVE VALUE
ACTIVE	Event that causes the notification is occurring. The notification is in the "notify" state.	TRUE
WAS_ACTIVE	Global Manager has disconnected from the underlying domain that was the Source for the notification. For more information regarding WAS_ACTIVE, see *The WAS_ACTIVE and SUSPENDED Event States* on page 45.	TRUE
SUSPENDED	Global Manager can no longer retrieve information about the notification. For more information regarding SUSPENDED, see *The WAS_ACTIVE and SUSPENDED Event States* on page 45.	TRUE
INACTIVE	Event that causes the notification is no longer occurring. The notification is in the "clear" state.	FALSE
UNINITIALIZED	State of the notification object when it is first created and before it has been sent to a client.	FALSE

Table 12: Description of Event State Notification Attribute

The WAS_ACTIVE and SUSPENDED Event States

The WAS_ACTIVE event state is used to identify those notifications that are active under two conditions:

- A Global Manager disconnects from the underlying domain that is the event source. In this case, all active notifications from the disconnected domain are marked WAS_ACTIVE.
- A Global Manager is started from a saved repository file. In this case, all active notifications are marked WAS_ACTIVE.

Notifications marked as WAS_ACTIVE remain in the WAS_ACTIVE state until the Global Manager can verify their status.

If the Global Manager is not able to reconnect to the underlying domain(s) that generated these notifications in 1800 seconds (30 minutes), the value of the Severity attribute is set to 4, changing the color of the notification in a Notification Log to blue. The value of 1800 seconds is referred to as the detachTime.

When the connection between the Global Manager and the underlying domain(s) is re-established, the Global Manager does the following:

- Notifications that are still active in the underlying domain have their Event State changed to ACTIVE and their Severity value updated accordingly.
- Notifications that have cleared in the underlying domain remain in the WAS_ACTIVE state until the attachTime has elapsed. The default value of attachTime is 6000 seconds (100 minutes). However, the actual value that is used to calculate when to clear the notifications is determined as follows:
    - When the uptime for the underlying domain is greater than the attachTime, then the value is 240 seconds plus the smoothing interval. The smoothing interval is specified in the DomainType section of the *ics.conf* file.
    - When the uptime for the underlying servers is less than the attachTime, the value is attachTime minus uptime. For temporary disconnects, this value is typically used to determine the state of WAS_ACTIVE notifications.

The SUSPENDED state indicates that the Global Manager is no longer able to retrieve information about an active notification. When an underlying InCharge domain is not able to get to the source of a notification, it suspends the notification. The InCharge domain sends this message to the Global Manager, which in turn suspends the notification. An InCharge domain may not be able to get to the source of a notification because an SNMP agent is not responding or because it received unexpected error values in an SNMP request.

Notifications suspended by the Global Manager have the value of the Severity attribute set to 4, changing the color of the notification in a Notification Log blue.

A Notification's Life Cycle

Table 13 shows the state and the value of certain notification attributes for NOTIFICATION-Router_R1_Down at different time intervals. The columns of the table mirror the columns an operator might see in the Global Console, with two exceptions: Time and Archived. The Time and the Archived columns are provided for this example.

NOTIFICATION-Router_R1_Down notification becomes active (notified) at 2:00. Fifteen minutes later, at 2:15, the notification becomes inactive (clears). At 2:18, the notification returns to the active state and clears again at 2:35. After remaining in the clear state for five minutes, the notification is acknowledged by the Global Manager at 2:40. After remaining clear and inactive for four hours, the notification is archived at 6:40. At 6:55 NOTIFICATION-Router_R1_Down becomes active. Because the previous instance of this notification was archived, the Global Manager resets the Count, First Notify, Last Notify, and Last Change fields.

TIME	ACTIVE	FIRST NOTIFY	LAST NOTIFY	LAST CHANGE	COUNT	ACKNOWLEDGED	ARCHIVED
2:00	TRUE	2:00	2:00	2:00	1	—	—
2:15	FALSE	2:00	2:00	2:15	1	—	—
2:18	TRUE	2:00	2:18	2:18	2	—	—
2:35	FALSE	2:00	2:18	2:35	2	—	—
2:40	FALSE	2:00	2:18	2:35	2	Acknowledged	—
6:40	FALSE	2:00	2:18	2:35	2	Acknowledged	Archived
6:55	TRUE	6:55	6:55	6:55	1	—	—

Table 13: States of NOTIFICATION-Router_R1_Down

Acknowledging and Archiving Notifications

Regardless of its state, a notification can be marked as Acknowledged. An operator can acknowledge a notification through the Global Console. When an operator acknowledges a notification, the operator becomes the owner of the notification. Acknowledging a notification does not change its state. If an inactive acknowledged notification is re-notified, the value of the Acknowledged attribute is set to FALSE and the value of the Owner attribute is cleared.

By default, the Global Manager automatically acknowledges cleared (inactive) and unowned notifications after five minutes. When Global Manager acknowledges a cleared notification, the owner is set to SYSTEM. You can configure this interval through the AutoAcknowledgementInterval.

Using a notification list, you can filter acknowledged notifications so that they do not appear in an operator's display.

After a notification has been acknowledged, it is eligible to be archived. An archived notification is removed from the Global Manager's repository and written to an archive file. The notification archive file is named *<global_manger>.archive* and located in the ***BASEDIR****/smarts/local/logs* directory.

When a notification is archived, the Global Manager treats a recurrence of that notification as though it were the first occurrence. The value of notification attributes start over again: new First Notify time, Count starts at 1, and so on.

The Global Manager only archives inactive and acknowledged notifications. You can configure the acknowledgement and archival of notifications through the AutoAcknowledgementInterval and InactiveAutoArchiveInterval settings.

Configuration Parameters for Acknowledging and Archiving Notifications

The following parameters, defined in the *ics.conf* file, control how and when the Global Manager acknowledges or archives notifications. Any changes made to the parameters of the *ics.conf* file require that you reload the *ics.conf* file to make those changes take effect.

For information about where to set these parameters, see *Defining Domain Parameters* on page 31.

AutoAcknowledgementInterval

AutoAcknowledgementInterval controls when a *cleared* (inactive) notification is automatically marked as acknowledged by the Global Manager. The interval is calculated from the time when a notification clears. If a notification recurs, or is unacknowledged, the notification is rescheduled for automatic acknowledgement. The default value is 300 seconds (5 minutes). When this interval is set to 0, the Global Manager will not automatically acknowledge cleared notifications.

InactiveAutoArchiveInterval

InactiveAutoArchiveInterval controls when a *cleared* (inactive) and *acknowledged* notification is archived by the Global Manager. The interval is calculated from the time a notification is acknowledged. If a notification recurs, or is unacknowledged, the notification is rescheduled for archival. The default value is 14400 seconds (4 hours). When this interval is set to 0, the Global Manager will not automatically archive cleared and acknowledged notifications.

Notification Types and Incrementing OccurrenceCount

The InCharge Information Model (ICIM) defines two types of notifications: momentary and durable. A momentary notification has no duration, it describes an event that happened at a specific time. For example, an authentication failure is an example of a momentary event. A durable notification describes an event that is active over a period of time. While the event is active, the problem it causes is still in effect. An example of a durable event is a link failure. You can determine a notification's type by checking the value of the EventType notification attribute.

A notification's type determines how the value of the OccurrenceCount attribute is increased by a Global Manager. In addition, the source of the notification and the conditions under which a notification is sent, also determine how a notification's count is increased.

- For durable notifications, the OccurrenceCount represents how many times the notification has become active (notified) during the notification's life cycle.

  For example, consider the scenario where notification N_1 is active in a Global Manager and the underlying domain where the event occurred disconnects and reconnects. If N_1 is still active when the Global Manager and the underlying domain re synchronize, the Global Manager does not increment the OccurrenceCount.

- For momentary notifications, the OccurrenceCount represents how many times the Global Manager has received a notify message regarding the notification. The current state of the notification has no effect.

If notification N_1 were a momentary notification in the preceding scenario, a Global Manager would increment the OccurrenceCount by one, regardless of whether the notification was active. In fact, if notification N_1 occurred 50 times in the underlying domain, the Global Manger would increase the value of the OccurrenceCount by 50.

Overview of Notification Lists

A notification list determines what notifications a client of the Global Manager receives. Clients that make use of notification lists include:

- Global Consoles —For users of the Global Console, their notification list is specified in the UserSection of the *ics.conf* file as described in "Defining User Profiles" on page 27. The notification list is defined in the NotificationListSection of the *ics.conf* file.
- Notification adapters (File, E-mail, Script, TIBCO Rendezvous, Trap, and SDI)— The notification list used by a notification adapter is specified in the NLsubscription section of the adapter's configuration file. The notification list is defined in the NotificationListSection of the *ics.conf* file.
- Global Managers—The Global Manager also uses a notification list when it receives events from an Open Integration server. The notification list used by Global Manager is specified in the *dxa-oi.conf* file. The notification list is defined in the NotificationListSection of the Open Integration server's *icoi.conf* configuration file.

Note: Because a notification list controls what notifications a client receives, it affects more than the display of notifications in a Notification Log. A notification list also affects the display of maps in the Global Console. For example, if a notification list filters out all notifications for a router, the status of that router will appear normal, regardless of its actual condition.

Notification lists for clients of the Global Manager are defined in the *ics.conf* file. The definition for a notification list includes one or more filters that specify what notifications the client receives. In addition to a filter, you can also change the default name of column headings in the Notification Log of the Global Console.

The filter(s) specified as part of a notification list should not be confused with the filtering capabilities of the Global Console. The Global Manager performs the filtering defined in a notification list. The notifications that are filtered out by a notification list are not sent to a client. Filters in the Global Console can be used to further refine the filter of a notification list. This filtering is performed by the Global Console and described in the *InCharge Service Assurance Manager Operator's Guide*.

If you edit the *ics.conf* file when the Global Manager is running, you must reconfigure the Global Manager to make the changes take effect. To reconfigure the Global Manager, invoke the following command from the ***BASEDIR****/smarts/bin* directory:

```
% sm_adapter -s <global_manager> ics/ICS_RemoteConfig.asl
```

For more information regarding this command, see *Reconfiguring the Global Manager* on page 24.

Note: Changes to the NotificationListSection are not available to console users until they restart the Global Console.

Defining Notification List Parameters

A notification list specifies the name of the notification list and one or more filters. Notification lists are defined in the NotificationListSection of the *ics.conf* file. The NotificationListSection is divided into one or more NotificationList subsections, each subsection defines a single notification list.

The following example shows the syntax of two NotificationList sections: Default and Maintenance. The Default notification list specifies an expression filter. The second notification list, Maintenance, does not specify a filter, meaning it matches all notifications. The Maintenance notification list also changes the heading of the Owner column to Responsible.

For the purpose and description of the default notification lists, see *About the Default Notification Lists* on page 58.

```
NotificationListSection
{
        NotificationList
        {
             Name      = "Default";
#             ASLFilter = "ics/nl-sample-filter.asl";
             Filter    = condition(
                            match("Owner", "~maint")
                            and
                            match("Owner", "~SYSTEM")
                            );
        }

        NotificationList
        {
             Name      = "Maintenance";

             ColumnHeading
             {
                   ColumnName = "Owner";
                   Heading    = "Responsible";
             {
        }
}
```

For information regarding the purpose of the default notification lists, see *About the Default User Profiles* on page 29 and *About the Default Notification Lists* on page 58.

Table 14 describes the fields of the NotificationList subsection.

FIELD	DESCRIPTION
Name	Name of the notification list. Must be unique within the NotificationListSection.
Filter and ASLFilter	[Optional] If the filter field is omitted, the notification list matches all notifications. You can specify two types of filters: • An expression filter with matching criteria. An expression filter is specified by the Filter field and is defined within the NotificationList subsection. For more information regarding expression filters, see *Expression Filters* on page 55. • An ASL filter is specified by the ASLFilter field that identifies the ASL filter file. The ASL filter file must be located in the ***BASEDIR****/smarts/local/rules/ics* directory. The name of the ASL filter must be preceded by by the name of the directory under ***BASEDIR****/smarts/local/rules* where it is located. For more information regarding ASL filters, see *ASL Filters* on page 57.
ColumnHeading	Indicates the beginning of a column heading section. Used to customize the names of column headings for notifications that pass by this notification filter. See *Customizing Column Headings in the Notification Log* on page 53.
ColumnName	Name of the column heading that is being changed. This must be the name of a notification attribute. For a list of notification attributes, see *Attributes for Matching Notification Properties* on page 112.
Heading	Name of the new column heading.

Table 14: NotificationList Fields

Customizing Column Headings in the Notification Log

You can also use the NotificationListSection of the *ics.conf* file to change the column headings used throughout the Global Console. This enables you to customize the existing column headings.

The following example illustrates the syntax for specifying customized column headings. The column Severity is changed to “Critical Routers”. As a result, “Critical Routers” will replace “Severity” throughout the Global Console.

```
NotificationListSection {
   NotificationList {
         Name   = "RouterNotifications";
         Filter = condition(
                        match("ElementClassName", "Router")
                           );
```

```
                ColumnHeading {
                        ColumnName = "Severity";
                        Heading    = "Critical Routers";
                              }
          }
      }
```

Note: If you rename one of the UserDefined column headings, you also need to modify the HookScript to populate these fields with information. For more information, see *Customizing User Defined Notification Attributes* on page 58.

Creating Filters for Notification Lists

You can use a notification list filter to send a subset of the available notifications to a particular client. For example, you can create one notification list that displays system notifications to system operators and another notification list that displays network notifications to network operators.

Notification lists are also used by non-console clients. For example, suppose you want a notification adapter to respond to a particular notification. You can create a notification list that only listens for that notification. You can create a notification list for each such adapter.

When the Global Manager receives events from an underlying Open Integration server, a notification list determines what events the Open Integration servers sends to the Global Manager. The *dxa-oi.conf* file specifies the name of the notification list. The parameters of the notification list, such as the filter, are specified in the *icoi.conf* file of the Open Integration server. If you want to change the notifications that Global Manager receives, you have two options:

- Change the filters of the notification list in the *icoi.conf* file that is used by Global Manager.
- Define a new notification list in the *icoi.conf* file and update the *dxa-oi.conf* file to use the new notification list.

Filter Types

You can define two types of filters for a notification list: an expression filter or an ASL filter. An expression filter, indicated by the Filter field name, is defined within the NotificationList subsection. An ASL filter, indicated by the ASLFilter field name, is defined in a separate ASL file.

Expression Filters

An expression filter uses matching criteria to filter against the value of a specified notification attribute. For the complete list of notification attributes, see *Attributes for Matching Notification Properties* on page 112.

Note: The match conditions use the attribute name, not the Notification Log column heading.

The following examples show the syntax of an expression filter.

```
Filter = condition(
                match("NotificationAttribute", "value")
                  );

Filter = condition(
                match("NotificationAttribute", "value")
                and
                match("NotificationAttribute", "value")
                  );
```

Table 15 describes the parameters of an expression filter.

PARAMETER	DESCRIPTION
Filter	Indicates that this is an expression filter.
condition	Keyword. It is case-sensitive.
match	Keyword that specifies a filter rule. The rule matches the matching pattern against the value of the specified notification attribute.
NotificationAttribute	Name of the notification attribute. Must be one of the attributes listed in *Attributes for Matching Notification Properties* on page 112.
value	Matching pattern that is compared against the value of the notification attribute. For more information regarding wildcards, see *Wildcard Patterns* on page 117.
and	Keyword used to specify multiple match conditions.

Table 15: Expression Filter Fields

The following examples illustrate different expression filters. The filter in the first example matches all notifications with a severity of one or two.

```
NotificationList
{
        Name   = "SeverityNotifications";
        Filter = condition(
                       match("Severity", "<1-2>")
                        );
}
```

The filter in the following example matches all notifications with a value of Router in the ElementClassName attribute.

```
NotificationList
{
        Name   = "RouterNotifications";
        Filter = condition(
                       match("ElementClassName", "Router")
                        );
}
```

The filter in the following example uses an implicit "or" condition. When you specify a list of filters as shown in the example, a notification passes the filter if it matches any of the conditions. In this example, a notification passes the filter if it has a severity of one or two *or* an impact greater than five.

```
NotificationList
{
        Name   = "TroubleTicket8145all";
        Filter = condition(
                       match("Severity", "<1-2>")
                        );
        Filter = condition(
                       match("Impact", "<5->")
                        );
}
```

The filter in the following example matches all notifications that have an owner assigned to them unless that owner is SYSTEM.

```
NotificationList
{
        Name   = "OwnerNotifications";
        Filter = condition(
                     match("Owner", "[A-z]*")
                     and
                     match("Owner", "~SYSTEM"
                      );
}
```

The filter in the following example matches all notifications for routers with a value of one in the severity field.

```
NotificationList
{
        Name   = "RouterNotifications";
        Filter = condition(
                       match("ElementClassName", "Router")
                       and
                       match("Severity", "1")
                         );
}
```

The key to creating a successful filter is understanding the types of values that an attribute can have. For example, the value of the Certainty field can be a floating point number. A matching pattern of <0-100> only matches whole numbers and would not match a value of 98.5. In this case, you need to create a matching pattern that treats each digit separately. A better matching pattern is <0-100>*.

ASL Filters

If an expression filter does not provide enough flexibility to define filtering criteria, you can use an ASL filter. An ASL filter is an ASL program that processes each notification.

The following example describes the syntax for specifying an ASL filter in a notification list.

```
ASLFilter = "ics/nl-sample-filter.asl";
```

Table 16 describes the parameters of an ASLFilter.

PARAMETER	DESCRIPTION
ASLFilter	Indicates that this notification list uses an ASL file to filter notifications.
ics/ns-sample-filter.asl	Name of the ASL program that filters notifications. This program must be located in the ***BASEDIR****/smarts/local/rules/ics* directory.

Table 16: ASLFilter Fields

An example ASL filter script, *nl-sample-filter.asl*, is included in the ***BASEDIR****/smarts/local/rules/ics* directory. The following example is the ASL filter from that file. It filters for notifications with the word Failure in the EventName attribute.

The name of the notification object is assigned to the variable NotificationName. If the value of EventName includes Failure, the value of Result is set to TRUE and the notification passes the filter.

```
START do {
      notification = object(notificationName);
      if (notification->EventName == "Failure")
      {
         result = TRUE;
      }
      else
      {
         result = FALSE;
      }
}
```

About the Default Notification Lists

Two notification lists, Default and Maintenance, are defined in the *ics.conf* file. The Default notification list is assigned to any client for which a notification list is not defined. It contains an expression filter that defines the following conditions. If a notification does not pass any one of the conditions it is rejected.

- The first condition matches any notification in which the value of the Owner attribute is not "maint".
- The second condition matches any notification in which the value of Owner attribute is not "SYSTEM".

The Maintenance notification list does not specify a filter. Because of this, it matches all notifications.

You can assign one of these pre-defined notification lists to users, as described in *Defining User Profiles* on page 27, or create your own. The purpose of the default notification lists, and how they are incorporated into the default user profiles is described in *About the Default User Profiles* on page 29.

Customizing User Defined Notification Attributes

A notification object also includes ten user-definable attributes. By default, these attributes have no value.

To populate the attributes with values, you must create an ASL program and specify it in the HookScript field of a DomainType subsection. See *Defining Domain Parameters* on page 31 for a description of the HookScript field.

A sample ASL program, *dxa-sample-hook.asl*, is included in the ***BASEDIR****/smarts/local/rules/ics* directory. This ASL program receives a handle to the notification object. You need to provide any additional functionality. Your ASL script should query the Global Manager or the underlying domain for additional information.

For example, you can populate one of the columns with the number of affected customers (instances of the ServiceSubscriber class) by counting the number of impacted subscribers.

To give the column names in the Global Console more meaningful names, use the method described in *Customizing Column Headings in the Notification Log* on page 53.

6

Topology Configuration for the Global Manager

This chapter describes the topology functions of the Global Manager. Topics include ensuring a consistent view of topology, organizing topology into groups, and creating topology for the Service and Application maps.

Topology Synchronization

A Global Manager imports topology information from the underlying domains specified in its *ics.conf* configuration file. To maintain an up-to-date representation topology, the Global Manager synchronizes its topology with an underlying domain when any one of the following occur:

- The underlying domain is disconnected or restarted.
- The Global Manager is started.
- The underlying domain performs a full or incremental discovery, rediscovers an object, or a manual discovery is initiated.
- A change is made is made to the DomainType section of the Global Manager's *ics.conf* configuration file. When the Global Manager is reconfigured, it automatically synchronizes its topology with all of the underlying domains.

Ensuring a Consistent Representation of Topology

A Global Manager imports topology information. Because the topology information comes from disparate sources, it is important that the Global Manager present a correct and consistent representation of the topology. This is especially true when two or more domains manage the same devices.

Newer versions of InCharge applications include an expanded and updated list of certified devices. This might mean that a device classified as Uncertified or Node by an older version of InCharge is classified as a Router or Switch by a newer version of InCharge.

For best results, SMARTS recommends that you upgrade existing InCharge applications to the most recent version. If that is not possible, use the same version of InCharge for all the underlying applications.

Two issues can arise when the Global Manager receives inconsistent topology information:

- The Global Manager imports information about two or more devices with the same name but the devices are classified differently in their respective underlying domains.
- The Global Manager imports the same device from two or more underlying domains but the device is named differently in each underlying domain.

Note: The Global Manager does not import instances of the Unsupported and Undiscovered classes. However, the topology of a Global Manager may include instances, such as Hosts, with a value of Undiscovered for their Certification attribute.

Same Device Classified Differently in Separate Underlying Domains

The Global Manager can receive conflicting topology information when two underlying domains discover the same device but classify it differently. The scenarios where this may occur are:

- A device has the same name but is an instance of different classes in two or more underlying domains. In addition, the device is classified as Uncertified or Node in one or more of the underlying domains. When this occurs, the Global Manager replaces an instance of a less specific class, Uncertified or Node, with an instance of a more specific class. Incoming event information for the device is consolidated to the instance in the Global Manager's topology.

  For example, *device1.mydomain.com* is classified as Uncertified in one underlying domain and classified as a Router in a second underlying domain. The Global Manager classifies the device as a Router in its topology. All incoming notifications related to *device1.smarts.com* are associated with the Router instance in the Global Manager's topology.

- A device has the same name but is an instance of different classes in two or more underlying domains. In this case, the device is not classified as Uncertified or Node in any underlying domain. The Global Manager classifies the device according to the first topology information it receives from an underlying domain. Relationship information for the device is updated during consecutive topology synchronizations. Incoming event information for the device is consolidated to the instance in the Global Manager's topology.

  For example, *device2.mydomain.com* is classified as a Host in one underlying domain and a Router in a second underlying domain. If the first underlying domain synchronizes first with the Global Manager, the Global Manager classifies the device as a Host in its topology. However, any relationship information for the router from the second underlying domain is preserved by the Global Manager and added to the Host instance. All incoming notifications related to *device2.mydomain.com* are associated with the Host instance in the Global Manager's topology.

Same Device Named Differently in Separate Underlying Domains

The Global Manager creates its topology based on the names of the devices it imports from the underlying domains. When the Global Manager imports the same device from two or more underlying domains with the same name, it creates a single corresponding device in its own topology. The Global Manager associates any incoming events from the underlying domains that are related to this device with the single device in its own topology.

When the Global Manager imports the same device from two or more underlying domains and each domain gives the device a different name, the Global Manager creates unique elements in its topology for each device.

For example, one underlying domain discovers a device and gives it the name *device3*. A second underlying domain discovers the same device but gives it the name *device3.mydomain.com.* When the Global Manager receives topology information from these underlying domains, it creates two instances in its topology—one named *device3* and one named *device3.smarts.com*.

For more information about InCharge discovery and the convention InCharge applications use to name devices, see the *InCharge Configuration and Administration Guide*.

Organizing Topology with Groups

Grouping provides a method by which you can organize topology elements. With Service Assurance, you can create groups and organize topology elements to help you more efficiently manage large numbers of elements.

Before you start, you may find it useful to devise a strategy around which you organize topology elements into groups. Common strategies include organizing by:

- Business units
- Geographical regions
- Resources

General Properties of Groups

A group is a user-defined collection of instances from the Global Manager's topology. A group consists of members or child groups. A *member* is a topological element such as a switch. A *child group* (or subgroup) is another group, which may be a collection of members or additional subgroups. A group that contains child groups is referred to as the *parent* group.

Parent and child groups are organized into a tree structure. At the root of each tree is a top-level group. Each top-level group is a distinct organization of groups and members—its configuration or removal does not effect other top-level groups. In the Map Console, top-level groups are displayed directly beneath the icon for the Global Manager.

- A member is also a member of the groups above it within the same group hierarchy.
- Within a single group hierarchy, an element cannot be a member of more than one group. An element can belong to two or more groups that descend from different top-level groups.
- You cannot create a circular group where a group is specified as a subgroup of itself or one of its subgroups.

Before we explain how to create groups, you may find it helpful to understand how groups are displayed in the Global Console.

How Groups Are Displayed in the Map Console

The Map Console only displays the members of a group when that group does not contain any child groups. If a group contains one or more child groups as well as members, only the child groups are displayed.

When you create groups to display them in the console maps, SMARTS recommends that you create a "catch-all" group within a parent group. The "catch-all" group should contain all the members that do not belong to any of the other subgroups. This technique prevents a parent group from containing both members and child groups.

Types of Groups

Service Assurance Manager supports two types of groups: *selective groups* and *hierarchical groups*. A selective group is a group whose members are determined by a matching pattern that you specify through the Global Console. Hierarchical groups are specified in a data file which is then imported by the Global Manager.

In addition to the methods by which they are created, there exist several other differences between selective and hierarchical groups. These differences are described in the following section.

Properties of Selective Groups

Selective groups contain three properties that distinguish them from hierarchical groups: priority, matching criteria, and target classes. These properties help to determine what elements become members of a selective group.

Matching Criteria

Matching criteria are attributes defined in the ICIM model that you use to determine what elements are eligible to become a member of a group. When you create a selective group, you specify a matching pattern that is compared against the attributes of the element. If the pattern matches the value of the specified attribute, the element becomes a member of the group. A matching pattern is comprised of one or more characters and wildcards. If you do not specify a matching pattern, all managed elements that pass the target class filter match the group; priority will determine if any elements become members.

Table 17 lists attributes against which you can apply a matching pattern. The group's target class determines which attributes are available to match against. For example, if the target class is IPNetwork, attributes that describe a managed system, such as Certification, are not listed.

ATTRIBUTE	DESCRIPTIONS
Certification	Level of certification assigned to this device during discovery. Possible values include: UNCERTIFIED, GENERIC, TEMPLATE, CERTIFIED, or VALIDATED.
CreationClassName	Name of the class of which the managed element is an instance. This is used as the ClassDisplayName attribute in notifications affecting this element.
Description	A brief description of the element.
DisplayClassName	Same as creation class name.
DisplayName	Name of the managed element. For systems, DisplayName and Name are usually the same.
IsManaged	Determines if the system is monitored by Global Manager. Note that unmanaged elements do not appear in the Global Manager topology. Value is TRUE or FALSE.
Location	A brief textual description of the system's physical location.
Model	Vendor's name for the system.
Name	Name of the managed element. For systems, Name and DisplayName are usually the same.
PrimaryOwnerContact	Information on how to contact the system's owner.
PrimaryOwnerName	Name of the system's owner.
ServiceName	Name of external system used to import attributes and events.

ATTRIBUTE	DESCRIPTIONS
SystemName	Name of the system that contains this managed element.
Type	Classifies the type of system. Possible values include: Bridge, Host, Hub, Node, Other, Probe, Router, RSFC, RSM, Switch, and TerminalServer.
Vendor	Name of the system's manufacturer.

Table 17: Attributes for Matching Criteria

For information regarding the wildcards you can use to build a matching pattern, see *Wildcard Patterns* on page 117.

Priority

Priority distinguishes between groups at the same level of the hierarchy with the same parent. Each such group is automatically assigned a different priority. When a topology element matches the pattern of two different groups, it becomes a member of the group with the higher priority. Because of this, you should assign a higher priority to a group with a stricter matching pattern. If a group has a high priority and it matches all the topology elements, it will contain all the available members.

Target Classes

A target class acts like a filter, allowing only those elements that are instances of the target class, or one of its subclasses, to become members of the group. Managed elements must pass the target class filter before they are compared against the matching criteria.

When you create a child group, the child group should have the same target class as its parent. The exception to this rule is when the new target class is a subclass of the parent group's target class. For example, the ICIM_ManagedSystemElement class is near the top of the ICIM hierarchy. All of the other target classes are a subclass of ICIM_ManagedSystemElement. If ICIM_ManagedSystemElement is the target class of a parent group, you can select a different target class for a child group.

Similarly, VLAN, NetworkConnection, and IPNetwork are subclasses of ICIM_LogicalLink. If ICIM_LogicalLink is the target class of the parent, you can choose one of these three classes as the target class for a child group.

Table 18 lists the target classes you can assign to a selective group when you create it. Classes are listed in the order that they appear in the ICIM class hierarchy. The description indicates when a class is a subclass of another target class.

TARGETCLASS	DESCRIPTION
ICIM_ManagedElement	Base class for the ICIM system element hierarchy. This is the broadest target class.
UnitaryComputerSystem	Represents a single computer system. This is the superclass for the Bridge, Host, Hub, Probe, MSFC, Router, RSFC, RSM, Switch, TerminalServer, and Node classes.
ApplicationService	Represents service provided by software. Examples include e-mail, Web server, and database applications.
LogicalLink	Represents a link between two endpoints. Examples include database transactions, IP networks, and cables.
VLAN	Virtual LAN typical in switched networks. VLAN is a subclass of LogicalLink.
NetworkConnection	A connection between two routers, typically a virtual circuit. NetworkConnection is a subclass of LogicalLink.
IPNetwork	Subnet of an IP network. IPNetwork is a subclass of LogicalLink.
ServiceSubscriber	Customer who receives services provided through a service offering.
ServiceOffering	A service provide to customers.

Table 18: Target Classes for Selective Groups

Creating Selective Groups

You create and edit selective groups using the Global Console. Creating or editing groups through the console requires administrator privileges from the Global Manager. Operators with monitoring privileges can view groups in the Topology Browser and Map Console but are not able to create or edit groups.

You should be aware of the following points when creating selective groups:

- Top-level selective groups cannot contain members, only child groups. As such, you need to create both a top-level group and one or more child groups to assign topology elements to a group.
- By default, each top-level group contains all the topology elements that match its target class and matching criteria.

- Each group is identified by a unique name. The name is displayed in group maps and in the Topology Browser. It is the value of the DisplayName attribute for the group instance.
- Hierarchical groups are visible in the Group Definition window but cannot be edited.

Layout of the Group Definition Window

You create and edit groups through the Group Definition window. You open this window by selecting **Groups** from the *Configure* menu of the Global Console.

The Group Definition window is divided into two panels. The left panel displays the Global Manager, groups, and the group members. When you select a group in the left panel, the Properties, Priorities, and Matching Criteria tabs display in the right panel of the Group Definition window.

Note: When you select a top-level group, only the Properties tab is displayed. Top-level selective groups do not have priority or matching criteria.

Priority and matching criteria determine which topology elements are members of each group. If you have worked with configuration groups (Polling Groups and Threshold Groups) for InCharge applications, the priority and matching criteria for topology groups function similarly.

For more information about how priority and matching criteria, see *Properties of Selective Groups* on page 65.

The toolbar of the Group Definition window contains four buttons:

- **Delete** removes the specified group from the topology of the Global Manager. If the deleted group contains child groups, the child groups are also deleted.

Note: The **Delete** command does not remove hierarchical groups.

- **Regroup** tells the Global Manager to rebuild the selective groups *from the selected group down to the bottom of the group hierarchy.* If you select the Global Manager icon and invoke Regroup, the Global Manager rebuilds all of the selective groups. This, however, is not usually necessary. It is more efficient to regroup a section of the group hierarchy when there are large number of topology elements in each group.

You need to invoke **Regroup** after you make changes to the priority or matching criteria of a group. The Global Manager automatically regroups the topology when it synchronizes its topology with the underlying domains.

Note: The **Regroup** command does not affect hierarchical groups.

- **Create Top Level Group** displays the New Group dialog where you specify the name, description, and target class for the group. This command is only available when the Global Manager is selected.
- **Create New Group** displays the New Group dialog where you specify the name, description, and target class for a child group. This command is available when a group is selected in the group tree hierarchy in the left panel of the Group Definition window.

You can also find these commands under the Group menu. In addition, the Group menu also contains the **Save Groups** command. When you invoke **Save Groups**, the Global Manager saves its in-memory database to the repository file.

Method for Creating Selective Groups

To create a selective group, use the Create Top Level Group or Create New Group command.

1. Select *Groups* from the *Configure* menu of the Global Console. This displays the Group Definition window.

   Note that this requires administrator privileges. The *Configure* menu is not displayed for users with monitoring privileges.

2. To create a top-level group, select the Global Manager icon and click the **Create Top Level Group** toolbar button. Alternatively, right-click on the Global Manager and select **Create Top Level Group** from the pop-up menu.

   To create a child group, select the parent group and click the **Create New Group** toolbar button. Alternatively, right-click on the parent group and select **Create New Group** from the pop-up menu.

   This displays the New Group dialog.

3. Specify a name, description, and target class for the group. After you finish specifying values for these three fields, click **OK**.

The new group displays in the Group Definition window. You can also see the group as an instance of the SelectiveGroup class in the Topology Browser Console or as a group in the Groups tab of the Map Console.

4. Change the priority of the new group. If there are other groups at the same level of the group hierarchy, they are listed under the Priorities tab. By default, a new group is assigned the lowest priority.

   To change a group's priority, select the group whose priority you wish to change. Under the Priorities tab, click the up arrow to give the group a higher priority or click the down arrow to give the group a lower priority.

5. Click the Matching Criteria tab and specify a matching pattern for the group. By default, a new group does not contain matching criteria, meaning it matches all possible elements.

   For example, if you want to create a group that includes systems from a certain geographical area then you might match against the value of the Type and Location attributes. First, specify a pattern that matches the type of systems you want to group. To create a group of routers you would add Type as a matching criteria attribute and specify a matching pattern of Router. Next, specify a pattern that matches against the geographical locale specified in the system's Location attribute. To match against routers in New York, you could add Location as a matching criteria attribute and specify a pattern of "*NY*|*New York*". This pattern would match against "NY" and "New York" anywhere in the Location attribute. Table 17 lists the classes whose values you can match against.

6. Select the parent for the newly created group, or the Global Manager, and click the **Regroup** toolbar button. You can also right-click on the parent and select **Regroup** from the pop-up menu.

Note: To create a "catch-all" group, create a group with no matching criteria and assign it the lowest priority.

Editing the Properties of a Selective Group

You can edit the description, target class, priority, and matching criteria of a selective group. Similar to creating a group, you must attach to Global Manager with administrative privileges and select *Groups* from the *Configure* menu.

You can edit all the properties of a group before applying the changes. However, if you select another group before clicking Apply, your changes are not applied. In this case, the console displays a dialog window asking if you want to abandon the changes that have not been applied.

1. Select the group whose properties you wish to edit. The properties of the group are displayed in the right panel of the Group Definition window.
2. The description and target class of the group are displayed under the Properties tab.
   - To change the target class, select a class from the pop-up menu.
   - To change the description, edit the text in the Description text box.
3. Select the Priority tab to change the priority of the group.
4. Select the Matching Criteria tab to change the matching pattern of the group.
5. Click **Apply**. If you change the target class, priority, or matching criteria, you also need to click **Regroup**.

Creating Hierarchical Groups

This section describes how to create hierarchical groups, including the syntax of the group data file and how to load the group information into the Global Manager.

Creating hierarchical groups is a three step process.

- Create one or more hierarchical group data files that specify the groups and their members.
- List the hierarchical group data files in the configuration file of the Global Manager.
- Reconfigure the Global Manager.

You create hierarchical groups by specifying the groups, child groups, and members in one or more data files. These data files must be located in the ***BASEDIR****/smarts/local/conf/ics* directory and their names specified in the BusinessSection of the *ics.conf* file.

After you edit the group data file to specify group information, import this information into the Global Manager by invoking the command to reconfigure the Global Managers or by invoking the regroup command to process only the BusinessSection of the *ics.conf* file. Administrative privileges for the Global Manager are required to invoke either command, as described in *Loading Hierarchical Group Data into the Global Manager* on page 75.

Creating Hierarchical Group Data Files

The syntax for group data files provide a means for specifying groups, children (subgroups), and members. The following example illustrates the syntax of a hierarchical group data file.

```
HierarchicalGroup NewYork children NY-Routers "NY Customers"
HierarchicalGroup NY-Routers members Router::nyc1
HierarchicalGroup "NY Customers" file:/opt/dev/incharge-sa/
smarts/conf/ics/ny-customers.data
```

This example illustrates the following syntactic rules of hierarchical group data files:

- Each line that specifies a hierarchical group must start with the keyword *HierarchicalGroup*.
- Elements of a hierarchical group must be separated by one or more spaces.
- The name of the group follows the keyword *HierarchicalGroup*. The name of the group must be unique for all existing groups.
- If the name of a group contains a space, it must be enclosed in double quotes.
- The keyword *children* indicates that this line specifies subgroups of the named group. For example, NY-Routers and "NY Customers" are child groups, or subgroups, of the group named NewYork.
- The keyword members indicates that this line specifies members of the named group. For example, the router nyc1 is a member of the group named NY-Routers. You must specify the class name and the instance name of the topology element, separating them with a double colon (::).
- You cannot specify child groups and members in the same line. You can, however, use multiple lines to specify members or children for the same group.

- You can specify a list of members in a member file. For example, the file *ny-customers.group.data* lists members of the "NY Customers" group. However, you must specify the full path to the file, using the correct syntax for the host operating system.

The following example shows the syntax of a file that lists the members of a group. Because the hierarchical group data file specifies the name of the group, you only need to specify the *<class>*::*<instance>* pairs, one per line, for each member.

```
Router::nycbrd1
Router::nycbrd3
Switch::nycs2
```

Syntax of the Group Data File in ics.conf

The name of the file or files that define the hierarchical group data must be specified in the *ics.conf* file, located in the ***BASEDIR****/smarts/local/conf/ics* directory. The following example shows the syntax of the BusinessSection of the *ics.conf* file.

```
BusinessSection
{
    Name = "topology-group.data.template";
    Name = "service.data.template";
}
```

The BusinessSection is used to specify the data files for hierarchical groups and service topology. You can use any number of files by specifying additional "Name" lines, as shown in the following example.

```
BusinessSection
{
    Name = "new-york-group.data";
    Name = "albany-group.data";
    Name = "san-francisco-group.data";
}
```

Creating a Hierarchical Group Data File

Service Assurance includes an example data file, *topology-group.data.template*, for specifying hierarchical groups. It is located in the ***BASEDIR****/smarts/local/conf/ics* directory. Any hierarchical group or service topology data files you create and specify in the *ics.conf* file must also be located in this directory.

To create a hierarchical group data file, make a copy of the example file and rename it appropriately. Create groups and populate them with topology elements from the topology of the Global Manager. Finally, edit the BusinessSection of the *ics.conf* file to specify your newly created file, as described in the following section.

Loading Hierarchical Group Data into the Global Manager

There are two methods for loading the hierarchical group information into the Global Manager: reconfiguring the Global Manager and regrouping the group data files. Remember the following to decide which command to use:

- Use the reconfigure command to load the group data files for the first time or to reload the *ics.conf* file if you edit the BusinessSection.
- Use the regroup command to regroup the group data from files previously loaded by the Global Manager. If you edit the hierarchical group data files already imported by the Global Manager, it is not necessary to reconfigure the Global Manager. If you edit the *ics.conf* configuration file, you must reconfigure the Global Manager.

When either method is invoked, the Global Manager processes the group and service topology data specified in the BusinessSection. If the Global Manager does not encounter any errors, it removes all of the current group and service information from its topology and replaces it with the information contained in the data files.

Note: Both reconfiguring and regrouping update the service topology, if any. For more information about importing service topology, see the *InCharge Service Assurance Manager User's Guide for Business Impact Manager*.

After the hierarchical group data is loaded into the Global Manager, you can view hierarchical groups through the Global Console.

Reconfiguring the Global Manager

To reconfigure the Global Manager, invoke the following command from the ***BASEDIR***/*smarts/bin* directory:

```
# sm_adapter -s <global_manager> ics/ICS_RemoteConfig.asl
```

Depending on your security configuration, you may be prompted for your InCharge user name and password.

When the Global Manager reloads the *ics.conf* file, it sends output to *stdout* verifying that it was able to read each section of the configuration file or report an error if one was encountered.

If the Global Manager encounters an error, it does not update its configuration.

Regrouping Hierarchical Group Data

To regroup the hierarchical group and service topology data files, invoke the following command from the ***BASEDIR****/smarts/bin* directory:

```
▼ % dmctl -s <global_manager> invoke GA_DaemonDriver::
   ICS-Group-Driver start ▲
```

Note: The command must typed as one line.

Depending on your security configuration, you may be prompted for your InCharge user name and password.

When the Global Manager regroups the topology information in the data files, it first checks for errors. If the Global Manager encounters an error, it writes output to the terminal or its log file and does not process the data files.

Building Topology for Services and Applications Maps

Using the Global Console, you can modify the topology of the Global Manager to extend the Services and Applications maps.

- You can create new instances of service and application topology and add them to Services and Applications maps.
- Create links between instances on Services and Applications maps.
- Add existing topology elements to Services and Applications maps.

The types, or classes, of elements you can create, link, or add depend on the type of map. For more information about Services and Applications maps, see the *InCharge Service Assurance Manager Operator's Guide*.

To perform these functions from the Global Console requires a user account with administrative privileges.

Note: Services maps are provided with InCharge Business Impact Manager and Applications maps are provided with InCharge Applications Services Manager. For more information about Business Impact Manager, see the *InCharge Service Assurance Manager User's Guide for Business Impact Manager* and for more information about Application Services Manager, see the *InCharge Service Assurance Manager Web Portal Configuration Guide*.

Toolbar Buttons for Editing Topology

Table 19 describes the toolbar buttons that are used to create, link, and add topology elements on Services and Applications maps. If a function is not available on a particular map, or the user does not have sufficient permission to perform the action, the toolbar buttons are gray.

BUTTON	DESCRIPTION
■+	Create Node makes a new topology element and adds it to the map.
■×	Remove Node deletes the selected topology element or link. You can only remove nodes and links that you have added or created with the Global Console.
🔗	Link Node creates a relationship between two nodes.
▦	Synchronize Topology makes the Global Manager send topology updates to its clients.

Table 19: Toolbar Buttons For Editing Topology

When you click the **Synchronize Topology** toolbar button, the Global Manager sends the changes you have made to the topology and maps to all of its clients. As a result, your changes become available to other Global Console users. If you plan to make several changes, invoke **Synchronize Topology** after you have completed your changes.

If you exit the Map Console, or reset the Services or Applications map you are working on, before invoking **Save Map** or **Synchronize Topology**, your changes are not saved.

Modifying Services Maps

You can edit the topology of a Services map by performing one or more of the following actions:

- Create instances of the ServiceOffering and ServiceSubscriber classes that are automatically added to the Services map and the topology.
- Add any topology element to a Services map.
- Link ServiceOffering nodes to any other node on a Services map.

Creating New Topology Elements for a Services Map

You can create new map nodes of the ServiceOffering and ServiceSubscriber classes. When you create a new node, it is automatically added to the topology and the active map.

1. Open the Services map that you want to modify.
2. Click the **Create Node** toolbar button. The Create Map Node dialog box displays.
3. Select the type of element you want to create from the Select Class drop down menu.
4. Type the name you want to give this node.
5. Click **Apply** to add this node to the map and create additional nodes, or click **OK** to add this node to the map and close the dialog box.
6. Link the new map node to an existing map node. For more information about linking nodes on a Services map, see *Linking Topology Elements on a Services Map* on page 79.

Adding Topology Elements to a Services Map

You can add any existing element from the topology to a Services map. This includes selective and hierarchical groups.

1. Open the Services map that you want to modify.
2. In the Map Console topology browser, right-click on the instance that you want to add to the map and select **Add to map** from the pop-up menu.
3. Link the new map node to an existing map node. For more information about linking nodes on a Services map, see *Linking Topology Elements on a Services Map* on page 79.

Linking Topology Elements on a Services Map

After you add nodes to a Services map, you should link them to related topology elements. The root-cause and impact analysis performed by InCharge software depends on the relationships (links) between topology elements.

Table 20 describes the types of links you can create on a Services map.

SOURCE NODE	RELATED NODE	RELATIONSHIP	EDGE
ServiceOffering	ServiceSubscriber	Subscribers/Subscriptions	Dotted arrow
ServiceOffering	Any	MemberOf/PartOf	Dotted arrow

Table 20: Links For Services Maps

To link nodes on a Services map, perform the following steps:

1. Press the **Ctrl** key and select both nodes between which you are establishing a link. You should select the source node first, which is where the link or relationship originates. For Services maps, the source node is always a Service Offering.
2. Click the **Link Nodes** toolbar button. The Global Console automatically creates the appropriate relationship, as described in Table 20.

Modifying Applications Maps

You can modify an Applications maps by performing one or more of the following actions:

- Create nodes of the Application and ApplicationCluster class that are automatically added to the Applications map and the topology.
- Add nodes of the ApplicationService and UnitaryComputerSystem classes, and their subclasses, to Applications maps.
- Link nodes of the ApplicationService class, and its subclasses, to other nodes on an Applications map.

Note: In most cases, application topology should be discovered by the system or application agents deployed to manage them. If you manually create application topology, you must ensure that the names you enter correspond to the notifications that management agents generate.

Creating New Topology Elements for an Applications Map

You can create new nodes of the following classes:

- ApplicationService
- MgmtService
- Application
- ApplicationCluster

When you create a new topology element, it is automatically added to the active map and listed in the Map Console topology browser.

1. Open the Applications map that you want to modify.
2. Click the **Create Node** toolbar button. The Create Map Node dialog box displays.
3. Select the type of element you want to create from the Select Class drop down menu.
4. Type the name you want to give this node.
5. Click **Apply** to add this node to the map and create additional nodes, or click **OK** to add this node to the map and close the dialog box.
6. Link the new map node to an existing map node. For more information about linking nodes on an Applications map, see *Linking Topology Elements on an Applications Map* on page 81.

Adding Topology Elements to an Applications Map

You can add instances of the ApplicationService and UnitaryComputerSystem classes, and their subclasses, to an Applications map.

1. Open the Applications map that you want to modify.
2. In the Map Console topology browser, right-click on an instance that you want to add to the map and select **Add to map** from the pop-up menu.
3. Link the new map node to an existing map node. For more information about linking nodes on a Services map, see *Linking Topology Elements on a Services Map* on page 79.

Linking Topology Elements on an Applications Map

After you add nodes to an Applications map, you should link them to related topology elements. The root-cause and impact analysis performed by InCharge software depends on the relationships (links) between topology elements.

Table 21 describes the types of links you can create on an Applications map. You can create links for instances of the classes specified by the Source Node and Related Node columns.

SOURCE NODE	RELATIONSHIP	RELATED NODE	EDGE
Application	PartOf/ComposedOf	ApplicationCluster	Dashed line
ApplicationService	HostedBy/HostsServices	UnitaryComputerSystem	Solid arrow
ApplicationService	Consumes/ConsumedBy	Transaction/Session	Solid arrow
Transaction/Session	ProducedBy/Produces	ApplicationService	Solid arrow

Table 21: Links For Applications Maps

To link nodes on an Applications map, perform the following steps:

1. Press the **Ctrl** key and select both nodes between which you are establishing a link. Select the source node first, which is where the link or relationship originates.
2. Click the **Link Nodes** toolbar button.
   - When only one type of link can be created, the Global Console automatically creates the appropriate relationship.
   - When more than one type of link can be created between the selected nodes, the Global Console displays the Link selected map nodes dialog box. Select the appropriate link from the drop-down menu.
3. Click **Apply** to create the link between the nodes and to create additional links, or click **OK** to create this link and close the dialog box.

Creating Elements from the Map Console Topology Browser

In addition to creating topology elements on a Services or Applications map, you can also create topology elements from the Topology Browser in the Map Console. You can create elements of the following classes:

- ServiceSubscriber
- ServiceOffering

- ApplicationService and its subclasses
  - Application
  - ApplicationCluster
  - MgmtService and its subclasses

To create a topology element of one these classes, right-click the class in the Map Console's Topology Browser and select **Create**.

Note: To remove a topology element created from the Map Console Topology Browser, you must first add it to a map.

Removing Map Nodes and Links

You can remove the nodes and links you created with the **Create Node**, **Add Node**, or **Link Node** toolbar buttons. Remember that the **Create Node** toolbar button creates an instance that is added to the topology. If you remove it, that instance is deleted from the topology.

The following steps describe how remove a new object:

1. Select the node or link that you want to remove. You can only remove links or nodes that were created through the Global Console.
2. Click the **Remove** button.
3. The node or link is removed from the map. If the node was created with the Create Node toolbar button, the instance is also deleted from the topology. If a link is removed, the relationship between the nodes is deleted from the topology.
4. Click the **Synchronize Topology** button.

7

Tool Configuration for the Global Manager

This chapter describes the programmatic tools you can create and use with InCharge Service Assurance Manager. Tools are programs that can be invoked automatically by the Global Manager or through the console by an operator. Topics include how to create, configure, and invoke tools.

In addition, InCharge Service Assurance Manager (Service Assurance) provides tools that integrate with the Remedy Action Request System (Remedy AR) by taking problem notifications from a Global Manager and translating them into open, get, or close ticket requests. These requests then get submitted to Remedy AR. This chapter describes how to configure and use these tools.

Types of Program Tools

A tool is a script that is typically invoked in response to a notification. Such a response might be to ping an affected device or to open a trouble ticket. Tools can also be invoked on topology elements. Service Assurance supports three types of tools:

- Server tools
- Client tools
- Automated tools

Server and client tools are invoked by an operator using the Global Console. Two tool submenus, Tools and Local Tools, are displayed in a pop-up menu when an operator right-clicks on a notification or a device.

- The Tools menu displays a list of available server tools. Server tools are invoked by the Global Manager on the host where the Global Manager is running. Server tools are available to any console attached to the Global Manager.
- The Local Tools menu displays a list of available client tools. Client tools are invoked on the host where the Global Console is running. Local tools are only available when the *tools.conf* file is configured on the host where the console is running. Because of this, different local tools can be made available on different hosts.

An automated tool is a type of server tool that is invoked using the **sm_adapter** command. Once started, an automated tool runs in the background and responds to notifications that match its notification list. For example, an automated tool could respond to a notification by opening a trouble ticket.

How Tools are Invoked

This section describes how to invoke tools and where the output of server and client tools is displayed.

Invoking Server and Client Tools

Server and client tools are invoked through the Global Console. Such tools are always invoked on a particular target object by the console operator. The target object can be a notification or a topology element such as a router. When an operator right-clicks on the target, the pop-up menu lists the available tools.

For information about invoking server and client tools from the Global Console, see the *InCharge Service Assurance Manager Operator's Guide*.

Viewing the Output of a Server or Client Tool

Server and client tools, depending on their purpose, may produce output. For example, the output of a ping tool can tell whether the ping completed successfully.

By default, all client tools display output. For server tools, you can configure whether a tool displays any output. When a server tool is configured to display its output, the output is echoed to *stdout* and collected by the Global Manager. The Global Manager sends the output back to the console where the tool was invoked.

The output of server and client tools is displayed in the Tool Output window. The Tool Output window opens when the tool is invoked. Output is written to the window while the tool is running. If a tool produces an error, the error message is also displayed in the Tool Output window. If an error occurs for a server tool that is not configured to display output, the Tool Output window displays with the return code of the tool.

Invoking Automated Tools

An automated tool runs as an adapter on the same host as the Global Manager. You start an automated tool by invoking the **sm_adapter** command and supplying a list of arguments. After you start an automated tool, no further intervention is required. The adapter uses a notification list to listen for particular notifications. When it receives a matching notification, it automatically responds. For more information about creating an automated tool, see *Configuring an Automated Tool* on page 97.

Information Recorded to a Notification's Audit Log

When a server or automated tool is invoked a notification, information about the tool is recorded to the notification's Audit Log. This information includes:

- Date and time when the tool was invoked.
- Name of user under which the tool process ran.
- Whether the tool completed successfully.
- Name of the tool.

Security Configuration for Tools

If a tool invokes a SMARTS utility, such as dmctl, or otherwise initiates a connection to the Global Manager, you must configure security to enable that connection. An essential piece of information is knowing the user name under which the tool is invoked.

- Client tools are invoked under the user name of the operator who started the Global Console.
- Server tools are invoked under the user name that the Global Manager is running under. Remember, server tools are invoked on the same host as the Global Manager.
- Automated tools are invoked under the user name that started the **sm_adapter** command.

If a client tool needs to connect to the Global Manager and perform an action that requires administrative privileges, you have two choices:

- Provide the operator with administrative privileges.
- Create a server tools that performs the action.

Because server tools run under the user name that invoked the Global Manger process, this user must have an authentication record in the *clientConnect.conf* and *serverConnect.conf* files on the host where the Global Manager is running. You must configure the authentication records for this account so that prompting is not required. Prompting should not be used for automated tools.

Creating a Tool Script

This section describes how to create a tool script. Server, client, and automated tools are all scripts that are invoked on a notification or an element.

Follow these steps to write a tool script:

1. Determine the action to be performed by the tool and write the script.
2. Save the script to the proper location.
3. Test the tool script to ensure that it executes properly.
4. Assign the proper read/execute permissions to the tool script.

Sample Tool Scripts

SMARTS provides a number of sample tool scripts that, with minor modifications, work on most systems. The main purpose of these scripts, however, is to provide examples that you can examine when developing your own tools.

Table 22 lists the sample server and automated tools and provides a brief description. Server and automated tools are located in the ***BASEDIR****/smarts/actions/server* directory.

To modify one of the sample tool scripts, open it with the *sm_edit* utility to create a local copy of the script.

Note: The tool names listed in Table 22 are the names of the tool scripts, not the names that are displayed in the console.

SERVER TOOL	DESCRIPTION
`ics-closetkt`	This tool inserts a static value into the field of a notification. The context and status criteria specify that this tool is only active for notifications with the text OPEN in the TroubleTicketID field.
`ics-opentkt`	This tool inserts a static value into the field of a notification. The context and status criteria specify that this tool is only active for notifications without the text OPEN in the TroubleTicketID field.
`ics-ping-interface`	This tool pings an IP interface. The context criteria specify that this tool is active for notifications with a value of "Interface" for the ClassName attribute.
`ics-ping-IP`	This tool pings an IP interface. The context criteria specify that this tool is active for notifications with a value of "IP" for the ClassName attribute.
`ics-ping-all`	This tool pings all the IP interfaces associated with a device. The context criteria specify that this tool is active when the target is a UnitaryComputerSystem. As such, valid targets include both notifications and instances. When invoked, this tool retrieves all of the IP addresses associated with the target instance from one or more underlying domains. Two variables that control the behavior of the tool: • PING_ALL_DOMAINS controls whether the tool retreives IP addresses from each underlying domain that manages the instance. When set to 0 (zero), the default, the tool retrieves IP addresses from a single domain. When set to 1 (one), the tool retreives IP addresses from all domains. • PING_ONCE controls whether the tool pings each retrieved IP address or whether it stops after the first successful ping. When set to 0 (zero), the tool stops after the first successful ping. When set to 1 (one), the default, the tool pings all available IP addresses.

SERVER TOOL	DESCRIPTION
`ics-ping-device`	This tool pings the IP address of the SNMP agent for the affected element.The context criteria specify that this tool is active when the target is a UnitaryComputerSystem. As such, valid targets include both notifications and instances. When invoked, this tool retrieves the IP address of the SNMP agent associated with the target instance from one or more underlying domains. The tools script includes one variable that controls the behavior of the tool: • PING_ALL_DOMAINS controls whether the tool retreives IP addresses from each underlying domain that manages the instance. When set to 0 (zero), the default, the tool retrieves IP addresses from a single domain. When set to 1 (one), the tool retreives IP addresses from all domains.
`ics-telnet`	This tool opens a telnet session with the affected device. The *.sh* version of this script pings an IP address of the affected device before attempting to telnet. If the ping fails, the script pings another IP address on the device, and so on until a ping succeeds or all the pings fail. When a ping succeeds, the script telnets to the IP address where the ping succeeded. If this tool is invoked from a console running over X Windows, the script opens a separate window on the user's display to invoke the telnet session.

Table 22: Sample Server Tool Scripts

Table 23 lists the sample client tools and provides a brief description of each. Client tools are located in the ***BASEDIR****/smarts/actions/client* directory.

CLIENT TOOL	DESCRIPTION
`SmGetEnv`	This tool parses the environment variables passed by the Global Manager to the tool script. It also prints all environment variables to the Tool Output window. A version of this tool script is also provided in Perl.
`SmLaunchPerlScript`	This tool launches the *SmGetEnv.pl* tool script to illustrate how you can launch Perl scripts from a tool. You must edit this script to specify the location where Service Assurance is installed.
`adminconsole`	This tool opens an InCharge Administration Console and automatically attaches to the source domain listed in the notification. The target for this script must be a notification because the source domain is not one of the default variables returned by the Global Manager for an element. You must edit this script to specify the location where the InCharge Console is installed.

CLIENT TOOL	DESCRIPTION
`browser`	This tool opens a Web browser. You must edit this script to specify the location where your browser is installed.
`ciscoworks`	This tool opens a Web browser that loads the administration page for CiscoWorks. You must edit this script to specify the location of your browser and the host where CiscoWorks is running.
`newconsole`	This tool opens a new Global Console. You must edit this script to specify the location where the Global Console is installed.
`newicconsole`	This tool opens an InCharge Administration Console. You must edit this script to specify the location where the InCharge Console is installed.
`pinger`	This tool pings the name of the specified element.
`Reporting`	This actions opens a Web browser with the URL set to the location of the Crystal Enterprise EportFolio application. You must edit this script to specify both the Web browser and the URL.

Table 23: Sample Client Tools Scripts

Remember that tool scripts are system-specific. For UNIX systems, a tool is typically invoked by a shell script (*/bin/sh*) while on Windows systems, a tool is typically invoked by the Windows command interpreter (cmd.exe). You can also write a tool script in a language such as Perl, which runs on different platforms with little or no modification.

How Information is Passed to a Tool Script

When a tool is invoked, the attributes of the tool target are automatically passed to the tool script. The attributes are passed to the tool script as environment variables in the form:

```
SM_OBJ_<NAME>=<value>
```

The `<NAME>` parameter identifies the attribute and the `<value>` parameter contains the attribute's value. For example, the ClassName attribute would be passed in the environment variable:

```
SM_OBJ_CLASSNAME=Router
```

Your tool script must parse these environment variables and extract the information required by the program invoked by the tool script. For example, if you want a tool that telnets to a device, you need to extract the environment variable that contains the name or IP address of the device.

The tool target can either be a notification or an instance. For both types of targets, a tool script receives the attributes listed in Table 24 as environment variables.

ENVIRONMENT VARIABLE	DESCRIPTION
SM_REMOTE_USER_NAME	Specifies the InCharge user name of the person that invoked the tool.
SM_SERVER_NAME	Specifies the name of the Global Manager that executes the tool script.

Table 24: Attributes Passed To All Tool Scripts

Table 25 lists the attributes that are passed to the tool script when the target is a topology element.

ELEMENT TARGET ATTRIBUTES	DESCRIPTION
SM_OBJ_CLASS_NAME	Class name of the system. For example, Router.
SM_OBJ_INSTANCE_NAME	Name of the system.
SM_OBJ_DOMAIN_NAME	Name of the Global Manager.

Table 25: Attributes for Element Targets

Table 26 lists the attributes that are passed to the tool script when the target is a notification.

NOTIFICATION TARGET ATTRIBUTES	DESCRIPTION
SM_OBJ_Acknowledged	Specifies whether the notification is acknowledged. TRUE if the notification has been acknowledged, FALSE if not.
SM_OBJ_Active	Specifies whether the notification is active. TRUE if the notification is active, FALSE if not.
SM_OBJ_Category	Type of notification sent by the Global Manager. Possible values include: • BackplaneUtilization • Error • Performance • PowerSupply • Resource • SystemConnectivity • Temperature • VLANConnectivity

Notification Target Attributes	Description
SM_OBJ_Certainty	Confidence that this notification is the correct diagnosis. Value ranges from 0 to 100
SM_OBJ_ClassDisplayName	Name of the ClassName that is displayed to the user.
SM_OBJ_ClassName	Name of the class where the event occurred. May not be the same as SM_OBJ_ClassDisplayName.
SM_OBJ_ElementClassName	Class name of the managed element most closely related to this event.
SM_OBJ_ElementName	Name of the managed element most closely related to this event.
SM_OBJ_EventDisplayName	Name of the EventName that is displayed to the user.
SM_OBJ_EventName	Name of the event. May not be the same as SM_OBJ_EventDisplayName.
SM_OBJ_EventText	A description of the notification.
SM_OBJ_EventType	MOMENTARY when the notification has no duration. DURABLE if the notification has a period for which it is active, such as a link failure.
SM_EventState	State of the event. Possible values include: • ACTIVE • WAS_ACTIVE • SUSPENDED • INACTIVE The UNITIALIZED state is not relevant here.
SM_OBJ_FirstNotifiedAt	Time, in seconds, when the notification first became active.
SM_OBJ_Impact	Numeric value that indicates the effect of this event on related elements.
SM_OBJ_InMaintenance	TRUE if the device is in maintenance mode, FALSE if not.
SM_OBJ_InstanceDisplayName	Name of the InstanceName that is displayed to the user.
SM_OBJ_InstanceName	Name of the instance where the event occurred. May not be the same as SM_OBJ_InstanceDisplayName.

Notification Target Attributes	Description
SM_OBJ_IsRoot	TRUE if the notification is a root cause, FALSE if not.
SM_OBJ_LastChangedAt	Time, in seconds, when the status of the notification last changed.
SM_OBJ_LastClearedAt	Time, in seconds, when the notification was last cleared.
SM_OBJ_LastNotifiedAt	Time, in seconds, when the notification was last notified
SM_OBJ_Name	InternalEventHandle for the notification.
SM_OBJ_OccurrenceCount	Number of times the notification has occurred.
SM_OBJ_Owner	Name of the person responsible for this notification. Value is SYSTEM when acknowledged by the Global Manager.
SM_OBJ_Severity	Level of severity for this notification. 1 = CRITICAL 2 = MAJOR 3 = MINOR 4 = UNKNOWN 5 = NORMAL
SM_OBJ_SourceDomainName	Name of the underlying domain that sent this notification. If more than one domain is listed, the names are separated by commas.
SM_OBJ_TroubleTicketID	Trouble-ticket number associated with this notification.
SM_OBJ_UserDefined1-10	Ten user-defined fields.You can populate these fields with data with an ASL hook script.

Table 26: Attributes for Notification Targets

In addition to the attributes listed above, a server tool script receives the DISPLAY environment variable. The value of this variable is used to determine the location of the user's X Window System display. For more information about invoking server tools through an X Windows System, see *Running Tools Over X Windows* on page 100.

Where to Save Tool Scripts

After you write the script, you need to save it to the proper location and ensure that it is executable. A server tool, which includes automated tools, must be located in the ***BASEDIR****/smarts/local/actions/server* directory on the host where the Global Manager is running. The Global Manager must be able to execute the script.

A client script must be located on the host where the Global Console is installed. The proper location is the ***BASEDIR****/smarts/local/actions/client* directory. The console user must be able to execute the script.

The next step is to configure your tool so that it appears in the Global Console. Server tools are configured in the *ics.conf* file and client tools are configured in a file named *tools.conf*.

Configuring a Server Tool

The configuration of a server tool is specified in the *ics.conf* file.This is the primary configuration file for the Global Manager; it is located in the ***BASEDIR****/smarts/local/conf/ics* directory.

If you edit the *ics.conf* file when the Global Manager is running, you must reconfigure the Global Manager to make the changes take effect. To reconfigure the Global Manager, invoke the following command from the ***BASEDIR****/smarts/bin* directory:

```
% sm_adapter -s <global_manager> ics/ICS_RemoteConfig.asl
```

For more information regarding this command, see *Reconfiguring the Global Manager* on page 24.

Note: Changes to the ToolSection are not available to console users until they restart the Global Console.

The following example shows the format of the ToolSection. You can specify as many entries as you wish. Server tools are displayed in the *Tools* menu of the Global Console in the order that they are listed in the ToolSection.

```
ToolSection {

   ProgramTool
   {
     Name = "Sample - Telnet";
     Program = "ics-telnet.sh";
     DisplayOutput = false;
     Timeout = 60;
     Context = condition(isa("UnitaryComputerSystem"));
   }

   ProgramTool
   {
     Name = "Sample - Open Trouble Ticket";
     Program = "ics-opentkt.sh";
     DisplayOutput = true;
     Timeout = 60;
     Context = condition(isa("ICIM_Notification"));
     Status = condition(match("TroubleTicketID", "~*OPEN*"));
   }
}
```

The ToolSection parameter identifies the section of the *ics.conf* file relevant to server tools. It is followed by an open curly brace ({); the last listed tool script must be followed by a corresponding close curly brace (}). Each parameter must be followed by a semicolon (;).

Table 27 describes the configuration parameters for each server tool.

PARAMETER	DESCRIPTION
Name	Name of the tool that is displayed in the console.
Program	Name of the program that the Global Manager executes when the tool is invoked. This parameter cannot contain path separators.
DisplayOutput (optional)	Specifies whether the tool script should display output to the console. If this parameter is set to true, the console will open the Tool Output window immediately after invoking the tool. If false, the console only opens the Tool Output window if the tool returns an error. If this parameter is omitted, its defaults to true.
Timeout	Specifies the maximum number of seconds to wait for the tool script to complete. If the tool script does not complete within the specified time-out interval, the Global Manager terminates the tool.

PARAMETER	DESCRIPTION
Context (optional)	Determines whether the tool appears in the console's pop-up menu when the user right-clicks on a tool target. A tool always appears in the console menu when this parameter is omitted. See *Setting the Context and Status Criteria for Server Tools* on page 95.
Status (optional)	Determines whether the tool is enabled or disabled in the console's pop-up menu. The tool is always enabled when this parameter is omitted. See *Setting the Context and Status Criteria for Server Tools* on page 95.

Table 27: Server Tool Configuration Parameters

Setting the Context and Status Criteria for Server Tools

Context and status criteria help determine whether a tool should be available for the selected target. If an operator selects an instance, then the target is an instance. If an operator selects a notification, the target can either be the notification or the instance where the notification occurred, as identified by the ElementClass and ElementInstance attributes.

Context criteria determine whether a tool is displayed in the Tools menu for a particular target. The context criteria is checked when the console receives the notification and it is not checked again until the notification is archived and re-notified. Because of this, you should apply context checking to attributes whose values do not change.

- Category
- ClassDisplayName
- ClassName
- ElementClassName
- ElementName
- EventDisplayName
- EventName
- EventType
- InstanceDisplayName
- InstanceName

Although you can apply context checking to other attributes, the results would be based on the initial value of the attribute, not its current value.

Status criteria determines whether a tool is active. The text for a disabled tool is gray in the Tools menu and cannot be invoked. You should apply status criteria to attributes whose value might change.

- Acknowledged
- Active
- Certainty
- ClearOnAcknowledge
- EventState
- EventText
- FirstNotifiedAt
- Impact
- InMaintenance
- IsRoot
- LastChangedAt
- LastClearedAt
- LastNotifiedAt
- OccurrenceCount
- Owner
- Priority
- Severity
- SourceDomainName
- TroubleTicketID

The tool target determines what attributes are passed to the tool to be matched against the context criteria. For example, when the target is an element, you can match against the attributes specified in Table 25. If the target is a notification, the matching criteria determine which attributes are passed to the context criteria. If the matching criteria specify a notification attribute, then the attributes of a notification, from Table 26, are passed. If the matching criteria specify an element, then the attributes of an element are passed.

Syntax of Context and Status Criteria

The Context and Status fields use the same syntax. Each field begins with keyword **condition**.

The following example illustrates the use of these two parameters.

```
Context:  condition (isa ("ICIM_Notification"));

Status:   condition (
                 match ("TroubleTicketID", "")
                 and
                 match ("InMaintenance", "FALSE")
);
```

For the Context parameter, the **isa** condition checks whether the tool target is a notification. If the selected target is a notification, the Context parameter matches and the tool is listed in the console.

For the Status parameter, the **match** conditions are checked against the values of two notification attributes. The value of the TroubleTicketID attribute is checked against a wildcard pattern. In this example, the condition checks to see if the TroubleTicketID field is empty, which would indicate whether a trouble ticket has been issued for this notification. The value of the InMaintenance attribute is checked to see if it is FALSE, which indicates whether the managed element is undergoing maintenance. If both conditions for the Status parameter are true, then this tool is active. If the Status is false, the tool will be listed in the Tools menu but it will not be active.

Configuring an Automated Tool

The configuration options for an automated tool are specified as arguments to the **sm_adapter** command when you invoke it at the command line. Table 28 lists the arguments you should specify when invoking an automated tool.

ARGUMENT	DEFAULT VALUE	DESCRIPTION
`--server=`	INCHARGE-SA	The name of the Global Manager from which the adapter receives its notification list and notifications.
`--subscribe=`	n/a	Name of the notification list received from the Global Manager. This value must correspond to a notification list specified in the *ics.conf* file.
`-DNotifyAction=`	n/a	Name of the tool script to invoke when a notification is active. This argument is optional.
`-DClearAction`	n/a	Name of the tool script to invoke when the notification clears. This argument is optional.
`-DLogActions`	FALSE	Specifies whether the adapter should log each action to *stdout*. This argument is optional.
`--output`	n/a	Redirects the output from the adapter to a log file in ***BASEDIR****/smarts/local/logs*. The `--DLogActions` argument must be true to enable output.
`ics/auto-action.asl`	n/a	This script is required to perform the automated action.

Table 28: Arguments to sm_adapter for Automated Action

The following example shows the arguments for an automated tool that invokes the "Open Trouble Ticket" script when a notification becomes active (is notified) and invokes the "Close Trouble Ticket" script when the notification becomes inactive (clears).

```
# BASEDIR/smarts/bin/sm_adapter--server=INCHARGE-SA \
                 --subscribe=TicketingNL  \
                 -DNotifyAction="Open Trouble Ticket"  \
                 -DClearAction="Close Trouble Ticket"  \
                 -DLogActions=TRUE  \
                 --output  \
                 ics/auto-action.asl
```

When the script that runs an automated tool is invoked, the context and status criteria for the tool are matched against the target. If the context and status criteria do not match, the tool is not invoked.

If you do not want to provide access to your automated script from the Global Console, you may prefer to use the Script Notifier to automate the action. For more information about the Script Notifier, see the *InCharge Notification Adapters User's Guide*.

Configuring a Client Tool

The configuration file for client tools is named *tools.conf* and it is located in the ***BASEDIR****/smarts/local/conf/console* directory on the host where the Global Console is running. You must properly configure this file on each host where the Global Console is used to invoke client tools.

The *tools.conf* configuration file provides the console with the following information:

- Name of tool script to display in the Local Tools pop-up menu
- Target on which the console user can invoke the tool script

Each tool script must be specified on a separate line. The specification for a client tool is divided into two parts. The first part identifies the tool target and provides the name of the tool script as it should appear in the console. The second part, following the equals sign (=) specifies the path to the tool script.

The following is an example of a tool definition in the *tools.conf* file for a client tool script that targets instances.

```
INSTANCE.SCRIPT.Local
Ping=//c/InCharge/smarts/local/actions/client/pinger.bat
```

The first part of the tool definition is INSTANCE.SCRIPT.Local Ping. INSTANCE tells the console that the target for this tool script is an instance. The "Local Ping" parameter is the name of the tool script that will be displayed in the console. Following the equals sign (=) is the path to the tool script and the name of the script itself, *pinger.bat*.

If you want a client tool script that targets notifications, substitute NOTIFICATION for INSTANCE.

Note: The location of client tools is fixed. Client tools must be located in the ***BASEDIR****/smarts/local/actions/client* directory on the host where the Global Console is installed.

Running Tools Over X Windows

If necessary, you can configure a server tool to invoke an X Windows application. The console user that invokes such an action must run an X server in order to display the X window. When the Global Console attaches to the Global Manager, it passes the user's display information to the Global Manager.

The Global Console determines a user's display using the following rules:

- If the environment variable SM_DISPLAY is set, its value is passed to the Global Manager.
- If the environment variable DISPLAY is set, its value is passed to the Global Manager.
- The value is set to ***<host>***:0.0 where <host> is the name of the user's system.

Because X Windows applications are typically long running, you should take special care when writing the tool script to avoid unwanted interactions with the tool time-out specified in the ToolSection of the *ics.conf* file. You should write the tool script so that the X Window application runs in the background. This allows the tool script to exit and the Global Manager to stop the time-out. If the tools script runs in the foreground, the Global Manager will terminate the process when the time-out expires.

The *sm_xcmd* command is a utility provided by SMARTS for executing an X Windows application in the background. The utility performs a basic connectivity test of the user's X Windows display. The *ics-telnet.sh* server tool is an example of a tool script that uses *sm_xcmd* to run an application over X Windows.

Configuring and Using Remedy Tools

InCharge Service Assurance Manager provides tools that integrate with the Remedy Action Request System (Remedy AR) by taking problem notifications from a Global Manager and translating them into open, get, or close ticket requests. These requests are submitted to Remedy AR.

There are two ways that you can invoke a Remedy tool:

- Manually, you select a notification in the Global Console and select the appropriate Remedy server tool to open, get, or close a ticket associated with that notification

- Automatically, in response to a notification or a cleared notification. You automate Remedy tools as you would any other server tool, using *sm_adapter* command line options.

Prerequisites

For UNIX operating systems, the Remedy tools require Remedy AR System version 3.2, 4.0, or 4.5.

For Windows systems, the Remedy tools require Remedy AR System version 4.5.2. Also, for Windows only, two runtime library requirements also apply:

- The Remedy runtime library files (or a copy of those files) installed with the Remedy AR System, must reside on the host running the Global Manager.
- The PATH environment variable must include the path where the Remedy runtime libraries reside.

SMARTS recommends you set the PATH during the invocation of the Remedy tool. To do this, add the following lines to the *remedy-config* script, which is located in the ***BASEDIR****/smarts/local/actions/server* directory. For example, on Windows systems you would specify the following:

```
set REMEDY_LIB_PATH=C:/remedy/bin
set PATH=%PATH%;%REMEDY_LIB_PATH%
```

Alternatively, you can set the Remedy runtime library path by editing your PATH environment variable to include the Remedy runtime library path. In Windows choose Environment Variables from the Advanced tab of the System Control Panel. Then, select the PATH variable and add the Remedy runtime library path to it (for example, *C:\remedy\bin*).

Overview of Integration with Remedy

There are three Service Assurance Remedy tools interface with the Remedy AR System.

- Open Remedy Ticket
- Get Remedy Ticket
- Close Remedy Ticket

Figure 3 illustrates the ways to invoke a Remedy tool and identifies how the tools work. All the Remedy tools use the *sm_arclient* command-line utility to communicate with the Remedy AR System.

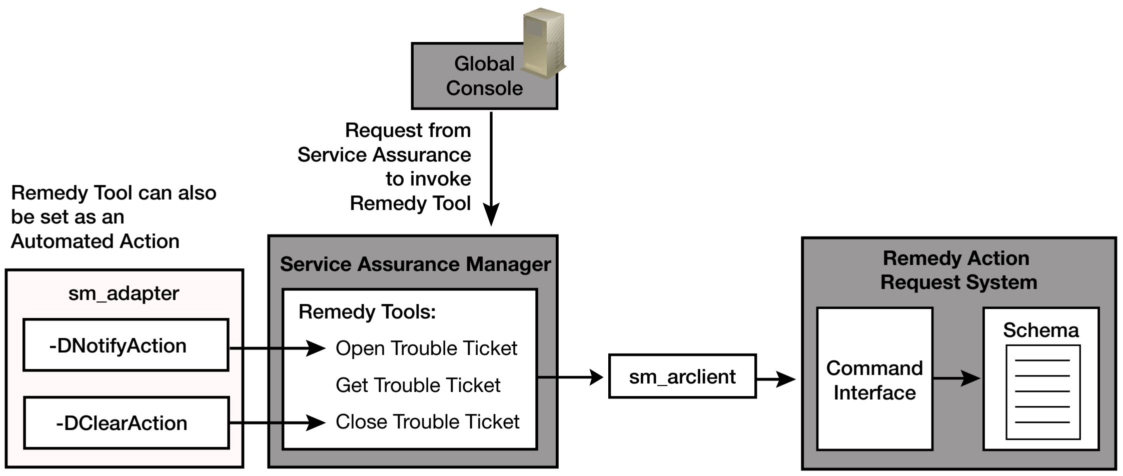
Global Console
Request from Service Assurance to invoke Remedy Tool
Remedy Tool can also be set as an Automated Action
sm_adapter
-DNotifyAction
-DClearAction
Service Assurance Manager
Remedy Tools:
Open Trouble Ticket
Get Trouble Ticket
Close Trouble Ticket
sm_arclient
Remedy Action Request System
Command Interface
Schema

Figure 3: Global Manager and the Remedy Action Request System

By default, the Remedy tools are configured to use the sample Remedy schema, *IC_SAM_Final.def,* which is located in the ***BASEDIR****/smarts/local/conf/remedy* directory. If you choose to use a different schema, you need to modify the tool accordingly. For more information see *Integrating the Remedy Tools With the Remedy AR System* on page 103.

Open Remedy Ticket Tool

The Open Remedy Ticket Tool first checks to see if a Ticket already exists for the selected notification. If a ticket is open, the tool changes the ticket status from closed to open. If a ticket does not exist for this notification, then the tool opens a Remedy ticket and populates the TroubleTicketID notification attribute with the Remedy Ticket ID.

Get Remedy Ticket Tool

The Get Remedy Ticket Tool retrieves Remedy ticket information associated with the selected notification, processes the ticket, and displays the information (mapping information to the appropriate ticket field names).

Close Remedy Ticket Tool

The Close Remedy Ticket Tool verifies that a Remedy ticket exists for the selected notification and changes the value TroubleTicketID attribute to indicate that the ticket is closed. The tool also changes the ticket's status in the Remedy system from open to closed.

Integrating the Remedy Tools With the Remedy AR System

Table 29 describes the files that are used to configure the Remedy tools. You may need to edit these files so that the Remedy tools function properly in your environment.

FILE NAME(S)	DESCRIPTION
remedy-opentkt	Tool that opens a Remedy Ticket for a given notification
remedy-closetkt	Tool that closes a Remedy Ticket associated with a given notification
remedy-gettkt	Tool that retrieves information about a Ticket and displays it if one exists for a given notification
remedy-config.sh	UNIX script that specifies host running Remedy AR System, name of Remedy schema, and Remedy system user name and password Service Assurance needs to access the AR System server. All other UNIX Remedy scripts call this script.
remedy-config.cmd	Windows script that specifies the host running Remedy AR System, name of Remedy schema, and Remedy system user name and password Service Assurance needs to access AR System Server. All other Windows Remedy scripts call this script.
remedy-fields.txt	A Windows-only file that defines mapping of Field IDs and Field Names defined in the Remedy schema. The *remedy-gettkt.cmd* uses this mapping to display ticket information by Field Name.
IC_SAM_final.def	A sample Remedy schema definition that includes information available from Service Assurance. If you use the sample schema, you need to import this definition file into your Remedy AR System. By default, all Remedy tool scripts use this schema.

Table 29: Remedy Integration Configuration, Script, and Schema Files

To configure the Remedy tools, complete the following steps:

1. Edit the *ics.conf* file to include the Remedy tools in the ToolSection. For details, refer to *Editing the ics.conf File* on page 104.
2. Change the configuration parameters in the *remedy-config* file to match your Remedy AR System configuration. If your operating system is UNIX then edit the *remedy-config.sh* file; if your operating system is Windows, edit the *remedy-config.cmd* file. For details, refer to *Configuring the remedy-conf File on page 104*.
3. If you choose to use a different Remedy schema from the one provided (*IC_SAM_final.def) and* you are operating in a UNIX environment, you need to perform the following additional steps to configure the tools:

- Edit the *remedy-config.sh* file to define the variable names used by the tools and map them to the Remedy Field IDs of your schema.
- Edit the Open Ticket and Get Ticket tools. Specifically, in the *remedy-opentkt.sh*, revise the MANDATORY_ARGS and/or OPTIONAL_ARGS in the `Main - Create Entry` section of the script so that these arguments represent your Remedy schema. Each argument is formatted as a triplet of Remedy Field ID, data type, and value.

  Also, in the *remedy-gettkt.sh*, replace the mapping in the `translateFieldName()` function with the Field IDs and Field Names you use in Remedy.

If you choose to use a different Remedy schema from the one provided (*IC_SAM_final.def*) *and* you are operating in a Windows environment, you need to perform the following additional steps to configure the tools:

- Make additional changes (same as mentioned above) to the *remedy-config.cmd* file.
- Edit the mapping of Remedy Field IDs and Field Names in the *remedy-fields.txt* file.
- Edit the Open Ticket tool. Make the same changes as mentioned above regarding the MANDATORY_ARGS and/or OPTIONAL_ARGS.

For general considerations, also refer to *Using a Different Remedy Schema* on page 105.

Editing the ics.conf File

By default, the Remedy tools are commented out. To use the Remedy tools, you need to remove the pound sign (#) from the beginning of each line in the ToolSection (ProgramTool subsections) that define the Remedy tools.

Note: Making changes to the ProgramTool section of the *ics.conf* file may require that the operator restart the Global Console.

Configuring the remedy-conf File

The ***BASEDIR***/*smarts/local/actions/server/remedy-config* file specifies information about your Remedy system. This information gets passed to the Remedy tools and allows the tools to communicate with the Remedy system.

For example, the *remedy-config* file specifies the host running the Remedy AR System, the Remedy schema, and the Remedy user name and password needed by the Global Manager to access the Remedy AR System Server. Table 30 describes the parameters you are required to provide in this file.

PARAMETER	DESCRIPTION
`AR_SERVER`	The host where the Remedy server is running.
`AR_SCHEMA`	The name of the Remedy schema. The default is the InCharge SAM schema.
`AR_USER`	The user specified in the Remedy system login.
`AR_PASSWORD`	The user password
`AR_SUBMITTER`	The user or system submitting the Remedy request. The default is InCharge SAM.

Table 30: remedy-conf Parameters

Using a Different Remedy Schema

If you choose to use a Remedy schema other than the one provided, consider the following before changing any of the Remedy scripts:

- Decide how to map your Remedy Fields to the Service Assurance notification attributes. You may want to provide certain values as constants and other values will need to come directly from specific notification attributes.
- Decide if you want to use variable names instead of Remedy Field IDs in the *remedy-config* file. Default variables are already defined in this file for the Service Assurance schema provided. If you choose to use variables, you will need to revise the variable names in the *remedy-config* file to meet your Remedy field requirements.
- Gain an understanding of how the *remedy-opentkt.sh* and *remedy-opentkt.sh* scripts use the **sm_arclient** command-line utility to communicate with the Remedy system. The open ticket scripts set up the Remedy fields as formatted triplets (defined as ARG variables) that contain a Field ID, data type, and value. You need to define triplets and ARG variables for the Remedy fields in your schema.

  For more information, refer to *About sm_arclient* on page 106.

- Gain an understanding of how the *remedy-gettkt* script replaces Remedy Field IDs with Field Names. For UNIX, the mapping is defined in the *remedy-gettkt.sh* file. For Windows, the mapping is defined in the *remedy-field.txt file.* You need to define the appropriate mapping for your Remedy schema.

About sm_arclient

Remedy tools use the *sm_arclient* utility to communicate with the Remedy AR System to create, delete, get and set information in Remedy. If you plan use your own Remedy schema, you need to understand how this utility interacts with Remedy.

The *sm_arclient* utility uses the following syntax:

```
% sm_arclient [options...] <command>
```

The *sm_arclient* options define parameters for the existing Remedy schema. Table 31 describes these options.

OPTION	DESCRIPTION	DEFAULT VALUE
`--server=<server>`	Remedy AR System server host.	localhost
`--user=<user>`	Remedy AR System user.	"Demo"
`--password=<password>`	Remedy AR System password.	""
`--language=<lang>`	Remedy AR System language.	""

Table 31: sm_arclient Options

There are four commands for *sm_arclient*. These commands are described in Table 32. Each command will include several required arguments.

COMMAND	DESCRIPTION
`createEntry`	Creates a Remedy Ticket
`deleteEntry`	Deletes a Remedy Ticket
`getEntry`	Writes a Remedy Ticket to a standard output.
`setEntry`	Updates one or more fields for a Remedy Ticket

Table 32: sm_arclient Commands

Each command should include arguments for the schema and field associated with each entry. For all commands except createEntry you must also include the AR entry ID parameter, which is the ticket ID returned when a ticket is created. Samples for each of these commands follow:

```
% createEntry <schema> <field>
% deleteEntry <schema> <entry>
% getEntry <schema> <entry> [<id>...]
% setEntry <schema> <entry> <field> ...
```

The <schema> argument refers to the AR schema name you are mapping to.

The <entry> argument refers to the AR entry ID (also referred to as the Field ID).

The <field> argument is a triplet comprised of <id>, <type>, and <value>.

The values for these are defined in Table 33.

FIELD	DESCRIPTION
id	AR entry id <string>
type	c - - CHAR
	d - - DIARY
	e - - ENUM
	i - - INTEGER
	n - - NULL
	r - - REAL
	t - - TIME
	u - - ULONG
value	Value of the field

Table 33: Description of <field> Arguments

Examples Using sm_arclient

This section provides sample options, commands, and arguments to the *sm_arclient* utility.

Sample 1

```
% sm_arclient -s <remedy_server> -u <username> -p <password>
createEntry "InCharge SAM Schema" 2 c "InCharge SA" 8 c
"Router Down" 1042601004 c "Router" 1042601003 c "moto-gw"
```

The preceding lines can be read as follows:

- `sm_arclient` – the comand to begin the action
- `-s <remedy_server>` – the name of the Remedy AR system host
- `-u <username>` – Remedy user name
- `-p <password>` – Remedy user password
- `createEntry` – the specific command for the AR action
- "InCharge SAM Schema" – the AR System schema name

Using the InCharge Service Assurance Schema, the entries created by the ID/Data type/Value triplets in the previous example are as follows:

- Field 2 (submitter) is "InCharge SA"
- Field 8 (short description) is "Router Down"
- Field 1042601004 (class name) is "Router"
- Field 1042601003 (instance name) is "moto-gw"

Note: The **sm_arclient** utility that interfaces with the Remedy system uses field IDs for arguments. We have substituted the field IDs with the actual parameters for readability.

Sample 2

```
% sm_arclient -s <remedy_server> -u <username> -p <password>
getEntry "InCharge SAM Schema" ticketidnnn
```

Sample 2 is the command to return a ticket ID (AR Entry ID) which is used to get or set one or more fields for this entry or to delete the entire entry. The ticket entry would be written to a standard output.

Sample 3

```
% sm_arclient -s <remedy_server> -u <username> -p <password>
deleteEntry "InCharge SAME Schema" ticketidnnn
```

Sample 3 is the command to use to delete a ticket entry from the Remedy AR server by naming the schema and the ticket ID number.

A

ICIM Classes Used with Matching Criteria

To configure the Global Manager, you may need to specify a matching pattern or matching criteria. The matching pattern or criteria are compared to the name and/or value of system and notification attributes. Tasks where you might specify a matching pattern include:

- Creating notification lists.
- Specifying tagging patterns.
- Determining the status or criteria of tools.
- Creating selective groups with the Global Console

Matching criteria are a string that is matched against the value of the specified attribute. The criteria can contain any combination of text, integers, and wildcards.

Classes and Attributes for Matching Managed Systems

Table 34 lists the attributes of the system class hierarchy. The ICIM_System class is at the top of the hierarchy. As you move down the table, the classes become increasingly more specialized until you reach classes that represent specific systems such as routers and switches. Note that classes lower in the hierarchy inherit the attributes of classes higher in the hierarchy.

System Class	Attributes	Type	Description
ICIM_System	CreationClassName	String	Name of the class from which the element was instantiated.
	DisplayClassName	String	Name of the system's class that is displayed in the console.
	DisplayName	String	Name of the system that is displayed in the console.
	Description	String	Description of the system.
	IsManaged	Boolean	Determines if the system is monitored by Global Manager. Note that unmanaged elements do not appear in the Global Manager topology.
	Name	String	Name of the system.
	PrimaryOwnerContact	String	Contact information for the primary owner of this system.
	PrimaryOwnerName	String	Name of the primary owner of this system.
	SystemName	String	Name of the system of which this component is a part of or this service is hosted by.
ICIM_ComputerSystem			ICIM_ComputerSystem does not add attributes to the system class hierarchy. A computer system is a collection of managed system elements such as file systems, processors, and memory.

SYSTEM CLASS	ATTRIBUTES	TYPE	DESCRIPTION
UnitaryComputerSystem	Certification	String	Level of certification assigned to this device during discovery. Possible values are: • CERTIFIED • VALIDATED • TEMPLATE • GENERIC • UNCERTIFIED • UNDISCOVERED • UNSUPPORTED
	Location	String	Description of the physical location of this system.
	Model	String	Vendor name for the system.
	Type	String	Type of the computer system. Possible values are: • BRIDGE • HOST • HUB • MSFC • NODE • PROBE • ROUTER • RSFC • RSM • SWITCH • TERMINALSERVER • UNCERTIFIED
	Vendor	String	Name of the system's manufacturer.
Bridge	Type	String	Unitary computer system that bridges packets between separate segments. Value is BRIDGE.
Host	Type	String	Unitary computer system that represents a workstation or server. Value is HOST.
Hub	Type	String	Unitary computer system that connects multiple segments. Value is HUB.
MSFC			Unitary computer system that represents a Multi Layer Switch Feature Card. An MSFC is a card installed into a switch to perform routing between VLANs. Value is MSFC.
Node	Type	String	Unitary computer system that has not yet been certified by discovery.
Probe	Type	String	Unitary computer system that monitors networks or systems. Value PROBE.

System Class	Attributes	Type	Description
Router	Type	String	Unitary computer system that routes packets between computer networks. Value is ROUTER.
RSFC	Type	String	Unitary computer system that represents a Router Switch Feature Card. An RSFC runs Cisco IOS router software and directly interfaces with Catalyst switches to provide inter-VLAN routing. Value is RSFC.
RSM	Type	String	Unitary computer system that represents a Router Switch Module. Often installed in switches to route packets between VLANs. Value is RSM.
Switch	Type	String	Unitary computer system that switches packets between separate segments. Value is Switch.
TerminalServer	Type	String	Unitary computer system that represents a terminal server or similar access device. Value is TERMINALSERVER.
Uncertified	Type	String	[Deprecated, replaced by Node] Unitary computer system that represents a system that has not yet been certified by discovery. Value is HOST.

Table 34: System Classes and their Attributes

Attributes for Matching Notification Properties

Table 35 lists the notification attributes that are contained in notifications generated by the Global Manager. When you create a matching pattern, use the name of the attribute, not the column heading from the Notification Log.

ATTRIBUTE	DEFAULT COLUMN NAME	VALUE TYPE	DESCRIPTION
Acknowledged	Acknowledged	Boolean	TRUE if the notification has been acknowledged, FALSE if not.
Active	Active	Boolean	TRUE if the notification is active, FALSE if not.
AuditTrail	n/a	String	Audit trail includes five fields separated by spaces. The fields include: • Serial number is a unique number that identifies the audit trail entry. • Timestamp identifies when this audit trail entry was written. • User is the InCharge user name associated with this audit trial entry. • Action type identifies the reason for the audit trail entry. • Text is a brief description of the audit trail entry.
Category	Category	String	Type of notification sent by the Global Manager. Possible values include: • Availability • Discovery • Error • IMPACT • Operational • Performance • PowerSupply • Resource • Temperature
Certainty	Certainty	Float	Confidence that this notification is the correct diagnosis. Value ranges from 0 to 100.
ClassDisplayName	Class	String	Name of the class that is displayed to the user.
ClassName	ClassName	String	Class name of the instance where this event occurred. This attribute, with InstanceName and EventName uniquely identifies this notification.
ClearOnAcknowledge	n/a	Boolean	Indicates that this event should be cleared when it is acknowledged. Default is FALSE.
ElementClassName	Element Class	String	Class name of the managed element most closely related to this event.
ElementName	Element Name	String	Name of the managed element most closely related to this event.

ATTRIBUTE	DEFAULT COLUMN NAME	VALUE TYPE	DESCRIPTION
EventDisplayName	Event	String	Name of the notification that is displayed to the user.
EventName	EventName	String	Name of this notification. This attribute, with ClassName and InstanceName, uniquely identify this notification.
EventState	Event State	String	Describes the state of the event. Value can be one of: • ACTIVE • WAS_ACTIVE • SUSPENDED • INACTIVE • UNITIALIZED
EventText	Event Text	String	A description of the notification.
EventType	Event Type	String	MOMENTARY when the notification has no duration. DURABLE if the notification has a period for which it is active, such as a link failure.
FirstNotifiedAt	First Notify	Integer	Time, in seconds, when the notification first became active.
Impact	Impact	Integer	Numeric value that indicates the effect of this event on related elements.
InMaintenance	In Maintenance	Boolean	TRUE if the device is in maintenance mode, FALSE if not.
InstanceDisplayName	Name	String	Name of the instance that is displayed to the user.
InstanceName	InstanceName	String	Name of the object where this notification occurred. This attribute, with ClassName and EventName, uniquely identify this notification.
IsRoot	IsRoot	Boolean	TRUE if the notification is an authentic problem (root cause), FALSE if not.
LastChangedAt	Last Change	Integer	Time, in seconds, when the status of the notification last changed.
LastClearedAt	Last Clear	Integer	Time, in seconds, when the notification was last cleared.
LastNotifiedAt	Last Notify	Integer	Time, in seconds, when the notification was last notified.
OccurrenceCount	Count	Integer	Number of times the notification has occurred.

ATTRIBUTE	DEFAULT COLUMN NAME	VALUE TYPE	DESCRIPTION
Owner	Owner	String	Name of the person responsible for this notification. Value is SYSTEM when acknowledged by the Global Manager.
Severity	Severity	Integer	Level of severity for this notification. 1 = CRITICAL 2 = MAJOR 3 = MINOR 4 = UNKNOWN 5 = NORMAL Note that only the numbers, not the text descriptions, are passed by the Global Manager.
SourceDomainName	Source	String	Name of the underlying domain that sent this notification. If more than one domain is listed, the names are separated by commas.
TroubleTicketID	Ticket ID	String	Trouble-ticket number associated with this notification.
UserDefined1-10	User Defined 1-10	String	Ten notification attributes can be defined by the user. You can set a value with a hook script specified in DomainType section of *ics.conf*.

Table 35: Notification Attributes

Note: For attributes that contain a time value, time is counted from Midnight, January 1st, 1970 (GMT). In the Global Console, these values are converted to a date and time.

B

Wildcard Patterns

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 36. 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 37.

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. 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. 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. For example, *server[789-].smarts.com* matches *server7.smarts.com* through *server9.smarts.com*, but not *server6.smarts.com*. It also matches *server-.smarts.com*. 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. For example, *98.49.<1-100>.10* matches a range of IP addresses from *98.49.1.10* through *98.49.100.10*. Example of an omitted high end of the range: <*50*-> matches any string of digits with a value greater than or equal to 50. Example of an omitted low end of the range: <-*150*> matches any value between zero and 150. 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. 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 36: Basic Wildcard Patterns

Special characters for compound wildcard patterns are summarized below.

&	"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. Example: *NY*&*Router* matches all strings which contain NY and also contain Router. 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. 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. Example: 10.20.30.*~10.20.30.50 matches all devices on network 10.20.30 except 10.20.30.50. 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 37: Compound Wildcard Patterns

Index